i.MX25 Multimedia Applications Processor Reference Manual

Supports
i.MX251 (MCIMX251)
i.MX253 (MCIMX253)
i.MX255 (MCIMX255)
i.MX257 (MCIMX257)
i.MX258 (MCIMX258)

IMX25RM
Rev. 2
01/2011

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: IMX25RM
Rev. 2, 01/2011

freescale™
semiconductor

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 18
Configurable Serial Peripheral Interface (CSPI)

Contents

Contents

Contents

Contents

Chapter 24
Enhanced SDRAM Controller (ESDRAMC)

Contents

Chapter 25
Fast Ethernet Controller (FEC)

Contents

Chapter 26
Controller Area Network (FlexCAN)

Contents

Contents

Chapter 30
IC Identification Module (IIM)

Contents

Contents

Contents

Chapter 34
ARM9 Platform Multi-Layer AHB Crossbar Switch (MAX)

Contents

Contents

Chapter 37
Pulse-Width Modulator (PWM)

Contents

Contents

Contents

Contents

Contents

Chapter 42
Shared Peripheral Bus Arbiter (SPBA)

Contents

Contents

Contents

Contents

Chapter 49
Wireless External Interface Module (WEIM)

Contents

Contents

i.MX25 Reference Manual
Book I

Rev. 2
01/2011

Chapter 1
IC Architecture Overview

This chapter introduces the i.MX25 architecture.

1.1 i.MX25 Overview

The i.MX25 is targeted for the automotive and general industrial markets. This multimedia applications processor has the right mix of high performance, low power, and integration to support the growing needs of the industrial and general embedded markets. The i.MX25 is partitioned into two major subsystems as shown in Figure 1-1: the ARM926 platform and the SDMA platform with an external memory interface (EMI).

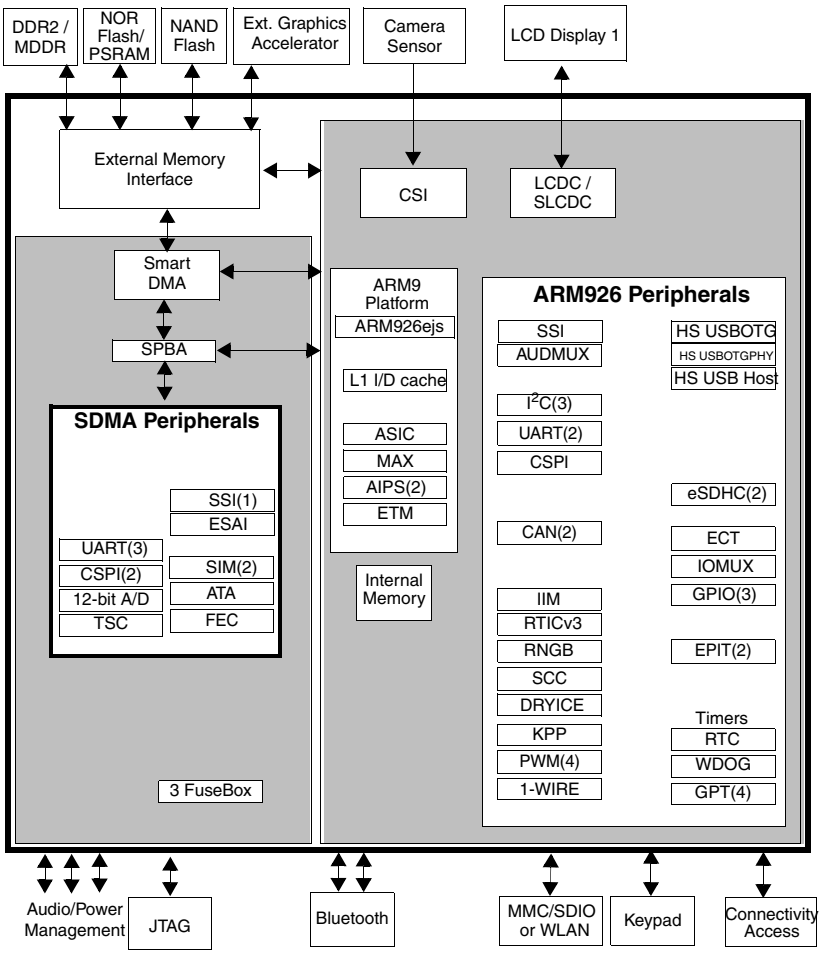


Figure 1-1. i.MX25 High-Level Block Diagram

1.1.1 Key Features

The i.MX25 is based on the ARM926 platform. The ARM926 platform has the following features:

- ARM926EJS processor
- 16 Kbyte L1 instruction cache
- 16 Kbyte L1 data cache
- 400 MHz maximum frequency of the core and L1 cache

To boost the performance, the following hardware accelerators are integrated:

- Alpha blending at the LCD controller
- 128 Kbyte internal SRAM
- A/D controller and integrated touch screen controller

Security functions are enabled and accelerated by the following hardware:

- Secure JTAG Controller (SJC). Protects JTAG from debug port attacks by regulating or blocking the access to the system debug features.
- Real-Time Integrity Checker type2 (RTICv3). RTIC type1, enhanced with SHA-256 engine.
- Secure RAM module and the Security Monitor (SCCv3). Provides 2 Kbytes of secure storage of sensitive information both in on-chip RAM and in off-chip, non-volatile memory.
- High Assurance Boot (HAB) with SHA-256.
- DryIce
  - Second level encrypted key with enhanced tamper detection erase capability
  - Tamper detectors
  - Secure real time clock
  - 32 KHz oscillator

The memory system consists of the following levels:

- Level-1 cache
  - Instruction (16 Kbyte)
  - Data (16 Kbyte)
- Level-2 memory
  - Boot ROM, including HAB (32 Kbyte)
  - Internal RAM (128 Kbyte)
  - Secure RAM (2 Kbyte)

The i.MX25 provides the following interfaces to external devices:

- Two Controller Area Network (CAN) interfaces
- Two CE-ATA, SDIO/MMC interfaces (up to 416 Mbps)
- Three Configurable Serial Peripheral Interfaces (CSPI) - supporting speeds up to 52 Mbps each
- DDR2, Mobile DDR, and SDRAM (up to 133 MHz) [DAH: Also supports PSDRAM]
- Ethernet 10/100 Mbps

- Flash controller - MLC, SLC NAND and NOR
- GPIO with interrupt capabilities
- Three I^2C (up to 400 Kbps each)
- JTAG
- Keypad port
- 1-Wire module
- Parallel camera sensor (4/8/10/16-bit data port for video color models: YCC, YUV, 30 MPix)
- Parallel display (primary up to 24-bit, 1024x1024)
- Parallel-ATA (P-ATA) -up to 66 MByte/s
- Four Pulse Width Modulators (PWM)
- Two Synchronous Serial Interfaces (SSI)
- One Enhanced Serial Audio Interface (eSAI)
- Five UART (up to 4.0 Mbps each)
- USB 2.0 Host with FS PHY
- USB 2.0 OTG (up to 480 Mbps) with HS PHY
- Two Subscriber Identification Modules (SIM)

1.2 Architecture Overview

1.2.1 Functional Domains Overview

The i.MX25 consists of the following major subsystems:

- ARM926 platform
- SDMA platform and EMI

1.2.1.1 ARM926 Platform Overview

The ARM926 platform is responsible for running the operating system and applications software, providing the user interface, and supplying access to integrated and external peripherals. The look and feel of the device depends on the software running on this processor, ultimately tying market acceptance to the availability of a wide variety of off-the-shelf, third-party software and development tools. Over the past couple of years, the ARM CPU family has emerged as the de-facto standard for mobile application processors. To leverage this growing software base, the ARM926 platform is based on the ARM architecture. The ARM926 platform is built around an ARM926EJS core with 16 Kbyte instruction and 16 Kbyte data L1 caches, an MMU, a multi-ported crossbar switch, and advanced debug and trace interfaces.

The ARM9 core is intended to operate at a maximum frequency of 400 MHz in order to support the required multimedia use cases, such as concurrent video playback QVGA at 30 fps and MP3 audio decode. Furthermore, an LCDC is integrated into the ARM926 platform to off-load the core from graphics overlay.

Peripheral functionality belonging to the ARM926 platform includes the user interface, connectivity, display, security, and memory interfaces and 128 Kbytes multipurpose SRAM. This SRAM can be used as audio RAM, scratch pad RAM by the ARM9, or it can be accessed by the LCDC for use as a display buffer.

1.2.1.2 SDMA Platform and EMI Overview

The shared domain is composed of the SDMA peripherals, a Smart DMA Engine (SDMA) and a number of miscellaneous modules. For maximum flexibility, some peripherals are directly accessible by the SDMA engine.

The external memory subsystem represents a significant investment in chipset cost and board area. High-performance processors such as the ARM9 require a significant amount of bus bandwidth to achieve their maximum potential. The challenge is to maximize the performance while minimizing the bandwidth. To achieve that, the i.MX25 includes a hierarchical memory architecture including L1 caches and M2 memory. This reduces the bandwidth demands for the external bus and external memory. The external memory subsystem supports a flexible external memory system, including support for SDRAM (SDR and DDR), DDR2, Mobile DDR (mDDR), and NAND flash.

1.2.2 Advanced Power Management Overview

To address the continuing need to reduce power consumption, the following techniques have been incorporated into the i.MX25 processor:

- Clock gating
- Power optimized synthesis
- Well biasing
- Dynamic Process and Temperature Compensation (DPTC)
- Dynamic Voltage and Frequency Scaling (DVFS)

Clock distribution circuits in digital ICs with the complexity of i.MX25 can consume as much as 40% of the total dissipated power. By inserting gating into the clock paths, unused portions of the chip can be disabled. Since static CMOS logic consumes only leakage power, significant power savings can be realized. The i.MX25 clock gating is inserted both manually on a large functional block basis and automatically within the blocks during logic synthesis. The i.MX25 integrates both clock gating and power optimization into the synthesis design flow.

Well biasing is applying a voltage that is greater than V_{dd} to the nwells and lower than V_{ss} to the P-wells. The effect of applying this well back bias voltage is to reduce the subthreshold channel leakage. For the 90-nm digital process, it is estimated that the subthreshold leakage is reduced by a factor of 10 over the nominal leakage.

Additionally, the supply voltage for internal logic can be reduced from 1.4 V to 1.2V to 1.0 V during periods of inactivity.

1.2.3 Modules Inventory

Table 1-1 describes the ARM926 core.

Table 1-1. Core Summary

Core Acronym	Core Name	Brief Description	Integrated Memory Includes
ARM9 or ARM926	ARM926 platform and memory	The ARM926 platform consists of the ARM926EJS™ core, the ETM real-time debug modules, a 5x5 multilayer AHB crossbar switch, and a “primary AHB” complex.	• 16-Kbyte Instruction L1 Cache • 16-Kbyte Data L1 Cache • 32-Kbyte ROM • 128-Kbyte RAM

Table 1-2 describes the modules on the chip, as well as a description of the functionality, as well as the names of the subsystem to which they belong.

Table 1-2. Digital and Analog Modules

Block Mnemonic	Block Name	Domain[1]	Sub-System	Brief Description	Chapter
ATA	ATA Module	SDMA	Connectivity Peripherals	The ATA block is a AT attachment host interface. Its main use is to interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device over a number of ATA signals.	—
AUDMUX	Digital Audio Mux	ARM	Multimedia Peripherals	The AUDMUX is a programmable interconnect for voice, audio, and synchronous data routing between host serial interfaces (SSIs) and peripheral serial interfaces (audio codecs). The AUDMUX has two sets of interfaces: internal ports to on-chip peripherals, and external ports to off-chip audio devices. Data is routed by configuring the appropriate internal and external ports.	—
CAN(2)	CAN Module	ARM	Connectivity Peripherals	The CAN protocol is primarily designed to be used as a vehicle serial data bus running at 1 Mbps.	—
CCM	Clock Control Module	ARM	Clocks	This block generates all clocks for the peripherals in the SDMA platform. The CCM also manages the ARM926 platform low power modes (WAIT, STOP), disabling peripheral clocks appropriately for power conservation, and provides alternate clock sources for the ARM926 and SDMA platforms.	—
CSPI(3)	Configurable Serial Peripheral Interface	SDMA, ARM	Connectivity Peripherals	This module is a serial interface equipped with data FIFOs, each master/slave-configurable SPI module is capable of interfacing to both serial port interface master and slave devices. The CSPI ready (SPI_RDY) and slave select (SS) control signals enable fast data communication with fewer software interrupts.	—

Table 1-2. Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Domain[1]	Sub-System	Brief Description	Chapter
EMI	External Memory Interface	SDMA	External Memory Interface	The EMI module provides access to external memory for the ARM and other masters. It is composed of main sub-modules: • M3IF provides arbitration between multiple masters requesting access to the external memory. • The SDRAM CTRL interfaces to DDR2 and SDR interfaces. • NANDFC provide an interface to NAND Flash memories. • The WEIM interfaces to NOR flash and PSRAM.	—
EPIT(2)	Enhanced Periodic Interrupt Timer	ARM	Timer Peripherals	Each EPIT is a 32-bit “set and forget” timer that starts counting after the EPIT is enabled by software. It is capable of providing precise interrupts at regular intervals with minimal processor intervention. It has a 12-bit prescaler to adjust the input clock frequency to the required time setting for the interrupts, and the counter value can be programmed on the fly.	—
ESAI	Enhanced Serial Audio Interface	SDMA	Connectivity Peripherals	The Enhanced Serial Audio Interface (ESAI) provides a full-duplex serial port for serial communication with a variety of serial devices, including industry-standard codecs, SPDIF transceivers, and other DSPs. The ESAI consists of independent transmitter and receiver sections, each section with its own clock generator.	—
FEC	Ethernet	SDMA	Connectivity Peripherals	The Ethernet Media Access Controller (MAC) is designed to support both 10- and 100- Mbps Ethernet/IEEE Std. 802.3™ networks. An external transceiver interface and transceiver function are required to complete the interface to the media.	—
GPIO(3)	General Purpose I/O Modules	ARM	Pins	Used for general purpose input/output to external ICs. Each GPIO module supports 32 bits of I/O.	—
GPT(4)	General Purpose Timers	ARM	Timer Peripherals	Each GPT is a 32-bit "free-running" or "set and forget" mode timer with programmable prescaler and compare and capture register. A timer counter value can be captured using an external event and can be configured to trigger a capture event on either the leading or trailing edges of an input pulse. When the timer is configured to operate in "set and forget" mode, it is capable of providing precise interrupts at regular intervals with minimal processor intervention. The counter has output compare logic to provide the status and interrupt at comparison. This timer can be configured to run either on an external clock or on an internal clock.	—

Table 1-2. Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Domain[1]	Sub-System	Brief Description	Chapter
I^2C(3)	I^2C Module	ARM	ARM926 Platform Peripherals	inter-IC Communication (I^2C) is an industry-standard, bi-directional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. I^2C is suitable for applications requiring occasional communications over a short distance between many devices. The interface operates up to 100 kbps with maximum bus loading and timing. The I^2C system is a true multiple-master bus, including arbitration and collision detection that prevents data corruption if multiple devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer.	—
IIM	IC Identification Module	ARM	Security Modules	The IIM provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Among the uses for the fuses are unique chip identifiers, mask revision numbers, cryptographic keys, and various control signals requiring a fixed value.	—
IOMUX	I/O Multiplexers	ARM	Pins	Each I/O multiplexer provides a flexible, scalable multiplexing solution with the following features: • Up to eight output sources multiplexed per pin • Up to four destinations for each input pin • Unselected input paths are held at constant level for reduced power consumption	—
KPP	KeyPad Port	ARM	Connectivity Peripherals	Can be used for either keypad matrix scanning or general purpose I/O.	—
OSC24M	OSC24 MHz Reference Oscillator	Analog	Clock	The OSC24M oscillator provides a stable frequency reference for the PLLs. This oscillator is designed to work in conjunction with an external 24 MHz crystal.	—
OSC32K	OSC32 kHz Reference Oscillator	Analog	Clock	The OSC32K oscillator provides a stable frequency reference for the RTC. This oscillator is designed to work in conjunction with an external 32 KHz crystal.	—
1-WIRE	1-Wire Interface	ARM	ARM926 Platform Peripherals	1-WIRE provides the communication line to a 1 Kbit Add-Only Memory (DS2502). The interface sends or receives one bit at a time. The required protocol for accessing the 1-Wire device, especially the DS2502, is defined by Dallas Semiconductor Corporation.™ The DS2502 holds battery characteristics information.	—
PWM(4)	Pulse Width Modulator	ARM	ARM926 Platform Peripherals	The pulse-width modulator (PWM) has a 16-bit counter and is optimized to generate sound from stored sample audio images. The PWM can also generate tones. It uses 16-bit resolution and a 4 × 16 data FIFO to generate sound.	—
RTC	Real Time Clock	ARM	Clocks	Provides the ARM926 platform with a clock function.	—

Table 1-2. Digital and Analog Modules (continued)

Block Mnemonic	Block Name	Domain[1]	Sub-System	Brief Description	Chapter
SIM(2)	Subscriber Identification Module	ARM	Security	The SIM is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards.	—
SDMA	Smart DMA Engine	SDMA	System Controls	The SDMA provides DMA capabilities inside the processor. It is a shared module that implements 32 DMA channels and has interface to connect to the ARM926 platform, EMI interface and the shared peripherals.	—
SJC	Secure JTAG Controller	ARM	Pins	The SJC provides debug and test control with maximum security.	—
SPBA	Shared Peripheral Bus Arbiter	SDMA	System Controls	The SPBA controls access to the shared peripherals. It supports shared peripheral ownership and access rights to an owned peripheral.	—
SSI(2)	Synchronous Serial Interface	SDMA, ARM	Connectivity peripherals	The SSI is a full-duplex serial port that allows the processor connected to it to communicate with a variety of serial protocols, including the Freescale Semiconductor SPI standard and the inter-IC sound bus standard (I^2S). The SSIs interface to the AUDMUX for flexible audio routing.	—
UART(5)	Universal Asynchronous Receiver/ Transmitters	SDMA, ARM	Connectivity Peripherals	Each UART provides serial communication capability with external devices through an RS-232 cable using the standard RS-232 non-return-to-zero (NRZ) encoding format. Each module transmits and receives characters containing either 7 or 8 bits (program selectable). Each UART can also provide low-speed IrDA compatibility through the use of external circuitry that converts infrared signals to electrical signals (for reception) or transforms electrical signals to signals that drive an infrared LED (for transmission).	—
USBOTG USBHOST	High-Speed USB On-The-Go	ARM	Connectivity Peripherals	The USB module provides high performance USB On-The-Go (OTG) and Host functionality (up to 480 Mbps), compatible with the USB 2.0 specification, the OTG supplement and the ULPI 1.0 Low Pin Count specification. The module has DMA capabilities handling data transfer between internal buffers and system memory. A OTG HS PHY and HOST FS PHY are also integrated.	—
WDOG	Watchdog Modules	ARM	Timer peripherals	Each module protects against system failures by providing a method of escaping from unexpected events or programming errors. Once activated, the timer must be serviced by software on a periodic basis. If servicing does not take place, the watchdog times out and then either asserts a system reset signal or an interrupt request signal, depending on the software configuration.	—

[1] ARM = ARM926 platform, SDMA = SDMA platform

Chapter 2
Memory Map

This chapter introduces the memory architecture of the i.MX25 device. The i.MX25 memory is hierarchical, with one level of cache between the microprocessors and the external memory.

The chapter is organized as follows:

- "Section 2.1, "System Memory Map," presents the system memory map.
- "Section 2.2, "SDMA Peripheral Memory Map," presents the SDMA master IP bus peripheral memory map.

NOTE

All referenced addresses are physical addresses.

2.1 System Memory Map

Table 2-1 shows the system memory map for the i.MX25 device.

Table 2-1. System Memory Map

Address Range		Size	Region
Start	End		
0x0000_0000	0x0000_3FFF	16 Kbytes	ROM (36 Kbytes)
0x0000_4000	0x0040_3FFF	4 Mbytes	Reserved
0x0040_4000	0x0040_8FFF	20 Kbytes	ROM (36 Kbytes)
0x0040_9000	0x0FFF_FFFF	252 Mbytes (minus 36 Kbytes)	Reserved
0x1000_0000	0x1FFF_FFFF	256 Mbytes	Reserved
0x2000_0000	0x2FFF_FFFF	256 Mbytes	Reserved
0x3000_0000	0x3FFF_FFFF	256 Mbytes	Reserved
0x4000_0000	0x43EF_FFFF	63 Mbytes	Reserved
0x43F0_0000	0x43F0_3FFF	16 Kbytes	AIPS A control registers
0x43F0_4000	0x43F0_7FFF	16 Kbytes	ARM926 platform MAX
0x43F0_8000	0x43F0_BFFF	16 Kbytes	ARM926 platform CLKCTL
0x43F0_C000	0x43F0_FFFF	16 Kbytes	ARM926 platform ETB registers
0x43F1_0000	0x43F1_3FFF	16 Kbytes	ARM926 platform ETB memory
0x43F1_4000	0x43F1_7FFF	16 Kbytes	ARM926 platform AAPE registers
0x43F1_8000	0x43F7_FFFF	416 Kbytes	Reserved

Table 2-1. System Memory Map (continued)

Address Range		Size	Region
Start	End		
0x43F8_0000	0x43F8_3FFF	16 Kbytes	I^2C-1
0x43F8_4000	0x43F8_7FFF	16 Kbytes	I^2C-3
0x43F8_8000	0x43F8_BFFF	16 Kbytes	CAN-1
0x43F8_C000	0x43F8_FFFF	16 Kbytes	CAN-2
0x43F9_0000	0x43F9_3FFF	16 Kbytes	UART-1
0x43F9_4000	0x43F9_7FFF	16 Kbytes	UART-2
0x43F9_8000	0x43F9_BFFF	16 Kbytes	I^2C-2
0x43F9_C000	0x43F9_FFFF	16 Kbytes	1-Wire
0x43FA_0000	0x43FA_3FFF	16 Kbytes	ATA (CPU side)
0x43FA_4000	0x43FA_7FFF	16 Kbytes	CSPI-1
0x43FA_8000	0x43FA_BFFF	16 Kbytes	KPP
0x43FA_C000	0x43FA_FFFF	16 Kbytes	IOMUXC
0x43FB_0000	0x43FB_3FFF	16 Kbytes	AUDMUX
0x43FB_4000	0x43FB_7FFF	16 Kbytes	Reserved
0x43FB_8000	0x43FB_BFFF	16 Kbytes	ECT (IP BUS A)
0x43FB_C000	0x43FB_FFFF	16 Kbytes	ECT (IP BUS B)
0x43FC_0000	0x43FF_FFFF	256 Kbytes	Reserved AIPS A off-platform slots
0x4400_0000	0x4FFF_FFFF	192 Mbytes	Reserved
0x5000_0000	0x5000_3FFF	16 Kbytes	SPBA base address
0x5000_4000	0x5000_7FFF	16 Kbytes	CSPI-3
0x5000_8000	0x5000_BFFF	16 Kbytes	UART-4
0x5000_C000	0x5000_FFFF	16 Kbytes	UART-3
0x5001_0000	0x5001_3FFF	16 Kbytes	CSPI-2
0x5001_4000	0x5001_7FFF	16 Kbytes	SSI-2
0x5001_8000	0x5001_BFFF	16 Kbytes	ESAI
0x5001_C000	0x5001_FFFF	16 Kbytes	Reserved
0x5002_0000	0x5002_3FFF	16 Kbytes	ATA
0x5002_4000	0x5002_7FFF	16 Kbytes	SIM-1
0x5002_8000	0x5002_BFFF	16 Kbytes	SIM-2
0x5002_C000	0x5002_FFFF	16 Kbytes	UART-5
0x5003_0000	0x5003_3FFF	16 Kbytes	TSC

Table 2-1. System Memory Map (continued)

Address Range		Size	Region
Start	End		
0x5003_4000	0x5003_7FFF	16 Kbytes	SSI-1
0x5003_8000	0x5003_BFFF	16 Kbytes	FEC
0x5003_C000	0x5003_FFFF	16 Kbytes	SPBA registers
0x5004_0000	0x51FF_FFFF	32 Mbytes (minus 256 Kbytes)	Reserved AIPS B
0x5200_0000	0x53EF_FFFF	31 Mbytes	Reserved
0x53F0_0000	0x53F0_3FFF	16 Kbytes	AIPS B control registers
0x53F0_4000	0x53F7_FFFF	496 Kbytes	Reserved
0x53F8_0000	0x53F8_3FFF	16 Kbytes	CCM
0x53F8_4000	0x53F8_7FFF	16 Kbytes	GPT-4
0x53F8_8000	0x53F8_BFFF	16 Kbytes	GPT-3
0x53F8_C000	0x53F8_FFFF	16 Kbytes	GPT-2
0x53F9_0000	0x53F9_3FFF	16 Kbytes	GPT-1
0x53F9_4000	0x53F9_7FFF	16 Kbytes	EPIT-1
0x53F9_8000	0x53F9_BFFF	16 Kbytes	EPIT-2
0x53F9_C000	0x53F9_FFFF	16 Kbytes	GPIO-4
0x53FA_0000	0x53FA_3FFF	16 Kbytes	PWM-2
0x53FA_4000	0x53FA_7FFF	16 Kbytes	GPIO-3
0x53FA_8000	0x53FA_BFFF	16 Kbytes	PWM-3
0x53FA_C000	0x53FA_FFFF	16 Kbytes	SCC
0x53FB_0000	0x53FB_3FFF	16 Kbytes	RNGB
0x53FB_4000	0x53FB_7FFF	16 Kbytes	eSDHC-1
0x53FB_8000	0x53FB_BFFF	16 Kbytes	eSDHC-2
0x53FB_C000	0x53FB_FFFF	16 Kbytes	LCDC
0x53FC_0000	0x53FC_3FFF	16 Kbytes	SLCDC
0x53FC_4000	0x53FC_7FFF	16 Kbytes	Reserved
0x53FC_8000	0x53FC_BFFF	16 Kbytes	PWM-4
0x53FC_C000	0x53FC_FFFF	16 Kbytes	GPIO-1
0x53FD_0000	0x53FD_3FFF	16 Kbytes	GPIO-2
0x53FD_4000	0x53FD_7FFF	16 Kbytes	SDMA
0x53FD_8000	0x53FD_BFFF	16 Kbytes	Reserved
0x53FD_C000	0x53FD_FFFF	16 Kbytes	WDOG

Table 2-1. System Memory Map (continued)

Address Range		Size	Region
Start	End		
0x53FE_0000	0x53FE_3FFF	16 Kbytes	PWM-1
0x53FE_4000	0x53FE_7FFF	16 Kbytes	Reserved
0x53FE_8000	0x53FE_BFFF	16 Kbytes	Reserved
0x53FE_C000	0x53FE_FFFF	16 Kbytes	RTICv3
0x53FF_0000	0x53FF_3FFF	16 Kbytes	IIM
0x53FF_4000	0x53FF_7FFF	16 Kbytes	USB
0x53FF_8000	0x53FF_BFFF	16 Kbytes	CSI
0x53FF_C000	0x53FF_FFFF	16 Kbytes	DryIce
0x5400_0000	0x5FFF_FFFF	192 Mbytes	Reserved (aliased AIPS B slots)
0x6000_0000	0x67FF_FFFF	128 Mbytes	ARM926 platform ROMPATCH
0x6800_0000	0x6FFF_FFFF	128 Mbytes	ARM926 platform ASIC
0x7000_0000	0x77FF_FFFF	128 Mbytes	Reserved
0x7800_0000	0x7801_FFFF	128 Kbytes	RAM
0x7802_0000	0x7FFF_FFFF	128 Mbytes (minus 128 Kbytes)	RAM alias
0x8000_0000	0x8FFF_FFFF	256 Mbytes	SDRAM bank 0
0x9000_0000	0x9FFF_FFFF	256 Mbytes	SDRAM bank 1
0xA000_0000	0xA7FF_FFFF	128 Mbytes	WEIM CS0 (flash 128)[1]
0xA800_0000	0xAFFF_FFFF	128 Mbytes	WEIM CS1 (flash 64)[1]
0xB000_0000	0xB1FF_FFFF	32 Mbytes	WEIM CS2 (SRAM)
0xB200_0000	0xB3FF_FFFF	32 Mbytes	WEIM CS3 (SRAM)
0xB400_0000	0xB5FF_FFFF	32 Mbytes	WEIM CS4
0xB600_0000	0xB7FF_FFFF	32 Mbytes	Reserved
0xB800_0000	0xB800_0FFF	4 Kbytes	Reserved
0xB800_1000	0xB800_1FFF	4 Kbytes	SDRAM control registers
0xB800_2000	0xB800_2FFF	4 Kbytes	WEIM control registers
0xB800_3000	0xB800_3FFF	4 Kbytes	M3IF control registers
0xB800_4000	0xB800_4FFF	4 Kbytes	EMI control registers
0xB800_5000	0xBAFF_FFFF	32 Mbytes (minus 20 Kbytes)	Reserved
0xBB00_0000	0xBB00_0FFF	4 Kbytes	NAND flash main area buffer
0xBB00_1000	0xBB00_11FF	512 B	NAND flash spare area buffer
0xBB00_1200	0xBB00_1DFF	3 Kbytes	Reserved

Table 2-1. System Memory Map (continued)

Address Range		Size	Region
Start	End		
0xBB00_1E00	0xBB00_1FFF	512 B	NAND flash control registers
0xBB01_2000	0xBFFF_FFFF	96 Mbytes (minus 8 Kbytes)	Reserved
0xC000_0000	0xFFFF_FFFF	1024 Mbytes	Reserved

[1] WEIM CS0 and CS1 can be collapsed to form a single 256-Mbyte region from 0xA000_0000 to 0xAFFF_FFFF

2.2 SDMA Peripheral Memory Map

SDMA scripts can read and write to the context RAM, data RAM, shared-peripheral registers, and internal registers. All of the data accessible to SDMA scripts make up the data memory space of the SDMA. The address range is 16 bits and the data width is 32 bits. Each address corresponds to a 32-bit data word.

The SDMA can perform only 32-bit access to shared-peripheral registers. The i.MX25, shared peripheral registers are treated as if byte addressed. This is a consequence of connections through the SPBA shared peripheral bus outside of the SDMA. The result is that although address space 4 Kwords (16 Kbytes) is allocated for each peripheral, only the first 4 Kbytes of the peripherals register space can be accessed. For example, the shared peripheral register at address 0x3000 is mapped also to addresses 0x3001, 0x3002, 0x3003. A read or write access to any of these 4 addresses will respond as if the access was to address 0x3000.

Table 2-2 shows the memory map of the shared peripherals to SDMA-accessible memory space.

Table 2-2. SDMA Peripheral Memory Map

Peripheral	Base Address	Size	Comments
SDMA internal memory	0x0000	4 Kwords	map to SDMA Internal ROM/RAM
CSPI3	0x1000	4 Kwords	map to 4 Kbytes of CSPI3 address space
UART4	0x2000	4 Kwords	map to 4 Kbytes of UART4 address space
UART3	0x3000	4 Kwords	map to 4 Kbytes of UART3 address space
CSPI2	0x4000	4 Kwords	map to 4 Kbytes of CSPI2 address space
SSI2	0x5000	4 Kwords	map to 4 Kbytes of SSI2 address space
ESAI	0x6000	4 Kwords	map to 4 Kbytes of ESAI address space
SDMA internal registers	0x7000	4 Kwords	map to SDMA registers
ATA	0x8000	4 Kwords	map to 4 Kbytes of ATA address space
SIM1	0x9000	4 Kwords	map to 4 Kbytes of SIM1 address space
SIM2	0xA000	4 Kwords	map to 4 Kbytes of SIM2 address space
UART5	0xB000	4 Kwords	map to 4 Kbytes of UART5 address space
TCH/SC/ADCTL	0xC000	4 Kwords	map to 4 Kbytes of TCH/SC/ADCTL address space

Table 2-2. SDMA Peripheral Memory Map (continued)

Peripheral	Base Address	Size	Comments
SSI1	0xD000	4 Kwords	map to 4 Kbytes of SSI1 address space
FEC	0xE000	4 Kwords	map to 4 Kbytes of FEC address space
SPBA Registers	0xF000	4 Kwords	map to 4 Kbytes of SPBA address space

Chapter 3
Interrupts and DMA Events

This chapter provides the assignments for the interrupt requests of the AP domains:

- Section 3.1, “ARM926 Platform Interrupts”

This chapter also defines the DMA events in the following sections:

- Section 3.2, “SDMA Event Mapping”

3.1 ARM926 Platform Interrupts

The ARM926EJ-S Simple Interrupt Controller (ASIC) is designed to handle up to 64 interrupt requests. Table 3-1 lists the ARM926 platform interrupt sources.

Table 3-1. ARM926 Platform Interrupt Sources

IRQ	Interrupt Source	Interrupt Description
0	CSPI-3	—
1	GPT-4	—
2	1-Wire	—
3	I^2C-1	—
4	I^2C-2	—
5	UART-4	—
6	RTIC	—
7	ESAI	—
8	ESDHC-2	—
9	ESDHC-1	—
10	I^2C-3	—
11	SSI-2	—
12	SSI-1	—
13	CSPI-2	—
14	CSPI-1	—
15	ATA	—
16	GPIO-3	Combined Interrupts - 1 Bit Int Or Of 32
17	CSI	—
18	UART-3	—

Table 3-1. ARM926 Platform Interrupt Sources (continued)

IRQ	Interrupt Source	Interrupt Description
19	IIM	—
20	SIM-1	—
21	SIM-2	—
22	RNGB	—
23	GPIO-4	Combined Interrupts - 1 Bit Int Or Of 32
24	KPP	Keypad Interrupt
25	DryIce	Consolidated RTC Interrupt
26	PWM-1	—
27	EPIT-2	—
28	EPIT-1	—
29	GPT-3	—
30	POWER FAIL	power fail interrupt from external PAD
31	CCM	—
32	UART-2	—
33	NANDFC	Consolidated Nand Flash Controller Interrupt
34	SDMA	“AND” of all 32 interrupts from all the channels
35	USB-HTG	—
36	PWM-2	—
37	USB-OTG	—
38	SLCDC	—
39	LCDC	—
40	UART-5	—
41	PWM-3	—
42	PWM-4	—
43	CAN-1	—
44	CAN-2	—
45	UART-1	—
46	TSC	—
47	Reserved	—
48	ECT	—
49	SCC SCM	SCM interrupt
50	SCC SMN	SMN interrupt
51	GPIO-2	Combined Interrupts - 1 Bit Int Or Of 32

Table 3-1. ARM926 Platform Interrupt Sources (continued)

IRQ	Interrupt Source	Interrupt Description
52	GPIO-1	Combined Interrupts - 1 Bit Int Or Of 32
53	GPT-2	—
54	GPT-1	—
55	WDOG	—
56	DryIce	security violation interrupt
57	FEC	—
58	EXT_INT5	External interrupt for Power Management using GPIO-1[5]
59	EXT_INT4	External interrupt for Temp using GPIO-1[4]
60	EXT_INT3	External interrupt for Sensor using GPIO-1[3]
61	EXT_INT2	External interrupt for Sensor using GPIO-1[2]
62	EXT_INT1	External interrupt for Watch-dog using GPIO-1[1]
63	EXT_INT0	External interrupt for TV using GPIO-1[0]

3.2 SDMA Event Mapping

Table 3-2 shows the SDMA event mapping.

Table 3-2. SDMA Event Mapping

Event Number	DMA Source	Description
0	EXT DMA 0	External DMA request 0 from GPIO-4:GPIO[27]
1	CCM	DPTC, DVFS event
2	ATA TXFER end	—
3	ATA TX FIFO	—
4	ATA RX FIFO	—
5	Reserved	—
6	CSP-2 RX	CSPI2 receive
7	CSPI-2 TX	CSPI2 transmit
8	CSPI-1 RX	CSPI1 receive
9	CSPI-1 TX	CSPI1 transmit
10	UART-3 RX	UART3 receive
11	UART-3 TX	UART3 transmit
12	UART-4 RX	UART4 receive
13	UART-4 TX	UART4 transmit
14	EXT DMA 1	External DMA request 1 from GPIO-4:GPIO[28]

Table 3-2. SDMA Event Mapping (continued)

Event Number	DMA Source	Description
15	EXT DMA 2	External DMA request 2 from GPIO-4:GPIO[29]
16	UART-2 RX	UART2 receive
17	UART-2 TX	UART2 transmit
18	UART-1 RX	UART1 receive
19	UART-1 TX	UART1 transmit
20	Reserved	—
21	Reserved	—
22	SSI-2 RX1	SSI2 receive channel 1
23	SSI-2 TX1	SSI2 transmit channel 1
24	SSI-2 RX0	SSI2 receive channel 0
25	SSI-2 TX0	SSI2 transmit channel 0
26	SSI-1 RX1	SSI1 receive channel 1
27	SSI-1 TX1	SSI1 transmit channel 1
28	SSI-1 RX0	SSI1 receive channel 0
29	SSI-1 TX0	SSI1 transmit channel 0
30	NANDFC	—
31	ECT	CTI trigger out
32	ESAI RX	—
33	ESAI TX	—
34	CSPI-3 RX	—
35	CSPI-3 TX	—
36	SIM-2 RX	—
37	SIM-2 TX	—
38	SIM-1 RX	—
39	SIM-1 TX	—
40	Reserved	—
41	Reserved	—
42	Reserved	—
43	Reserved	—
44	TSC GCQ	—
45	TSC TCQ	—
46	UART-5 RX	—
47	UART-5 TX	—

Chapter 4
External Signals and Pin Multiplexing

The i.MX25 has a limited number of pins, and most pins are shared among multiple signals. This chapter describes the external I/O signals and pin multiplexing (muxing) in order to help users to understand pin assignment and the input-output multiplexer (IOMUX). It is also intended to enable software developers to do appropriate muxing programming.

4.1 IOMUX Overview

The muxing hardware in i.MX25 is composed of the following:

- IOMUX—Combinational logic that does the muxing.
- IOMUX_CTL—Muxing controller which controls signal muxing, interrupt observability, and pin settings.

The IOMUX module deals with signal multiplexing, and consists of combinatorial logic built with a basic IOMUX cell. Each pin, which is shared by multiple signals, has a related IOMUX cell to handle signal multiplexing.

The IOMUX_CTL module consists of registers and deals with interrupt and pin characteristic settings, such as I/O driver voltage, slew rate, drive strength, open drain and pull/keeper, and so on.

Figure 4-1 shows a simplified SoC block diagram of pin muxing with two typical cases:

- Case 1 is a regular muxing example: Module A, Module B, and Module GPIO share the same pin through the IOMUX, and muxing is controlled by the IOMUX_CTL.
- Case 2 is an example of no muxing: the pin is dedicated to Module C, signals are directly connected between Module C and the pin, and no IOMUX cell is involved.

Each IOMUX cell can support up to eight muxing modes (ALT0–ALT7), which means that each pin can ultimately be shared by eight signals.

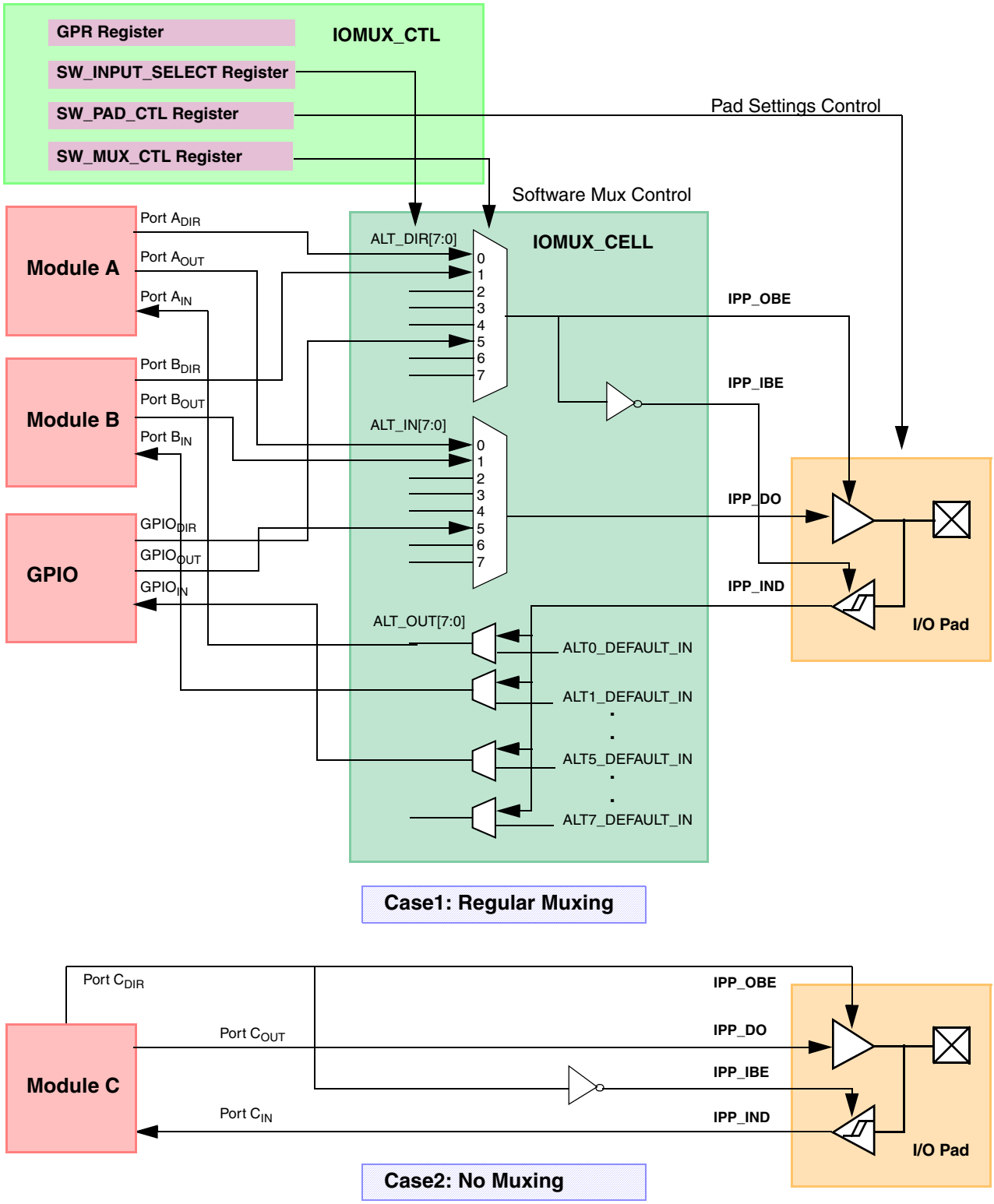


Figure 4-1. Pin Multiplexing Block Diagram

4.2 Pin-Muxing Control

This section describes how to select the muxing mode for each pin through software programming. After chip reset, each pin works in its primary mode, ALT0. Boot code or software programming are required to enable the alternate modes (ALT1–ALT7).

Each pin's signal muxing is controlled by a software mux control (SW_MUX_CTL) register. Each SW_MUX_CTL register handles pin muxing for only one pin. The SW_MUX_CTL register also has the ability to enable a loopback feature. See Section 4.2.1, "Software Mux Control Registers (SW_MUX_CTL)," for more information.

The software select input (SW_SELECT_INPUT) register is user-accessible during IOMUX programming which is required when there are multiple pins as muxing options. See Section 4.2.2, "SW_SELECT_INPUT Register Definition," for more information. Section 4.6, "Daisy Chain List," also provides information about SW_SELECT_INPUT settings.

4.2.1 Software Mux Control Registers (SW_MUX_CTL)

Each 32-bit SW_MUX_CTL register controls the signal multiplexing for one IOMUX cell. The SW_MUX_CTL register is partitioned into two fields: mux mode select field (MUX_MODE) and software input on field (SION). MUX_MODE controls the signal muxing on each pin, and SION controls loopback and GPIO capture feature, although not all individual pins have the SION bit option.

When the SION bit is cleared, the pin works in normal mode and the I/O direction is determined by the selected muxing mode. When the SION bit is set to 1, the pin operates in loopback mode. All modes' input path is forced on by software, and pin value feeds through into all modes' input. In loopback mode, using a GPIO module, users can capture the pin values.

Figure 4-2 shows the general format of an SW_MUX_CTL register. Table 4-1 shows the field descriptions.

SW_MUX_CTL registers follow the naming convention of

IOMUXC_SW_MUX_CTL_PAD_<*Pin_Name*>,

where <*Pin_Name*> is the pin name, which is different for each register.

IOMUXC_SW_MUX_CTL_PAD_<*Pin_Name*> Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	0	—	0		

Figure 4-2. SW_MUX_CTL Register Generic Format

Table 4-1. SW_MUX_CTL Field Descriptions

Field	Description
31–5	Reserved
4 SION	Software Input On Field. 0 Normal mode: IO is determined by the selected muxing mode. (default) 1 Loopback mode: Input feedthrough, loopback feature enabled.
3	Reserved
2–0 MUX_MODE	MUX Mode Select Field. Select one of the IOMUX modes to be used for the pin: <*Pin_Name*> 000 Select ALT0 mux mode (default) 001 Select ALT1 mux mode 010 Select ALT2 mux mode 011 Select ALT3 mux mode 100 Select ALT4 mux mode 101 Select ALT5 mux mode 110 Select ALT6 mux mode 111 Select ALT7 mux mode

4.2.1.1 Loopback Examples

The following examples illustrate how to use the loopback feature for bus-status detection and GPIO capture.

Example 1: Bus-Status Detection

Figure 4-3 shows the case where module A drives the pin while simultaneously receiving or detecting the pin value or bus status.

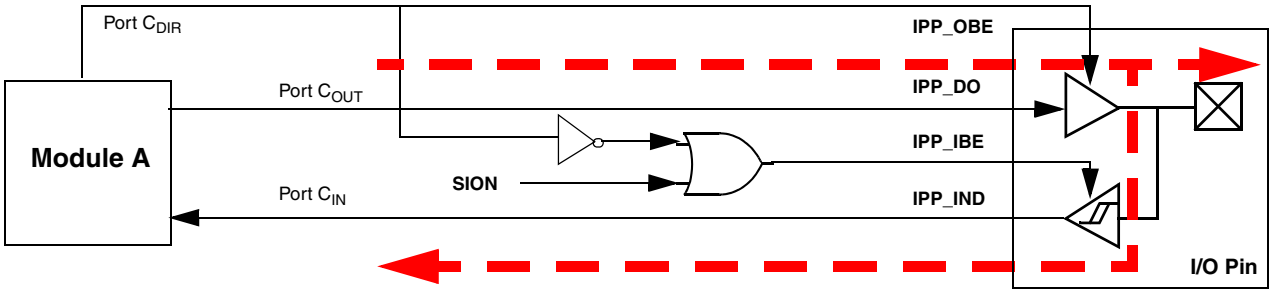


Figure 4-3. Loopback Example 1: Bus-Status Detection

Example 2: GPIO Capture

Figure 4-4 shows the case where module A drives the pin while the GPIO module captures the pin value or bus status.

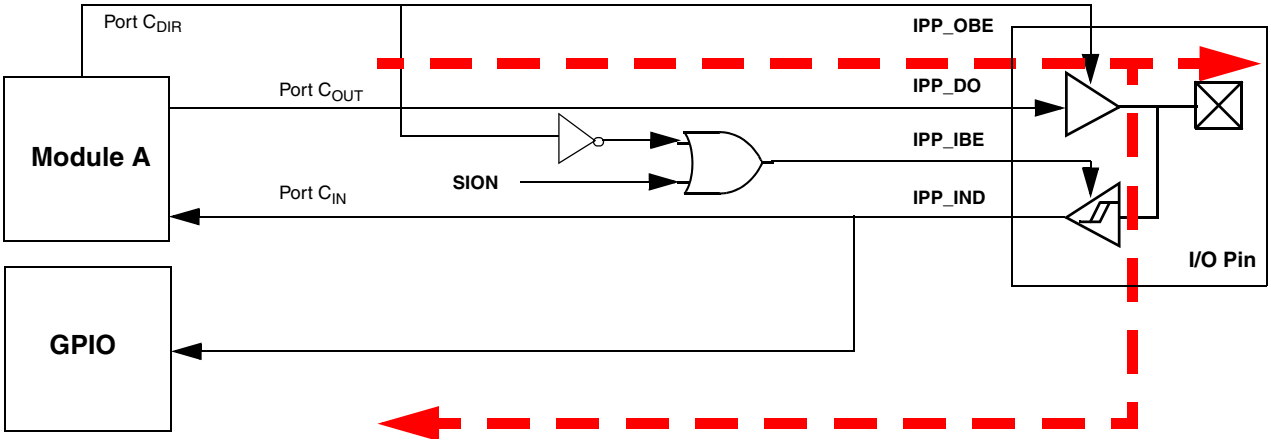


Figure 4-4. Loopback Example 2: GPIO Capture

4.2.2 SW_SELECT_INPUT Register Definition

Each 32-bit Software Select Input Control (SW_SELECT_INPUT) register controls the input path to avoid multi-driving problem. This register is only required when multiple pins drive the same internal port. This usually happens when one input port has multiple pin-muxing options on different pins.

For example, in Figure 4-5, both pin A and pin B could be used as the input pin of an on-chip IP Instance, and serve as Port C options. In such cases, a SW_SELECT_INPUT register is used to select the driving pin. Section 4.6, "Daisy Chain List," provides all instances and ports that are involved in the daisy chain, and also lists pins and their corresponding SW_SELECT_INPUT register values.

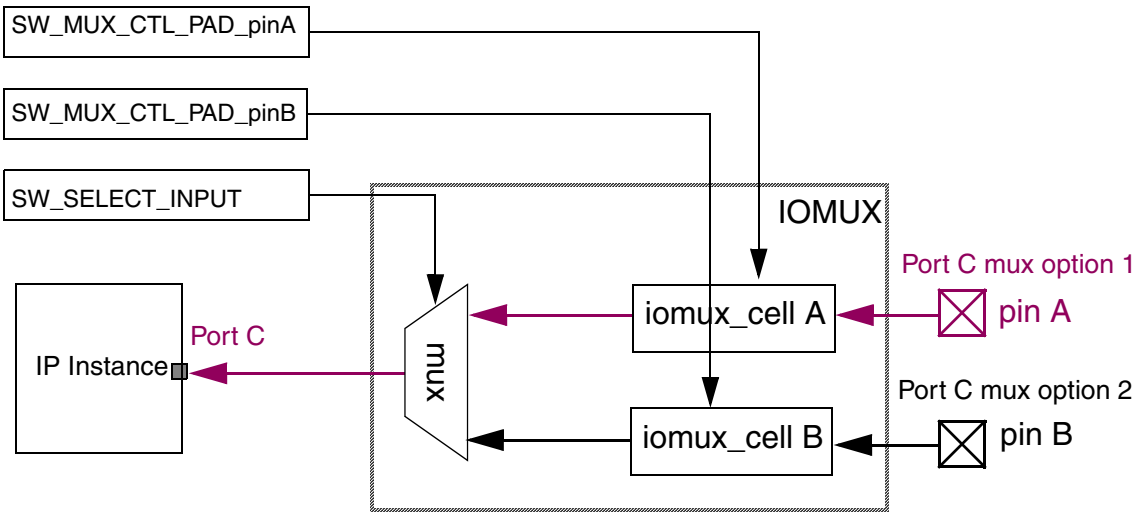


Figure 4-5. Select Input from Pins

Two steps are required during IOMUX software programming in such cases. For example, in Figure 4-5, if pin A is selected to work as the driving pin of Port C, then the following IOMUX programming steps are required:

1. Set IOMUXC_SW_MUX_CTL_PAD_<*pinA*> to select the correct muxing mode for Port C.
2. Set IOMUXC_<*Instance*>_<*PortC*>_SELECT_INPUT to select the input driving from pin A.

SW_SELECT_INPUT registers follow the naming convention:

IOMUXC_<*Instance*>_<*Port*>_SELECT_INPUT.

Figure 4-6 shows the general format of an SW_SELECT_INPUT register. Table 4-2 shows the field descriptions.

IOMUXC_<*Instance*>_<*Port*>_SELECT_INPUT — Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	Daisy		
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	0	0	0

Figure 4-6. SW_SELECT_INPUT Register Generic Format

Table 4-2. SW_SELECT_INPUT Field Descriptions

Field	Description
31–3	Reserved
2–0 Daisy	The actual width of the Daisy field is equal to the number of driving pins. For example, if there are two driving pins then bits [2:1] are reserved, and bit 0 is the actual Daisy field. Users can select a specific drive pin for an input port of a module to solve the multi-driving problem (this is referred to as “daisy chaining”).

All the instances and ports that are involved in Daisy Chain are listed in Section 4.6, “Daisy Chain List,” users can find the pins and correct values to program SW_SELECT_INPUT registers in Table 4-16.

4.3 Pin-Setting Control

This section describes how to set characteristic settings of each pin through software programming. The software pad control registers (SW_PAD_CTL) control I/O driver voltage, slew rate, drive strength, open drain, pull/keeper, DDR type, and so on. See Section 4.3.1, “Software Pad Control Registers (SW_PAD_CTL),” for a full description.

Settings for some pins are controlled in groups instead of pin-by-pin. This is implemented by the software pad group control registers (SW_PAD_CTL_GRP). See Section 4.3.2, "Software Pad Group Control Registers (SW_PAD_CTL_GRP)," for more information.

4.3.1 Software Pad Control Registers (SW_PAD_CTL)

Each 32-bit SW_PAD_CTL register controls the software-configurable characteristics of an I/O pin. The configurable settings include Slew Rate, Drive Strength, Open Drain, Pull/Keeper, Hysteresis, and so on.

DDR pins have different characteristics from GPIO pins. The generic format of the SW_PAD_CTL register for GPIO pins is shown in Figure 4-7 and the format for DDR pins is shown in Figure 4-8.

Not all SW_PAD_CTL registers implement every bit defined in the generic definition. Some characteristics of specific pins may have constant settings that are not software configurable. Only those characteristics with CFG ( ) in the detailed pin muxing Table 4-18 are software-configurable.

Characteristics for some pins are configured in groups, not pin-by-pin. This is defined in Section 4.3.2, "Software Pad Group Control Registers (SW_PAD_CTL_GRP)." If a characteristic is configurable as a group, then the corresponding bit in the SW_PAD_CTL register is not used.

SW_PAD_CTL registers follow the naming convention: IOMUXC_SW_PAD_CTL_PAD_<*Pin_Name*>.

Figure 4-7 and Table 4-3 show the register definition for GPIO pins. Reset values are different for different registers, depending on the registers' functionality.

IOMUXC_SW_PAD_CTL_PAD_<*Pin_Name*> Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset[1]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	HYS	PKE	PUE	PUS		ODE	DSE		SRE
W																
Reset[1]	—	—	*n*	—	—	—	—	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

Figure 4-7. SW_PAD_CTL Register Generic Format (GPIO Pins)

[1] Reset values differ for different registers.

Table 4-3. SW_PAD_CTL Field Descriptions (GPIO Pins)

Field	Description
31–14	Reserved
13 DVS	Driver Voltage Select Bit. Select the correct driver for different I/O supplies[1]. 0 3.3 V I/O Driver 1 1.8 V I/O Driver

Table 4-3. SW_PAD_CTL Field Descriptions (GPIO Pins) (continued)

Field	Description
12–9	Reserved
8 HYS	Hysteresis enable bit. Select Schmitt trigger or CMOS input: 0 Hysteresis Disabled, CMOS input. 1 Hysteresis Enabled, Schmitt trigger input.
7 PKE	Pull/Keeper enable bit. when PKE = 0, PUE and PUS have no functionality. 0 Pull/Keeper Disabled 1 Pull/Keeper Enabled
6 PUE	Pull or keeper select bit. when PUE = 0, PUS has no functionality. 0 Keeper Enabled, Pull Disabled 1 Keeper Disabled, Pull Enabled
5–4 PUS	Pull up or down strength select bits. 00 100 KΩ Pull-down 01 47 KΩ Pull-up 10 100 KΩ Pull-up 11 22 KΩ Pull-up
3 ODE	Open drain enable bit. This bit selects open drain or CMOS output: 0 Output is CMOS 1 Output is Open Drain
2–1 DSE	Drive Strength Control Bits. These bits select standard, high or max pin drive strength[2]. 00 Nominal or standard drive strength 01 High drive strength 10 Max drive strength 11 Max drive strength
0 SRE	Slew Rate Control Bit. This bit selects between FAST/SLOW slew rate output. 0 Slow slew rate 1 Fast slew rate, used for high frequency designs

[1] The DVS bit is used to select the different output drivers, 1.8 V (±10%) or 3.3 V (±10%). Each GPIO pin has two internal output-buffer units designed for 1.8 V and 3.3 V respectively. For the 2.5 V supply case, the 1.8 V driver is applicable. Normally, the DVS bit should be set to match the I/O supply.

[2] I/O Drive strength depends on the application conditions. For GPIO pins, at a temperature of 25 °C and in the typical case, the following drive strength can be referenced: Standard (2 mA), High (4 mA) and Max (8 mA) in SLOW mode at 3.3 V supply with the equivalent impedances of 100 Ω, 50 Ω, and 25 Ω respectively. And, Standard (4 mA), High (6 mA) and Max (8 mA) in FAST mode at 3.3 V supply with the equivalent impedances of 50 Ω, 33 Ω, and 25 Ω respectively.

DDR pins have different characteristics than GPIO pins. Figure 4-8 and Table 4-4 shows the register definitions related to DDR pins. Reset values are different for different registers, depending on the registers' functionality.

IOMUXC_SW_PAD_CTL_PAD_<*Pin_Name*> Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset[1]	—	—	—	—	—	—	—	—	*n*	—	—	—	—	—	—	—

Figure 4-8. SW_PAD_CTL Register Generic Format (DDR Pins)

1 Reset values differ for different registers.

Table 4-4. SW_PAD_CTL Field Descriptions (DDR Pins)

Field	Description
31–8	Reserved
7 PKE	Keeper Enable Bit. This bit enables the internal keeper capability of the DDR pin. The DDR pin has no Pull capability. 0 Internal Keeper Disabled 1 Internal Keeper Enabled
6–0	Reserved

Note: For DDR pins, there are three working modes with different drive strength. And in this device, the DDR mode and drive strength of all DDR pins are controlled as a group, not on a pin-by-pin basis. For more information, see Section 4.3.2, "Software Pad Group Control Registers (SW_PAD_CTL_GRP)".

4.3.2 Software Pad Group Control Registers (SW_PAD_CTL_GRP)

The 32-bit software pad group control registers (SW_PAD_CTL_GRP) configure specific characteristics for a pin group, where a pin group is defined as a set of pins sharing a common I/O pin settings. These registers thus differ from the SW_PAD_CTL registers, which only configure characteristics for a single pin.

In the i.MX25 device there are three characteristics that are controlled in groups: DDR Type, Drive Voltage and Drive Strength.

4.3.2.1 DDR Type Group Control Register (IOMUXC_SW_PAD_CTL_GRP_DDRTYPE)

The IOMUXC_SW_PAD_CTL_GRP_DDRTYPE register controls the DDR type setting of all DDR pins. Register settings only apply to DDR pins.

The register's format is shown in Figure 4-9, and the register fields are described in Table 4-5. Users can select the DDR type by programming the DDR_TYPE field in this register. Different DDR types correspond to different drive strengths and impedance equivalence, as follows:

- The default setting for DDR_TYPE is 0b00, which is the setting that is suitable for most systems using mobile DDR (mDDR) or DDR2 that do not require termination. mDDR and DDR2 types are designed for 1.8 V (± 5%) applications. For DDRTYPE=00 at 25^{o} C in the typical case, the following drive strengths can be referenced: Standard (3.6 mA), High (7.2 mA), Max (10.8 mA) with equivalent impedances of 90 Ω, 45 Ω, and 20 Ω respectively.
  - For mDDR and DDR2 configurations with 1 load (single chip), DDRTYPE=00 and Standard (3.6mA) drive strength is the typical setting and normally does not require termination
  - For mDDR and DDR2 configurations with 2 loads (2 parallel chips), DDRTYPE=00 and High (7.2mA) drive strength is the typical setting and normally does not require termination
- The DDR_TYPE setting of 0b01 corresponds to SDRAM. SDRAM type is designed for 3.3 V (± 5%) applications, with drive strengths Standard (4 mA), High (8 mA), Max (12 mA) and equivalent impedances of 90 Ω, 45 Ω, and 30 Ω respectively.
- The DDR_TYPE setting of 0b10 corresponds to maximum drive strength (13.4 mA) with equivalent impedance 20 Ω. Under this setting, the drive strength is not configurable, and the DSE bits (drive strength group control register) have no functionality.

IOMUXC_SW_PAD_CTL_GRP_DDRTYPE — Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	DDR_TYPE		0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-9. SW_PAD_CTL_GRP_DDRTYPE Register Format

Table 4-5. SW_PAD_CTL_GRP_DDRTYPE Field Descriptions

Field	Description
31–13	Reserved
12–11 DDR_TYPE	DDR_TYPE Field Select the working mode of all DDR pins. (Only valid for DDR pins.) 00 1.8 V mDDR and DDR2 type 01 3.3 V SDRAM type 10 Maximum drive 11 Reserved
10–0	Reserved

4.3.2.2 Drive Voltage Select Group Control Registers (SW_PAD_CTL_GRP_DVS)

The general format for Driver Voltage Select (DVS) group control registers is shown in Figure 4-10 and Table 4-6. This register allows users to select the different drivers for 1.8 V or 3.3 V. Table 4-7 shows the list of i.MX25 registers for the DVS group. DVS control is only valid for GPIO pins.

IOMUXC_SW_PAD_CTL_GRP_DVS_<GROUP_NAME> Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-10. SW_PAD_CTL_GRP_DVS Generic Format

Table 4-6. SW_PAD_CTL_GRP_DVS Field Descriptions

Field	Description
31–14	Reserved
13 DVS	Driver Voltage Select Bit. Selects the correct output driver for different I/O supplies. (Only valid for GPIO pins) Each GPIO pin has two internal output-buffer units designed for 1.8 V (±10%) and 3.3 V (±10%) respectively. For the 2.5 V supply case, the 1.8 V driver is applicable. Normally, the DVS bit should be set to match with the I/O supply. 0 3.3 V I/O Driver 1 1.8 V I/O Driver
12–0	Reserved

Table 4-7. SW_PAD_CTL_GRP_DVS Register List

Register Name	Comment
IOMUXC_SW_PAD_CTL_GRP_DVS_CSI	CSI power bank

Table 4-7. SW_PAD_CTL_GRP_DVS Register List (continued)

IOMUXC_SW_PAD_CTL_GRP_DVS_CCM	CCM power bank
IOMUXC_SW_PAD_CTL_GRP_DVS_JTAG	JTAG power bank
IOMUXC_SW_PAD_CTL_GRP_DVS_LCD	LCD power bank
IOMUXC_SW_PAD_CTL_GRP_DVS_MISC	MISC power bank
IOMUXC_SW_PAD_CTL_GRP_DVS_NFC	NFC power bank
IOMUXC_SW_PAD_CTL_GRP_DVS_SDHC1	SDHC1 power bank

4.3.2.3 Drive Strength Group Control Registers (SW_PAD_CTL_GRP_DSE)

The general format for drive strength group control registers is shown in Figure 4-11 and Table 4-8. The DSE control field is valid for both DDR and GPIO pins. Table 4-9 shows the list of i.MX25 registers for the drive strength group.

IOMUXC_SW_PAD_CTL_GRP_DSE_*<GROUP_NAME>* — Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-11. SW_PAD_CTL_GRP_DSE Generic Format

Table 4-8. SW_PAD_CTL_GRP_DSE Field Descriptions

Field	Description
31–3	Reserved
2–1 DSE	Drive Strength Control Bits. These bits select Standard, High or Max drive strength. (Valid for both DDR and GPIO pins.) 00 Nominal or standard drive strength 01 High drive strength 10 Max drive strength 11 Max drive strength
0	Reserved

Table 4-9. SW_PAD_CTL_GRP_DSE Register List

Register Name	Comment
IOMUXC_SW_PAD_CTL_GRP_DSE_CSI	Drive Strength Control of CSI group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_CSPI1	Drive Strength Control of CSPI1 group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_DDR	Drive Strength Control of DDR group pins

Table 4-9. SW_PAD_CTL_GRP_DSE Register List

IOMUXC_SW_PAD_CTL_GRP_DSE_FEC	Drive Strength Control of FEC group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_KPP	Drive Strength Control of KPP group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_LCD	Drive Strength Control of LCD group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_NFC	Drive Strength Control of NFC group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_UART	Drive Strength Control of UART group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_SDHC1	Drive Strength Control of SDHC1 group pins
IOMUXC_SW_PAD_CTL_GRP_DSE_WEIM	Drive Strength Control of WEIM group pins

4.4 Special Functionality

This section includes detailed descriptions of the following:

- General purpose register
- Interrupt observe control register
- Loopback and GPIO capture
- Boot-related pins
- Special pins

4.4.1 General Purpose Register (IOMUXC_GPR1)

The 32-bit general purpose register (IOMUXC_GPR1), shown in Figure 4-12, configures two EMI chip select options.

IOMUXC_GPR1 — Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SDCTL_CSD1_SEL_B	SDCTL_CSD0_SEL_B
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-12. IOMUXC_GPR1 Register

Table 4-10. IOMUXC_GPR1 Field Descriptions

Field	Description
31–2	Reserved

Table 4-10. IOMUXC_GPR1 Field Descriptions (continued)

Field	Description
1 SDCTL_CSD1_SEL_B	EMI Chip Select Control Bit for SDCTL over WEIM. Select multiplexed chip select (CS3) between SDRAM/DDR and WEIM in EMI. 0 SDCTL chip select 1 WEIM chip select
0 SDCTL_CSD0_SEL_B	EMI Chip Select Control Bit for SDCTL over WEIM. Select multiplexed chip select (CS2) between S DRAM/DDR and WEIM in EMI. 0 SDCTL chip select 1 WEIM chip select

4.4.2 Interrupt Observe Control Register (IOMUXC_OBSERVE_INT_MUX)

The interrupt observe control register, shown in Figure 4-13, is used for interrupt observability control, which controls the IOMUX to route specific internal interrupts to pin GPIO_A. The observed interrupt could be used for debugging or software development.

IOMUXC_OBSERVE_INT_MUX — Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	OBSRV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-13. IOMUXC_OBSERVE_INT_MUX Register

Table 4-11. IOMUXC_OBSERVE_INT_MUX Field Descriptions

Field	Description
31–7	Reserved
6–0 OBSRV	This field is used to select the observed interrupt out or debug ports. The selected interrupt can be observed on pin GPIO_A.

Table 4-12. Observed Interrupts List

OBSRV Bits[6:0]	Observed Interrupt Source
0000000	CSPI-3
0000001	GPT-4
0000010	1-Wire
0000011	I^2C-1
0000100	I^2C-2

Table 4-12. Observed Interrupts List (continued)

OBSRV Bits[6:0]	Observed Interrupt Source
0000101	UART-4
0000110	RTIC
0000111	ESAI
0001000	ESDHC-1
0001001	ESDHC-2
0001010	I^2C-3
0001011	SSI-2
0001100	SSI-1
0001101	CSPI-2
0001110	CSPI-1
0001111	ATA
0010000	GPIO-3
0010001	CSI
0010010	UART-3
0010011	IIM
0010100	SIM-1
0010101	SIM-2
0010110	RNGB
0010111	GPIO-4
0011000	KPP
0011001	DRYICE
0011010	PWM-1
0011011	EPIT-2
0011100	EPIT-1
0011101	GPT-3
0011110	POWER FAIL
0011111	CCM
0100000	UART-2
0100001	NANDFC
0100010	SDMA
0100011	USB-HOST
0100100	PWM-2
0100101	USB-OTG

Table 4-12. Observed Interrupts List (continued)

OBSRV Bits[6:0]	Observed Interrupt Source
0100110	SLCDC
0100111	LCDC
0101000	UART-5
0101001	PWM-3
0101010	PWM-4
0101011	CAN-1
0101100	CAN-2
0101101	UART-1
0101110	TSC
0101111	Reserved as VDD
0110000	ECT
0110001	SCC SCM
0110010	SCC SMN
0110011	GPIO-2
0110100	GPIO-1
0110101	GPT-2
0110110	GPT-1
0110111	WDOG
0111000	DRYICE
0111001	FEC
0111010	EXT_INT5 using GPIO1[5]
0111011	EXT_INT4 using GPIO1[4]
0111100	EXT_INT3 using GPIO1[3]
0111101	EXT_INT2 using GPIO1[2]
0111110	EXT_INT1 using GPIO1[1]
0111111	EXT_INT0 using GPIO1[0]
1000000	SDMA Pin debug_yield
1000001	SDMA Pin debug_core_run
1000010	SDMA Pin debug_mode
1000011	SDMA Pin debug_bus_error
1000100	SDMA Pin debug_bus_device[0]
1000101	SDMA Pin debug_bus_device[1]
1000110	SDMA Pin debug_bus_device[2]

Table 4-12. Observed Interrupts List (continued)

OBSRV Bits[6:0]	Observed Interrupt Source
1000111	SDMA Pin debug_bus_device[3]
1001000	SDMA Pin debug_bus_device[4]
1001001	SDMA Pin debug_bus_rwb
1001010	SDMA Pin debug_matched_dmbus
1001011	SDMA Pin debug_rtbuffer_write
1001100	SDMA Pin debug_evt_chn_lines[0]
1001101	SDMA Pin debug_evt_chn_lines[1]
1001110	SDMA Pin debug_evt_chn_lines[2]
1001111	SDMA Pin debug_evt_chn_lines[3]
1010000	SDMA Pin debug_evt_chn_lines[4]
1010001	SDMA Pin debug_evt_chn_lines[5]
1010010	SDMA Pin debug_evt_chn_lines[6]
1010011	SDMA Pin debug_evt_chn_lines[7]
1010100	ARM926P platform Pin etm_portsize[0]
1010101	ARM926P platform Pin etm_portsize[1]
1010110	ARM926P platform Pin etm_portsize[2]
1010111	FEC Pin miigsk_int_col
1011000	FEC Pin miigsk_int_crs
1011001	FEC Pin miigsk_int_rx_clk
1011010	FEC Pin miigsk_int_rx_dv
1011011	FEC Pin miigsk_int_tx_clk
1011100	FEC Pin miigsk_mux_cnt_mii_mode
1011101	FEC Pin miigsk_slow_en

4.4.3 Loopback and GPIO Capture

Loopback is when the module drives the pin and also receives the pin value as an input. GPIO Capture is when the module drives the pin and the value is captured by GPIO module.

These features are available through the SION (Software Input On) bit of the SW_MUX_CTL register (Section 4.2.1, “Software Mux Control Registers (SW_MUX_CTL)). When the SION bit is set, it enables the input path of all muxing modes on the pin by forcing IPP_IBE = 1 (see Figure 4-1), regardless of the original pin direction control.

4.4.4 Boot-Related Pins

The boot modes are primarily controlled by two dedicated boot pins BOOT_MODE0 and BOOT_MODE1. There are also a number of eFUSEs to further determine the specific boot mode and path during the boot process. The fuse value can be determined by IIM or GPIO pins with the control of the GPIO_BT_SEL fuse. When the GPIO_BT_SEL fuse is cleared, the fuse value is determined by the GPIO pins. (See the Boot chapter for details.) The corresponding relationship between GPIO boot pins and eFUSEs are listed in Table 4-13.

Table 4-13. Boot Related Pins

Pin Name	Direction at Boot	eFUSE Name	Details
BOOT_MODE0	INPUT	N/A	Boot mode select pins
BOOT_MODE1	INPUT	N/A	
LD0	INPUT	BT_MEM_CTRL[0]	Boot memory device select
LD1	INPUT	BT_MEM_CTRL[1]	
LD2	INPUT	BT_MEM_TYPE[0]	Boot memory type select
LD3	INPUT	BT_MEM_TYPE[1]	
LD4	INPUT	BT_PAGE_SIZE[0]	NAND Flash page size select
LD5	INPUT	BT_PAGE_SIZE[1]	
LD6	INPUT	BT_BUS_WIDTH[0]	NAND bus width select
LD7	INPUT	BT_BUS_WIDTH[1]	
LD8	INPUT	BT_USB_SRC[0]	USB PHY selection select
LD9	INPUT	BT_USB_SRC[1]	
LD10	INPUT	BT_MLC_SEL	MLC/SLC NAND Flash Select
LD11	INPUT	BT_SPARE_SIZE	NAND Flash spare byte size
LD12	INPUT	BT_SRC[0]	Boot Select for eSDHC, CSPI, and I2C
LD13	INPUT	BT_SRC[1]	
LD14	INPUT	BT_EEPROM_CFG	EEPROM config
LD15	INPUT	BT_UART_SRC[0]	UART boot select
HSYNC	INPUT	BT_UART_SRC[1]	
VSYNC	INPUT	BT_UART_SRC[2]	
LSCLK	INPUT	BT_LPB_FREQ[0]	LPB ARM core frequency select
OE_ACD	INPUT	BT_LPB_FREQ[1]	
PWM	INPUT	BT_LPB_FREQ[2]	

Table 4-13. Boot Related Pins (continued)

Pin Name	Direction at Boot	eFUSE Name	Details
CSI_MCLK	INPUT	BT_RES[0]	Reserved boot options
CSI_VSYNC	INPUT	BT_RES[1]	
CSI_HSYNC	INPUT	BT_RES[2]	
CSI_PIXCLK	INPUT	BT_RES[3]	

Note: In this device, the GPIO boot pins are latched by the CCM on chip reset when the fuse GPIO_BT_SEL is set to 0; no extra software pin-muxing programming is required. After reset, these pins work as normal GPIO pins in its default functionality (ALT0 mode). If you still need the pins to work as boot pins, then the users need to configure IOMUX to select the correct muxing mode.

4.4.5 Special Pins

Table 4-14 lists the special pins supported in the device.

Table 4-14. Special Pins List

Pin Name	Mux Mode	Function Name	Detailed Description
EXT_ARMCLK	ALT0	External ARM clock	External ARM clock input, used for function test when internal ARM clock is bypassed.
UPLL_BYPCLK	ALT0	PLL bypass control	UPLL bypass clock, used when internal UPLL is bypassed.
CLKO	ALT0	Clock out	Clock out pin from CCM, clock source is controllable and can also be used for debug.
VSTBY_ACK	ALT0	CCM hreset_b out	This pin routes out CCM hreset_b for test and debugging in ALT0 mode.
VSTBY_ACK	ALT1	PMIC signal	Standby acknowledge from external PMIC
VSTBY_REQ	ALT0		Standby request from chip to PMIC
POWER_FAIL	ALT0		Power fail interrupt from PMIC
TEST_MODE	ALT0	TCU	Chip enter into test mode when TEST_MODE pin is asserted.
OSC24M_EXTAL	ALT0	24M Oscillator	24 MHz oscillator analog pin
OSC24M_XTAL	ALT0		24 MHz oscillator analog pin
OSC32K_EXTAL	ALT0	32K Oscillator	32 kHz oscillator analog pin
OSC32K_XTAL	ALT0		32 kHz oscillator analog pin
OSC_BYP	ALT0		32 kHz oscillator bypass-control pin

4.5 Register Memory Map

The IOMUX consists of three modules:

- IOMUX Module which contains the signals muxing logic
- Observe Module which contains the interrupt observability logic
- IOMUX_CTL Module which contains the muxing modes, observe and pin settings control logic

Table 4-15 shows the base addresses of IOMUX_CTL and the detailed register memory map. See the system memory map for the IOMUXC_BASE_ADDR.

Table 4-15. IOMUX_CTL Memory Map

Base Address Offset	Register Name	Comment
0x0000	IOMUXC_GPR1	GPR register
0x0004	IOMUXC_OBSERVE_INT_MUX	INT_OBSRV register
0x0008	IOMUXC_SW_MUX_CTL_PAD_A10	SW_MUX_CTL register
0x000c	IOMUXC_SW_MUX_CTL_PAD_A13	—
0x0010	IOMUXC_SW_MUX_CTL_PAD_A14	—
0x0014	IOMUXC_SW_MUX_CTL_PAD_A15	—
0x0018	IOMUXC_SW_MUX_CTL_PAD_A16	—
0x001c	IOMUXC_SW_MUX_CTL_PAD_A17	—
0x0020	IOMUXC_SW_MUX_CTL_PAD_A18	—
0x0024	IOMUXC_SW_MUX_CTL_PAD_A19	—
0x0028	IOMUXC_SW_MUX_CTL_PAD_A20	—
0x002c	IOMUXC_SW_MUX_CTL_PAD_A21	—
0x0030	IOMUXC_SW_MUX_CTL_PAD_A22	—
0x0034	IOMUXC_SW_MUX_CTL_PAD_A23	—
0x0038	IOMUXC_SW_MUX_CTL_PAD_A24	—
0x003c	IOMUXC_SW_MUX_CTL_PAD_A25	—
0x0040	IOMUXC_SW_MUX_CTL_PAD_EB0	—
0x0044	IOMUXC_SW_MUX_CTL_PAD_EB1	—
0x0048	IOMUXC_SW_MUX_CTL_PAD_OE	—
0x004c	IOMUXC_SW_MUX_CTL_PAD_CS0	—
0x0050	IOMUXC_SW_MUX_CTL_PAD_CS1	—
0x0054	IOMUXC_SW_MUX_CTL_PAD_CS4	—
0x0058	IOMUXC_SW_MUX_CTL_PAD_CS5	—
0x005c	IOMUXC_SW_MUX_CTL_PAD_NF_CE0	—
0x0060	IOMUXC_SW_MUX_CTL_PAD_ECB	—
0x0064	IOMUXC_SW_MUX_CTL_PAD_LBA	—
0x0068	IOMUXC_SW_MUX_CTL_PAD_BCLK	—
0x006c	IOMUXC_SW_MUX_CTL_PAD_RW	—
0x0070	IOMUXC_SW_MUX_CTL_PAD_NFWE_B	—
0x0074	IOMUXC_SW_MUX_CTL_PAD_NFRE_B	—
0x0078	IOMUXC_SW_MUX_CTL_PAD_NFALE	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x007c	IOMUXC_SW_MUX_CTL_PAD_NFCLE	—
0x0080	IOMUXC_SW_MUX_CTL_PAD_NFWP_B	—
0x0084	IOMUXC_SW_MUX_CTL_PAD_NFRB	—
0x0088	IOMUXC_SW_MUX_CTL_PAD_D15	—
0x008c	IOMUXC_SW_MUX_CTL_PAD_D14	—
0x0090	IOMUXC_SW_MUX_CTL_PAD_D13	—
0x0094	IOMUXC_SW_MUX_CTL_PAD_D12	—
0x0098	IOMUXC_SW_MUX_CTL_PAD_D11	—
0x009c	IOMUXC_SW_MUX_CTL_PAD_D10	—
0x00a0	IOMUXC_SW_MUX_CTL_PAD_D9	—
0x00a4	IOMUXC_SW_MUX_CTL_PAD_D8	—
0x00a8	IOMUXC_SW_MUX_CTL_PAD_D7	—
0x00ac	IOMUXC_SW_MUX_CTL_PAD_D6	—
0x00b0	IOMUXC_SW_MUX_CTL_PAD_D5	—
0x00b4	IOMUXC_SW_MUX_CTL_PAD_D4	—
0x00b8	IOMUXC_SW_MUX_CTL_PAD_D3	—
0x00bc	IOMUXC_SW_MUX_CTL_PAD_D2	—
0x00c0	IOMUXC_SW_MUX_CTL_PAD_D1	—
0x00c4	IOMUXC_SW_MUX_CTL_PAD_D0	—
0x00c8	IOMUXC_SW_MUX_CTL_PAD_LD0	—
0x00cc	IOMUXC_SW_MUX_CTL_PAD_LD1	—
0x00d0	IOMUXC_SW_MUX_CTL_PAD_LD2	—
0x00d4	IOMUXC_SW_MUX_CTL_PAD_LD3	—
0x00d8	IOMUXC_SW_MUX_CTL_PAD_LD4	—
0x00dc	IOMUXC_SW_MUX_CTL_PAD_LD5	—
0x00e0	IOMUXC_SW_MUX_CTL_PAD_LD6	—
0x00e4	IOMUXC_SW_MUX_CTL_PAD_LD7	—
0x00e8	IOMUXC_SW_MUX_CTL_PAD_LD8	—
0x00ec	IOMUXC_SW_MUX_CTL_PAD_LD9	—
0x00f0	IOMUXC_SW_MUX_CTL_PAD_LD10	—
0x00f4	IOMUXC_SW_MUX_CTL_PAD_LD11	—
0x00f8	IOMUXC_SW_MUX_CTL_PAD_LD12	—
0x00fc	IOMUXC_SW_MUX_CTL_PAD_LD13	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0100	IOMUXC_SW_MUX_CTL_PAD_LD14	—
0x0104	IOMUXC_SW_MUX_CTL_PAD_LD15	—
0x0108	IOMUXC_SW_MUX_CTL_PAD_HSYNC	—
0x010c	IOMUXC_SW_MUX_CTL_PAD_VSYNC	—
0x0110	IOMUXC_SW_MUX_CTL_PAD_LSCLK	—
0x0114	IOMUXC_SW_MUX_CTL_PAD_OE_ACD	—
0x0118	IOMUXC_SW_MUX_CTL_PAD_CONTRAST	—
0x011c	IOMUXC_SW_MUX_CTL_PAD_PWM	—
0x0120	IOMUXC_SW_MUX_CTL_PAD_CSI_D2	—
0x0124	IOMUXC_SW_MUX_CTL_PAD_CSI_D3	—
0x0128	IOMUXC_SW_MUX_CTL_PAD_CSI_D4	—
0x012c	IOMUXC_SW_MUX_CTL_PAD_CSI_D5	—
0x0130	IOMUXC_SW_MUX_CTL_PAD_CSI_D6	—
0x0134	IOMUXC_SW_MUX_CTL_PAD_CSI_D7	—
0x0138	IOMUXC_SW_MUX_CTL_PAD_CSI_D8	—
0x013c	IOMUXC_SW_MUX_CTL_PAD_CSI_D9	—
0x0140	IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK	—
0x0144	IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC	—
0x0148	IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC	—
0x014c	IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK	—
0x0150	IOMUXC_SW_MUX_CTL_PAD_I2C1_CLK	—
0x0154	IOMUXC_SW_MUX_CTL_PAD_I2C1_DAT	—
0x0158	IOMUXC_SW_MUX_CTL_PAD_CSPI1_MOSI	—
0x015c	IOMUXC_SW_MUX_CTL_PAD_CSPI1_MISO	—
0x0160	IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS0	—
0x0164	IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS1	—
0x0168	IOMUXC_SW_MUX_CTL_PAD_CSPI1_SCLK	—
0x016c	IOMUXC_SW_MUX_CTL_PAD_CSPI1_RDY	—
0x0170	IOMUXC_SW_MUX_CTL_PAD_UART1_RXD	—
0x0174	IOMUXC_SW_MUX_CTL_PAD_UART1_TXD	—
0x0178	IOMUXC_SW_MUX_CTL_PAD_UART1_RTS	—
0x017c	IOMUXC_SW_MUX_CTL_PAD_UART1_CTS	—
0x0180	IOMUXC_SW_MUX_CTL_PAD_UART2_RXD	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0184	IOMUXC_SW_MUX_CTL_PAD_UART2_TXD	—
0x0188	IOMUXC_SW_MUX_CTL_PAD_UART2_RTS	—
0x018c	IOMUXC_SW_MUX_CTL_PAD_UART2_CTS	—
0x0190	IOMUXC_SW_MUX_CTL_PAD_SD1_CMD	—
0x0194	IOMUXC_SW_MUX_CTL_PAD_SD1_CLK	—
0x0198	IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0	—
0x019c	IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1	—
0x01a0	IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2	—
0x01a4	IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3	—
0x01a8	IOMUXC_SW_MUX_CTL_PAD_KPP_ROW0	—
0x01ac	IOMUXC_SW_MUX_CTL_PAD_KPP_ROW1	—
0x01b0	IOMUXC_SW_MUX_CTL_PAD_KPP_ROW2	—
0x01b4	IOMUXC_SW_MUX_CTL_PAD_KPP_ROW3	—
0x01b8	IOMUXC_SW_MUX_CTL_PAD_KPP_COL0	—
0x01bc	IOMUXC_SW_MUX_CTL_PAD_KPP_COL1	—
0x01c0	IOMUXC_SW_MUX_CTL_PAD_KPP_COL2	—
0x01c4	IOMUXC_SW_MUX_CTL_PAD_KPP_COL3	—
0x01c8	IOMUXC_SW_MUX_CTL_PAD_FEC_MDC	—
0x01cc	IOMUXC_SW_MUX_CTL_PAD_FEC_MDIO	—
0x01d0	IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA0	—
0x01d4	IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA1	—
0x01d8	IOMUXC_SW_MUX_CTL_PAD_FEC_TX_EN	—
0x01dc	IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA0	—
0x01e0	IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA1	—
0x01e4	IOMUXC_SW_MUX_CTL_PAD_FEC_RX_DV	—
0x01e8	IOMUXC_SW_MUX_CTL_PAD_FEC_TX_CLK	—
0x01ec	IOMUXC_SW_MUX_CTL_PAD_RTCK	—
0x01f0	IOMUXC_SW_MUX_CTL_PAD_DE_B	—
0x01f4	IOMUXC_SW_MUX_CTL_PAD_GPIO_A	—
0x01f8	IOMUXC_SW_MUX_CTL_PAD_GPIO_B	—
0x01fc	IOMUXC_SW_MUX_CTL_PAD_GPIO_C	—
0x0200	IOMUXC_SW_MUX_CTL_PAD_GPIO_D	—
0x0204	IOMUXC_SW_MUX_CTL_PAD_GPIO_E	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0208	IOMUXC_SW_MUX_CTL_PAD_GPIO_F	—
0x020c	IOMUXC_SW_MUX_CTL_PAD_EXT_ARMCLK	—
0x0210	IOMUXC_SW_MUX_CTL_PAD_UPLL_BYPCLK	—
0x0214	IOMUXC_SW_MUX_CTL_PAD_VSTBY_REQ	—
0x0218	IOMUXC_SW_MUX_CTL_PAD_VSTBY_ACK	—
0x021c	IOMUXC_SW_MUX_CTL_PAD_POWER_FAIL	—
0x0220	IOMUXC_SW_MUX_CTL_PAD_CLKO	—
0x0224	IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0	—
0x0228	IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1	—
0x022c	IOMUXC_SW_PAD_CTL_PAD_A13	SW_PAD_CTL register
0x0230	IOMUXC_SW_PAD_CTL_PAD_A14	—
0x0234	IOMUXC_SW_PAD_CTL_PAD_A15	—
0x0238	IOMUXC_SW_PAD_CTL_PAD_A17	—
0x023c	IOMUXC_SW_PAD_CTL_PAD_A18	—
0x0240	IOMUXC_SW_PAD_CTL_PAD_A19	—
0x0244	IOMUXC_SW_PAD_CTL_PAD_A20	—
0x0248	IOMUXC_SW_PAD_CTL_PAD_A21	—
0x024c	IOMUXC_SW_PAD_CTL_PAD_A23	—
0x0250	IOMUXC_SW_PAD_CTL_PAD_A24	—
0x0254	IOMUXC_SW_PAD_CTL_PAD_A25	—
0x0258	IOMUXC_SW_PAD_CTL_PAD_EB0	—
0x025c	IOMUXC_SW_PAD_CTL_PAD_EB1	—
0x0260	IOMUXC_SW_PAD_CTL_PAD_OE	—
0x0264	IOMUXC_SW_PAD_CTL_PAD_CS4	—
0x0268	IOMUXC_SW_PAD_CTL_PAD_CS5	—
0x026c	IOMUXC_SW_PAD_CTL_PAD_NF_CE0	—
0x0270	IOMUXC_SW_PAD_CTL_PAD_ECB	—
0x0274	IOMUXC_SW_PAD_CTL_PAD_LBA	—
0x0278	IOMUXC_SW_PAD_CTL_PAD_RW	—
0x027c	IOMUXC_SW_PAD_CTL_PAD_NFRB	—
0x0280	IOMUXC_SW_PAD_CTL_PAD_D15	—
0x0284	IOMUXC_SW_PAD_CTL_PAD_D14	—
0x0288	IOMUXC_SW_PAD_CTL_PAD_D13	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x028c	IOMUXC_SW_PAD_CTL_PAD_D12	—
0x0290	IOMUXC_SW_PAD_CTL_PAD_D11	—
0x0294	IOMUXC_SW_PAD_CTL_PAD_D10	—
0x0298	IOMUXC_SW_PAD_CTL_PAD_D9	—
0x029c	IOMUXC_SW_PAD_CTL_PAD_D8	—
0x02a0	IOMUXC_SW_PAD_CTL_PAD_D7	—
0x02a4	IOMUXC_SW_PAD_CTL_PAD_D6	—
0x02a8	IOMUXC_SW_PAD_CTL_PAD_D5	—
0x02ac	IOMUXC_SW_PAD_CTL_PAD_D4	—
0x02b0	IOMUXC_SW_PAD_CTL_PAD_D3	—
0x02b4	IOMUXC_SW_PAD_CTL_PAD_D2	—
0x02b8	IOMUXC_SW_PAD_CTL_PAD_D1	—
0x02bc	IOMUXC_SW_PAD_CTL_PAD_D0	—
0x02c0	IOMUXC_SW_PAD_CTL_PAD_LD0	—
0x02c4	IOMUXC_SW_PAD_CTL_PAD_LD1	—
0x02c8	IOMUXC_SW_PAD_CTL_PAD_LD2	—
0x02cc	IOMUXC_SW_PAD_CTL_PAD_LD3	—
0x02d0	IOMUXC_SW_PAD_CTL_PAD_LD4	—
0x02d4	IOMUXC_SW_PAD_CTL_PAD_LD5	—
0x02d8	IOMUXC_SW_PAD_CTL_PAD_LD6	—
0x02dc	IOMUXC_SW_PAD_CTL_PAD_LD7	—
0x02e0	IOMUXC_SW_PAD_CTL_PAD_LD8	—
0x02e4	IOMUXC_SW_PAD_CTL_PAD_LD9	—
0x02e8	IOMUXC_SW_PAD_CTL_PAD_LD10	—
0x02ec	IOMUXC_SW_PAD_CTL_PAD_LD11	—
0x02f0	IOMUXC_SW_PAD_CTL_PAD_LD12	—
0x02f4	IOMUXC_SW_PAD_CTL_PAD_LD13	—
0x02f8	IOMUXC_SW_PAD_CTL_PAD_LD14	—
0x02fc	IOMUXC_SW_PAD_CTL_PAD_LD15	—
0x0300	IOMUXC_SW_PAD_CTL_PAD_HSYNC	—
0x0304	IOMUXC_SW_PAD_CTL_PAD_VSYNC	—
0x0308	IOMUXC_SW_PAD_CTL_PAD_LSCLK	—
0x030c	IOMUXC_SW_PAD_CTL_PAD_OE_ACD	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0310	IOMUXC_SW_PAD_CTL_PAD_CONTRAST	—
0x0314	IOMUXC_SW_PAD_CTL_PAD_PWM	—
0x0318	IOMUXC_SW_PAD_CTL_PAD_CSI_D2	—
0x031c	IOMUXC_SW_PAD_CTL_PAD_CSI_D3	—
0x0320	IOMUXC_SW_PAD_CTL_PAD_CSI_D4	—
0x0324	IOMUXC_SW_PAD_CTL_PAD_CSI_D5	—
0x0328	IOMUXC_SW_PAD_CTL_PAD_CSI_D6	—
0x032c	IOMUXC_SW_PAD_CTL_PAD_CSI_D7	—
0x0330	IOMUXC_SW_PAD_CTL_PAD_CSI_D8	—
0x0334	IOMUXC_SW_PAD_CTL_PAD_CSI_D9	—
0x0338	IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK	—
0x033c	IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC	—
0x0340	IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC	—
0x0344	IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK	—
0x0348	IOMUXC_SW_PAD_CTL_PAD_I2C1_CLK	—
0x034c	IOMUXC_SW_PAD_CTL_PAD_I2C1_DAT	—
0x0350	IOMUXC_SW_PAD_CTL_PAD_CSPI1_MOSI	—
0x0354	IOMUXC_SW_PAD_CTL_PAD_CSPI1_MISO	—
0x0358	IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS0	—
0x035c	IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS1	—
0x0360	IOMUXC_SW_PAD_CTL_PAD_CSPI1_SCLK	—
0x0364	IOMUXC_SW_PAD_CTL_PAD_CSPI1_RDY	—
0x0368	IOMUXC_SW_PAD_CTL_PAD_UART1_RXD	—
0x036c	IOMUXC_SW_PAD_CTL_PAD_UART1_TXD	—
0x0370	IOMUXC_SW_PAD_CTL_PAD_UART1_RTS	—
0x0374	IOMUXC_SW_PAD_CTL_PAD_UART1_CTS	—
0x0378	IOMUXC_SW_PAD_CTL_PAD_UART2_RXD	—
0x037c	IOMUXC_SW_PAD_CTL_PAD_UART2_TXD	—
0x0380	IOMUXC_SW_PAD_CTL_PAD_UART2_RTS	—
0x0384	IOMUXC_SW_PAD_CTL_PAD_UART2_CTS	—
0x0388	IOMUXC_SW_PAD_CTL_PAD_SD1_CMD	—
0x038c	IOMUXC_SW_PAD_CTL_PAD_SD1_CLK	—
0x0390	IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0394	IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1	—
0x0398	IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2	—
0x039c	IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3	—
0x03a0	IOMUXC_SW_PAD_CTL_PAD_KPP_ROW0	—
0x03a4	IOMUXC_SW_PAD_CTL_PAD_KPP_ROW1	—
0x03a8	IOMUXC_SW_PAD_CTL_PAD_KPP_ROW2	—
0x03ac	IOMUXC_SW_PAD_CTL_PAD_KPP_ROW3	—
0x03b0	IOMUXC_SW_PAD_CTL_PAD_KPP_COL0	—
0x03b4	IOMUXC_SW_PAD_CTL_PAD_KPP_COL1	—
0x03b8	IOMUXC_SW_PAD_CTL_PAD_KPP_COL2	—
0x03bc	IOMUXC_SW_PAD_CTL_PAD_KPP_COL3	—
0x03c0	IOMUXC_SW_PAD_CTL_PAD_FEC_MDC	—
0x03c4	IOMUXC_SW_PAD_CTL_PAD_FEC_MDIO	—
0x03c8	IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA0	—
0x03cc	IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA1	—
0x03d0	IOMUXC_SW_PAD_CTL_PAD_FEC_TX_EN	—
0x03d4	IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA0	—
0x03d8	IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA1	—
0x03dc	IOMUXC_SW_PAD_CTL_PAD_FEC_RX_DV	—
0x03e0	IOMUXC_SW_PAD_CTL_PAD_FEC_TX_CLK	—
0x03e4	IOMUXC_SW_PAD_CTL_PAD_RTCK	—
0x03e8	IOMUXC_SW_PAD_CTL_PAD_TDO	—
0x03ec	IOMUXC_SW_PAD_CTL_PAD_DE_B	—
0x03f0	IOMUXC_SW_PAD_CTL_PAD_GPIO_A	—
0x03f4	IOMUXC_SW_PAD_CTL_PAD_GPIO_B	—
0x03f8	IOMUXC_SW_PAD_CTL_PAD_GPIO_C	—
0x03fc	IOMUXC_SW_PAD_CTL_PAD_GPIO_D	—
0x0400	IOMUXC_SW_PAD_CTL_PAD_GPIO_E	—
0x0404	IOMUXC_SW_PAD_CTL_PAD_GPIO_F	—
0x0408	IOMUXC_SW_PAD_CTL_PAD_VSTBY_REQ	—
0x040c	IOMUXC_SW_PAD_CTL_PAD_VSTBY_ACK	—
0x0410	IOMUXC_SW_PAD_CTL_PAD_POWER_FAIL	—
0x0414	IOMUXC_SW_PAD_CTL_PAD_CLKO	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0418	IOMUXC_SW_PAD_CTL_GRP_DVS_MISC	SW_PAD_CTL_GRP register
0x041c	IOMUXC_SW_PAD_CTL_GRP_DSE_FEC	—
0x0420	IOMUXC_SW_PAD_CTL_GRP_DVS_JTAG	—
0x0424	IOMUXC_SW_PAD_CTL_GRP_DSE_NFC	—
0x0428	IOMUXC_SW_PAD_CTL_GRP_DSE_CSI	—
0x042c	IOMUXC_SW_PAD_CTL_GRP_DSE_WEIM	—
0x0430	IOMUXC_SW_PAD_CTL_GRP_DSE_DDR	—
0x0434	IOMUXC_SW_PAD_CTL_GRP_DVS_CCM	—
0x0438	IOMUXC_SW_PAD_CTL_GRP_DSE_KPP	—
0x043c	IOMUXC_SW_PAD_CTL_GRP_DSE_SDHC1	—
0x0440	IOMUXC_SW_PAD_CTL_GRP_DSE_LCD	—
0x0444	IOMUXC_SW_PAD_CTL_GRP_DSE_UART	—
0x0448	IOMUXC_SW_PAD_CTL_GRP_DVS_NFC	—
0x044c	IOMUXC_SW_PAD_CTL_GRP_DVS_CSI	—
0x0450	IOMUXC_SW_PAD_CTL_GRP_DSE_CSPI1	—
0x0454	IOMUXC_SW_PAD_CTL_GRP_DDRTYPE	—
0x0458	IOMUXC_SW_PAD_CTL_GRP_DVS_SDHC1	—
0x045c	IOMUXC_SW_PAD_CTL_GRP_DVS_LCD	—
0x0460	IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT	SW_SELECT_INPUT register
0x0464	IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT	—
0x0468	IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT	—
0x046c	IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT	—
0x0470	IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT	—
0x0474	IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT	—
0x0478	IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT	—
0x047c	IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT	—
0x0480	IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT	—
0x0484	IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT	—
0x0488	IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT	—
0x048c	IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT	—
0x0490	IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT	—
0x0494	IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT	—
0x0498	IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x049c	IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT	—
0x04a0	IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT	—
0x04a4	IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT	—
0x04a8	IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT	—
0x04ac	IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT	—
0x04b0	IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT	—
0x04b4	IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT	—
0x04b8	IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT	—
0x04bc	IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT	—
0x04c0	IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT	—
0x04c4	IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT	—
0x04c8	IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT	—
0x04cc	IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT	—
0x04d0	IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT	—
0x04d4	IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT	—
0x04d8	IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT	—
0x04dc	IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT	—
0x04e0	IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT	—
0x04e4	IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT	—
0x04e8	IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT	—
0x04ec	IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT	—
0x04f0	IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT	—
0x04f4	IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT	—
0x04f8	IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT	—
0x04fc	IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT	—
0x0500	IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT	—
0x0504	IOMUXC_FEC_FEC_COL_SELECT_INPUT	—
0x0508	IOMUXC_FEC_FEC_CRS_SELECT_INPUT	—
0x050c	IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT	—
0x0510	IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT	—
0x0514	IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT	—
0x0518	IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT	—
0x051c	IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT	—

Table 4-15. IOMUX_CTL Memory Map (continued)

Base Address Offset	Register Name	Comment
0x0520	IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT	—
0x0524	IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT	—
0x0528	IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT	—
0x052c	IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT	—
0x0530	IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT	—
0x0534	IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT	—
0x0538	IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT	—
0x053c	IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT	—
0x0540	IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT	—
0x0544	IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT	—
0x0548	IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT	—
0x054c	IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT	—
0x0550	IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT	—
0x0554	IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT	—
0x0558	IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT	—
0x055c	IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT	—
0x0560	IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT	—
0x0564	IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT	—
0x0568	IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT	—
0x056c	IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT	—
0x0570	IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT	—
0x0574	IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT	—
0x0578	IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT	—
0x057c	IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT	—
0x0580	IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT	—

4.6 Daisy Chain List

Table 4-16 lists all the instances and ports which are involved in Daisy Chain, and also the corresponding values of SW_SELECT_INPUT registers. SW_SELECT_INPUT registers follow the naming convention of: IOMUXC_<*Instance*>_<*Port*>_SELECT_INPUT.

For example, if pin EB1 need to work as the input port p4_input_da_amx of instance AUDMUX, then configure pin EB1 to work in ALT4 mode, and configure register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT to value 0x0 to solve Daisy Chain.

Table 4-16. Daisy Chain List

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
AUDMUX	p4_input_da_amx	EB1	ALT4	0x0
		FEC_MDIO	ALT2	0x1
	p4_input_db_amx	EB0	ALT4	0x0
		FEC_MDC	ALT2	0x1
	p4_input_rxclk_amx	CS4	ALT4	0x0
		FEC_TX_EN	ALT2	0x1
	p4_input_rxfs_amx	CS5	ALT4	0x0
		FEC_RDATA0	ALT2	0x1
	p4_input_txclk_amx	OE	ALT4	0x0
		FEC_TDATA0	ALT2	0x1
	p4_input_txfs_amx	RW	ALT4	0x0
		FEC_TDATA1	ALT2	0x1
	p7_input_da_amx	SD1_DATA1	ALT3	0x0
		POWER_FAIL	ALT4	0x1
	p7_input_txfs_amx	SD1_DATA0	ALT3	0x0
		VSTBY_REQ	ALT4	0x1
CAN1	ipp_ind_canrx	FEC_RDATA0	ALT4	0x0
		GPIO_B	ALT6	0x1
CAN2	ipp_ind_canrx	FEC_RX_DV	ALT4	0x0
		GPIO_D	ALT6	0x1
CSI	ipp_csi_d[0]	LD0	ALT2	0x0
		UART1_RTS	ALT1	0x1
		KPP_ROW2	ALT3	0x2
	ipp_csi_d[1]	LD1	ALT2	0x0
		UART1_CTS	ALT1	0x1
		KPP_ROW3	ALT3	0x2
CSPI1	ipp_ind_ss3_b	NF_CE0	ALT1	0x0
		VSTBY_ACK	ALT2	0x1

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
CSPI2	ipp_cspi_clk_in	LD14	ALT2	0x0
		SD1_DATA0	ALT1	0x1
	ipp_ind_dataready_b	LD15	ALT2	0x0
		SD1_DATA1	ALT1	0x1
	ipp_ind_miso	LD13	ALT2	0x0
		SD1_CLK	ALT1	0x1
	ipp_ind_mosi	LD12	ALT2	0x0
		SD1_CMD	ALT1	0x1
	ipp_ind_ss0_b	OE_ACD	ALT2	0x0
		SD1_DATA2	ALT1	0x1
	ipp_ind_ss1_b	CONTRAST	ALT2	0x0
		SD1_DATA3	ALT1	0x1
CSPI3	ipp_cspi_clk_in	ECB	ALT6	0x0
		CSI_D4	ALT7	0x1
	ipp_ind_dataready_b	LBA	ALT6	0x0
		CSI_D5	ALT7	0x1
	ipp_ind_miso	CS5	ALT6	0x0
		CSI_D3	ALT7	0x1
	ipp_ind_mosi	CS4	ALT6	0x0
		CSI_D2	ALT7	0x1
	ipp_ind_ss0_b	EB0	ALT6	0x0
		CSI_D6	ALT7	0x1
	ipp_ind_ss1_b	EB1	ALT6	0x0
		CSI_D7	ALT7	0x1
	ipp_ind_ss2_b	CSI_D8	ALT7	0x0
		GPIO_D	ALT7	0x1
	ipp_ind_ss3_b	CSI_D9	ALT7	0x0
		UART2_CTS	ALT6	0x1

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
ESDHC1	ipp_dat4_in	D12	ALT6	0x0
		UART2_CTS	ALT1	0x1
	ipp_dat5_in	D13	ALT6	0x0
		UART2_RTS	ALT1	0x1
	ipp_dat6_in	D14	ALT6	0x0
		UART2_TXD	ALT1	0x1
	ipp_dat7_in	D15	ALT6	0x0
		UART2_RXD	ALT1	0x1

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
ESDHC2	ipp_card_clk_in	LD9	ALT6	0x0
		CSI_D7	ALT2	0x1
		FEC_MDIO	ALT1	0x2
	ipp_cmd_in	LD8	ALT6	0x0
		CSI_D6	ALT2	0x1
		FEC_MDC	ALT1	0x2
	ipp_dat0_in	LD10	ALT6	0x0
		CSI_MCLK	ALT2	0x1
		FEC_TDATA0	ALT1	0x2
	ipp_dat1_in	LD11	ALT6	0x0
		CSI_VSYNC	ALT2	0x1
		FEC_TDATA1	ALT1	0x2
	ipp_dat2_in	LD12	ALT6	0x0
		CSI_HSYNC	ALT2	0x1
		FEC_TX_EN	ALT1	0x2
	ipp_dat3_in	LD13	ALT6	0x0
		CSI_PIXCLK	ALT2	0x1
		FEC_RDATA0	ALT1	0x2
	ipp_dat4_in	CSI_D2	ALT2	0x0
		FEC_RDATA1	ALT2	0x1
	ipp_dat5_in	CSI_D3	ALT2	0x0
		FEC_RX_DV	ALT2	0x1
	ipp_dat6_in	CSI_D4	ALT2	0x0
		FEC_TX_CLK	ALT2	0x1
	ipp_dat7_in	CSI_D5	ALT2	0x0
		RTCK	ALT2	0x1

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
FEC	fec_col	A18	ALT7	0x0
		LD9	ALT5	0x1
		UART2_RTS	ALT2	0x2
	fec_crs	A25	ALT7	0x0
		CONTRAST	ALT5	0x1
		SD1_DATA3	ALT2	0x2
	fec_rdata[2]	A20	ALT7	0x0
		LD11	ALT5	0x1
		SD1_CMD	ALT2	0x2
	fec_rdata[3]	A21	ALT7	0x0
		LD12	ALT5	0x1
		SD1_CLK	ALT2	0x2
	fec_rx_clk	A24	ALT7	0x0
		LD15	ALT5	0x1
		SD1_DATA2	ALT2	0x2
	fec_rx_er	A19	ALT7	0x0
		LD10	ALT5	0x1
		UART2_CTS	ALT2	0x2
I2C2	ipp_scl_in	FEC_RDATA1	ALT1	0x0
		GPIO_C	ALT2	0x1
I2C2	ipp_sda_in	FEC_RX_DV	ALT1	0x0
		GPIO_D	ALT2	0x1
I2C3	ipp_scl_in	HSYNC	ALT2	0x0
		GPIO_A	ALT4	0x1
		GPIO_E	ALT1	0x2
I2C3	ipp_sda_in	VSYNC	ALT2	0x0
		CSPI1_SS1	ALT1	0x1
		GPIO_B	ALT4	0x2

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
KPP	ipp_ind_col[4]	FEC_RDATA1	ALT6	0x0
		GPIO_C	ALT3	0x1
	ipp_ind_col[5]	FEC_RX_DV	ALT6	0x0
		GPIO_D	ALT3	0x1
	ipp_ind_col[6]	LD14	ALT4	0x0
		CSI_D8	ALT1	0x1
	ipp_ind_col[7]	LD15	ALT4	0x0
		CSI_D9	ALT1	0x1
	ipp_ind_row[4]	FEC_TX_EN	ALT6	0x0
		GPIO_A	ALT3	0x1
	ipp_ind_row[5]	FEC_RDATA0	ALT6	0x0
		GPIO_B	ALT3	0x1
	ipp_ind_row[6]	LD12	ALT4	0x0
		CSI_D6	ALT1	0x1
	ipp_ind_row[7]	LD13	ALT4	0x0
		CSI_D7	ALT1	0x1
SIM1	pin_sim_rcvd1_in	A19	ALT6	0x0
		LD5	ALT4	0x1
	pin_sim_simpd1	A18	ALT6	0x0
		LD4	ALT4	0x1
	sim_rcvd1_io	A17	ALT6	0x0
		LD3	ALT4	0x1
SIM2	pin_sim_rcvd1_in	A25	ALT6	0x0
		OE_ACD	ALT4	0x1
	pin_sim_simpd1	A24	ALT6	0x0
		LSCLK	ALT4	0x1
	sim_rcvd1_io	A23	ALT6	0x0
		VSYNC	ALT4	0x1
UART3	ipp_uart_rts_b	CSPI1_SS1	ALT2	0x0
		KPP_ROW2	ALT1	0x1
	ipp_uart_rxd_mux	CSPI1_MOSI	ALT2	0x0
		KPP_ROW0	ALT1	0x1

Table 4-16. Daisy Chain List (continued)

Instance	Input Port	Pin	Mode	SW_SELECT_INPUT Value
UART4	ipp_uart_rts_b	LD10	ALT2	0x0
		KPP_COL2	ALT1	0x1
		VSTBY_REQ	ALT6	0x2
	ipp_uart_rxd_mux	LD8	ALT2	0x0
		KPP_COL0	ALT1	0x1
		GPIO_E	ALT6	0x2
UART5	ipp_uart_rts_b	CS5	ALT3	0x0
		CSI_D4	ALT1	0x1
	ipp_uart_rxd_mux	LBA	ALT3	0x0
		CSI_D2	ALT1	0x1
USB_top	ipp_ind_otg_usb_oc	D10	ALT6	0x0
		GPIO_B	ALT2	0x1
	ipp_ind_uh2_usb_oc	D8	ALT6	0x0
		PWM	ALT6	0x1

4.7 Pin Multiplexing

This section shows pin muxing options. Both a high-level picture and detailed pin muxing options are presented.

4.7.1 Pin Multiplexing Overview

Table 4-17 shows the main high-level pin muxing options.

Table 4-17. i.MX25 Simplified High-level Pin Muxing

i.MX25	Default Mode	Mux Mode						
Pin Num	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
21	LCDC	SLCDC (16)	CSI (8)	PATA (21)	SIM1 P1(6)	GPIO	USB ULPI (8)	GPIO_BOOT (20)
					SIM2 P1(2)	GPIO		
			UART4 (4)		SSI-P3 (4)	FEC (9)	SDIO2 (6)	
			CSPI2 (4)		KPP (4)		SSI-P3 (2)	
		—	I2C3 (2)		SIM2 P1 (4)	GPIO	USB ULPI (6)	
		SLCDC(2)	CSPI2 (2)			GPIO		
		GPT4 (2)	—		PWM (1)	GPIO		WDOG (1)
1	PWM1		—	—	—	GPIO		GPIO_BOOT (1)

Table 4-17. i.MX25 Simplified High-level Pin Muxing (continued)

i.MX25	Default Mode	Mux Mode						
Pin Num	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
12	CSI	UART5 (4)	SDIO2 (6)	ESAI (12)	SIM1 P0 (6)	GPIO	OTG ULPI (12)	CSPI3 (8)
		KPP (4)	SSI-P6 (2)		—	GPIO		
		SSI-P6 (4)	SDIO2 (4)		SIM2 P0 (6)	GPIO		GPIO_BOOT (4)
2	I2C1	—	—	—	—	GPIO	SLCDC (12)	—
6	CSPI1	—	UART3 (4)	—	—	GPIO		ETM (6)
		LCDC_D[16] (1)	PWM2 (1)	—	—	GPIO		
		I2C3 (1)	—	—	—	GPIO		
4	UART1	CSI_D[1:0] (2)	GPT3 (2)	—	—	GPIO		—
4	UART2	SDIO1 (4)	FEC (9)	GPT1 (2)	EPIT2 (1)	GPIO	CSPI2/CSPI3 SS (2)	DMA_EVT (3)
6	SDIO1	CSPI2 (6)		—	—	GPIO	SLCDC (6)	ETM (6)
8	KPP	UART3	SSI-P5	—	—	GPIO	—	—
		UART4		CSI_D[1:0] (2)	—	GPIO	—	—
9	FEC	SDIO2 (6)	SSI-P4 (6)	PATA (8)	CAN1	GPIO	KPP (4)	LCDC (8)
		I2C2 (2)	SDIO2 (3)		CAN2	GPIO		
		PWM3 (1)		LCDC_D[16] (1)	—	GPIO	—	—
6	GPIO	PWM2 (1)	USB PWR/OC (2)	KPP (4)	I2C3 (2)	—	CAN1 (2)	—
		PWM3 (1)				—		—
		PWM4 (1)	I2C2 (2)		GPT2 (2)	CSPI1 SS (2)	CAN2 (2)	CSPI2 SS (1)
		WDOG (1)				—		CSPI3 SS (1)
		I2C3 (1)	LCDC_D[16] (1)	—	SSI-P7 (4)	—	UART4 (4)	ECT (2)
		—	LCDC_D[17] (1)	EPIT1 (1)		—		
11	CCM	—	CSPI1 SS (1)	EPIT1 (1)		GPIO		-
1	TCU	—	—	—	—	—	—	—
8	JTAG	O-WIRE (1)	SDIO2 (1)	—	—	—	—	—
30	DDR/SDRAM	—	—	—	—	—	—	—

Table 4-17. i.MX25 Simplified High-level Pin Muxing (continued)

i.MX25	Default Mode	Mux Mode						
Pin Num	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
16	WEIM DATA	LCDC_D[23:16]	—	—	—	—	eSDHC1 (4)	—
			—	—	—	GPIO	USB PWR/OC (4)	—
26	WEIM ADDR	—	—	—	—	GPIO	SIM1 P1 (6)	—
		—	—	—	—	GPIO	SIM2 P1 (6)	—
7	WEIM CTL	—	—	—	SSI-P4 (6)	GPIO	CSPI3 (6)	—
7	WEIM CS	NFC CS (3)	—	—		GPIO		ETM (3)
		CSPI1 SS (1)	—	—	—	GPIO	—	
6	NFC	—	—	—	—	GPIO	—	ETM
2	FS USBPHY	—	—	—	—	—	—	—
5	HS USBPHY	—	—	—	—	—	—	—
9	ADC	—	—	—	—	—	—	—
7	DRYICE	—	—	—	—	—	—	—
2	OSC24M	—	—	—	—	—	—	—

4.7.2 Detailed Pin Multiplexing Description

Table 4-18 shows in detail the pin muxing and pad settings of each pin. The Pin Name column lists the i.MX25 pins and the Instance column lists the instantiated on-chip IP modules. For each pin that supports pin muxing, there is a related IOMUX cell for muxing control. Each IOMUX cell can support up to eight muxing modes: ALT0–ALT7, where ALT0 is the default mode. The Pad Settings column shows the pad characteristic setting of each pin.

Following is an example to clarify Table 4-18.

- Muxing Mode:

  Pin A10 serves as the WEIM address pin of EMI module (ALT0 mode) or as a GPIO (ALT5 mode). Select the desired working mode by programing the SW_MUX_CTL register of pin A10, see Section 4.2.1, “Software Mux Control Registers (SW_MUX_CTL).”

- Pad Settings:

  The A10 pin has following settings:

  — *Drive Strength—CFG (High)*, which means the default drive strength is set as high, and software-configurable.

  — *Keeper—Enabled*, which means internal keeper is enabled, and non-software-configurable.

  — *DDR Type—CFG(mDDR)*, which means the DDR type is set as mDDR mode at default, and software-configurable.

Only characteristics with CFG ( ) are software-configurable, and then users can program SW_PAD_CTL registers in IOMUX_CTL to select different setting values.

Table 4-18. i.MX25 Detailed Pin Muxing

Pin Name	Mode	Instance	Port	Pad Settings
A0	No Muxing (ALT0)	EMI	EIM_DA_L[0]	Drive Strength—CFG (High) Keeper—Enabled DDR Type—CFG (mDDR)
A1		EMI	EIM_DA_L[1]	
A2		EMI	EIM_DA_L[2]	
A3		EMI	EIM_DA_L[3]	
A4		EMI	EIM_DA_L[4]	
A5		EMI	EIM_DA_L[5]	
A6		EMI	EIM_DA_L[6]	
A7		EMI	EIM_DA_L[7]	
A8		EMI	EIM_DA_H[8]	
A9		EMI	EIM_DA_H[9]	
A10	ALT0	EMI	EIM_DA_H[10]	Drive Strength—CFG (High) Keeper—Enabled DDR Type—CFG (mDDR)
	ALT5	GPIO4	GPIO[0]	
MA10	No Muxing (ALT0)	EMI	MA10	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
A11	No Muxing (ALT0)	EMI	EIM_DA_H[11]	Drive Strength—CFG (High) Keeper—Enabled DDR Type—CFG (mDDR)
A12		EMI	EIM_DA_H[12]	
A13	ALT0	EMI	EIM_DA_H[13]	Drive Strength—CFG (High) Keeper—CFG (Enabled) DDR Type—CFG (mDDR)
	ALT5	GPIO4	GPIO[1]	
	ALT7	LCDC	LCDC_CLS	
A14	ALT0	EMI	EIM_DA_H2[14]	Drive Strength—CFG (High) Keeper—CFG (Enabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[0]	
	ALT6	SIM1	CLK1	
	ALT7	LCDC	LCDC_SPL	
A15	ALT0	EMI	EIM_DA_H2[15]	Drive Strength—CFG (High) Keeper—CFG (Enabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[1]	
	ALT6	SIM1	RST1	
	ALT7	LCDC	LCDC_PS	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
A16	ALT0	EMI	EIM_A[16]	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[2]	
	ALT6	SIM1	VEN1	
	ALT7	LCDC	LCDC_REV	
A17	ALT0	EMI	EIM_A[17]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[3]	
	ALT6	SIM1	TX1	
	ALT7	FEC	TX_ERR	
A18	ALT0	EMI	EIM_A[18]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[4]	
	ALT6	SIM1	PD1	
	ALT7	FEC	COL	
A19	ALT0	EMI	EIM_A[19]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[5]	
	ALT6	SIM1	RX1	
	ALT7	FEC	RX_ERR	
A20	ALT0	EMI	EIM_A[20]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[6]	
	ALT6	SIM2	CLK1	
	ALT7	FEC	RDATA[2]	
A21	ALT0	EMI	EIM_A[21]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[7]	
	ALT6	SIM2	RST1	
	ALT7	FEC	RDATA[3]	
A22	ALT0	EMI	EIM_A[22]	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[8]	
	ALT6	SIM2	VEN1	
	ALT7	FEC	TDATA[2]	
A23	ALT0	EMI	EIM_A[23]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[9]	
	ALT6	SIM2	TX1	
	ALT7	FEC	TDATA[3]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
A24	ALT0	EMI	EIM_A[24]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[10]	
	ALT6	SIM2	PD1	
	ALT7	FEC	RX_CLK	
A25	ALT0	EMI	EIM_A[25]	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT5	GPIO2	GPIO[11]	
	ALT6	SIM2	RX1	
	ALT7	FEC	CRS	
SD0	No Muxing (ALT0)	EMI	DRAM_D[0]	Drive Strength—CFG (High) Keeper—Enabled DDR Type—CFG (mDDR)
SD1		EMI	DRAM_D[1]	
SD2		EMI	DRAM_D[2]	
SD3		EMI	DRAM_D[3]	
SD4		EMI	DRAM_D[4]	
SD5		EMI	DRAM_D[5]	
SD6		EMI	DRAM_D[6]	
SD7		EMI	DRAM_D[7]	
SD8		EMI	DRAM_D[8]	
SD9		EMI	DRAM_D[9]	
SD10		EMI	DRAM_D[10]	
SD11		EMI	DRAM_D[11]	
SD12		EMI	DRAM_D[12]	
SD13		EMI	DRAM_D[13]	
SD14		EMI	DRAM_D[14]	
SD15		EMI	DRAM_D[15]	
SDBA1	No Muxing (ALT0)	EMI	EIM_SDBA1	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
SDBA0		EMI	EIM_SDBA0	
DQM0		EMI	DRAM_DQM[0]	
DQM1		EMI	DRAM_DQM[1]	
RAS		EMI	DRAM_RAS	
CAS		EMI	DRAM_CAS	
SDWE		EMI	DRAM_SDWE	
SDCKE0		EMI	DRAM_SDCKE[0]	
SDCKE1		EMI	DRAM_SDCKE[1]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
SDCLK	No Muxing (ALT0)	EMI	DRAM_SDCLK	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
SDQS0	No Muxing (ALT0)	EMI	DRAM_SDQS[0]	Drive Strength—CFG (High) Keeper—Enabled DDR Type—CFG (mDDR)
SDQS1		EMI	DRAM_SDQS[1]	
EB0	ALT0	EMI	EIM_EB0_B	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT4	AUDMUX	AUD4_TXD	
	ALT5	GPIO2	GPIO[12]	
	ALT6	CSPI3	SS0	
EB1	ALT0	EMI	EIM_EB1_B	
	ALT4	AUDMUX	AUD4_RXD	
	ALT5	GPIO2	GPIO[13]	
	ALT6	CSPI3	SS1	
OE	ALT0	EMI	EIM_OE	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT4	AUDMUX	AUD4_TXC	
	ALT5	GPIO2	GPIO[14]	
CS0	ALT0	EMI	EIM_CS0	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
	ALT5	GPIO4	GPIO[2]	
CS1	ALT0	EMI	EIM_CS1	
	ALT1	EMI	NANDF_CE3	
	ALT5	GPIO4	GPIO[3]	
CS2	No Muxing (ALT0)	EMI	EIM_CS2	
CS3		EMI	EIM_CS3	
CS4	ALT0	EMI	EIM_CS4	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configure—CFG (100 KΩ PD) Open Drain Enable—Disabled Drive Voltage Select—CFG (1.8 V) Pull/Keep Select—Pull Slew Rate—CFG (FAST)
	ALT1	EMI	NANDF_CE1	
	ALT3	UART5	CTS	
	ALT4	AUDMUX	AUD4_RXC	
	ALT5	GPIO3	GPIO[20]	
	ALT6	CSPI3	MOSI	
	ALT7	ARM926P_PLATFORM	TRSYNC	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CS5	ALT0	EMI	EIM_CS5	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—Disabled Drive Voltage Select—CFG (1.8 V) Pull/Keep Select—Pull Slew Rate—CFG (FAST)
	ALT1	EMI	NANDF_CE2	
	ALT2	EMI	DTACK_B	
	ALT3	UART5	RTS	
	ALT4	AUDMUX	AUD4_RXFS	
	ALT5	GPIO3	GPIO[21]	
	ALT6	CSPI3	MISO	
	ALT7	ARM926P_PLATFORM	TRCLK	
NF_CE0	ALT0	EMI	NANDF_CE0	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (FAST)
	ALT1	CSPI1	SS3	
	ALT5	GPIO3	GPIO[22]	
	ALT7	ARM926P_PLATFORM	TRACE[3]	
ECB	ALT0	EMI	EIM_ECB	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (1.8 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT3	UART5	TXD_MUX	
	ALT5	GPIO3	GPIO[23]	
	ALT6	CSPI3	SCLK	
LBA	ALT0	EMI	EIM_LBA	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT3	UART5	RXD_MUX	
	ALT5	GPIO3	GPIO[24]	
	ALT6	CSPI3	RDY	
BCLK	ALT0	EMI	EIM_BCLK	Drive Strength—CFG (High) Keeper—Disabled DDR Type—CFG (mDDR)
	ALT5	GPIO4	GPIO[4]	
RW	ALT0	EMI	EIM_RW	Drive Strength—CFG (High) Keeper—CFG (Disabled) DDR Type—CFG (mDDR)
	ALT4	AUDMUX	AUD4_TXFS	
	ALT5	GPIO3	GPIO[25]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
NFWE_B	ALT0	EMI	NANDF_WE_B	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—Disabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—FAST
	ALT5	GPIO3	GPIO[26]	
	ALT7	ARM926P_PLATFORM	PIPESTAT[2]	
NFRE_B	ALT0	EMI	NANDF_RE_B	
	ALT5	GPIO3	GPIO[27]	
	ALT7	ARM926P_PLATFORM	PIPESTAT[1]	
NFALE	ALT0	EMI	NANDF_ALE	
	ALT5	GPIO3	GPIO[28]	
	ALT7	ARM926P_PLATFORM	PIPESTAT[0]	
NFCLE	ALT0	EMI	NANDF_CLE	
	ALT5	GPIO3	GPIO[29]	
	ALT7	ARM926P_PLATFORM	TRACE[0]	
NFWP_B	ALT0	EMI	NANDF_WP_B	
	ALT5	GPIO3	GPIO[30]	
	ALT7	ARM926P_PLATFORM	TRACE[1]	
NFRB	ALT0	EMI	NANDF_RB	Hysteresis Enable—Disabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—FAST
	ALT5	GPIO3	GPIO[31]	
	ALT7	ARM926P_PLATFORM	TRACE[2]	
D15	ALT0	EMI	EIM_D[15]	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[16]	
	ALT5	GPIO4	GPIO[5]	
	ALT6	ESDHC1	DAT7	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
D14	ALT0	EMI	EIM_D[14]	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[17]	
	ALT5	GPIO4	GPIO[6]	
	ALT6	ESDHC1	DAT6	
D13	ALT0	EMI	EIM_D[13]	
	ALT1	LCDC	LCDC_LD[18]	
	ALT5	GPIO4	GPIO[7]	
	ALT6	ESDHC1	DAT5	
D12	ALT0	EMI	EIM_D[12]	
	ALT1	LCDC	LCDC_LD[19]	
	ALT5	GPIO4	GPIO[8]	
	ALT6	ESDHC1	DAT4	
D11	ALT0	EMI	EIM_D[11]	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[20]	
	ALT5	GPIO4	GPIO[9]	
	ALT6	USB_TOP	USBOTG_PWR	
D10	ALT0	EMI	EIM_D[10]	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[21]	
	ALT5	GPIO4	GPIO[10]	
	ALT6	USB_TOP	USBOTG_OC	
D9	ALT0	EMI	EIM_D[9]	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[22]	
	ALT5	GPIO4	GPIO[11]	
	ALT6	USB_TOP	USBH2_PWR	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
D8	ALT0	EMI	EIM_D[8]	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	LCDC	LCDC_LD[23]	
	ALT5	GPIO4	GPIO[12]	
	ALT6	USB_TOP	USBH2_OC	
D7	ALT0	EMI	EIM_D[7]	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—FAST
	ALT5	GPIO4	GPIO[13]	
D6	ALT0	EMI	EIM_D[6]	
	ALT5	GPIO4	GPIO[14]	
D5	ALT0	EMI	EIM_D[5]	
	ALT5	GPIO4	GPIO[15]	
D4	ALT0	EMI	EIM_D[4]	
	ALT5	GPIO4	GPIO[16]	
D3	ALT0	EMI	EIM_D[3]	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—FAST
	ALT5	GPIO4	GPIO[17]	
D2	ALT0	EMI	EIM_D[2]	
	ALT5	GPIO4	GPIO[18]	
D1	ALT0	EMI	EIM_D[1]	
	ALT5	GPIO4	GPIO[19]	
D0	ALT0	EMI	EIM_D[0]	
	ALT5	GPIO4	GPIO[20]	
LD0	ALT0	LCDC	LCDC_LD[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	SLCDC	SLCDC_DATA[0]	
	ALT2	CSI	CSI_D[0]	
	ALT3	ATA	DATA[0]	
	ALT4	SIM1	CLK1	
	ALT5	GPIO2	GPIO[15]	
	ALT6	USB_TOP	USBH2_CLK	
	ALT7	CCM	BT_MEM_CTRL[0]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
LD1	ALT0	LCDC	LCDC_LD[1]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	SLCDC	SLCDC_DATA[1]	
	ALT2	CSI	CSI_D[1]	
	ALT3	ATA	DATA[1]	
	ALT4	SIM1	RST1	
	ALT5	GPIO2	GPIO[16]	
	ALT6	USB_TOP	USBH2_DIR	
	ALT7	CCM	BT_MEM_CTRL[1]	
LD2	ALT0	LCDC	LCDC_LD[2]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[2]	
	ALT2	CSI	CSI_D[15]	
	ALT3	ATA	DATA[2]	
	ALT4	SIM1	VEN1	
	ALT5	GPIO2	GPIO[17]	
	ALT6	USB_TOP	USBH2_STP	
	ALT7	CCM	BT_MEM_TYPE[0]	
LD3	ALT0	LCDC	LCDC_LD[3]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	SLCDC	SLCDC_DATA[3]	
	ALT2	CSI	CSI_D[14]	
	ALT3	ATA	DATA[3]	
	ALT4	SIM1	TX1	
	ALT5	GPIO2	GPIO[18]	
	ALT6	USB_TOP	USBH2_NXT	
	ALT7	CCM	BT_MEM_TYPE[1]	
LD4	ALT0	LCDC	LCDC_LD[4]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[4]	
	ALT2	CSI	CSI_D[13]	
	ALT3	ATA	DATA[4]	
	ALT4	SIM1	PD1	
	ALT5	GPIO2	GPIO[19]	
	ALT6	USB_TOP	USBH2_DATA[0]	
	ALT7	CCM	BT_PAGE_SIZE[0]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
LD5	ALT0	LCDC	LCDC_LD[5]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[5]	
	ALT2	CSI	CSI_D[12]	
	ALT3	ATA	DATA[5]	
	ALT4	SIM1	RX1	
	ALT5	GPIO1	GPIO[19]	
	ALT6	USB_TOP	USBH2_DATA[1]	
	ALT7	CCM	BT_PAGE_SIZE[1]	
LD6	ALT0	LCDC	LCDC_LD[6]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[6]	
	ALT2	CSI	CSI_D[11]	
	ALT3	ATA	DATA[6]	
	ALT4	SIM2	CLK1	
	ALT5	GPIO1	GPIO[20]	
	ALT6	USB_TOP	USBH2_DATA[2]	
	ALT7	CCM	BT_BUS_WIDTH[0]	
LD7	ALT0	LCDC	LCDC_LD[7]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[7]	
	ALT2	CSI	CSI_D[10]	
	ALT3	ATA	DATA[7]	
	ALT4	SIM2	RST1	
	ALT5	GPIO1	GPIO[21]	
	ALT6	USB_TOP	USBH2_DATA[3]	
	ALT7	CCM	BT_BUS_WIDTH[1]	
LD8	ALT0	LCDC	LCDC_LD[8]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[8]	
	ALT2	UART4	RXD_MUX	
	ALT3	ATA	DATA[8]	
	ALT4	AUDMUX	AUD3_TXD	
	ALT5	FEC	TX_ERR	
	ALT6	ESDHC2	CMD	
	ALT7	CCM	BT_USB_SRC[0]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
LD9	ALT0	LCDC	LCDC_LD[9]	Hysteresis Enable—CFG (Enabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[9]	
	ALT2	UART4	TXD_MUX	
	ALT3	ATA	DATA[9]	
	ALT4	AUDMUX	AUD3_RXD	
	ALT5	FEC	COL	
	ALT6	ESDHC2	CLK	
	ALT7	CCM	BT_USB_SRC[1]	
LD10	ALT0	LCDC	LCDC_LD[10]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[10]	
	ALT2	UART4	RTS	
	ALT3	ATA	DATA[10]	
	ALT4	AUDMUX	AUD3_TXC	
	ALT5	FEC	RX_ERR	
	ALT6	ESDHC2	DAT0	
	ALT7	CCM	BT_MLC_SEL	
LD11	ALT0	LCDC	LCDC_LD[11]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[11]	
	ALT2	UART4	CTS	
	ALT3	ATA	DATA[11]	
	ALT4	AUDMUX	AUD3_TXFS	
	ALT5	FEC	RDATA[2]	
	ALT6	ESDHC2	DAT1	
	ALT7	CCM	BT_SPARE_SIZE	
LD12	ALT0	LCDC	LCDC_LD[12]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[12]	
	ALT2	CSPI2	MOSI	
	ALT3	ATA	DATA[12]	
	ALT4	KPP	ROW[6]	
	ALT5	FEC	RDATA[3]	
	ALT6	ESDHC2	DAT2	
	ALT7	CCM	BT_SRC[0]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
LD13	ALT0	LCDC	LCDC_LD[13]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_DATA[13]	
	ALT2	CSPI2	MISO	
	ALT3	ATA	DATA[13]	
	ALT4	KPP	ROW[7]	
	ALT5	FEC	TDATA[2]	
	ALT6	ESDHC2	DAT3	
	ALT7	CCM	BT_SRC[1]	
LD14	ALT0	LCDC	LCDC_LD[14]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	SLCDC	SLCDC_DATA[14]	
	ALT2	CSPI2	SCLK	
	ALT3	ATA	DATA[14]	
	ALT4	KPP	COL[6]	
	ALT5	FEC	TDATA[3]	
	ALT6	AUDMUX	AUD3_RXC	
	ALT7	CCM	BT_EEPROM_CFG	
LD15	ALT0	LCDC	LCDC_LD[15]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	SLCDC	SLCDC_DATA[15]	
	ALT2	CSPI2	RDY	
	ALT3	ATA	DATA[15]	
	ALT4	KPP	COL[7]	
	ALT5	FEC	RX_CLK	
	ALT6	AUDMUX	AUD3_RXFS	
	ALT7	CCM	BT_UART_SRC[0]	
HSYNC	ALT0	LCDC	LCDC_HSYN	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT2	I2C3	SCL	
	ALT3	ATA	BUFFER_EN	
	ALT4	SIM2	VEN1	
	ALT5	GPIO1	GPIO[22]	
	ALT6	USB_TOP	USBH2_DATA[4]	
	ALT7	CCM	BT_UART_SRC[1]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
VSYNC	ALT0	LCDC	LCDC_VSYN	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT2	I2C3	SDA	
	ALT3	ATA	DMARQ	
	ALT4	SIM2	TX1	
	ALT5	GPIO1	GPIO[23]	
	ALT6	USB_TOP	USBH2_DATA[5]	
	ALT7	CCM	BT_UART_SRC[2]	
LSCLK	ALT0	LCDC	LCDC_LSCLK	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	SLCDC	SLCDC_CS	
	ALT3	ATA	DA_0	
	ALT4	SIM2	PD1	
	ALT5	GPIO1	GPIO[24]	
	ALT6	USB_TOP	USBH2_DATA[6]	
	ALT7	CCM	BT_LPB_FREQ[0]	
OE_ACD	ALT0	LCDC	LCDC_OE_ACD	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	SLCDC	SLCDC_RS	
	ALT2	CSPI2	SS0	
	ALT3	ATA	DA_1	
	ALT4	SIM2	RX1	
	ALT5	GPIO1	GPIO[25]	
	ALT6	USB_TOP	USBH2_DATA[7]	
	ALT7	CCM	BT_LPB_FREQ[1]	
CONTRAST	ALT0	LCDC	LCDC_CONTRAST	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	GPT4	CAPIN1	
	ALT2	CSPI2	SS1	
	ALT3	ATA	DA_2	
	ALT4	PWM4	PWMO	
	ALT5	FEC	CRS	
	ALT6	USB_TOP	USBH2_PWR	
	ALT7	WDOG	WDOG_B	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
PWM	ALT0	PWM1	PWMO	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	GPT4	CMPOUT1	
	ALT5	GPIO1	GPIO[26]	
	ALT6	USB_TOP	USBH2_OC	
	ALT7	CCM	BT_LPB_FREQ[2]	
CSI_D2	ALT0	CSI	CSI_D[2]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	UART5	RXD_MUX	
	ALT2	ESDHC2	DAT4	
	ALT3	ESAI	SCKR	
	ALT4	SIM1	CLK0	
	ALT5	GPIO1	GPIO[27]	
	ALT6	USB_TOP	USBOTG_DATA[0]	
	ALT7	CSPI3	MOSI	
CSI_D3	ALT0	CSI	CSI_D[3]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	UART5	TXD_MUX	
	ALT2	ESDHC2	DAT5	
	ALT3	ESAI	FSR	
	ALT4	SIM1	RST0	
	ALT5	GPIO1	GPIO[28]	
	ALT6	USB_TOP	USBOTG_DATA[1]	
	ALT7	CSPI3	MISO	
CSI_D4	ALT0	CSI	CSI_D[4]	Hysteresis Enable—CFG (Enabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (FAST)
	ALT1	UART5	RTS	
	ALT2	ESDHC2	DAT6	
	ALT3	ESAI	HCKR	
	ALT4	SIM1	VEN0	
	ALT5	GPIO1	GPIO[29]	
	ALT6	USB_TOP	USBOTG_DATA[2]	
	ALT7	CSPI3	SCLK	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CSI_D5	ALT0	CSI	CSI_D[5]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	UART5	CTS	
	ALT2	ESDHC2	DAT7	
	ALT3	ESAI	SCKT	
	ALT4	SIM1	TX0	
	ALT5	GPIO1	GPIO[30]	
	ALT6	USB_TOP	USBOTG_DATA[3]	
	ALT7	CSPI3	RDY	
CSI_D6	ALT0	CSI	CSI_D[6]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	KPP	ROW[6]	
	ALT2	ESDHC2	CMD	
	ALT3	ESAI	FST	
	ALT4	SIM1	PD0	
	ALT5	GPIO1	GPIO[31]	
	ALT6	USB_TOP	USBOTG_DATA[4]	
	ALT7	CSPI3	SS0	
CSI_D7	ALT0	CSI	CSI_D[7]	Hysteresis Enable—CFG (Enabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	KPP	ROW[7]	
	ALT2	ESDHC2	CLK	
	ALT3	ESAI	HCKT	
	ALT4	SIM1	RX0	
	ALT5	GPIO1	GPIO[6]	
	ALT6	USB_TOP	USBOTG_DATA[5]	
	ALT7	CSPI3	SS1	
CSI_D8	ALT0	CSI	CSI_D[8]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	KPP	COL[6]	
	ALT2	AUDMUX	AUD6_RXC	
	ALT3	ESAI	TX5_RX0	
	ALT4	SIM2	CLK0	
	ALT5	GPIO1	GPIO[7]	
	ALT6	USB_TOP	USBOTG_DATA[6]	
	ALT7	CSPI3	SS2	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CSI_D9	ALT0	CSI	CSI_D[9]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	KPP	COL[7]	
	ALT2	AUDMUX	AUD6_RXFS	
	ALT3	ESAI	TX4_RX1	
	ALT4	SIM2	RST0	
	ALT5	GPIO4	GPIO[21]	
	ALT6	USB_TOP	USBOTG_DATA[7]	
	ALT7	CSPI3	SS3	
CSI_MCLK	ALT0	CSI	CSI_MCLK	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	AUDMUX	AUD6_TXD	
	ALT2	ESDHC2	DAT0	
	ALT3	ESAI	TX3_RX2	
	ALT4	SIM2	VEN0	
	ALT5	GPIO1	GPIO[8]	
	ALT6	USB_TOP	USBOTG_DIR	
	ALT7	CCM	BT_RES[0]	
CSI_VSYNC	ALT0	CSI	CSI_VSYNC	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	AUDMUX	AUD6_RXD	
	ALT2	ESDHC2	DAT1	
	ALT3	ESAI	TX2_RX3	
	ALT4	SIM2	TX0	
	ALT5	GPIO1	GPIO[9]	
	ALT6	USB_TOP	USBOTG_STP	
	ALT7	CCM	BT_RES[1]	
CSI_HSYNC	ALT0	CSI	CSI_HSYNC	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	AUDMUX	AUD6_TXC	
	ALT2	ESDHC2	DAT2	
	ALT3	ESAI	TX1	
	ALT4	SIM2	PD0	
	ALT5	GPIO1	GPIO[10]	
	ALT6	USB_TOP	USBOTG_NXT	
	ALT7	CCM	BT_RES[2]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CSI_PIXCLK	ALT0	CSI	CSI_PIXCLK	Hysteresis Enable—CFG (Enabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Keep) Slew Rate—CFG (SLOW)
	ALT1	AUDMUX	AUD6_TXFS	
	ALT2	ESDHC2	DAT3	
	ALT3	ESAI	TX0	
	ALT4	SIM2	RX0	
	ALT5	GPIO1	GPIO[11]	
	ALT6	USB_TOP	USBOTG_CLK	
	ALT7	CCM	BT_RES[3]	
I2C1_CLK	ALT0	I2C1	SCL	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Enabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT5	GPIO1	GPIO[12]	
	ALT6	SLCDC	SLCDC_DATA[6]	
I2C1_DAT	ALT0	I2C1	SDA	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Enabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT5	GPIO1	GPIO[13]	
	ALT6	SLCDC	SLCDC_DATA[7]	
CSPI1_MOSI	ALT0	CSPI1	MOSI	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (SLOW)
	ALT2	UART3	RXD_MUX	
	ALT4	SDMA	SDMA_DBG_EVT_0	
	ALT5	GPIO1	GPIO[14]	
	ALT6	SLCDC	SLCDC_DATA[12]	
	ALT7	ARM926P_PLATFORM	TRACE[4]	
CSPI1_MISO	ALT0	CSPI1	MISO	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (SLOW)
	ALT2	UART3	TXD_MUX	
	ALT4	SDMA	SDMA_DBG_EVT_1	
	ALT5	GPIO1	GPIO[15]	
	ALT6	SLCDC	SLCDC_DATA[13]	
	ALT7	ARM926P_PLATFORM	TRACE[5]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CSPI1_SS0	ALT0	CSPI1	SS0	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	LCDC	LCDC_LD[16]	
	ALT2	PWM2	PWMO	
	ALT4	SDMA	SDMA_DBG_EVT_2	
	ALT5	GPIO1	GPIO[16]	
	ALT6	SLCDC	SLCDC_CS	
	ALT7	ARM926P_PLATFORM	TRACE[6]	
CSPI1_SS1	ALT0	CSPI1	SS1	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (SLOW)
	ALT1	I2C3	SDA	
	ALT2	UART3	RTS	
	ALT4	SDMA	SDMA_DBG_EVT_3	
	ALT5	GPIO1	GPIO[17]	
	ALT6	SLCDC	SLCDC_RS	
	ALT7	ARM926P_PLATFORM	TRACE[7]	
CSPI1_SCLK	ALT0	CSPI1	SCLK	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (SLOW)
	ALT2	UART3	CTS	
	ALT4	SDMA	SDMA_DBG_EVT_4	
	ALT5	GPIO1	GPIO[18]	
	ALT6	SLCDC	SLCDC_DATA[14]	
	ALT7	ARM926P_PLATFORM	TRACE[8]	
CSPI1_RDY	ALT0	CSPI1	RDY	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—CFG (SLOW)
	ALT4	SDMA	SDMA_DBG_EVT_5	
	ALT5	GPIO2	GPIO[22]	
	ALT6	SLCDC	SLCDC_DATA[15]	
	ALT7	ARM926P_PLATFORM	TRACE[9]	
UART1_RXD	ALT0	UART1	RXD_MUX	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT3	UART2	DTR	
	ALT4	LCDC	LCDC_CLS	
	ALT5	GPIO4	GPIO[22]	
	ALT6	SLCDC	SLCDC_DATA[8]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
UART1_TXD	ALT0	UART1	TXD_MUX	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT3	UART2	DSR	
	ALT4	LCDC	LCDC_SPL	
	ALT5	GPIO4	GPIO[23]	
	ALT6	SLCDC	SLCDC_DATA[9]	
UART1_RTS	ALT0	UART1	RTS	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	CSI	CSI_D[0]	
	ALT2	GPT3	CAPIN1	
	ALT3	UART2	DCD	
	ALT4	LCDC	LCDC_PS	
	ALT5	GPIO4	GPIO[24]	
	ALT6	SLCDC	SLCDC_DATA[10]	
UART1_CTS	ALT0	UART1	CTS	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	CSI	CSI_D[1]	
	ALT2	GPT3	CMPOUT1	
	ALT3	UART2	RI	
	ALT4	LCDC	LCDC_REV	
	ALT5	GPIO4	GPIO[25]	
	ALT6	SLCDC	SLCDC_DATA[11]	
UART2_RXD	ALT0	UART2	RXD_MUX	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—FAST
	ALT1	ESDHC1	DAT7	
	ALT5	GPIO4	GPIO[26]	
UART2_TXD	ALT0	UART2	TXD_MUX	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC1	DAT6	
	ALT2	FEC	TX_ERR	
	ALT5	GPIO4	GPIO[27]	
	ALT7	SDMA	EXTDMA_0	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
UART2_RTS	ALT0	UART2	RTS	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	ESDHC1	DAT5	
	ALT2	FEC	COL	
	ALT3	GPT1	CAPIN1	
	ALT4	EPIT2	DO_EPITO	
	ALT5	GPIO4	GPIO[28]	
	ALT6	CSPI2	SS3	
	ALT7	SDMA	EXTDMA_1	
UART2_CTS	ALT0	UART2	CTS	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC1	DAT4	
	ALT2	FEC	RX_ERR	
	ALT3	GPT1	CMPOUT1	
	ALT5	GPIO4	GPIO[29]	
	ALT6	CSPI3	SS3	
	ALT7	SDMA	EXTDMA_2	
SD1_CMD	ALT0	ESDHC1	CMD	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	MOSI	
	ALT2	FEC	RDATA[2]	
	ALT4	SDMA	SDMA_DBG_EVT_SEL	
	ALT5	GPIO2	GPIO[23]	
	ALT6	SLCDC	SLCDC_DATA[0]	
	ALT7	ARM926P_PLATFORM	TRACE[10]	
SD1_CLK	ALT0	ESDHC1	CLK	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	MISO	
	ALT2	FEC	RDATA[3]	
	ALT4	SDMA	SDMA_DBG_STAT_0	
	ALT5	GPIO2	GPIO[24]	
	ALT6	SLCDC	SLCDC_DATA[1]	
	ALT7	ARM926P_PLATFORM	TRACE[11]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
SD1_DATA0	ALT0	ESDHC1	DAT0	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	SCLK	
	ALT2	FEC	TDATA[2]	
	ALT3	AUDMUX	AUD7_TXFS	
	ALT4	SDMA	SDMA_DBG_STAT_1	
	ALT5	GPIO2	GPIO[25]	
	ALT6	SLCDC	SLCDC_DATA[2]	
	ALT7	ARM926P_PLATFORM	TRACE[12]	
SD1_DATA1	ALT0	ESDHC1	DAT1	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	RDY	
	ALT2	FEC	TDATA[3]	
	ALT3	AUDMUX	AUD7_RXD	
	ALT4	SDMA	SDMA_DBG_STAT_2	
	ALT5	GPIO2	GPIO[26]	
	ALT6	SLCDC	SLCDC_DATA[3]	
	ALT7	ARM926P_PLATFORM	TRACE[13]	
SD1_DATA2	ALT0	ESDHC1	DAT2	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	SS0	
	ALT2	FEC	RX_CLK	
	ALT3	AUDMUX	AUD7_RXC	
	ALT4	SDMA	SDMA_DBG_STAT_3	
	ALT5	GPIO2	GPIO[27]	
	ALT6	SLCDC	SLCDC_DATA[4]	
	ALT7	ARM926P_PLATFORM	TRACE[14]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
SD1_DATA3	ALT0	ESDHC1	DAT3	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (47 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	CSPI2	SS1	
	ALT2	FEC	CRS	
	ALT3	AUDMUX	AUD7_RXFS	
	ALT5	GPIO2	GPIO[28]	
	ALT6	SLCDC	SLCDC_DATA[5]	
	ALT7	ARM926P_PLATFORM	TRACE[15]	
KPP_ROW0	ALT0	KPP	ROW[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	UART3	RXD_MUX	
	ALT4	UART1	DTR	
	ALT5	GPIO2	GPIO[29]	
	ALT6	SDMA	SDMA_DBG_PC_0	
KPP_ROW1	ALT0	KPP	ROW[1]	
	ALT1	UART3	TXD_MUX	
	ALT4	UART1	DSR	
	ALT5	GPIO2	GPIO[30]	
	ALT6	SDMA	SDMA_DBG_PC_1	
KPP_ROW2	ALT0	KPP	ROW[2]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	UART3	RTS	
	ALT2	AUDMUX	AUD5_RXC	
	ALT3	CSI	CSI_D[0]	
	ALT4	UART1	DCD	
	ALT5	GPIO2	GPIO[31]	
	ALT6	SDMA	SDMA_DBG_PC_2	
KPP_ROW3	ALT0	KPP	ROW[3]	
	ALT1	UART3	CTS	
	ALT2	AUDMUX	AUD5_RXFS	
	ALT3	CSI	CSI_D[1]	
	ALT4	UART1	RI	
	ALT5	GPIO3	GPIO[0]	
	ALT6	SDMA	SDMA_DBG_PC_3	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
KPP_COL0	ALT0	KPP	COL[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Enabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	UART4	RXD_MUX	
	ALT2	AUDMUX	AUD5_TXD	
	ALT5	GPIO3	GPIO[1]	
	ALT6	SDMA	SDMA_DBG_PC_4	
KPP_COL1	ALT0	KPP	COL[1]	
	ALT1	UART4	TXD_MUX	
	ALT2	AUDMUX	AUD5_RXD	
	ALT5	GPIO3	GPIO[2]	
	ALT6	SDMA	SDMA_DBG_PC_5	
KPP_COL2	ALT0	KPP	COL[2]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Enabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	UART4	RTS	
	ALT2	AUDMUX	AUD5_TXC	
	ALT5	GPIO3	GPIO[3]	
	ALT6	SDMA	SDMA_DBG_PC_6	
	ALT7	EMI	M3IF_CHOSEN_MASTER_1	
KPP_COL3	ALT0	KPP	COL[3]	
	ALT1	UART4	CTS	
	ALT2	AUDMUX	AUD5_TXFS	
	ALT5	GPIO3	GPIO[4]	
	ALT6	SDMA	SDMA_DBG_PC_7	
	ALT7	EMI	M3IF_CHOSEN_MASTER_2	
FEC_MDC	ALT0	FEC	MDC	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC2	CMD	
	ALT2	AUDMUX	AUD4_TXD	
	ALT3	ATA	DIOR	
	ALT5	GPIO3	GPIO[5]	
	ALT6	SDMA	SDMA_DBG_PC_8	
	ALT7	LCDC	LCDC_LD[16]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
FEC_MDIO	ALT0	FEC	MDIO	Hysteresis Enable—CFG (Enabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (22 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC2	CLK	
	ALT2	AUDMUX	AUD4_RXD	
	ALT3	ATA	DIOW	
	ALT5	GPIO3	GPIO[6]	
	ALT6	SDMA	SDMA_DBG_PC_9	
	ALT7	LCDC	LCDC_LD[17]	
FEC_TDATA0	ALT0	FEC	TDATA[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC2	DAT0	
	ALT2	AUDMUX	AUD4_TXC	
	ALT3	ATA	DMACK	
	ALT5	GPIO3	GPIO[7]	
	ALT6	SDMA	SDMA_DBG_PC_10	
	ALT7	LCDC	LCDC_LD[18]	
FEC_TDATA1	ALT0	FEC	TDATA[1]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC2	DAT1	
	ALT2	AUDMUX	AUD4_TXFS	
	ALT3	ATA	RESET_B	
	ALT5	GPIO3	GPIO[8]	
	ALT6	SDMA	SDMA_DBG_PC_11	
	ALT7	LCDC	LCDC_LD[19]	
FEC_TX_EN	ALT0	FEC	TX_EN	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	ESDHC2	DAT2	
	ALT2	AUDMUX	AUD4_RXC	
	ALT3	ATA	IORDY	
	ALT4	CAN1	TXCAN	
	ALT5	GPIO3	GPIO[9]	
	ALT6	KPP	ROW[4]	
	ALT7	LCDC	LCDC_LD[20]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
FEC_RDATA0	ALT0	FEC	RDATA[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (FAST)
	ALT1	ESDHC2	DAT3	
	ALT2	AUDMUX	AUD4_RXFS	
	ALT3	ATA	INTRQ	
	ALT4	CAN1	RXCAN	
	ALT5	GPIO3	GPIO[10]	
	ALT6	KPP	ROW[5]	
	ALT7	LCDC	LCDC_LD[21]	
FEC_RDATA1	ALT0	FEC	RDATA[1]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	I2C2	SCL	
	ALT2	ESDHC2	DAT4	
	ALT3	ATA	CS0	
	ALT4	CAN2	TXCAN	
	ALT5	GPIO3	GPIO[11]	
	ALT6	KPP	COL[4]	
	ALT7	LCDC	LCDC_LD[22]	
FEC_RX_DV	ALT0	FEC	RX_DV	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	I2C2	SDA	
	ALT2	ESDHC2	DAT5	
	ALT3	ATA	CS1	
	ALT4	CAN2	RXCAN	
	ALT5	GPIO3	GPIO[12]	
	ALT6	KPP	COL[5]	
	ALT7	LCDC	LCDC_LD[23]	
FEC_TX_CLK	ALT0	FEC	TX_CLK	Hysteresis Enable—CFG (Disabled) Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	PWM3	PWMO	
	ALT2	ESDHC2	DAT6	
	ALT3	LCDC	LCDC_LD[16]	
	ALT5	GPIO3	GPIO[13]	
	ALT6	SDMA	SDMA_DBG_PC_12	
	ALT7	EMI	M3IF_CHOSEN_MASTER_0	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
RTCK	ALT0	ARM926P_PLATFORM	RTCK	Hysteresis Enable—Enabled Drive Strength—CFG (High) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—CFG (SLOW)
	ALT1	OWIRE	LINE	
	ALT2	ESDHC2	DAT7	
	ALT5	GPIO3	GPIO[14]	
	ALT6	SDMA	SDMA_DBG_PC_13	
TCK	No Muxing (ALT0)	SJC	TCK	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PD Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
TMS	No Muxing (ALT0)	SJC	TMS	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—47 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
TDI		SJC	TDI	
TDO		SJC	TDO	Hysteresis Enable—Disabled Drive Strength—CFG (High) Pull/Keep Enable—Disabled Pull Up/Down Configuration—47 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
TRSTB	No Muxing (ALT0)	SJC	TRSTB	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—47 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
DE_B	ALT0	SJC	DE_B	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—Enabled Pull Up/Down Configuration—47 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT5	GPIO2	GPIO[20]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
SJC_MOD	No Muxing (ALT0)	SJC	MOD	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
USBPHY1_VBUS		USBPHY_UTMI	USBPHY1_VBUS	—
USBPHY1_DP		USBPHY_UTMI	USBPHY1_DP	—
USBPHY1_DM		USBPHY_UTMI	USBPHY1_DM	—
USBPHY1_UID		USBPHY_UTMI	USBPHY1_UID	—
USBPHY1_RREF		USBPHY_UTMI	USBPHY1_RREF	—
USBPHY2_DM		USBXCVR	USBPHY2_DM	—
USBPHY2_DP		USBXCVR	USBPHY2_DP	—
GPIO_A	ALT0	GPIO1	GPIO[0]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	PWM2	PWMO	
	ALT2	USB_TOP	USBOTG_PWR	
	ALT3	KPP	ROW[4]	
	ALT4	I2C3	SCL	
	ALT6	CAN1	TXCAN	
	ALT7	OBSRV_MODULE	INT_MUX_OUT	
GPIO_B	ALT0	GPIO1	GPIO[1]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	PWM3	PWMO	
	ALT2	USB_TOP	USBOTG_OC	
	ALT3	KPP	ROW[5]	
	ALT4	I2C3	SDA	
	ALT6	CAN1	RXCAN	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
GPIO_C	ALT0	GPIO1	GPIO[2]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—CFG (Pull) Slew Rate—SLOW
	ALT1	PWM4	PWMO	
	ALT2	I2C2	SCL	
	ALT3	KPP	COL[4]	
	ALT4	GPT2	CAPIN1	
	ALT5	CSPI1	SS2	
	ALT6	CAN2	TXCAN	
	ALT7	CSPI2	SS2	
GPIO_D	ALT0	GPIO1	GPIO[3]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Disabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	WDOG	WDOG_B	
	ALT2	I2C2	SDA	
	ALT3	KPP	COL[5]	
	ALT4	GPT2	CMPOUT1	
	ALT6	CAN2	RXCAN	
	ALT7	CSPI3	SS2	
GPIO_E	ALT0	GPIO1	GPIO[4]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—CFG (Enabled) Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	I2C3	SCL	
	ALT2	LCDC	LCDC_LD[16]	
	ALT4	AUDMUX	AUD7_TXD	
	ALT6	UART4	RXD_MUX	
	ALT7	ECT	CTI_TRIG_IN0_6	
GPIO_F	ALT0	GPIO1	GPIO[5]	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—CFG (100 KΩ PU) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT2	LCDC	LCDC_LD[17]	
	ALT3	EPIT1	DO_EPITO	
	ALT4	AUDMUX	AUD7_TXC	
	ALT6	UART4	TXD_MUX	
	ALT7	ECT	CTI_TRIG_OUT0_6	
EXT_ARMCLK	ALT0	CCM	EXT_ARMCLK	Hysteresis Enable—Disabled Drive Strength—Nominal Pull/Keep Enable—Disabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Keep Slew Rate—FAST
	ALT5	GPIO3	GPIO[15]	
UPLL_BYPCLK	ALT0	CCM	UPLL_BYPCLK	
	ALT5	GPIO3	GPIO[16]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
VSTBY_REQ	ALT0	CCM	VSTBY_REQ	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Disabled) Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT4	AUDMUX	AUD7_TXFS	
	ALT5	GPIO3	GPIO[17]	
	ALT6	UART4	RTS	
VSTBY_ACK	ALT0	CCM	HRESET_B	Hysteresis Enable—Enabled Drive Strength—CFG (Nominal) Pull/Keep Enable—CFG (Enabled) Pull Up/Down Configuration—CFG (100 KΩ PD) Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—SLOW
	ALT1	CCM	VSTBY_ACK	
	ALT2	CSPI1	SS3	
	ALT3	EPIT1	DO_EPITO	
	ALT5	GPIO3	GPIO[18]	
POWER_FAIL	ALT0	CCM	POWER_FAIL_INT	
	ALT4	AUDMUX	AUD7_RXD	
	ALT5	GPIO3	GPIO[19]	
	ALT6	UART4	CTS	
RESET_B	No Muxing (ALT0)	CCM	RESET_B	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
POR_B		CCM	POR_B	
CLKO	ALT0	CCM	CLKO	Hysteresis Enable—NA Drive Strength—CFG (Max) Pull/Keep Enable—Disabled Pull Up/Down Configuration—100 KΩ PU Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—FAST
	ALT5	GPIO2	GPIO[21]	
BOOT_MODE0	ALT0	CCM	BOOT_MODE[0]	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PD Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
	ALT5	GPIO4	GPIO[30]	
BOOT_MODE1	ALT0	CCM	BOOT_MODE[1]	
	ALT5	GPIO4	GPIO[31]	

Table 4-18. i.MX25 Detailed Pin Muxing (continued)

Pin Name	Mode	Instance	Port	Pad Settings
CLK_SEL	No Muxing (ALT0)	CCM	CLK_SEL	Hysteresis Enable—Enabled Drive Strength—Nominal Pull/Keep Enable—Enabled Pull Up/Down Configuration—100 KΩ PD Open Drain Enable—Disabled Drive Voltage Select—CFG (3.3 V) Pull/Keep Select—Pull Slew Rate—NA
TEST_MODE		TCU	TEST_MODE	
OSC24M_EXTAL	No Muxing (ALT0)	OSC24M	EXTAL24M	—
OSC24M_XTAL		OSC24M	XTAL24M	—
OSC32K_EXTAL		DRYICE	EXT32K	—
OSC32K_XTAL		DRYICE	XTAL32K	—
TAMPER_A		DRYICE	TAMPER_A	—
TAMPER_B		DRYICE	TAMPER_B	—
MESH_C	No Muxing (ALT0)	DRYICE	MESH_C	—
MESH_D		DRYICE	MESH_D	—
OSC_BYP		DRYICE	OSC_BYP	—
XP		ADC	XP	—
XN		ADC	XN	—
YP		ADC	YP	—
YN		ADC	YN	—
WIPER		ADC	WIPER	—
INAUX0		ADC	INAUX0	—
INAUX1		ADC	INAUX1	—
INAUX2		ADC	INAUX2	—
REF		ADC	REF	—

Chapter 5
Clock Distribution

This chapter covers clocking distribution for the i.MX25 device.

Figure 5-1 shows a simplified block diagram of the clock distribution on the i.MX25 device.

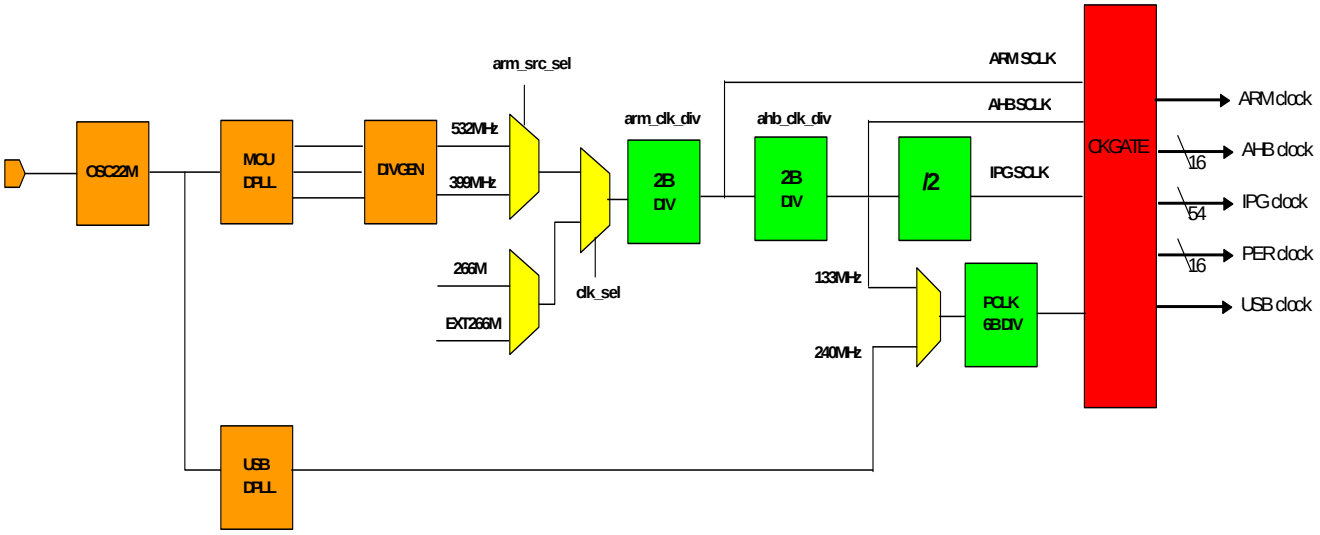


Figure 5-1. i.MX25 Clock Distribution

The clock control module (CCM) generates the clocks for all the modules on i.MX25. The CCM has the following features:

- Low-power mode (LPM) entry/exit control
- Clock distribution
- IP Bus accessible registers
- Reset control to the cores and peripherals
- Boot mode control logic

5.1 External Clock Sources

The OSC24M oscillator provides a frequency reference using a 24 MHz crystal with its corresponding integrated biasing resistor and loading capacitors. This oscillator is designed to supply the USBPHY which has very strict jitter requirements. The i.MX25 also uses this clock source as the primary PLL for the ARM platform.

The OSC32K oscillator provides a second frequency reference using a 32 kHz crystal with its corresponding integrated biasing resistor and loading capacitors. This oscillator is designed to supply the real time clock (RTC) in DryIce.

The i.MX25 uses two clocks as the reference clocks in the system:

- CKIL - Real time clock. This clock must be active at all times. The frequency must be 32.768KHz. CKIL must be present during reset.
- CKIH - High frequency input clock with a frequency between 13 MHz and 40 MHz. CKIH generally is the 24 MHz output from the OSC24M and must be present for the i.MX25 to come out of reset.

5.2 PLLs

There are two PLLs in the i.MX25 device as described in Table 5-1.

Table 5-1. i.MX25 PLLs

PLL Name	Reference Source Options	Default Output Frequency (MHz)	Comments	Type	Default State
Core PLL	CKIH	400	ARM platform clocks	FracN, Dithering	On
USB PLL	CKIH	300	USB clock	FracN, Dithering	On

5.3 Clock Gating

Two levels of clock gating are implemented in the i.MX25, as follows:

- Clock tree root, as implemented in the clock controller module (CCM)
- Clock tree leaf nodes, as implemented in the modules themselves

Clock tree roots are gated off by programming the CCM or by putting the domain into a low-power mode.

Clock tree branches are gated off as follows:

- Automatically, when the modules on a branch request its clock be disabled
- By programming the CCM, or putting the domain into a low-power mode

There is an override bit in the CCM modules for each clock branch, which forces the clock tree and branch to remain on.

5.4 Core PLL Clock Generation

The core PLL (also denoted MPLL) creates the clocks for the ARM platform and the SDMA platform. The high frequency bus (AHB) clock and the low frequency bus (IP) clock of the ARM platform are synchronous to the AP core clock. The clock frequency limitations and clock ratio restrictions between the core, AHB, and IP clocks frequencies are:

- Maximum AP core clock frequency is 399 MHz
- Maximum AHB bus frequency is 133 MHz
- Maximum IP bus frequency is 66.5 MHz

- Clock ratios must be integers between Core:AHB, Core:IP, AHB:IP
- AHB:IP clock ratio is fixed at 2:1
- Core:AHB and Core:IP clock ratio can be 3:1, 2:1, or 1:1

Some examples of allowable clock frequencies are shown in Table 5-2.

Table 5-2. Clock Frequency Examples

Example	Core (MHz)	AHB (MHz)	IP (MHz)
1	399	133	66.5
2	266	133	66.5
3	133	133	66.5
4	133	66	66.5
5	300	100	50

Chapter 6
Reset

This chapter describes the following:

- Reset types
- Reset sources
- Reset state machine
- Reset sequence

6.1 Reset Types

Table 6-1 summarizes the three reset types (Power-on reset (POR), cold and warm resets).

Table 6-1. Reset Types

Reset Type	Reset I/O Pin	Impact Modules
Power-on reset (POR)	POR_B	ARM core, OSC24M, DPLLs, fusebox, CCM, test logic and all other peripheral modules.
Cold reset	RESET_B	ARM core and all other peripheral modules.
Warm reset	N/A	ARM core and all other peripheral modules.

6.2 Reset Sources

Table 6-2 shows reset sources, qualification conditions, and resulting reset types. Qualified resets are qualified on the unsynchronized CKIL clock.

Table 6-2. Reset Sources and Qualification Conditions

Reset Source	Qualification Conditions	Resulting Reset Type
POR_B	Unqualified	Power-on reset (POR)
External low condition on RESET_B	Qualified for 4 CKIL clocks.	Cold reset
External low condition on internal reset from WDOG	Qualified for 1 CKIL clock.	Warm reset
Reset signal from the external JTAG connector	Unqualified	Warm reset
Software-initiated JTAG reset	Unqualified	Warm reset

6.3 Reset State Machine

The reset state machine requires the presence of the CKIL clock.The reset sequence uses the IIM fuse-read completion flag and EMI completion flag handshake signals to gate the progress of the state machine, to ensure that these sequences complete before progressing.

6.4 Reset Sequence

The reset module controls or distributes all of the system reset signals used by i.MX25. Figure 6-1 gives a simplified block diagram of the reset sequence. The signals and timing parameters shown in Figure 6-1 are described in Table 6-3 and Table 6-4, respectively.

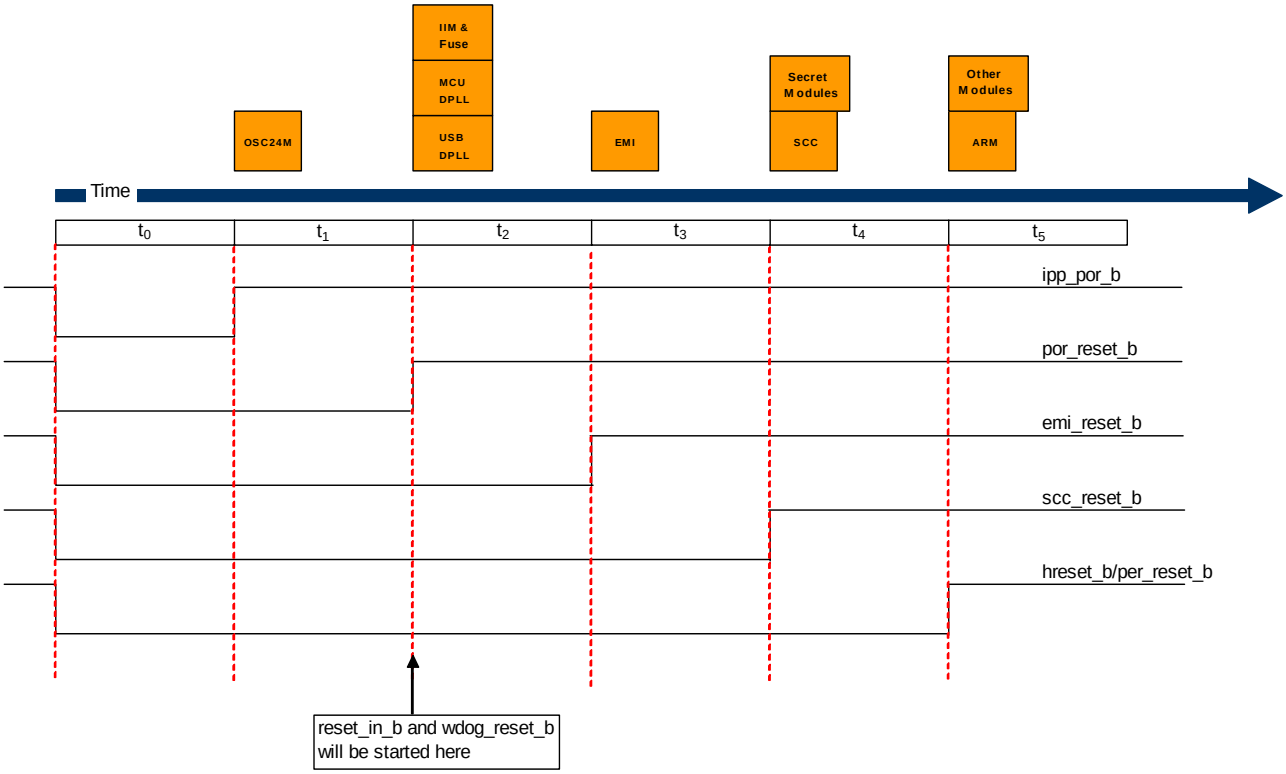


Figure 6-1. Reset Module Clock Diagram

Table 6-3. Reset Module Signals

Signal Name	Description
ipp_por_b	Resets OSC24M, which is input from external source
por_reset_b	Resets DPLLs, fusebox and IIM
emi_reset_b	Resets EMI module
scc_reset_b	Resets SCC and security modules
hreset_b	Resets ARM platform
per_reset	Resets peripheral modules

Table 6-4. Reset Module Timing Parameters

Timing Parameter	Description
t_0	Time for the power-up sequence of all the supplies. After t_0, OSC24M starts to work.
t_1	t_1 depends on boot mode: • If boot mode is PROD mode (0b01) then t_1 is 8 32-kHz cycles • If boot mode is not PROD mode, then t_1 is 256 32-kHz cycles After t_1, IIM, fusebox, and the DPLLs start to work
t_2	4 32-kHz cycles+ 1 HCLK cycle. After t_2, EMI starts to work.
t_3	• For NAND boot, t_3 is 32 32-kHz cycles + 1 HCLK cycle • Otherwise t_3 is 4 cycles 32-kHz + 1 HCLK cycle After t_3, the security modules and SCC start to work.
t_4	• 1 HCLK cycle for hreset_b • 1 ipg_clk cycle for per_reset_b After t_4, all other modules start to work
t_5	Out of reset

Chapter 7
System Boot

7.1 Overview

This chapter describes the system boot sequence of the i.MX25 application processor, and describes the various boot options.

The i.MX25 out-of-reset boot sequence makes use of the High-Assurance Boot (HAB) library to provide a secure boot environment. The HAB process utilizes a combination of hardware and software, including a public key infrastructure (PKI) protocol to protect the system from executing unauthorized images or programs. In order for the HAB to allow user code to run, the code must be signed by the private key holder which matches with the public key on i.MX25. The HAB library in the i.MX25 boot ROM also provides a number of API functions, which allow the user to authenticate any defined region and signature at run-time.

The i.MX25 also supports a HAB-bypass mode or direct external boot, in which the processor boots directly from external memory (as traditional microprocessors do).

The boot ROM also provides a mechanism to download and flash new code using a serial connection. Typically, a downloader application is downloaded to RAM, which facilitates the flash programming. The download is performed over either the USB or UART connection.

The boot capabilities differ between i.MX25 packages, depending on the HAB-type security configuration. Full flexibility is supported in the development (or engineering) configuration, while significant limitations are imposed on the production (or secure) configuration.

7.2 Boot Sources

The i.MX25 boot process utilizes the following memory/device sources:

- NOR Flash memory through WEIM interface, chip select 0 (CS0):
  - 16-bit slow asynchronous mode
  - Supports muxed address/data modes
  - 24-bit address is available for boot
- OneNAND memory through WEIM interface
- SLC (Binary) and MLC NAND Flash memory, through the NFC interface:
  - 512-byte, 2-Kbyte and 4-Kbyte page sizes
  - 4/8 bit ECC.
- LBA NAND devices through NFC interface (configured as SLC)
- SD/MMC cards through both eSDHC interfaces:
  - Support for high capacity SD, eSD (Embedded SD, versions 2.0 and 2.1 Draft Rev. 0.3) and MMC/eMMC (version 4.3 (MoviNAND), JEDSD84-A43) cards.
- EEPROM/serial Flash devices boot through any SPI interface
- EEPROM boot through any I^2C interface

- ‘Streamed in’ (serial downloader) boot through USB OTG/UART interfaces.

7.3 Boot Modes

Three boot modes are supported in the i.MX25. Table 7-1 describes these boot modes in a summary.

Table 7-1. Boot Mode Summary

BOOT_MODE[1:0]	Boot Type[1]	Descriptions
0 0	Internal Boot	When power-on reset, i.MX25 executes the boot ROM code to load the boot image from different boot sources. See Section 7.3.3 for more details.
0 1	Reserved	Reserved
1 0	External (Direct) Boot	Direct boot from WEIM interface, independent of boot ROM code. See Section 7.3.4 for more details.
1 1	USB/UART Serial Boot	Load and execute code using serial devices: • USB OTG (Full-Speed, using integrated PHY or external PHY) • UART See Section 7.3.5 for more details.

1 Boot type is determined by the values of boot mode contacts BOOT_MODE[1:0] sampled during the out-of-reset sequence and stored in the Clock Controller Module (CCM) Status Register (RCSR)

Other boot configuration settings are obtained from the programmable eFUSEs or by sampling the contacts during the out-of-reset sequence. In the i.MX25, a fuse, GPIO_BT_SEL, is provided for flexibility between these two configuration settings.

- If GPIO_BT_SEL is blown, all boot options are configured by eFUSEs as detailed in Table 7-2 below. Boot ROM software may read the values from the RCSR, or from the e-fuses, using the IIM module. It is the recommended configuration for deployed products.
- If GPIO_BT_SEL is left unblown, the various boot options are determined by sampling dedicated contacts at out-of-reset. Every e-fuse option is associated with a dedicated pin(s), such that same functionality is available for both boot options. For this case, regardless of fuse values, Boot ROM code must read the options' values from RCSR register of the CCM module. Table 7-3 lists boot option contacts.

7.3.1 Boot Configuration

This section lists the various boot modes and boot configurations as defined by the eFUSE values or contacts sampled at out-of-reset.

7.3.1.1 eFUSE Configuration

eFUSE is described in detail in Chapter 30, “IC Identification Module (IIM).” In this section, only the part related to boot is specifically listed here.

Table 7-2 shows the eFUSE settings used by the boot ROM code in the boot process.

Table 7-2. Fuse Description

eFUSE	Definition	Settings: Intact reads as 0 Blown reads as 1

BT_SRC[1:0]	Chooses the specific device for booting.	If BT_MEM_CTL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=00 (SD/MMC/MoviNAND) then
		00 eSDHC-1
		01 eSDHC-2
		10 Reserved
		11 Reserved
		If BT_MEM_CTL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=10 (Serial ROM using I^2C) then
		00 I2C1
		01 I2C2
		10 I2C3
		11 Reserved
		If BT_MEM_CTL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=11 (Serial ROM using SPI) then
		00 CSPI1
		01 CSPI2
		10 CSPI3
		11 Reserved
		Otherwise Reserved

BT_UART_SRC[2:0]	Choosing the specific UART controller for booting.	000 UART1 001 UART2 010 UART3 011 UART4 100 UART5 Otherwise Reserved
BT_MLC_SEL	SLC/MLC NAND device.	0 SLC NAND device 1 MLC NAND device
BT_SPARE_SIZE[1]	Specifies the size of spare bytes for 4-Kbyte page size NAND Flash devices.	0 128 bytes spare (Samsung) 1 218 bytes spare (Micron, Toshiba)
BT_USB_SRC[1:0]	USB boot source selection	00 USB OTG Internal UTMI PHY 01 USB OTG External ULPI PHY 10 Reserved 11 Reserved
BT_RES1, BT_RES2, BT_RES3, BT_RES4	Reserved for boot options Has a corresponded GPIO contact, including a place in SRC SBMR register	— —
BT_PAGE_SIZE[1:0]	NAND Flash page size. This field is used in conjunction with the BT_MEM_CTL[1:0] setting	If BT_MEM_CTL = NAND Flash then 00 512 bytes 01 2 Kbytes 10 4 Kbytes 11 Reserved

Table 7-2. Fuse Description (continued)

eFUSE	Definition	Settings: Intact reads as 0 Blown reads as 1
BT_EEPROM_CFG	Selects whether EEPROM device is used for load of device configuration data (DCD), prior to boot from other devices (not applicable when using EEPROM as boot device)	0 Use EEPROM DCD 1 Don't use EEPROM DCD
GPIO_BT_SEL	GPIO boot select. Determines, whether certain boot fuse values are controlled from GPIO contacts or fuses.	0 Boot mode configuration is set by contacts. 1 Boot mode configuration is set by fuses.
HAB_TYPE[2:0]	High assurance boot security type	001 Engineering—allows any code to be flashed and executed, even if it has no valid signature (default) 100 Security disabled Others Production (Security On)
BT_MEM_TYPE[1:0]	Boot memory type. Interpreted by boot ROM software according to BT_MEM_CTL setting.	If BT_MEM_CTL = WEIM then 00 NOR 01 Reserved 10 OneNAND 11 Reserved If BT_MEM_CTL = NAND Flash 00 3 address cycles 01 4 address cycles 10 5 address cycles 11 Reserved If BT_MEM_CTL = Expansion Card Device 00 SD/MMC/MoviNAND 01 Reserved 10 Serial ROM using I^2C 11 Serial ROM using SPI
BT_BUS_WIDTH[1:0]	Selects boot device bus mode.	BT_MEM_CTL[1:0] = NAND Flash 00 8-bit bus 01 16-bit bus 10 Reserved 11 Reserved BT_MEM_CTL[1:0] = WEIM (NOR) 00 Reserved 01 16-bit address/data unmultiplexed interface 10 Reserved 11 Reserved BT_MEM_CTL[1:0] = Expansion Device (SPI) 00 2-Address word SPI device (16-bit) 01 3-Address word SPI device (24-bit) 10 Reserved 11 Reserved

Table 7-2. Fuse Description (continued)

eFUSE	Definition	Settings: Intact reads as 0 Blown reads as 1
BT_MEM_CTL[1:0]	Boot memory control type (memory device)	00 WEIM 01 NAND Flash 10 Reserved 11 Expansion Device (SD/MMC/MoviNAND, support high storage, EEPROMs. See BT_MEM_TYPE[1:0] settings for details).
DIR_BT_DIS	Direct external memory boot disable	0 Direct boot from external memory is allowed 1 Direct boot from external memory is not allowed
SRK_HASH[255:0]	Super root key hash	Varies—used by high-assurance boot (HAB)
HAB_CUS[7:0]	HAB customer code	Varies—used by HAB
BT_LPB[1:0]	Options for low power boot mode (for more details see Section 7.6	00 Generic PMIC and one GPIO input (low battery detection) 01 Generic PMIC and two GPIO inputs (low battery and charger detection) 10 Reserved 11 AP power management IC.
BT_LPB_FREQ[2:0]	LPB ARM core frequency	000 133 MHz (Default) 001 CKIH 010 55.33 MHz 011 66 MHz 100 83 MHz 101 166 MHz 110 266 MHz 111 normal boot frequency
DIE-X-CORDINATE[7:0] DIE-Y-CORDINATE[7:0] WAFER_NO[4:0] LOT_NO_ENC[2][42:0]	Manufacturing Information. Used as 64-bit Unique part ID and Secure JTAG Challenge Value. Burnt by Freescale during the tester stage. 43-bit LOT_NO_ENC field encodes LOT ID STD II, including FAB ID inside.	Varies—used by HAB

1 512-byte page devices have 16 bytes spare area size, 2-Kbyte page devices have 64 bytes spare area size.

2 Lot Number Encoded field is 43-bit value, contains encoded ‘STD II’ lot ID.

7.3.1.2 Boot Fuses and Associated Contacts

Table 7-3 lists fuses and associated contacts used for boot. The input contacts listed are latched during POR; however, the value is stored once POR is released. It overrides fuse values when the GPIO_BT_SEL fuse is unblown. Also listed are boot mode contacts, and the functionality of other boot-related fuses and contacts.

Table 7-3. Fuses and Associated Contacts Used for Boot

Contact	Direction at Boot	eFUSE Name	Details
BOOT_MODE[1]	Input	N/A	Boot Mode selection (See Table 7-1)
BOOT_MODE[0]			
VSYNC	Input	BT_UART_SRC[2]	Boot Options, Contact value overrides fuse settings for GPIO_BT_SEL = 0
HSYNC	Input	BT_UART_SRC[1]	
LD15	Input	BT_UART_SRC[0]	
LD14	Input	BT_EEPROM_CFG	
LD[13:12]	Input	BT_SRC[1:0]	
LD11	Input	BT_SPARE_SIZE	
LD10	Input	BT_MLC_SEL	
LD[9:8]	Input	BT_USB_SRC[1:0]	
LD[7:6]	Input	BT_BUS_WIDTH[1:0]	
LD[5:4]	Input	BT_PAGE_SIZE[1:0]	
LD[3:2]	Input	BT_MEM_TYPE[1:0]	
LD[1:0]	Input	BT_MEM_CTL[1:0]	
PWM	Input	BT_LPB_FREQ[2]	
OE_ACD	Input	BT_LPB_FREQ[1]	
LSCLK	Input	BT_LPB_FREQ[0]	
CSI_PIXCLK	Input	BT_RES4	
CSI_HSYNC	Input	BT_RES3	
CSI_VSYNC	Input	BT_RES2	
CSI_MCLK	Input	BT_RES1	
VSTBY_ACK	Output	N/A	Toggle on this contacts indicates finish of internal system reset. After reset, this contact can be used for other purposes.

7.3.2 Boot Flow Diagram

Figure 7-1 shows the i.MX25 boot sequence flow. Color coding (see legend) indicates the controlling sources (hardware only, boot contacts, or eFUSEs).

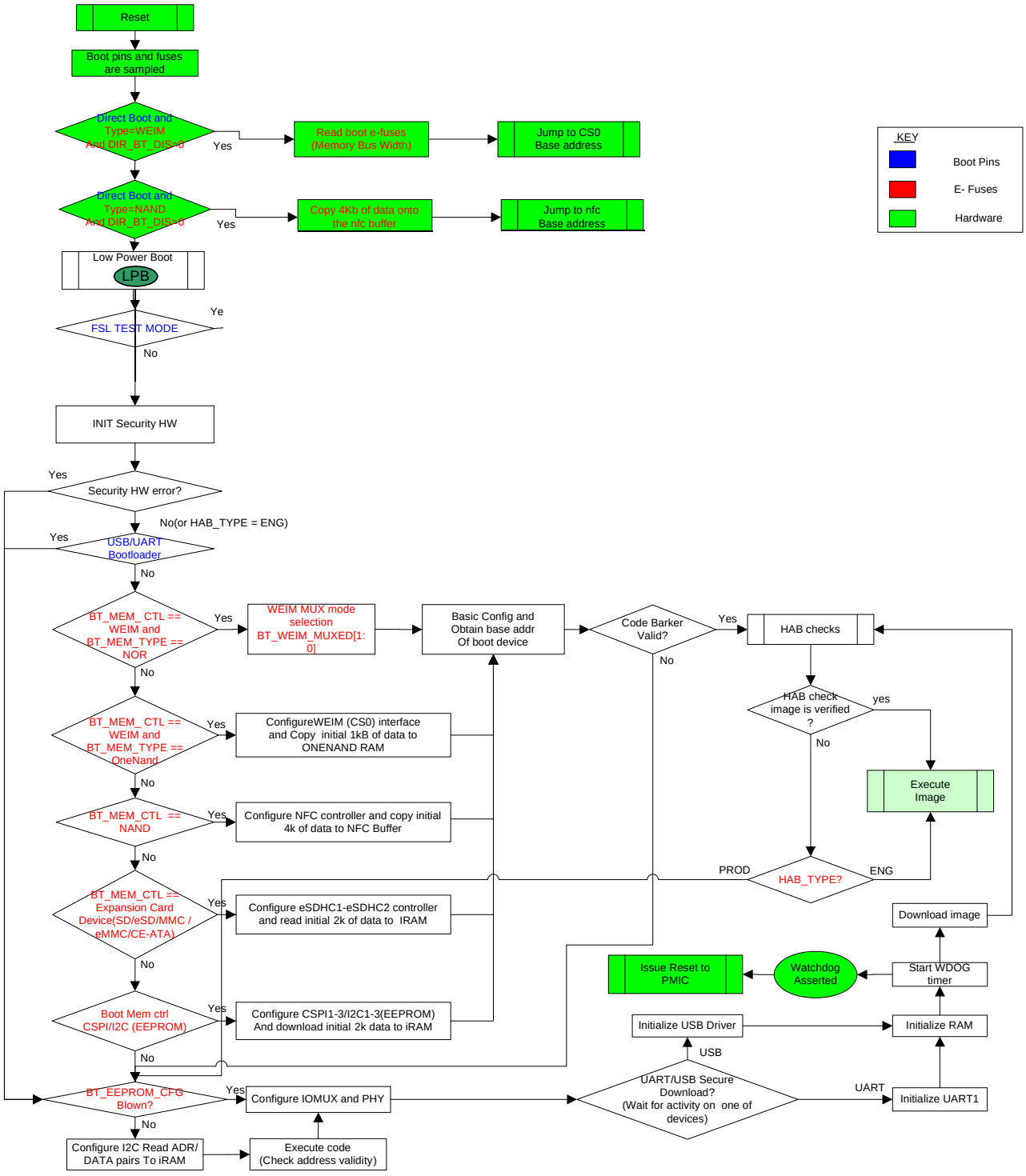


Figure 7-1. i.MX25 Boot Flow

7.3.3 Internal Boot Mode (BOOT_MODE[1:0]=00)

Internal boot is selected by driving the value ‘00’ on the BOOT_MODE[1:0] contacts, at device power-up. In this mode, the i.MX25 boots from internal boot ROM. The boot ROM code performs hardware initialization, application image validation using the high-assurance boot (HAB) library, and then jumps to an address derived from the application image. If any error occurs during internal boot the boot code jumps to the UART/USB secure download.

Internal boot mode is the only mode in which a secure boot of the i.MX25 is possible.

7.3.3.1 Security Settings

External boot modes are considered non-secure by definition: the software in Flash is executed regardless of its authenticity, and the SCC is automatically put into Non-Secure state so that it cannot be used to decrypt information with the device-unique secret key.

Internal boot modes can have one of two security levels:

- **Production:** This level is intended for use with shipped products. All HAB functions are executed and security hardware is initialized (the SCC enters Secure state), device configuration data (DCD) is processed if present, and software in Flash or downloaded to RAM is authenticated by HAB prior to its execution. (See Section 7.3.3.10, “Device Configuration Data (DCD),” for more information.) The first error detected is logged, then the boot flow is aborted with control passing to the download mode. With this level, execution does not leave the internal ROM unless the target executable image has been authenticated.
- **Engineering:** This level is intended for use during the development phases of a product or if secure boot is not required, but an internal boot to the ROM is still needed. All HAB functions are executed as for a production device, security hardware is initialized (except the SCC is left in Non-Secure state), DCD is processed if present, and software in Flash or downloaded to RAM is authenticated by HAB prior to its execution. First error detected is logged, but without any change to the boot flow. A Command Sequence File (CSF) must be present in this mode, even if it is invalid or the pointer in the application header is NULL. See Section 7.3.3.9, “Flash Header,” for more information. It is possible with this level to develop software without requiring that each build be signed for HAB authentication, since the device will boot even if the code signatures are missing.

7.3.3.2 Basic Initialization

On reset, the ARM has access to all shared peripherals.

By default, most of the module clocks are gated off in i.MX25. Table 7-4 shows the module clocks which are enabled by default. All other modules’ clocks are enabled by ROM as described in Section 7.3.3.2.1, “Normal Mode” and Section 7.3.3.2.2, “Low-Power Boot Mode.”

Table 7-4. Module Clocks Enabled by Default

Module	Clock Source
BROM, EMI, FEC	AHB
NFC	PER
FEC, IIM, SCC, DryIce, and SPBA	IPG

7.3.3.2.1 Normal Mode

Table 7-5 shows the clock configurations specified by ROM for normal mode.

Table 7-5. Normal Mode Clock Configuration

Clock Output	Speed
MPLL	399 MHz
UPLL	240 MHz
ARM core	199.5 MHz
AHB	99.75 MHz
IPG	49.875 MHz

7.3.3.2.2 Low-Power Boot Mode

In low-power boot (LPB) mode, MPLL is used as the source of the ARM core clock, serial bus clock (I2C, CSPI, eSDHC, UART and so on), IPG, perclk_root, AHB bus clocks. ROM configures the ARM core frequency value based on fuse BT_LPB_FREQ[2:0]. The maximal LPB ARM core frequency is customized and stored in fuses BT_LPB_FREQ[2:0].

Table 7-6. ARM Core Frequency Encoding for BT_LPB_FREQ[2:0]

Code	Value
0	133 MHz (Default)
1	CKIH
2	55.33 MHz
3	66 MHz
4	83 MHz
5	166 MHz
6	266 MHz
7	Normal boot frequency

Table 7-7. Module Frequency List for LPB Frequencies

BT_LPB_FREQ[2:0]	0	1	2	3	4	5	6	7
PLL1 frequency	532	—	166	266	166	166	266	399
ARM frequency	133	24	55.33	66	83	166	266	199.5
ahb_clk_root	66.5	12	27.665	33	41.5	83	133	99.75
ipg_clk_root	33.25	6	13.8325	16.5	20.75	41.5	66.5	49.875
emi_slow_clk_root	66.5	12	27.665	33	41.5	83	133	99.75
nfc_clk_root	16.625	6	13.8325	16.5	10.375	13.8333	16.625	20
perclk_root	33.25	6	13.8325	16.5	20.75	41.5	66.5	49.875
esdhc_clk_root	16.625	3	6.91625	8.25	10.375	13.8333	16.625	49.875
cspi_clk_root	33.25	6	13.8325	16.5	20.75	41.5	66.5	49.875
i2c_clk_root	33.25	6	13.8325	16.5	20.75	41.5	66.5	49.875

7.3.3.3 External Device Selection

The i.MX25 supports the following devices for internal boot mode:

- NOR Flash with WEIM Interface, located on CS0, bus width of 16 bits.
- OneNAND.
- MLC NAND and SLC NAND Flash with NFC interface. Page sizes of 512 bytes, 2KB or 4KB, bus width of 8-bit or 16-bit.
- SD/MMC/eSD/eMMC using all eSDHC interface, supporting all types of cards.
- eSD FAST BOOT and eMMC Boot Mode (FAST BOOT) are supported using all the eSDHC ports.
- EEPROM boot using SPI and I^2C.
- Serial Flash using SPI.

The selection of external flash device type is determined by BT_MEM_CTL[1:0] and BT_MEM_TYPE[1:0] eFUSEs. See Table 7-2 for more details.

7.3.3.4 NAND Flash Support

Several MLC/SLC NAND Flash devices from different vendors are supported by the boot ROM. The error correction and control (ECC) module is used to detect errors. Both 8-bit ECC and 4-bit ECC are supported.

Table 7-8 shows the parameters used to configure the external NAND Flash. In particular, NAND Flash boot requires that BT_MEM_CTL be set to 0b01. These parameters are either provided by eFUSEs or sampled on input contacts during booting.

Table 7-8. Parameter Settings for External NAND Flash

Parameter	Definition	Settings
BT_MLC_SEL	SLC/MLC NAND device.	0 SLC NAND device 1 MLC NAND device
BT_SPARE_SIZE[1]	Specifies the size of spare bytes for 4-Kbyte page size NAND Flash devices.	0 128 bytes spare (Samsung) 1 218 bytes spare (Micron, Toshiba)
BT_PAGE_SIZE[1:0]	NAND Flash page size. This field is used with conjunction with the BT_MEM_CTL[1:0] setting	00 512 bytes 01 2 Kbytes 10 4 Kbytes 11 Reserved
BT_MEM_TYPE[1:0]	Boot memory type.	00 3 address cycles 01 4 address cycles 10 5 address cycles 11 Reserved
BT_BUS_WIDTH[1:0]	Selects boot device bus mode	00 8 bit bus 01 16 bit bus 10 Reserved 11 Reserved
BT_MEM_CTL[1:0]	Boot memory control type (memory device)	01 NAND Flash

1 512-byte page devices have 16 bytes spare area size, 2-Kbyte page devices have 64 bytes spare area size.

Since MLC NAND Flash devices do not guarantee error-free boot blocks, the i.MX25 boot code supports boot redundancy to provide for the case when an unrecoverable error is detected within the first 4 Kbytes. To make use of this feature, the user must duplicate the first 4 Kbytes of boot code (contained in NAND Flash Block-0) to NAND Flash Block-1 in order to serve as a second-copy option.

The boot ROM code makes use of the duplicate boot code according to the following procedure:

- On device power-on, the boot ROM copies the first 4 Kbytes of boot code from the NAND Flash to the NFC buffer.
- ECC checks the first 4 Kbytes of boot data from NAND Flash Block-0
  - If no ECC errors are detected, the downloaded 4 Kbytes image as well as rest of the image is copied to application destination pointer location (specified in the HAB header) and secure boot is performed. The application image length is specified in the image header so the boot ROM reads this to know how much image data to download.
  - If ECC Error is detected in first 4 Kbytes of boot data from Block-0, the boot ROM code copies the duplicated 4k boot data from the NAND flash block-1.
  - If an error is detected in the subsequent boot block, boot ROM code logs an error and jumps to the USB/UART bootloader. The logged error can be queried using the serial protocol.
  - If there is no error in the data copy, then the boot ROM performs secure internal boot.

7.3.3.5 OneNAND Flash Boot Operation

OneNAND Flash devices are a 16-bit interface. The OneNAND Flash driver in boot ROM obtains the device page size by issuing a software command and collecting the response from device.

At system power-up, OneNAND automatically copies the first 1 Kbyte of data from the Flash array (sector 0 and sector 1, page 0, block 0) to its internal RAM. This 1 Kbyte area in OneNAND's internal RAM is referred to in the following as "boot RAM". The boot ROM copies the boot RAM contents to a destination address determined by the app_dest_ptr entry of the application header (Section 7.3.3.9, "Flash Header"), and decrements the length of the image to be read from OneNAND by 1 Kbyte. The boot RAM is memory-mapped, and the copying operation is simple memory copying operation. The length of the image to be read from the OneNAND device is specified in the Flash header structure shown in Table 7-10. Any failure in the data load operation from OneNAND Flash forces the boot ROM to switch to USB OTG/UART serial download.

At system power-up, the voltage detector in the device detects the rising edge of Vcc and activates the internal power-on reset (POR) signal. This in turn triggers boot code loading, so the OneNAND's boot loader copies the first 1 Kbyte of the Flash array to boot RAM. The boot code copy operation starts 400 µs after POR activation, and takes about 70 µs.The INT bit of Interrupt status register is then set on the condition of 'boot code copy done' and RP rising edge.

In the OneNAND boot operation, the boot ROM instructs an on-chip General Purpose Timer (GPT) to introduce a delay of around 500 µs, then waits for INT bit of the interrupt status register to be set. After the INT bit is set, the boot ROM then proceeds with OneNAND initialization.

7.3.3.6 Serial ROM Support using SPI and I^2C

The i.MX25 supports boot from serial memory devices such as EEPROM, and Serial Flash using the SPI (Chip Select #1) and I^2C interfaces.

The Boot ROM code determines the device type from the following parameters which are either provided by eFUSEs or sampled on I/O contacts during boot:

- BT_MEM_TYPE[1:0]—SPI/I^2C select.
- For SPI, "BT_BUS_WIDTH" selects between 2/3-address word devices. BT_SRC[1:0] selects between CSPI1, CSPI2 and CSPI3.
- For I^2C, BT_SRC[1:0] selects between I2C-1, I2C-2, or I2C-3.

See Table 7-3 for detailed settings.

7.3.3.6.1 CSPI Boot

Any of the CSPI modules can be used as boot device for booting from a serial device. The CSPI interface is configured in Master mode. The serial device is connected to CSPI interface as slave.

CSPI boot proceeds as follows:

- Boot ROM copies 2 Kbytes of data from the serial device to internal RAM.
- DCD verification is attempted

— If DCD verification is successful, the ROM code copies the initial 2 Kbytes of data as well as the rest of image directly to the application destination, extracted from application image.

— If DCD verification fails, ROM code execution logs and error and jumps to USB/UART serial downloader.

The CSPI can read data from the EEPROM using either 2- or 3-byte addressing, with a burst length of 32 bytes.

NOTE

The Serial ROM is required to reside on Chip Select #1 of the CSPI module. Using SPI as boot device, the i.MX25 supports boot from both Serial EEPROMS, and Serial Flash devices.

7.3.3.6.2 I^2C Boot

Any of the I^2C modules can be used as boot device using the I^2C interface for serial EEPROM boot. The I^2C interface is configured to operate at speeds up to 389.6 Kbps.

The I^2C boot flow proceeds as follows:

1. The boot ROM code reads from fuses BT_SRC[1:0] and BT_MEM_TYPE to determine that the I^2C interface is to be used for boot, and to specify which I^2C module is used.
2. The boot ROM then copies 2 Kbytes of data from the EEPROM device to internal RAM.
3. DCD verification is performed:
   — If DCD verification is successful, the ROM code copies the initial 2 Kbytes of data as well as the rest of image directly to application destination extracted from the application image.
   — If DCD verification fails, ROM code execution logs the error and jumps to the USB/UART serial downloader.

Table 7-9 shows the device select codes used by the i.MX25 to boot from EEPROM.

Table 7-9. EEPROM using I^2C Device Select Code

	Device Type Identifier				Chip Enable Address[1]			$\overline{RW}$
Bits	7	6	5	4	3	2	1	0
Device Select Code	1	0	1	0	0	0	0	$\overline{RW}$

[1] These address bits, should be configured at the memory device, to match this ‘000’ value.

7.3.3.7 NOR Flash (using WEIM) Support

The i.MX25 device supports booting from the WEIM NOR Flash interface. The device configures the NOR boot mode by hardware, due to existence of a direct NOR Flash boot option. The boot ROM code is not required to change these options. The IOMUX settings at POR support NOR boot by default. The NOR Flash interface works in asynchronous mode, and supports 16-bit muxed Address/Data and non-muxed schemes, based on the BT_BUS_WIDTH[1:0] fuse settings.

24-bit address is available for boot. When booting, if there is no device contention on EIM[27:24], the downloaded NOR driver can configure IOMUX to switch on 28-bit NOR address mode.

7.3.3.8 Expansion Card Device Support (SD/MMC)

The i.MX25 device supports booting from MMC and SD Card-compatible devices. Boot is also supported for mass storage cards (high-capacity cards, SD and MMC version 4.3), SanDisk's iNAND cards, and new eSD (Embedded SD, version 2.0 and 2.1 Draft Rev 0.2, June 20th 2007)–type cards.

SD/MMC Boot can be performed using either eSDHC-1 or eSDHC-2, based on the BT_SRC[1:0] fuse value or its associated input contact value at out-of-reset. See Table 7-2 for detailed settings. Low-voltage devices (1.8 V) can also be supported using the application code, by control of the power supply level. This task is not handled by the boot ROM code.

MMC/SD/eSD can be connected to either eSDHC-1 or eSDHC-2, and booting is done by copying first 2 Kbytes of data from MMC/SD/eSD device to internal RAM. After verifying the Barker value (0xB1) from the boot image, the ROM code performs a DCD check. If successful, the ROM code then extracts the destination pointer and the length of image to be copied to the RAM device where code execution occurs.

7.3.3.8.1 Secure Digital (SD) and Embedded Secure Digital (eSD) Cards)

The SD or eSD card frequency should be set to 311.71kHz for the negotiation phase. During negotiation phase SD and eSD card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings and if it fails it checks with low voltage settings. Capacity of card is also checked. Boot code supports high capacity and low capacity SD and eSD cards. After voltage validation, card initialization is done. During card initialization boot code tries to set boot partition for both SD and eSD device. If it fails, boot code assumes card as normal SD card otherwise as eSD card. After initialization phase is over, boot code switches to a higher frequency, 16.625MHz. ROM also support FAST_BOOT mode booting from eSD card. This mode can be selected by BT_SPARE_SIZE fuse.

The boot ROM uses 1-bit accesses to card. Application code can switch to 4- or 8-bit mode.

Figure 7-2 through Figure 7-4 show the SD/eSD boot flow.

7.3.3.8.2 MMC and eMMC

The MMC frequency is set to 311.71 kHz for the negotiation phase. During negotiation phase MMC, card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings, if it fails it checks with Dual voltage settings and capacity of card is checked. Boot code supports both high-capacity and low-capacity MMC cards. After voltage validation, card initialization is complete, and boot code can switch to a higher frequency, 16.625 MHz.

eMMC is also interfaced using eSDHC and follows same flow as done by MMC. Boot partition can be selected for eMMC cards, after the card initialization has been done. ROM reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in BOOT_PARTITION_ENABLE field or the user partition has been mentioned, ROM boots from the user partition. eMMC device support special "Boot mode", which can be initiated by issuing CMD0 with 0xFFFFFFFA. If BOOT ACK is enabled, the eMMC device sends the BOOT ACK using DATA0 and ROM can read the BOOT ACK [S010E] to identify the eMMC device.

eMMC device with “Boot mode” feature is supported using eSDHC-1 and eSDHC-2 with BOOT ACK enabled. ROM waits for 50ms to get the BOOT ACK and if BOOT ACK is received by ROM, then only eMMC is booted in “Boot mode”, otherwise eMMC boots as a normal MMC card from the selected boot partition. This boot mode can be selected by BT_MLC_SEL fuse. Only 1-bit bus width is used in ROM.

MoviNAND is also interfaced using eSDHC and follows the same flow as the MMC.

Figure 7-2 through Figure 7-4 show the boot flow for MMC and eMMC.

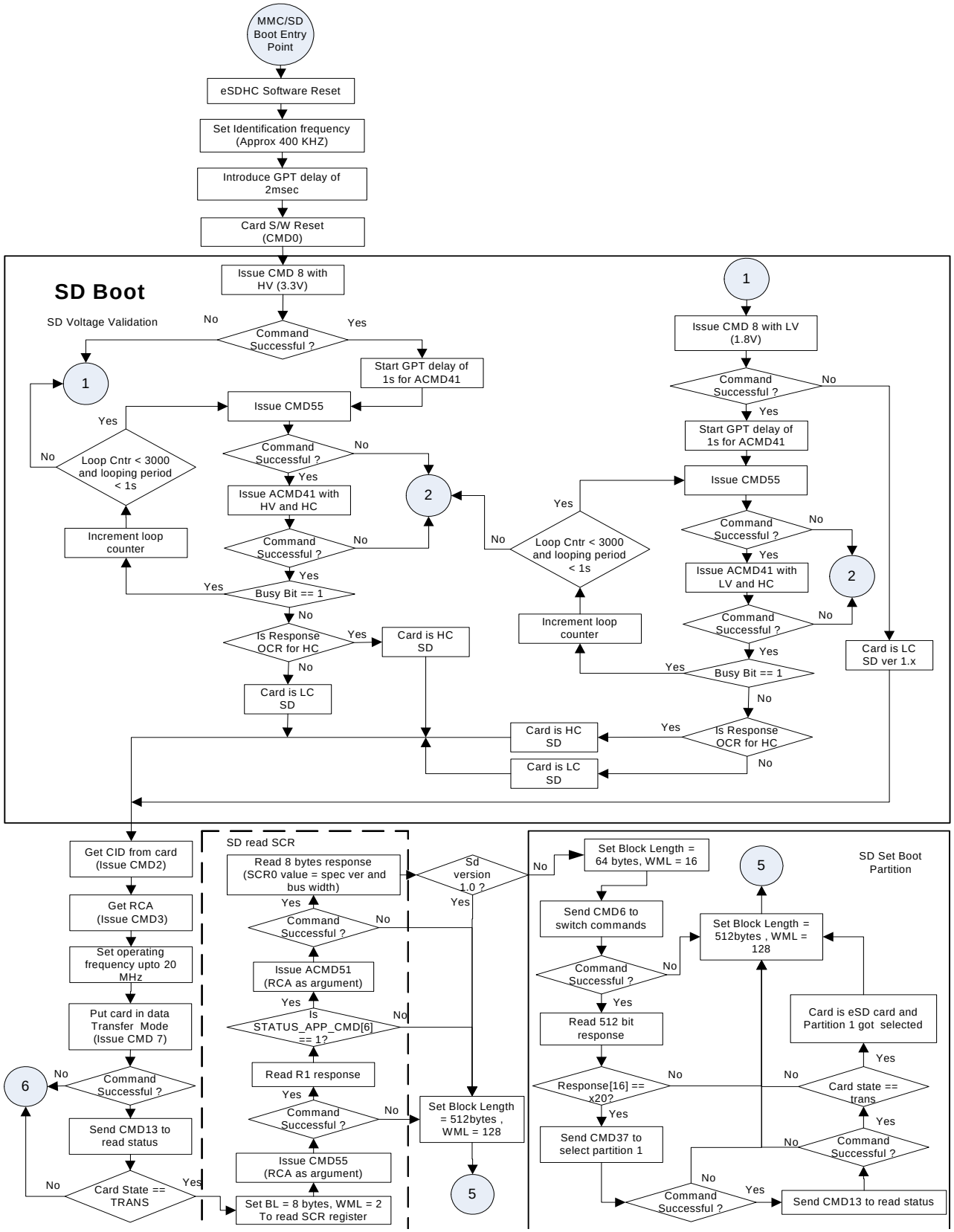
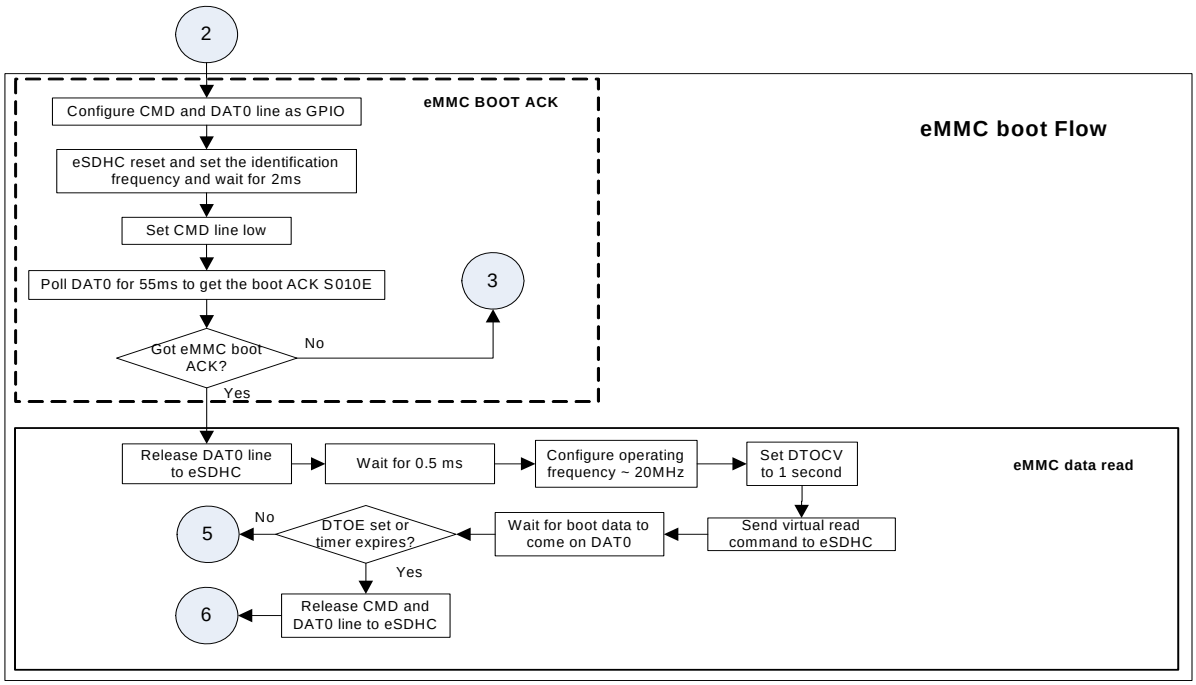


Figure 7-2. SD and MMC Boot Flow



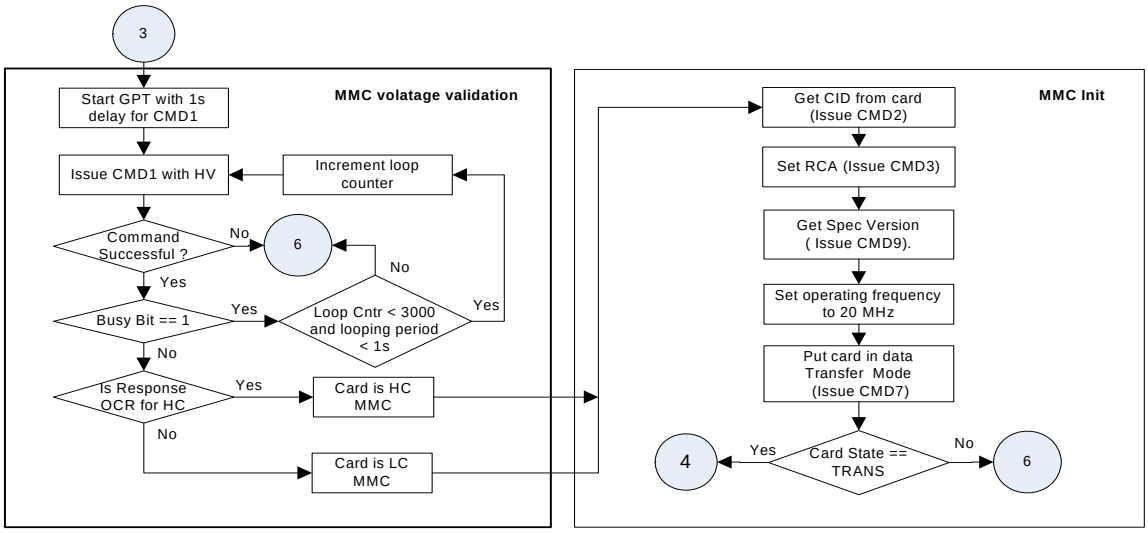


Figure 7-3. SD and MMC Boot Flow (continued)

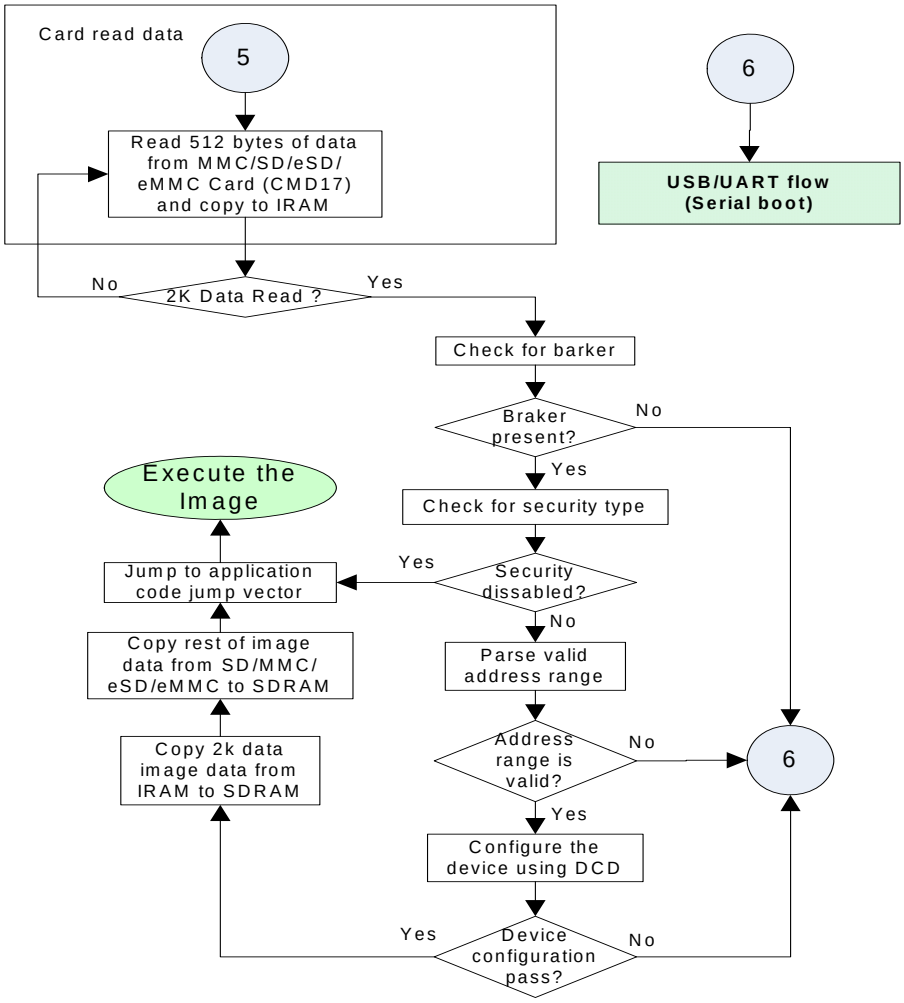


Figure 7-4. SD and MMC Boot Flow (continued)

7.3.3.9 Flash Header

The Flash header is a data structure that the boot code reads from Flash which provides information about the application. The Flash header must be located at a known fixed address depending on the type of external Flash device connected to i.MX25. The Flash header is only required when an internal boot mode is selected from BOOT_MODE[0:1]. The required offsets of the Flash header for each device type are described in Table 7-10.

Table 7-10. Flash Header Offset

Flash Type	Offset from Base Address
NOR	4 Kbyte = 0x1000 bytes
NAND	1 Kbyte = 0x400 bytes
OneNAND	256 bytes = 0x100 bytes

Table 7-10. Flash Header Offset (continued)

Flash Type	Offset from Base Address
SD/eSD/MMC/eMMC	1 kbyte = 0x400 bytes
I2C/CSPI EEPROM	1 kbyte = 0x400 bytes

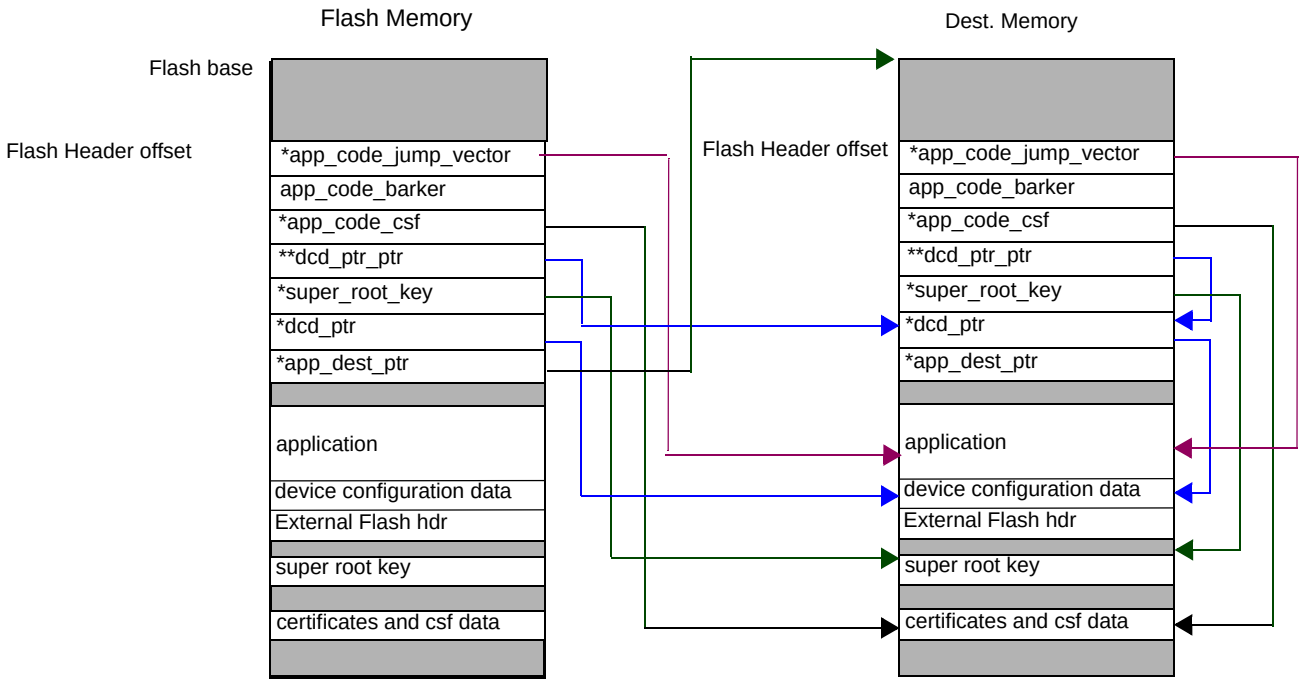


Figure 7-5. Flash Header Example for Devices Other Than NOR Flash

The Flash header must conform to the structure defined below:

```
typedef struct
{
    UINT32              *app_code_jump_vector;
    UINT32              app_code_barker;
    UINT32              *app_code_csf;
    DCD_T               **dcd_ptr_ptr;
    hab_rsa_public_key  *super_root_key;
    DCD_T               *dcd_ptr;
    UINT32              *app_dest_ptr;
} FLASH_HDR_T;

where:
UINT32 is a 32 bit unsigned integer
DCD_T is a structure that defines the device configuration table (see below for details)
hab_rsa_public_key is a structure defines the super root key
```

Table 7-11 shows the significance of the variables listed in the Flash header structure.

Table 7-11. Flash Header Structure Fields

<table>
<tr><th>Field Abbreviation</th><th>Significance</th></tr>
<tr><td>app_code_jump_vector pointer</td><td>Points to the address of the first instruction of the application</td></tr>
<tr><td>app_code_barker</td><td>A value that the ROM uses to determine that the Flash device has been programmed. For i.MX25 the app_code_barker value must be set to 0x000000B1.</td></tr>
<tr><td>app_code_csf pointer</td><td>Points to the certificate and command sequence file data. This data is used by the High Assurance Boot (HAB) library included in the ROM to verify that the application is authentic. Should be set to 0x0 when not used in a non-secure boot.</td></tr>
<tr><td>dcd_ptr_ptr double pointer</td><td>This pointer must be set to point to the dcd_ptr also contained in the Flash header structure.</td></tr>
<tr><td>super_root_key</td><td>Pointer to the super root key data. Should be set to 0x0 when not used in a non-secure boot. The super root key data should conform to the structure shown in the following code excerpt.
<pre>typedef struct
{
      UINT8   rsa_exponent[MAX_EXP_SIZE];   /* RSA public exponent */
      UINT8   *rsa_modulus;   /* RSA modulus pointer */
      UINT16 exponent_size;   /* Exponent size in bytes */
      UINT16 modulus_size;   /* Modulus size in bytes*/
      BOOLEAN init_flag;   /* Indicates if key initialized
      */
} hab_rsa_public_key;
where:
         UINT8 is an 8 bit unsigned integer
         UINT16 is a 16 bit unsigned integer
         BOOLEAN is an 8 bit flag indicating TRUE or FALSE</pre>
The variables in this code excerpt have significance as shown in the table:
<table>
<tr><th>Variable</th><th>Significance</th></tr>
<tr><td>rsa_exponent</td><td>Exponent of the RSA key. Maximum exponent size is 4.</td></tr>
<tr><td>rsa_modulus</td><td>Pointer to the RSA key modulus.</td></tr>
<tr><td>exponent size</td><td>Exponent size in bytes. Must be less than or equal to the maximum exponent size.</td></tr>
<tr><td>modulus size:</td><td>Modulus size in bytes. Must be greater than or equal to 128 and less than or equal to 256.</td></tr>
</table></td></tr>
<tr><td>dcd_ptr</td><td>Points to the device configuration data (DCD) table. See Section 7.3.3.10, “Device Configuration Data (DCD),” for further details on the DCD table.</td></tr>
<tr><td>app_dest_ptr</td><td>Used by the ROM for NAND/MMC/eMMC/SD/eSD/SPI(EEPROM)/OneNAND boot. During boot the ROM copies the application data from boot Flash memory to destination RAM. This pointer defines the location of destination memory where the ROM copies the application.</td></tr>
</table>

7.3.3.9.1 Flash Header for NAND Boot Devices

In the case of NAND boot, an additional Flash header must be defined as shown in the code example below. This additional Flash header must be located immediately after the DCD table and contains the length of the data to be read from boot device.

```
/* Flash Header Structure */
typedef struct {
    UINT32  length;         /* Length of data to be read */
} FLASH_CFG_PARMS_T;

where: UINT32 is a 32 bit unsigned integer.
```

In this Flash header structure, the parameter ‘length’ determines the length of image to copy to RAM.

Address Cycle Values for NAND Boot Devices

The Address cycle value for the connected NAND device is obtained from the BT_MEM_TYPE eFUSEs. See Table 7-8 for more information.

Error Detection and Correction

NFC automatically generates an ECC code for both main and spare data during NFC data load/read to/from NAND Flash, and NFC updates ECC in the ECC status Register. NFC performs error detection and error correction. If the number of ECC errors does not exceed the allowable limit (four for 4-bit ECC, eight for 8-bit ECC), then NFC corrects those errors.

On device power on, the copy of first 4 Kbytes boot data from NAND Flash device Block-0 to NFC buffer is done by boot ROM. Since in NAND Flash devices there are possibility of getting errors while reading first 4 Kbytes boot data from first block (NAND device Block-0), the user is required to duplicate the first 4 Kbytes boot code to the subsequent block (NAND device Block-1), to serve as a second copy option.

The boot ROM code utilizes the duplicate boot code according to the following flow:

1. If no ECC errors are detected in first boot block, boot execution performs the secure internal boot.
2. If ECC Error is detected in first 4 Kbytes boot data from the first block, the boot ROM code copies the 4 Kbytes boot data from the subsequent block:
   - If no error detected in the subsequent boot block, boot flow continues performing the secure internal boot.
   - If an error is detected in the subsequent boot block, the boot ROM code logs an error and jumps to USB/UART bootloader. The logged error can be queried using the serial protocol.

Figure 7-6 shows the arrangement of data in NAND Flash, in the case of duplicate boot code.

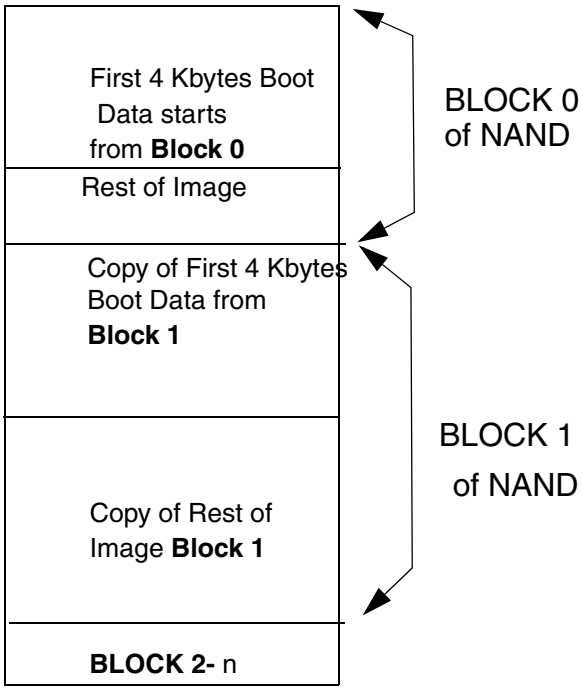


Figure 7-6. Duplicate Boot Code Data in NAND Flash Device

7.3.3.9.2 Flash Header for SD/eSD/MMC/eMMC/MoviNAND Boot Devices

The SD/eSD/MMC/eMMC/MoviNAND boot block size is fixed at 2 Kbytes. An additional header must be defined (just as in the NAND boot case), as described in Section 7.3.3.9.1, "Flash Header for NAND Boot Devices." See Figure 7-7. The header must be located immediately after the DCD table, and contains the length of the data to be read from boot device. eSDHC automatically generates CRC code during eSDHC data write/read to/from SD/eSD/MMC/eMMC/MoviNAND, and eSDHC updates the CRC in the CRC status register. Boot software performs following operations:

1. Copy the 2 Kbytes boot block data into internal RAM.
2. Check the CRC for errors:
   a) If no CRC error, then jump to address (base address plus the offset mentioned in Table 7-10 for SD/eSD/MMC/eMMC/MoviNAND) of internal RAM to authenticate the application if required.
   b) If CRC error occurs, boot ROM code will log an error and jumps to USB/UART bootloader. The logged error can be queried using the serial protocol.

Figure 7-7. Representation of Data in SD/MMC Card

7.3.3.9.3 Flash Header for I2C/CSPI EEPROM Boot Devices

The I^2C/CSPI EEPROM boot block size is fixed at 2 Kbytes. An additional header must be defined as in the NAND boot case (described in Section 7.3.3.9.1, "Flash Header for NAND Boot Devices"): the header is located immediately after the DCD table, and contains the length of the data to be read from the boot device. Representation of image data in I^2C/CSPI EEPROM is as shown in Figure 7-8.

The boot software performs the following operations:

1. Copy the 2 Kbytes boot block data into internal RAM.
2. Check the error using the Barker value:
   a) If no error is found, then jump to the address (base address plus the offset described in Table 7-10 for I^2C/CSPI EEPROM) of internal RAM to authenticate the application if required.
   b) If an error is found, then boot ROM code will log an error and jumps to USB/UART bootloader. The logged error can be queried using the serial protocol.

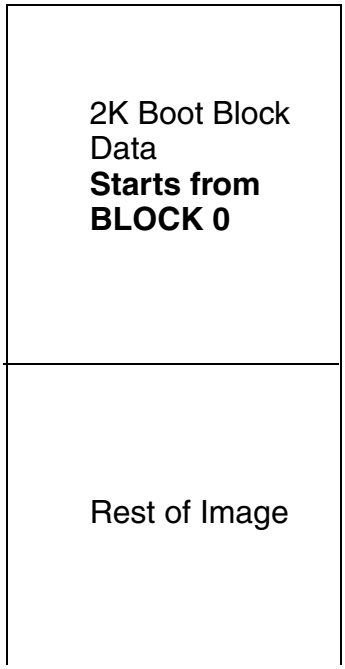


Figure 7-8. Representation of Data in I2C/CSPI EEPROM

7.3.3.9.4 Flash Header for OneNAND Boot Devices

In the OneNAND boot case, an additional header must be defined as for NAND boot (described in Section 7.3.3.9.1, "Flash Header for NAND Boot Devices"). The header must be located immediately after the DCD table, and contains the length of the data to be read from boot device.

The boot software performs the following operations:

1. At system power-up, OneNAND automatically copies 1 Kbyte data from the start of the Flash array (sector 0 and sector 1, page 0, block 0) to its boot RAM, so that boot RAM contains the OneNAND Flash header.
2. Boot ROM then copies the 1 Kbyte OneNAND boot RAM contents to the destination address located in the app_dest_ptr entry of application header, and decrements length of the image to be read from OneNAND by 1 Kbyte.

   Boot RAM area is memory-mapped and copying operation will be simple memory copying operation. The length of image to be read from OneNAND device is specified in the Flash header structure described in Figure 7-9. Any failure in the data loading from OneNAND Flash forces the processor boot ROM to switch to serial download.

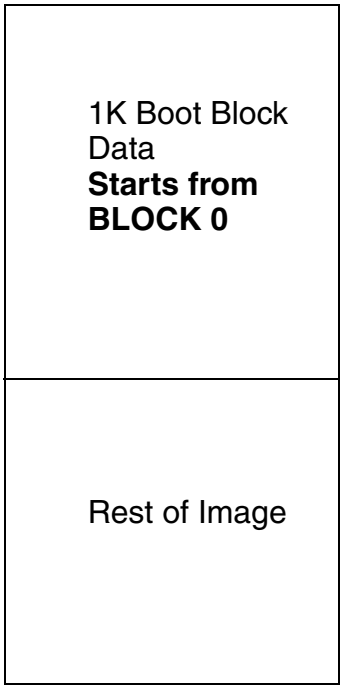


Figure 7-9. Representation of Data in OneNAND Flash

7.3.3.10 Device Configuration Data (DCD)

Some peripherals need to be configured before they can be used efficiently. The corresponding device specific configuration data is called DCD.

Upon reset, the i.MX25 uses the default register values for all peripherals in the system. These settings typically are not ideal for achieving optimal system performance.

For example, the EMI default settings allow core to interface to a NOR flash device immediately out of reset. This allows to interface with any NOR flash device, but has the cost of slow performance. Besides some peripherals like SDRAM etc. might require some sequence of register programming as part of configuration before it's ready to be used. It is assumed that EMI registers and eSDRAMC registers will be set up using DCD.

ROM bootstrap has a provision for accommodating solutions to above scenarios. To allow users to configure for better performance, the ROM reads a DCD table from the flash device. The boot ROM determines the location of the DCD table based on information located in the flash header shown in Figure 7-5 above. This DCD table is an array of structures containing three elements (access type, address and value) preceded by a barker code and length field. The number of DCD structure entries is limited to 60.

Table 7-12. DCD Block Description

Field Name	Description	Size (Bytes)	Value
DCD-barker	Barker for sanity check	4	0xB17219E9
DCD-block-length	The total length in bytes of theDCD block (excluding this length field as well as the barker code)	4	—
Followed by an array of structures with following fields			
DCD access type	Type of pointer (byte=0x1 halfword=0x2, word=0x4) in the following field	4	—
DCD-address	The absolute address of the register to be programmed	4	—
DCD-value	The value to be programmed at above address	4	—

Note: The DCD is read as 32-bit words, and must therefore be aligned on a word boundary.

The set of registers programmable with DCD must be restricted for security. Since the device configuration block is only post-authenticated (that is. after it has already been used), the restriction has to be very tight. ROM bootstrap performs boundary checking of DCD-address in the device configuration block against the valid range of addresses. The register sets allowed by the device configuration block include the following:

- Clock Controller Module (CCM)
- UART-1–5
- Universal Synchronous Bus (USB)
- IOMUX (only for the drive-strength registers SW_PAD_CTL)
- Watchdog (WDOG)
- NAND flash controller (NFC)
- Enhanced synchronous dynamic RAM controller (eSDRAMC)
- Wireless external interface module (WEIM)
- Enhanced SD host controller (eSDHC-1 and eSDHC-2)
- I^2C-1–3
- Configurable serial peripheral interface (CSPI) 1–3
- CS0–4
- CSD0–1
- Real Time Integrity Checker (RTIC)

ROM bootstrap has a look-up table of lower and upper addresses for each module allowed to be programmed through the DCD. This table is utilized for verifying any attempt to configure a register. ROM bootstrap determines the size of the block by reading DCD-block-length and copies it into IRAM. It is considered to be an error if any of the following is true:

- Any register address is outside the pre-defined range.

- The length of the DCD exceeds the maximum acceptable size. That is MAX_HW_CFG_SIZE (in bytes). Assuming maximum of sixty elements for the array it amounts to seven hundred and twenty bytes.
- Any other failure.

If successful, ROM bootstrap configures the HW registers and failure leads to non-initialization of the intended modules. A DCD error (for example, writing to a non-supported address such as ROMPATCH) causes the ROM boot code to enter the boot loader (serial download).

The DCD table must follow the structure below:

```
typedef struct
{
DCD_PREAMBLE_T preamble; /* Preamble */
/* Type / Address / data elements */
DCD_TYPE_ADDR_DATA_T type_addr_data[count]; /*where count would be some hardcoded value
less than 60*/
} DCD_T;
```

Where:

```
typedef struct
{
UINT32 barker; /* Barker for sanity check */
UINT32 length; /* Device configuration structure length (not including preamble) */
} DCD_PREAMBLE_T;

typedef struct
{
UINT32 type; /* Type of pointer (byte=0x1, halfword=0x2, word=0x4) */
UINT32 *addr; /* Address to write to */
UINT32 data; /* Data to write */
} DCD_TYPE_ADDR_DATA_T;
```

DCD_T typically contains the configuration data for WDOG, SDRAM, and EIM etc.

Example code:

```
DCD_T device_config_data =
{
    {
     IROM_DCD_BARKER, /* assuming this is pre-defined as macro */
     (18 * sizeof(DCD_TYPE_ADDR_DATA_T))
    },

    {
        {0x00000002, (UINT32 *)0x53fdc000, 0x00003F7E}, /* WDOG WCR */
        {0x00000002, (UINT32 *)0x53fdc004, 0x00005555}, /* WDOG WSR */
        {0x00000002, (UINT32 *)0x53fdc004, 0x0000aaaa}, /* WDOG WSR */
        {0x00000004, (UINT32 *)0xb8002000, 0x14110802}, /* EIM CTL H 0x11134722 */
      {0x00000004, (UINT32 *)0xb8002004, 0x80330d01}, /* EIM CTL L -0x50331D01- 16 Bit */
 {0x00000004, (UINT32 *)0xb8002008, 0x00000800}, /* EIM CTL A */
        {0x00000002, (UINT32 *)0xa0002394, 0x00000060}, /* EIM CS0 -6F0C- 16 Bit */
        {0x00000002, (UINT32 *)0xa0002394, 0x00000003}, /* EIM CS0 -6F0C- 16 Bit */
        {0x00000002, (UINT32 *)0xa0000000, 0x000000ff}, /* EIM CS0 - R/A - 16 Bit */
        {0x00000004, (UINT32 *)0xb8001004, 0x000ac7a8}, /* SDRAM CS0 CFG0 - Timing */
```

```
            {0x00000004, (UINT32 *)0xb8001000, 0x92110080}, /* SDRAM CS0 CTL0 - Enable */
            {0x00000004, (UINT32 *)0x80000400, 0x00000000}, /* SDRAM CS0 - Precharge */
            {0x00000004, (UINT32 *)0xB8001000, 0xa2110080}, /* SDRAM CS0 - Refresh */
            {0x00000004, (UINT32 *)0x80000000, 0x00000000}, /* SDRAM CS0 - Refresh */
            {0x00000004, (UINT32 *)0x80000000, 0x00000000}, /* SDRAM CS0 - Refresh */
            {0x00000004, (UINT32 *)0xB8001000, 0xb2110080}, /* SDRAM CS0 - Load Mode */
            {0x00000001, (UINT32 *)0x80000033, 0x00000000}, /* SDRAM CS0 - Load Mode */
            {0x00000004, (UINT32 *)0xB8001000, 0x82114c80}, /* SDRAM CS0 - Norm Mode */
    }
    };
```

The code above is based on Table 7-13.

Table 7-13. Valid DCD Address Range

Address Range	Start Address	Length (in bytes)
CCM register set	0x53F80000	0x00004000
CSD0 memory	0x80000000	0x10000000
CSD1 memory	0x90000000	0x10000000
WEIM register set	0xB8002000	0x00001000
NFC register set	0xBB000000	0x00002000
CS0 memory	0xA0000000	0x08000000
CS1 memory	0xA8000000	0x08000000
CS2 memory	0xB0000000	0x02000000
CS3 memory	0xB2000000	0x02000000
CS4 memory	0xB4000000	0x02000000
USB memory	0x53FF4000	0x00004000
IOMUXC registers	0x43FAC000	0x00004000
UART1 register	0x43F90000	0x00004000
UART2 register	0x43F94000	0x00004000
UART3 register	0x5000C000	0x00004000
UART4 register	0x50008000	0x00004000
UART5 register	0x5002C000	0x00004000
I2C1 register	0x43F80000	0x00004000
I2C2 register	0x43F98000	0x00004000
I2C3 register	0x43F84000	0x00004000
CSPI1 register	0x43FA4000	0x00004000
CSPI2 register	0x50010000	0x00004000
CSPI3 register	0x50004000	0x00004000
WDOG register	0x53FDC000	0x00004000
eSDHC1 register	0x53FB4000	0x00004000

Table 7-13. Valid DCD Address Range

Address Range	Start Address	Length (in bytes)
eSDHC2 register	0x53FB8000	0x00004000
eSDRAM controller	0xB8001000	0x00001000
RTIC register	0x53FEC000	0x00004000

In the boot flow, the DCD data is used prior to being verified.The DCD data must be included as part of the memory regions verified as indicated in one of the HAB CSF commands.

7.3.4 External Boot Mode (BOOT_MODE[1:0] = 10)

External boot is selected by driving value of ‘10’ on the BMOD[1:0] pins at device power up and fuse DIR_BT_DIS is not blown. This mode is non-secure mode. In this mode, the core boots directly from external memory (in the case of WEIM) or running NAND-flash boot. The supported direct external flash types include the following:

- NOR flash using WEIM
- NAND flash

7.3.4.1 NOR Flash using WEIM

i.MX25 supports either muxed or non-muxed address data boot from WEIM interface.

NOTE

The boot from WEIM is reserved for debugging/testing purposes, and the WEIM signals are muxed with the display port. If still user’s uses this mode, it is up to the loaded code, to switch back the IO muxing to the display port, on finish of boot.

7.3.4.2 NAND Flash

i.MX25 supports external boot from a NAND device. The initial 4 Kbytes of data is copied from the Flash onto the NFC buffer and jumps to the base address of the NFC buffer to execute it.

7.3.5 UART/USB Serial Download Mode (BOOT_MODE[1:0]=11)

The UART/USB serial download mode is selected by driving a value of ‘11’ on the BOOT_MODE[1:0] contacts at device power-up. Bootable UART is selected by BT_UART_SRC[2:0] fuses. Selection between UART and USB download boot device is made by polling the UART and USB controllers in turn. Whichever device shows activity first is selected.

This mode is also invoked when the external Flash device is not programmed or when a failure is encountered during the boot flow process. It is invoked in any of the following conditions:

- ROM is in internal boot and none of the fuses for external Flash are satisfied.
- Security hardware failure

- Run time exception occurs
- Error returned by the HAB functions in production mode.

To determine the active serial port either UART or USB, the processor ROM polls UART and USB status register for about 90 seconds. If there is no activity on either port within predefined polling loop time then ROM will reset the device using WDOG. In USB/UART bootloader valid case the WDOG will be serviced periodically. WDOG will expire and reset the device if the communication between the Host and the IC hangs for more than 90 s or if the processor goes into an endless loop.

If UART1–5 (selected depending on the BT_UART_SRC[1:0] eFUSE values) is selected as the communication path then UART1–5 is configured for baud rate 115.2 kbps, parity disable, 1 stop bit and 8-bit TX-RX character length and data is downloaded using the serial protocol. Otherwise, if USB is selected, then the USB core and either the internal or external transceiver are configured.

For the UART, activity is detected using the receive data ready (RDR) flag showing at least one character has been read into the FIFO.

For the USB, either the integrated on-chip PHY or the external PHY (using ULPI interface) can be used. Activity is detected using the setup endpoint status register showing that the setup transaction is received.

The USB/UART boot flow is shown in Figure 7-10.

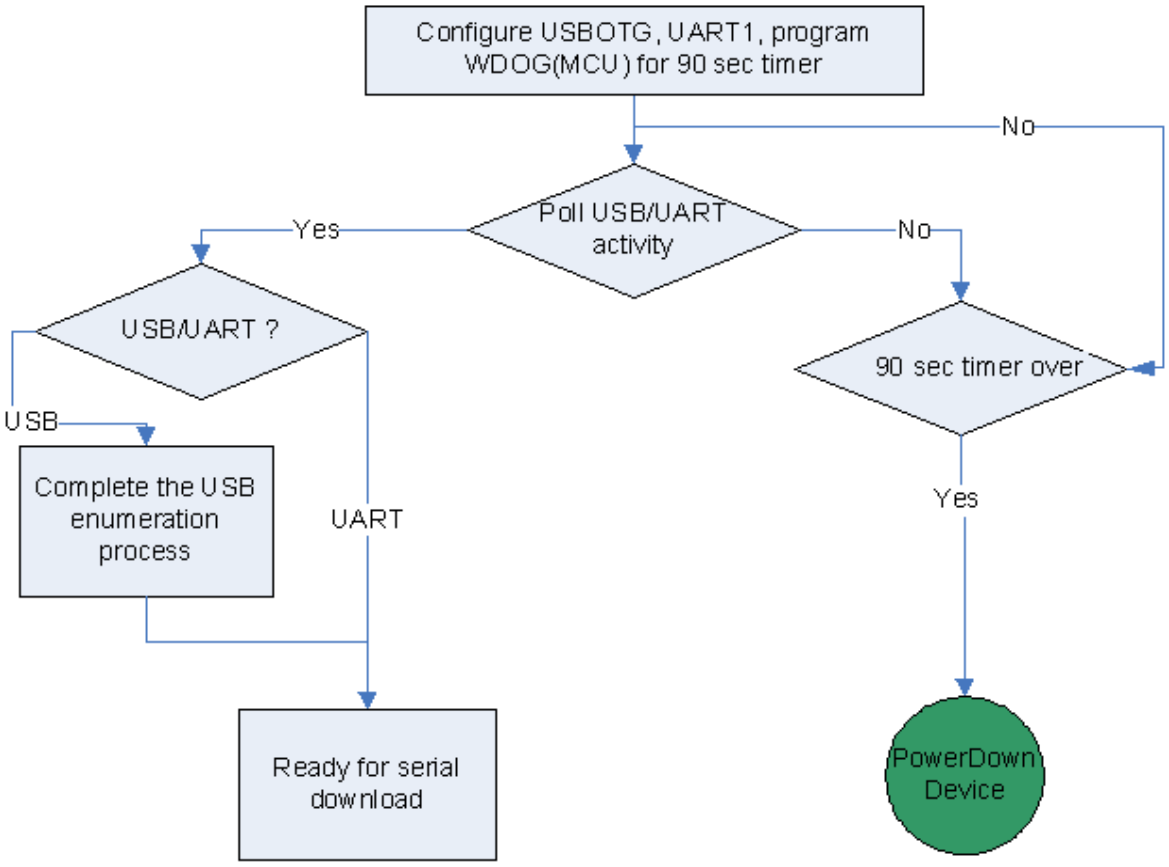


Figure 7-10. USB/UART Boot Flow

7.3.5.1 USB

The i.MX25 USB support is provided by the USB module (high performance USB On-The-Go (OTG) functionality, compatible with the USB 2.0 specification, the OTG supplement and the ULPI specification). The module consists of 2 independent USB cores (OTG and HOST), each with Serial and ULPI USB ports. The OTG core also supply the UTMI interface for the internal UTMI PHY.

USB OTG port is the bootable USB device. Additional USB hosts are intended for intra-platform communication, and are not bootable.

Since boot image loading through USB interface takes negligible time, the boot using USB is supported by USB Full Speed mode only.

To enumerate the ROM boot USB function the host PC needs a compatible windows driver and a specific application “ADS Tool kit”. Once enumerated, the USB boot device uses a specific protocol “Download protocol” to download an application, DCD block or command sequence file (CSF) and launch the application.

For USB boot, the USBOTG controller is used either with the integrated PHY (typical case) or using ULPI interface to an external PHY: selection is determined either by fuses, or by sampling contacts at boot as described in Table 7-15. The boot ROM USB driver configures the USB controller to function in Full Speed mode. The control endpoints EP0IN and EP0OUT are configured for control transfer and for data transfer in bulk mode. EP1OUT and EP2IN are configured as IN and OUT transactions respectively.

The supported transceivers are listed in Table 7-14.

Table 7-14. Supported Transceivers

Transceiver	Boot Speed	Interface	USB Controller Mode
Integral PHY	Full Speed (FS)	UTMI	FS Device
External PHY	Full Speed (FS)	ULPI	FS Device

7.3.5.1.1 USB Configuration Details

A full-speed low-level USB OTG function device driver is supported. Supported USB transceivers include:

- USB OTG Internal PHY.
- USB OTG External ULPI PHY.

Table 7-15. USB BOOT mode fuse setting

BOOT MODE	Senna Fuse
USB OTG Internal PHY	BT_USB_SRC=00
USB OTG External ULPI PHY	BT_USB_SRC=01

The VID/PID and strings for USB device driver are listed in Table 7-16.

Table 7-16. VID/PID and Strings for USB Device Driver

Descriptor	Value	Remarks
VID	0x15A2 (Freescale vendor ID)	
PID	003A	Allocated Based on BPN (Before Part Number)
String Descriptor1 (iManufacturer)	Freescale Semiconductor Inc.	
String Descriptor2 (iProduct)	SP Blank SENNA SE Blank SENNA NS Blank SENNA	
String Descriptor4	Freescale Flash	
String Descriptor5	Freescale Flash	

A typical USB boot flow is shown in Figure 7-11.

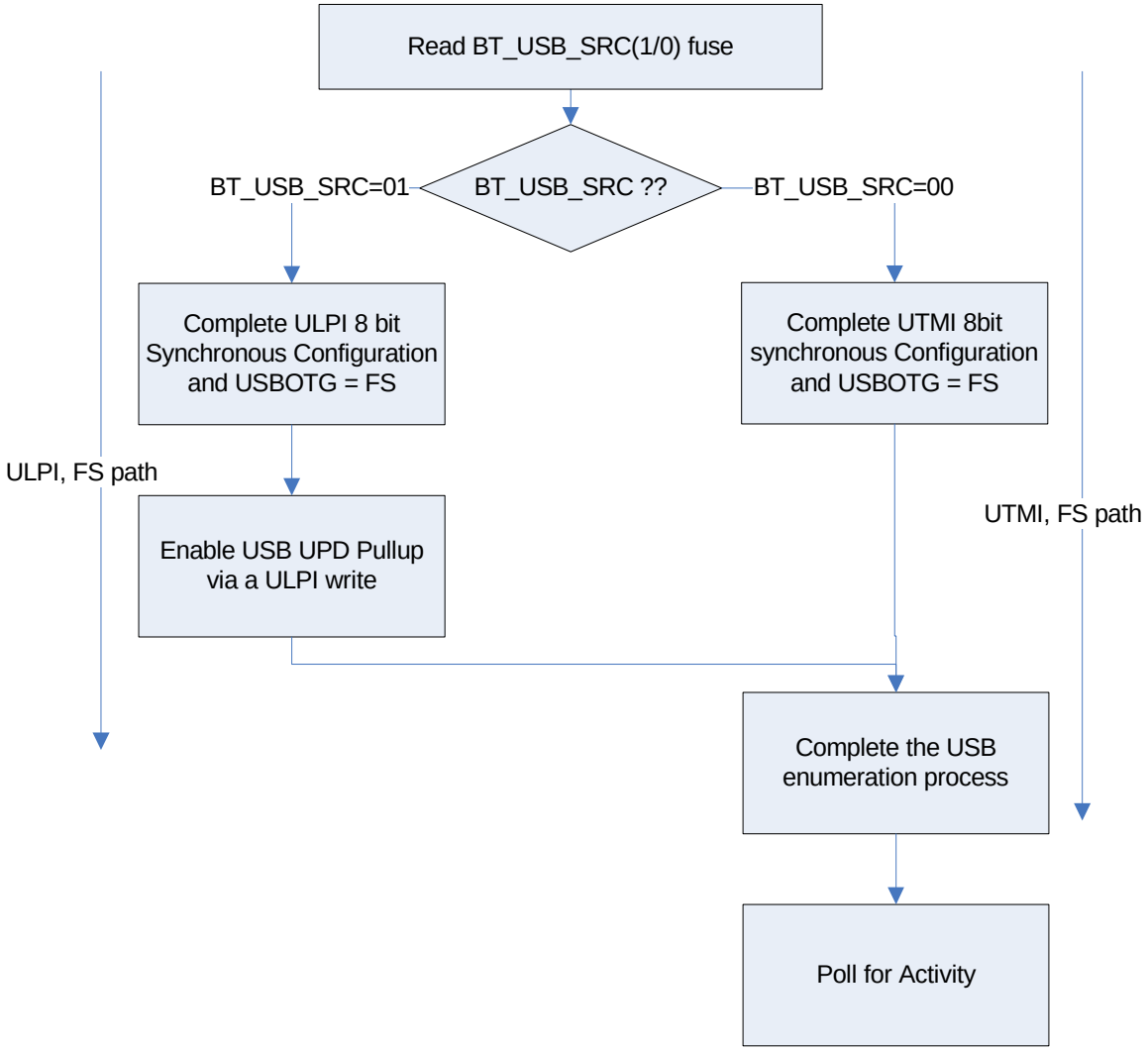


Figure 7-11. USB Boot Flow

7.3.5.2 UART

All UART ports are bootable.

Once initialized, the UART boot device uses a specific protocol “Serial Downloader Protocol” to download an application, DCD block or command sequence file (CSF) and launch the application.

The UART driver uses the communication parameters given in the Table 7-17 below.

Table 7-17. UART Configuration

Parameter	Value
Baud rate	115.2 kbps
Parity check	Disabled
Word size	8 bits

Table 7-17. UART Configuration (continued)

Parameter	Value
Stop bits	1 bit
RTS	Ignored
CTS	Controlled by host
Receive contact	RxD
Transmit contact	TxD

7.4 Exception Handling

No special interrupt handling routines are required during bootup. Instead, the iRAM exception table is filled with the address of the USB/UART bootloader module, so that any exception or interrupt will result in the RAM path being executed. Interrupts are disabled during boot ROM code execution, and can be enabled after boot ROM code execution. Flash or RAM applications image are responsible for enabling of interrupts as required.

7.5 Error Logging

All ROM errors are logged to address 0x7800_18D4.

7.6 USB Low-Power Boot Mode

i.MX25 supports low-power boot (LPB) mode with depleted or disconnected battery, from USB power supply only.

ROM involvement in LPB is required to be minimal, while most required configuration operations are performed by downloaded power management IC (PMIC) and USB drivers.

The platform current consumption during LPB ROM stage is less than 100 mA of VUSB. To meet this requirement, the DRAM module can be disabled and the primary boot image downloaded to iRAM.

The BT_LPB fuse settings determine the LPB mode as shown in Table 7-18.

Table 7-18. BT_LPB eFUSE Settings and Associated LPB Modes

BT_LPB[1:0]	LPB Mode
00	LPB disabled
01	Generic PMIC and one GPIO input (low battery detection)
10	Generic PMIC and two GPIO inputs (low battery and charger detection)
11	Atlas AP PMIC

The PMICs indicate power conditions through the GPIO_C contact. For example, in the basic case (BT_LPB[1:0]=01) where PMIC is used for low battery detection, then the signal is driven high is the battery is ready for normal boot; the signal is driven low if the battery is low, depleted, or disconnected. In the case where the PMIC is used for both battery and charger detection (BT_LPB[1:0]=01), then GPIO_F

is used to indicate the charger status. For better flexibility, in BT_LPB[1:0]=01 mode, GPIO_F receives an indication about charger presence in the system.

If the i.MX25 internal USB PHY has been selected (BT_USB_SRC=0), then the ROM receives a USB charger indication from the internal USB PHY.

The BT_LPB_FREQ eFUSE settings determine the maximal LPB ARM frequency as shown in Table 7-19.

Table 7-19. LPB_FREQ eFUSE Settings and Associated LPB ARM Frequencies

LPB_FREQ[2:0]	Frequency (MHz)
000	133 (default)
001	CKIH
010	55.33
011	66
100	83
101	166
110	266
111	Normal boot frequency

ROM code supports all primary boot devices in LPB mode. However the assumption that UART and USB boot devices will not be used by customers in LPB mode.

The LPB negotiation flowchart is shown in Figure 7-12.

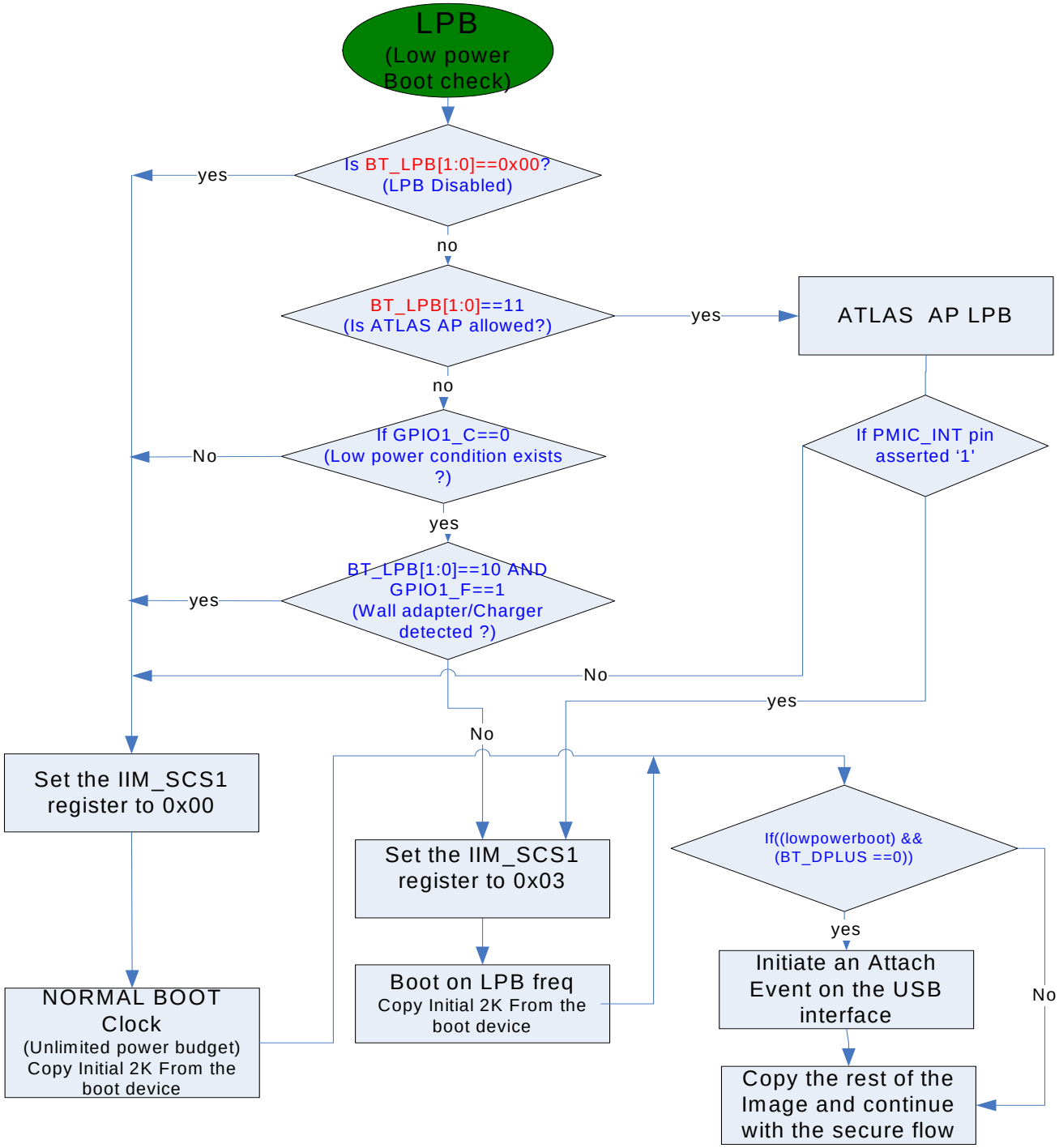


Figure 7-12. USB Low-Power Boot Flow

7.7 High Assurance Boot (HAB)

The boot flow handles all the security aspects, to ensure complete secure boot (based on fuse settings). This includes handling HAB modes, performing various code checks and initializing the security hardware secure real time clock (SRTC).

The high assurance boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available. The integration of the HAB feature with ROM code ensures that i.MX25 does not enter an operational state if the existing hardware security modules have detected a condition that may be a security compromise, or if areas of memory deemed to be important have been modified.

Figure 7-13 illustrates the components used during a secure boot. The processor security components are RTICv3, DryIce and SCCv3. They are supported through the HAB Library. The HAB uses either the RTICv3 hardware accelerator or its internal software implementation to support the SHA256 message digest operations that are part of the signature authentication process.

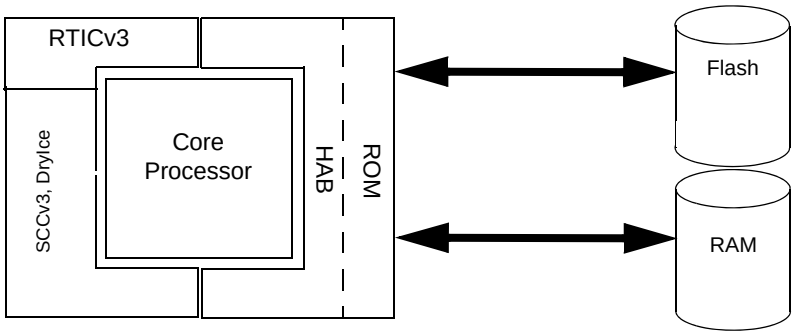


Figure 7-13. Secure Boot Components

7.7.1 High-Assurance Boot (HAB) Security Types

The HAB requires the location of two components in the boot device: the starting address of the super root key (SRK) data and the starting address of the command sequence file (CSF) data. Both of these components are defined in the Flash header as illustrated in Figure 7-5. The SRK defined in the application data located in the boot device is validated by the HAB. The HAB will perform a SHA-256 hash digest of the SRK provided in the application data. It will then compare the computed hash with that stored in the i.MX25 eFUSEs. If validation of the SRK is unsuccessful, the ROM will enter the serial bootloader.

The CSF provides the signature and certificate information of the piece of software to authenticate. The CSF is generated by using a client/server code signing tool (CST). Contact your Freescale representative for further information regarding code signing for HAB.

The boot flow is influenced by the device's HAB TYPE as specified in Table 7-20.

Table 7-20. HAB Security Types

HAB TYPE	HAB_TYPE[2:0] Fuse Value	Action
HAB_ENGINEERING	0x1(0b001) (default)	Initialize secure hardware, execute DCD block and authenticate applications prior to their execution. Any errors detected are logged, but have no influence on the boot flow.
HAB_PRODUCT	Any value in range 0b000–0b111 except for 0b001 and 0b100	Initialize secure hardware, execute DCD block and authenticate applications prior to their execution. Any errors detected are logged and control passes to the USB/UART Bootloader.

7.7.1.1 HAB_ENGINEERING Security Type

This level is intended for use during the development phases of a product. All HAB functions are executed as for a production device—security hardware is initialized (except the SCC is left in non-secure state), the DCD is processed if present, and the software in Flash or downloaded to RAM is authenticated by HAB prior to its execution. Any errors detected are logged, but have no influence on the boot flow. With this level it is possible to develop software without requiring that each build be signed for HAB authentication, since the device will boot even if the code signatures are missing.

7.7.1.2 HAB_PRODUCTION Security Type

This level is intended for use with shipping products. All HAB functions are executed – security hardware is initialized (the SCC enters Secure state), DCD is processed if present, and software in Flash or downloaded to RAM is authenticated by HAB prior to its execution. Any errors detected are logged, and the boot flow is aborted with control passing to the download mode. With this level, execution does not leave the internal ROM unless the target executable image has been authenticated.

In HAB_PRODUCTION mode, all the ROM errors are logged to the pu_irom_error_status variable (see Section 7.5, "Error Logging"). The address of pu_irom_error_status is 0x7800_18C0. All the HAB errors/status can be found in Table 7-21.

Table 7-21. HAB Status Codes

Name	Description	Value
HAB_DATA_OUT_OF_BOUNDS	Data specified is out of bounds.	0x8D
HAB_FAIL_ASSERT	Error during Assert Verification.	0x55
HAB_FAIL_HASH_VERIFICATION	Hash verification failed (including hash verification on certificates).	0x36
HAB_FAIL_PK_VER	Certificate parsing failed, or the certificate contained an unsupported key (including unsupported key length).	0x33
HAB_FAIL_SIG_VERIFICATION	Signature verification failed (including signature verification on certificate).	0x35
HAB_FAIL_SUPER_ROOT_INSTALL	Super-Root key installation failed.	0x47
HAB_FAILURE	Failure not matching any other description.	0x39
HAB_INVALID_CSF_COMMAND	CSF Command Sequence contains an unsupported command identifier.	0x4B

Table 7-21. HAB Status Codes (continued)

Name	Description	Value
HAB_INVALID_CSF_HEADER	Absence of expected CSF Header (including mismatched HAB Version).	0x4E
HAB_INVALID_CSF_LENGTH	CSF length is unsupported.	0x4D
HAB_INVALID_CSF_TYPE	CSF TYPE does not match processor TYPE.	0x2E
HAB_INVALID_CSF_UID	CSF UID does not match either processor UID or generic UID.	0x2D
HAB_INVALID_CSF_CODE	CSF customer/product code does not match processor Customer/Product code.	0x3A
HAB_INVALID_KEY_INDEX	Key index is either unsupported, or an attempt is made to overwrite the Super-Root key from a CSF command, or an attempt to write a non-CSF subordinate key to HAB_CSF_KEY_INDEX	0x87
HAB_PASSED	Successful operation completion.	0xF0
HAB_SCC_NOT_SECURE	SCC unexpectedly not in Secure State.	0x17
HAB_SECURE_RAM_BAD_KEY	Secure RAM secret key invalid.	0x1E
HAB_SECURE_RAM_CLR_FAIL	Secure RAM initialization failure.	0x1D
HAB_SECURE_RAM_FAIL	Secure RAM Self Test failure.	0x1B
HAB_SCC_FAIL	SCC unexpectedly not in Non-Secure State	0x53
HAB_SECURE_RAM_INT_ERR	Secure RAM internal failure.	0x2b
HAB_SECURE_RAM_SEC_KEY	Secure RAM secret key unexpectedly in use.	0x27
HAB_SAHARA_FAIL	SAHARA failure.	0x3C
HAB_SAHARA_SCC_FAIL	SAHARA/SCC connectivity failure.	0x59
HAB_RTIC_REGION_FAIL	All RTIC regions are allocated.	0xA3
HAB_RTIC_SCC_FAIL	RTIC/SCC connectivity failure.	0x93
HAB_RNG_SCC_FAIL	RNG/SCC connectivity failure.	0x95
HAB_RNG_SELF_TEST_FAIL	RNG self test failure.	0x99
HAB_DRYICE_BAD_RANDOM_KEY	DryIce Random key test failure	0x9C
HAB_DRYICE_BAD_PROGRAMMED_KEY	DryIce Programmed key test failure	0x9A
HAB_DRYICE_REG_WRITE_FAIL	DryIce register write fail	0xAA
HAB_DRYICE_BAD_XOR_KEY	DryIce XORed key test failure	0xA5
HAB_SHW_DISABLED	SHW is not enabled.	0x0F
HAB_UNINITIALISED_KEY_INDEX	An attempt is made to read a key from the list of subordinate public keys at a location where no key is installed.	0x8B
HAB_UNSUPPORTED_ALGORITHM	Algorithm type is either invalid or otherwise unsupported.	0x8E
HAB_INVALID_WRITE_REG	Write operation to register failed.	0x66

7.7.2 Function Prototypes

The following three HAB functions are available to application code residing outside of the ROM. These functions can be considered trustworthy since they are located in ROM and cannot be changed. Note that the HAB API functions may only be called from external images that have not yet enabled the MMU.

7.7.2.1 pu_irom_boot_decision

Syntax:

```
void pu_irom_boot_decision(void)
```

Description:

Entry point for the serial downloader. Performs the check of whether to use UART or USB for the serial bootloader.

Inputs:

None.

Returned Value:

None.

PreConditions/Assumptions:

GPIO for selecting USB Serial PHY and ULPI PHY is set appropriately.

Post Conditions:

None.

7.7.2.2 hab_csf_check

Syntax:

```
hab_result hab_csf_check(
                                    UINT8    csf_count,
                                    UINT32   *csf_list);
```

Description

Performs integrity checks on software in programmable memory as instructed by CSFs.

Inputs:

The number of CSFs to be processed and their locations

Returned Value:

The structure:

```
typedef struct
{
                                    unsigned char status; /* Status code */
                                    unsigned char type; /* HAB type from Table 7-20 */
} hab_result;
```

with processor TYPE and one of the following status codes:

- HAB_PASSED if all CSFs are valid and all verifications in all CSFs are satisfied.
- HAB_DATA_OUT_OF_BOUNDS if csf_count is 0, csf_count exceeds maximum length of CSF chain or csf_list is NULL.
- Appropriate error code for other failures as stated in Table 7-21.
- HAB_FAILURE otherwise.

Pre Condition/Assumptions:

- Verified blocks list and subordinate keys list are initialized. That means the hab_health_check() has been called prior to invoking hab_csf_check.
- The parameter csf_list points to a list of length csf_count.

Post Conditions:

- The subordinate keys list contains successfully validated subordinate keys.
- The verified blocks list contains the list of successfully verified data blocks, padded at the end of the list to indicate that no more verified block is available. The list of verified block is verified using hab_assert_verification.

7.7.2.3 hab_assert_verification

Syntax:

```
hab_result hab_assert_verification(
                                    UINT8    *block_start,
                                    UINT32  block_length);
```

Description:

Perform only after a CSF Check. Determines if a block of data lies within the regions of the pre-authenticated block or the regions verified during CSF Check. If SCC is supported, the state of the SCC (if enabled and the processor TYPE is not engineering) is also tested to ensure that the hardware is secure.

Inputs:

Starting address and length (in bytes) of the data block.

Returned Value:

The structure:

```
typedef struct
{
                                    unsigned char status; /* Status code */
                                    unsigned char type; /* HAB type from Table 7-20 */
} hab_result;
```

with processor TYPE and one of the following status codes:

— HAB_PASSED if all tests pass,

— HAB_FAIL_ASSERT if block is not pre-authenticated or in regions verified during CSF Check,

— HAB_SCC_NOT_SECURE if SCC supported and is not in the secure state and processor TYPE is not engineering,

— HAB_FAILURE otherwise.

Pre Condition/Assumptions:

The verified blocks list contains (start, length) pairs of 32 bit values, padded at the end of the list to indicate that no more verified block is available. If the given block has been verified or pre-authenticated, it is assumed to lie wholly within the boundaries of a single block in the verified blocks list.

Post Conditions:

None.

7.7.3 API Jump Table Addresses

The address of the HAB functions are available using a jump table defined in the ROM. Table 7-22 lists the functions and the associated jump table address.

Table 7-22. HAB API Jump Table Addresses

HAB API Function	Jump Table Address
hab_csf_check	0x88
hab_assert_verification	0x8C
pu_irom_boot_decision	0x00406968

7.7.4 DryIce Initialization

The DryIce module is initialized by the ROM as part of the HAB library. The DryIce secured registers cannot be configured by non-secure software. Non-secure software is considered as an application image executing after a secure boot that is not in supervisor mode and should not be confused with image data or code verified by HAB.

A dedicated non-secure access (NSA) bit in the DryIce control register can allow unsecured software to update the DryIce secured registers. This bit can only be set by secured software. In any case DryIce, secured registers like DryIce key select, random key, programmed key, timer counter and monotonic counter registers cannot be configured once the corresponding lock bits are set in DryIce control register. The initialization steps performed by HAB as part of boot ROM depends on HAB_TYPE and DryIce state. No DryIce programming is performed by the HAB when the HAB_TYPE is Engineering or Security disabled.

For more information on the DryIce module, see your Freescale representative.

7.7.4.1 HAB Support for DryIce in Valid/Non-Valid State

Prior to exiting from the ROM, the HAB ensures that the DryIce is configured appropriately for the given DryIce state.

For secure applications HAB enables an image to configure the DryIce register settings. When the HAB command sequence file included in the application includes the following WRITE_TO_REGISTER (WTR) command:

WRITE_TO_REGISTER <DryIce General Purpose Reg (GPR)> 4 0xCA69_3569

Then HAB will refrain from setting any of the lock bits in DryIce. However, if the CSF does not include this specific WTR command, then the HAB does not allow DryIce to be configured by setting lock bits in the DryIce control register prior to leaving the boot ROM.

Note that when using the above WTR command to leave the DryIce unlocked, it is recommended that this be the first command in the CSF.

7.7.4.1.1 Normal Boot Support

For a normal application, the HAB always sets the soft lock bits for the timer and monotonic counter in the DryIce control register. The HAB also sets other soft lock bits based on the value of the secure key select (SKS) bit in the DryIce key select register.

- If SKS is set to random key or random key XORed with IIM key, the HAB sets the soft lock bits for key select and random key.
- If SKS is set to programmed key or programmed key XORed with IIM key, the HAB sets the soft lock bits for key select, programmed key read and programmed key write.
- If the above two conditions are not met, the default SKS is the IIM key. In this case, the HAB sets the soft lock bit for key select.

7.7.4.1.2 External Boot Support

Direct (non-secure) boot modes (for example, direct NAND Flash boot) require the boot ROM to be executed first. If the DryIce module is in the Valid/Non-Valid state, the HAB sets the SWR bit in the DryIce control register which performs a POR on the DryIce. During the POR, the DryIce state is changed to the FAIL state.

7.7.4.2 HAB Support for DryIce in Fail State

When DryIce is in the FAIL state, no specific DryIce Programming is performed by HAB.

7.8 Serial Download protocol

This section describes the serial download protocol used for all boot devices in the serial bootloader on i.MX25.

Each stage in the protocol begins with a command issued by the host to the device, followed by a response from the device to the host. For most commands, that completes the protocol stage. The exception is the Write File command, which has an additional data stream sent from the host to the device after the response. The protocol is terminated by the host issuing a Write File command with application file type. After processing the Write File commands, the device interprets the next command as a Completed command, and resumes execution of the boot flow in order to authenticate the downloaded application (if required) and then execute it.

7.8.1 Get Status

The Get Status command retrieves the error log stored during a failed boot (or the success code when the serial bootloader is selected deliberately).

Table 7-23. Get Status Command

Command	0x05	0x05	–	–	–	–	–	–	–	–	–
Response	SC	SC	SC	SC	SC						

The fields have the following interpretation:

- SC = Status Code. Four copies of the status code are sent.
- – = Don’t Care (but must be present)

7.8.2 Read Memory

The Read Memory command reads a stream of bytes, half-words or words of data starting from a given address in memory. At production-level security, this command is ignored.

Table 7-24. Read Memory Command

Command	0x01	0x01	A[3:0]		DS	C[3:0]	–	–	–	–	–
Response (failure)	ACK[3:0]		Data stream starting from lowest address								
Response (success)	ACK[3:0]										

The fields have the following interpretation:

- A[3:0] = Target address (most-significant byte first).
- DS = Data Size (0x08 = byte, 0x10 = half-word, 0x20 = word). Unsupported DS values result in the failure response.
- C[3:0] = Number of data elements (bytes, half-words or words as appropriate) in data stream (most-significant byte first).
- ACK[3:0] = 0x56, 0x78, 0x78, 0x56.
- – = Don’t Care (but must be present)

7.8.3 Write Memory

The Write Memory command reads a byte, half-word, or word of data to a given address.

Table 7-25. Write Memory Command

Command	0x02	0x02	A[3:0]		DS	–	–	–	–	D[3:0]	–
Response (failure)	ACK[3:0]		0x12	0x8A	0x8A	0x12					
Response (success)	ACK[3:0]										

The fields have the following interpretation:

- A[3:0] = Target address (most-significant byte first).
- DS = Data Size (0x08 = byte, 0x10 = half-word, 0x20 = word). Unsupported DS values result in the failure response.
- D[3:0] = Data to write (most-significant byte first). For DS = 0x08, only D[0] is used. For DS = 0x10, only D[1:0] is used.
- ACK[3:0] = 0x12, 0x34, 0x34, 0x12 for production-level security, and 0x56, 0x78, 0x78, 0x56 otherwise.
- – = Don’t Care (but must be present)

At production-level security, the address ranges that may be written are restricted to those listed for DCD. Attempts to write outside of the valid ranges are ignored and result in the failure response. For other security levels, no restrictions apply to the target address.

7.8.4 Re-enumerate

The re-enumerate command resets the USB connection with an updated descriptor.

Table 7-26. Re-Enumerate Command

Command	0x09	0x09	–	–	–	–	–	–	–	SN[3:0]	–
Response	0x89	0x23	0x23	0x89							

The fields have the following interpretation:

- SN[3:0] = Serial number for enumeration descriptor.
- – = Don’t Care (but must be present)

7.8.5 Write File

The Write File command writes a stream of bytes to a given address in memory. The byte stream may be assigned a file type in order to distinguish CSF, DCD and application files. An application file type leads to the termination the serial download protocol, with the next command (whatever the command byte values) being treated as the Completed command.

Table 7-27. Write File Command

Command	0x04	0x04	A[3:0]	–	Q[3:0]	–	–	–	–	FT
Response	ACK[3:0]									
Data	Byte stream with data for lowest address first									

The fields have the following interpretation:

- A[3:0] = Target address (most-significant byte first).
- C[3:0] = Number of bytes in data stream (most-significant byte first).
- FT= File Type (0xAA = application (terminates protocol), 0xCC = CSF, 0xEE = DCD). With unrecognized FT values, the file is still downloaded, but the pointers to the three essential files are not modified.
- ACK[3:0] = 0x12, 0x34, 0x34, 0x12 for production-level security, and 0x56, 0x78, 0x78, 0x56 otherwise.
- – = Don’t Care (but must be present)
- At production-level security, the address ranges which may be written are restricted to those listed for DCD. Attempts to write outside of the valid ranges are ignored and result in the failure response. For other security levels, no restrictions apply to the target address.
- In case of application file type, the Flash header should still be provided. It must be located with a 0x0 offset, meaning right before the application code that will be authenticated (if secure boot), and executed.

7.8.6 Completed

The Completed command is required after the Write File command with application file type (0xAA) in order to terminate the protocol. The content of the command is irrelevant, but a command must be sent. This command triggers authentication and DCD processing (if required) followed by execution of the application. At production-level security, if application authentication fails, the serial download protocol resumes, with the status code for authentication failure available through the Get Status command.

Table 7-28. Completed Command

Command	–	–	–	–	–	–	–	–	–	–	–
Response	0x88	0x88	0x88	0x88							

The fields have the following interpretation:

- – = Don’t Care (but must be present)

7.9 Flash Code Image Detection

7.9.1 Overview

If the HAB security type is set as HAB_ENGINEERING then any code to be flashed and executed, even if it has no valid signature.

In the production process, there are two ways to flash an image into Flash memory:

- Flash the memory IC before it is installed on the PCB
- Use the i.MX25 ROM's serial downloader library.

In production, the serial downloader mode can not be entered using boot contact configuration. Instead, the following code-flashing functionality is provided:

1. The ROM boot code first checks a known fixed Flash address for a Barker code.
2. If the Barker code is missing, the boot assumes an empty Flash and calls the serial downloader routine.

7.9.2 Impact of Flash Code Image Detection

Flash code image detection has no impact on customers that are using secure parts, since they already have to restructure the Flash image map to accommodate the HAB signature.

Customers that are not using security and do not want to use the external boot option are responsible to ensure that the predefined address in Flash contains a Barker code.

7.10 Boot Image Redundancy on NAND Devices

Table 7-29 shows the address/command scheme implemented by the i.MX25 for accessing various types of NAND devices. When using the redundancy feature, the image code is limited to a size of one block.

Table 7-29. NAND Device Accesses

Device	Number of address cycles	Page Per Block	Notes	Command	Address					Command
					#1	#2	#3	#4	#5	
1/2-Kbyte page, SLC/MLC	4	32	5 LSB bits of Row1 give offset in block & 13 remaining bits choose block number	0x00	Col1	Row1	Row2	Row3	None	None
2KB SLC	5	62	6 LSB bits of Row1 give offset in block & 11 remaining bits choose block number	0x00	Col1	Col2	Row1	Row2	Row3	0x30

Table 7-29. NAND Device Accesses (continued)

Device	Number of address cycles	Page Per Block	Notes	Command	Address					Command
					#1	#2	#3	#4	#5	
2KB MLC	4	128	7 LSB bits of Row1 give offset in block & 9 remaining bits choose block number	0x00	Col1	Col2	Row1	Row2	None	0x30
4KB SLC/MLC	5	128	7 LSB bits of Row1 give offset in block & 12 remaining bits choose block number	0x00	Col1	Col2	Row1	Row2	Row3	0x30

Table 7-30 below provides details on block addressing:

Table 7-30. NAND Device Addressing

Device	Page Copying for 8 KByte Block	Address Cycles
0.5-Kbyte Page SLC/MLC	Page 1	0x00,0x00,0x00,0x00
	Page 2	0x00,0x01,0x00,0x00
	Page 3	0x00,0x02,0x00,0x00
	Page 4	0x00,0x03,0x00,0x00
	Page 5	0x00,0x04,0x00,0x00
	Page 6	0x00,0x05,0x00,0x00
	Page 7	0x00.0x06,0x00,0x00
	Page 8	0x00,0x07,0x00,0x00
	Next Block Accesses (upon Error)	
	Page 1	0x00,0x20,0x00,0x00
	Page 2	0x00,0x21,0x00,0x00
	Page 3	0x00,0x22,0x00,0x00
	Page 4	0x00,0x23,0x00,0x00
	Page 5	0x00,0x24,0x00,0x00
	Page 6	0x00,0x25,0x00,0x00
	Page 7	0x00.0x26,0x00,0x00
	Page 8	0x00,0x27,0x00,0x00

Table 7-30. NAND Device Addressing (continued)

Device	Page Copying for 8 KByte Block	Address Cycles
2-Kbyte Page SLC	Page 1	0x00,0x00,0x00,0x00,0x00
	Page 2	0x00,0x00,0x01,0x00,0x00
	Next Block Accesses (upon Error)	
	Page 1	0x00,0x00,0x40,0x00,0x00
	Page 2	0x00,0x00,0x41,0x00,0x00
2-Kbyte Page MLC	Page 1	—
	Page 2	—
	Next Block Accesses (upon Error)	
	Page 1	—
	Page 2	—
4-Kbyte Page SLC/MLC	Page 1	0x00,0x00,0x00,0x00,0x00
	Next Block Accesses (Upon Error)	
	Page 1	0x00,0x00,0x80,0x00,0x00

7.11 Booting From a NAND Device

7.11.1 Overview

This section explains the eFUSE settings required for booting from a NAND device. The Boot ROM copies only the Initial Program Loader (IPL), authenticates the IPL and jumps to execute the IPL. The IPL is a small program responsible for downloading and executing the rest of the program image. Also a brief description is given on the various components of the IPL (Initial Program Loader).

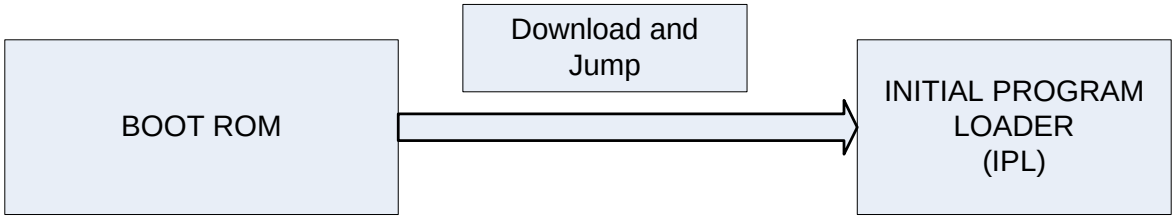


Figure 7-14. ROM Copies The IPL And Jump To Execute It

The various components that needs to be present in the IPL is dependent on the HAB_TYPE[2:0] fuse values. Figure 7-15 below shows the typical example about the IPL image layout.

- DCD DATA: Device configuration data.This block consist of the barker value (0xB17219E9), length of the DCD, and the access-type address-value triplets.

- External Flash Header: The Length of the IPL has to be present in this field. It is expected to be present at the location where DCD data ends.
- APPLN: Customer application Code.
- Flash Header: This Block Contains the Information about the IPL.This is expected to be present at a fixed offset. For NAND, this is at 0x400 byte offset from the base address.
- CSF DATA: Certificate and Command sequence file data
- SRK DATA: Super Root key data.

The required IPL fields depend on the HAB_TYPE[2:0] fuse values. Figure 7-15 shows a typical layout for the IPL image.

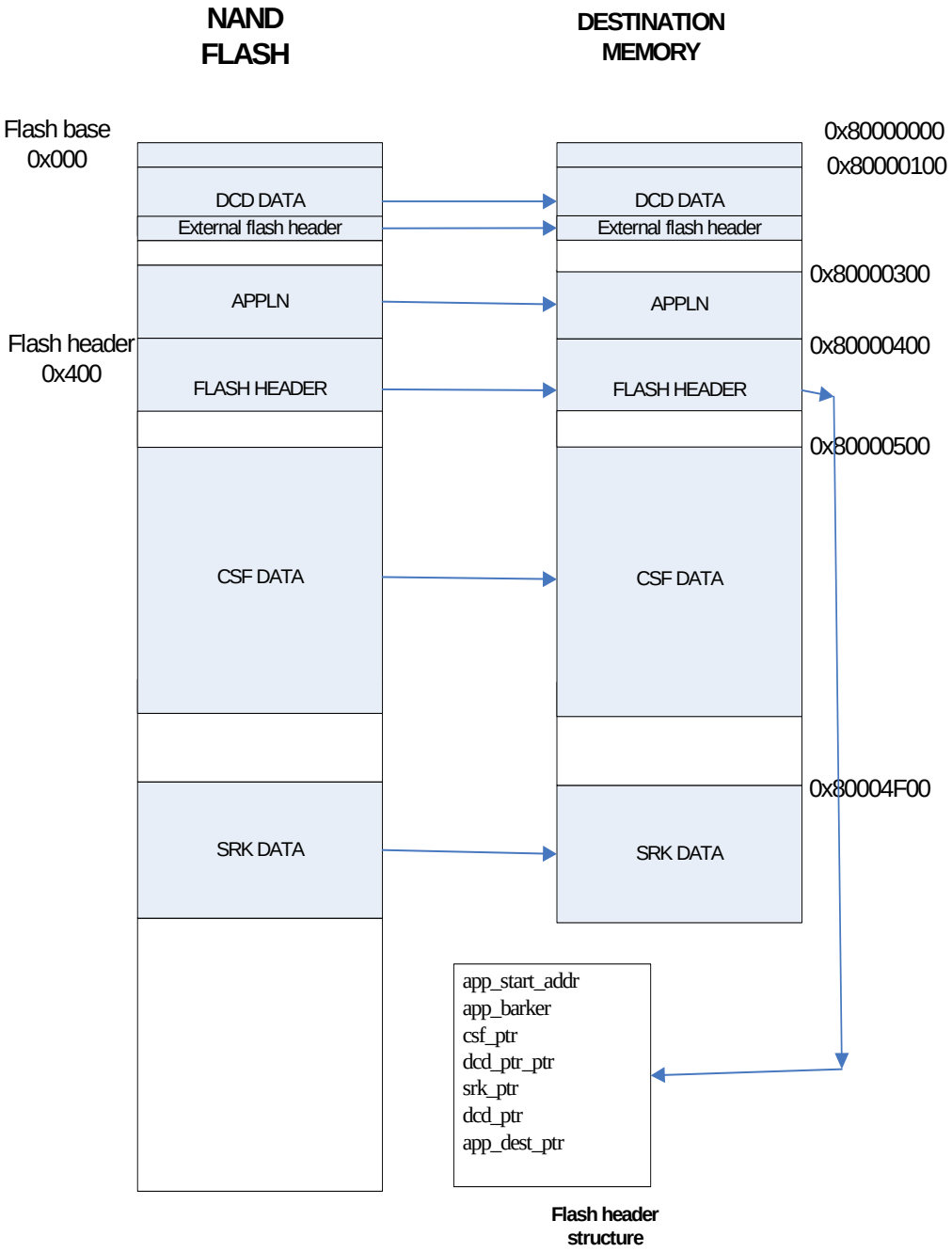


Figure 7-15. IPL Component Layout in Flash and Destination Memory

Table 7-31 shows the significance of the fields shown in Figure 7-15.

Table 7-31. IPL Fields

Field Name	Significance
DCD DATA	Device configuration data.This block consist of the Barker value (0xB172_19E9), length of the DCD, then the access type- address -value triplets.
External flash header	The length of the IPL must be present in this field, at the location where DCD data ends.
APPLN	Customer application code
FLASH HEADER	This block contains the information about the IPL.This is expected to be present at a fixed offset from the base address. For NAND the offset is 0x400, measured in bytes.
CSF DATA	Certificate and command sequence file (CSF) data
SRK DATA	Super Root key data

The IPL should consist of the following components for various HAB TYPE values:

- HAB TYPE = HAB_ENGEERING (0b001)
  - FLASH HEADER
  - APPLN
  - CSF DATA—not mandatory
  - SRK DATA—not mandatory
  - DCD DATA—not mandatory
- Case 2: HAB TYPE = HAB_PRODUCTION (Any value from 0b000–0b111 other than 0b001 or 0b100)
  - FLASH HEADER
  - APPLN
  - CSF DATA
  - SRK DATA
  - DCD DATA—not mandatory
- Case 3: HAB TYPE = HAB_SEC_DISABLED (0b100)
  - FLASH HEADER
  - APPLN

Table 7-32 shows eFUSE settings required for SLC NAND Flash (page size 512 bytes, bus width 8 bits, 3 address cycles, spare area 128 bytes).

Table 7-32. eFUSE Settings for SLC NAND Flash Boot

Field Name	Setting
BT_MEM_CTL[1:0]	01
BT_MEM_TYPE[1:0]	00
BT_BUS_WIDTH[1:0]	00

Table 7-32. eFUSE Settings for SLC NAND Flash Boot (continued)

Field Name	Setting
BT_PAGE_SIZE[1:0]	00
BT_MLC_SEL	0
BT_SPARE_SIZE	0

In addition to the settings shown in Table 7-32, the HAB TYPE[2:0] eFUSEs and BOOT_MODE contacts must be set according to the intended case.

For internal boot mode operation, the IPL for NAND must have the Flash header placed at an offset of 0x400 from the base address.

7.11.2 Flash Header Details

7.11.2.1 HAB_ENGEERING Mode

In internal boot mode with HAB security type HAB_ENGINEERING (BOOT_MODE[1:0] = 00 and HAB TYPE[2:0] = 010), the Flash header needs to have a proper app_start_addr, app_barker (0x0000_00B1), dcd_ptr_ptr, dcd_ptr, or app_dest_ptr. The other pointers in the Flash header can be NULL.

The dcd_ptr_ptr must point to the dcd_ptr in the Flash header structure, which in turn points to the DCD data. The dcd_ptr_ptr field must be set to the FLASH_HEADER start address + 0x14; for example, if the Flash header starts at 0x1000_2400, then the dcd_ptr_ptr setting must be 0x1000_2414).

The ROM copies the initial 4 Kbytes of data from the Flash to the NFC buffer, installs the DCD data, then copies the initial 4 Kbytes of data from the NFC buffer to the destination address pointed to by the app_dest_ptr. In the IPL, the IPL length field is expected at the end of the DCD table. The ROM uses this field to copy the rest of the image from the NAND device to the destination RAM indicated by app_dest_ptr. If the app_dest_ptr is an internal RAM address, then no DCD needs to be installed. In this case, the DCD data can contain only the DCD Barker field and the DCD block length field, whose value is equal to zero.

The format of the srk_ptr can be found from the Flash header section of the System boot chapter.

A typical Flash header structure for Engineering mode is shown in the following example code:

```
struct flash_hdr {
    UINT32 app_start_addr;
    UINT32 app_barker;
    UINT32 csf_ptr;
    UINT32 dcd_ptr_ptr;
    const hab_rsa_public_key *srk_ptr;
    UINT32 dcd_ptr;
    UINT32 app_dest_ptr;
}const start_app = {0x80000300, 0x000000B1, 0x00, (0x80000400+0x14), 0x00,
0x80000100,0x80000000};
```

7.11.2.2 HAB_PRODUCTION Mode

In internal boot mode with HAB security type HAB_PRODUCTION (BOOT_MODE[1:0] = 00 and HAB TYPE[2:0] any value except 0b001 and 0b100),then all Flash header fields must be properly filled. The ROM copies the initial 4 Kbytes data from the Flash to the NFC buffer, installs the DCD data, then copies the initial 4 Kbytes data from the NFC buffer to the destination address indicated by app_dest_ptr.

In the IPL, the IPL length is expected by the ROM code at the end of the DCD table. The ROM uses this field to copy the rest of the image from the NAND device to the RAM destination indicated by app_dest_ptr. Then, the Security check is performed on the downloaded code.

The following regions in the IPL must be signed:

- Flash Header
- Application Code
- DCD data.

These regions must be included in the CSF certificate during the signing.

The SRK hash values must be blown onto the Fuses SRK_HASH[255:0].

The SCC key values must be blown onto the fuses SCC_KEY[0:163]. If unblown, then the SCC will move to an insecure state, and the production test case fails.

The Unique ID UID[0:63] and Customer Code HAB_CUS[7:0] fuses also must be blown to appropriate values that match with those included in the CSF file.

A typical Flash header structure for HAB_PRODUCTION mode is shown in the following code example:

```
struct flash_hdr {
    UINT32 app_start_addr;
    UINT32 app_barker;
    UINT32 csf_ptr;
    UINT32 dcd_ptr_ptr;
    const hab_rsa_public_key *srk_ptr;
    UINT32 dcd_ptr;
    UINT32 app_dest_ptr;
}const start_app = {0x80000300, 0x000000B1, 0x80000500, (0x80000400+0x14)), 0x80004F00,
0x80000100,0x80000000};
```

7.11.2.3 HAB_SEC_DISABLED Mode

In internal boot mode with HAB security disabled (BOOT_MODE[1:0] = 00 and HAB_TYPE[2:0] = 100), then the Flash header only requires the app_start_addr and the app_barker fields (app_barker has value 0x0000_00B1). The ROM code first copies the initial 4 Kbytes of data to the NFC buffer, checks the Barker value, then jumps to app_start_addr.

A typical flash header is as follows:

```
struct flash_hdr {
    UINT32 app_start_addr;
    UINT32 app_barker;
    UINT32 csf_ptr;
    UINT32 dcd_ptr_ptr;
    const hab_rsa_public_key *srk_ptr;
    UINT32 dcd_ptr;
```

```
    UINT32 app_dest_ptr;
}const start_app = {0xBB000000, 0x000000B1, 0x0, 0x0, 0x0, 0x0,0xBB000000};
```

Chapter 8
Power Management

This chapter describes the power savings methodology, power supply requirements, and power-up and power-down sequences for the device.

8.1 Power Domains

Power management for the device is organized into the following power domains:

- Digital logic domain. This includes the core logic domain and I/O domain.
- Analog domain, including the OSC24M, PLLs, USBPHY, fusebox, and touch-screen controller (TSC).

8.1.1 Power-Supply Requirements

See the device datasheet's "DC Operating Characteristics" for power supply requirements.

8.2 Power Saving Methodology

The clock-control module (CCM) supports several power management techniques to reduce active and static power consumption, as follows:

- Active power savings, including clock gating, software controlled dynamic voltage and frequency scaling (DVFS), dynamic process and temperature compensation (DPTC).
- Leakage power savings, including active well bias (AWB)
- Low-power modes, including wait, doze, stop, and sleep modes

8.2.1 Active Power Savings

Active power savings features include the following:

- Dynamic voltage and frequency scaling (DVFS) allows dynamic software control of voltage and frequency scaling. The core clock domain frequency and chip voltage can be changed on-the-fly while all modules (including the core) continue normal operation.
- Dynamic process and temperature compensation (DPTC) enables part-specific voltage reduction in the digital domain. Voltage reduction for each part is determined by device characteristics that vary due to manufacturing process variations and temperature.
- Two levels of clock gating of idle modules:
  - Clock tree roots, as implemented in the clock controller module (CCM)
  - Clock tree leaf nodes, as implemented in the modules themselves

8.2.2 Leakage Power Saving

Active well bias (AWB) is used to control leakage in low-power modes. See the section on well bias support in the CCM chapter.

8.2.3 Power Modes

Table 8-1 shows the device's power modes.

Table 8-1. Power Modes

Power Mode	Conditions
Run	• ARM core and platform are active • Well bias is off • Clocks are on • Modules are active
Wait	• ARM core is in wait-for-interrupt mode (clock off) • Well bias is off • Core PLL is on • USB PLL is off (optional) • OSC24M is on • OSC32K is on • All other modules are off (optional) **Note:** Turning off the USB PLL and other modules makes the device use less power.
Doze	• ARM platform clock is off • Well bias is on • Core PLL is on • USB PLL is off (optional) • OSC24M is on • OSC32K is on • All the other modules are off (optional) **Note:** Turning off the USB PLL and other modules makes the device use less power.
Stop	• All PLLs are off • Well bias is on • OSC24M is off • OSC32K is on • All the other modules are off
Sleep	• All PLLs are off • Well bias is on • OSC24M is off • OSC32K is on • All the other modules are off • Core voltage is reduced to 1 V

8.2.3.1 SDRAM Operation in Low Power Modes

When the SDRAM controller (SDRAMC) is enabled, the external SDRAM operates in distributed-refresh mode or in self-refresh mode. The SDRAM wake-up latency is approximately 20 system clock cycles (HCLK). The SDRAMC can send a command to wake up the SDRAM from self-refresh mode in an SDRAM cycle and SDRAMC cannot access the SDRAM in at least the following two cycles or more,

based on the device type. (based on device type, see the low power mode table in the SDRAMC block guide.)

In wait, doze, and run modes, the power-down timers within the SDRAMC can be enabled to cause the SDRAM to enter power-down mode if no activity is detected. The SDRAMC still controls the refresh, and when necessary temporarily takes the SDRAM out of power-down mode to perform a refresh. In power-down mode, the clock to the SDRAM is gated off and the CKE pin (SDRAM device input) goes low.

When the system enters sleep mode, the SDRAM enters self-refresh mode. Self-refresh mode is exited when the chip exits sleep mode and re-enables the CCM MPLL enable bit (MPEN).

8.3 Power-Up and Power-Down Sequence

Any i.MX25 board design must comply with the power-up and power-down sequence guidelines given in this section to ensure reliable operation of the device. Recommended power-up and power-down sequences are given in the following subsections.

CAUTION

Deviations from the guidelines in this section may result in the following situations:

- Excessive current during power-up phase
- Prevention of the device from booting
- Irreversible damage to the i.MX25 (worst-case scenario)

NOTE

For security applications, the coin battery must be connected during both power-up and power-down sequences to ensure that security keys are not unintentionally erased.

8.3.1 Power-Up Sequence

The following power-up sequence is recommended:

1. Assert power on reset (POR).
2. Turn on digital logic domain and I/O power supplies VDD*n* and NVCC*x*.
3. Turn on all other analog power supplies, including USBPHY1_VDDA_BIAS, USBPHY1_UPLL_VDD, USBPHY1_VDDA, USBPHY2_VDD, OSC24M_VDD, MPPLL_VDD, UPLL_VDD, NVCC_ADC, and FUSEVDD (FUSEVDD is tied to GND if fuses are not being programmed). The minimum time between turning on each power supply is the time it takes for the previous supply to be stable.
4. Negate the POR signal.

NOTE

- The user is advised to connect FUSEVDD to GND except when fuses are being programmed, in order to prevent unintentional blowing of fuses.

- Other power-up sequences may be possible; however, the above sequence has been verified and is recommended.
- There is a 1-ms minimum time between supplies coming up, and a 1-ms minimum time between POR_B assert and deassert.

i.MX25 Reference Manual
Book II

Rev. 2
01/2011

Chapter 9
1-Wire Module (1-Wire)

9.1 Overview

The 1-Wire module provides the communication link to a generic 1-Kbit add-only memory. The module sends or receives one bit at a time with an option for software to manage the data using bytes. The required protocol for accessing the generic 1-Wire device is defined by Maxim-Dallas. The generic 1-Wire device holds battery characteristics information.

Figure 9-1 shows a block diagram of the 1-Wire module.

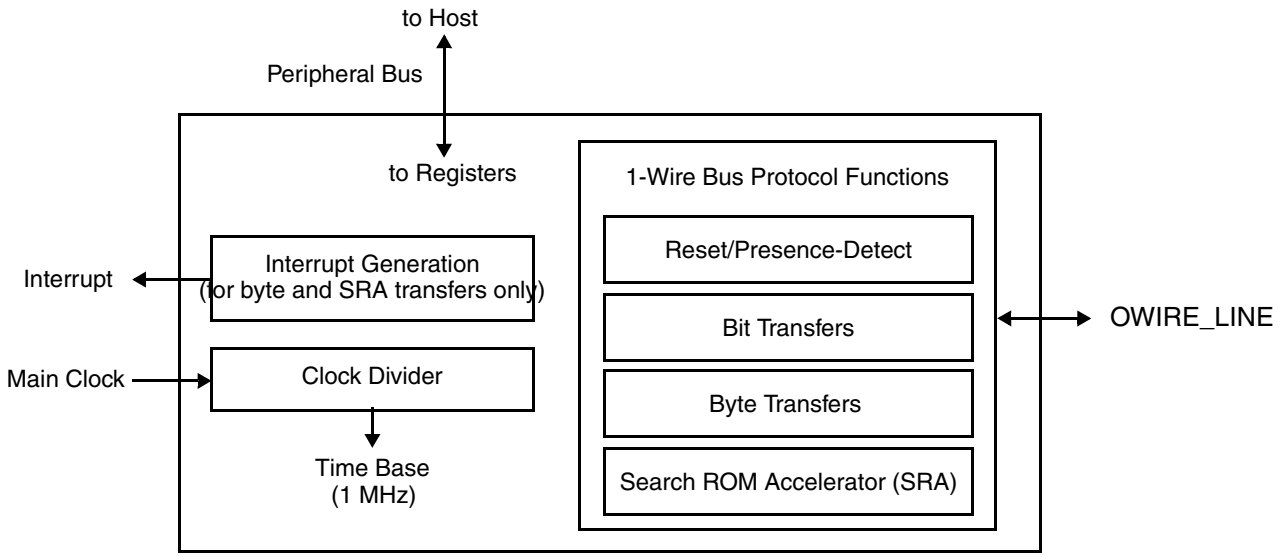


Figure 9-1. 1-Wire Module Block Diagram

9.1.1 Features

The 1-Wire module includes the following features:

- Performs the 1-Wire bus protocol to communicate with an external 1-Wire device.
- Provides a clock divider to a generate a 1-Wire bus reference clock (derived from the main clock provided internally to the module).
- Supports byte transfers with optional interrupts for more efficient programming.
- Provides search ROM accelerator mode to speed the search ROM protocol.

9.1.2 Modes of Operation

The 1-Wire module supports the following operations:

- Normal Operating Modes (See on page 9-8.)
  - Bit or Byte Transfers
  - Reset/Presence-detect Pulse
  - Search ROM Accelerator Mode
- Low Power Mode (See on page 9-11.)

9.2 External Signals

Table 9-1 shows the signal that interfaces with a generic 1-Wire device.

Table 9-1. 1-Wire Module Signal

Signal	I/O	Function
OWIRE_LINE	I/O	1-Wire bus Requires an external pull-up resistor. The recommended resistor value is specified by the generic 1-Wire device used in a given system.

9.3 Memory Map and Register Definition

This section provides the module memory map and detailed descriptions of all registers.

9.3.1 Memory Map

Table 9-2 shows the 1-Wire memory map.

Table 9-2. 1-Wire Memory Map

Base Address Offset	Register	Access	Reset Value	Section/Page
0x0000 (CONTROL)	Control register	R/W	0x0000	9.3.2.1/9-3
0x0002 (TIME_DIVIDER)	Time Divider register	R/W	0x0000	9.3.2.2/9-4
0x0004 (RESET)	Reset register	R/W	0x0000	9.3.2.3/9-4
0x0006 (COMMAND)	Command Register	R/W	0x0000	9.3.2.4/9-5
0x0008 (TX/RX)	Transmit/Receive Register	R/W	0x0000	9.3.2.5/9-5
0x000A (INTERRUPT)	Interrupt Register	R	0x000E	9.3.2.6/9-6
0x000C (INTERRUPT_EN)	Interrupt Enable Register	R/W	0x0000	9.3.2.7/9-7

9.3.2 Register Descriptions

This section provides the detailed descriptions for the registers. All registers are byte-addressable.

9.3.2.1 Control Register (CONTROL)

The control register is used to initiate the reset/presence-detect sequence and bit transfers. The register also provides the presence-detect status and bit-read status.

Figure 9-2 shows the register. Table 9-3 describes the register fields.

Offset 0x0000 (CONTROL) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RPP	PST	WR0	WR1	RDST	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-2. Control Register

Table 9-3. Control Register Field Descriptions

Field	Description
15–8	Reserved
7 RPP	Reset/Presence-detect Pulse. This bit is self-clearing and is cleared after the presence or absence of an external device is determined. See Section 9.4.1.1, “Reset/Presence-Detect Pulse.” When writing: 0 Do nothing. 1 Generate Reset Pulse and sample the bus for the presence pulse from the external device. When reading: 0 Reset pulse complete. 1 Sequence not complete.
6 PST	Presence Status. This bit is valid after the RPP bit is self-cleared. 0 Device is not present. 1 Device is present.
5 WR0	Write 0. This bit is self-clearing and is cleared when the write of the bit is complete. See Section 9.4.1.2.1, “Write-0 Sequence.” When writing: 0 Do nothing. 1 Write a 0 bit to the interface. When reading: 0 Write sequence complete. 1 Sequence not complete.
4 WR1	Write 1 / Read. This bit is self-clearing and is cleared when the write sequence is complete. See Section 9.4.1.2.2, “Write-1/Read Sequence.” When writing: 0 Do nothing 1 Write a 1 bit to the interface and sample the bus. When reading: 0 Sequence complete. 1 Sequence not complete.

Table 9-3. Control Register Field Descriptions (Continued)

Field	Description
3 RDST	Read Status. This bit is valid after the WR1 bit is self cleared. 0 A 0 has been sampled. 1 A 1 has been sampled.
2–0	Reserved

9.3.2.2 Time Divider Register (TIME_DIVIDER)

The time divider register is used for dividing the main clock (ipg_clk) input down to 1 MHz to generate the module's time base.

Figure 9-3 shows the register. Table 9-4 describes the register fields.

Offset 0x0002 (TIME_DIVIDER) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DVDR							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-3. Time Divider Register

Table 9-4. Time Divider Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DVDR	Divider Factor. The internal clock divider uses this field to generate the required time base for the module. See Section 9.4.3, "Clocks." 0x00 1 (default) 0x01 2 - - - - - - 0xFF 256

9.3.2.3 Reset Register (RESET)

The reset register is used to perform a software reset of the 1-Wire module. Figure 9-4 shows the register. Table 9-5 describes the register fields.

Offset 0x0004 (RESET) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RST
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-4. Reset Register

Table 9-5. Reset Register Field Descriptions

Field	Description
15–1	Reserved
0 RST	Software Reset. See Section 9.4.4.2, “Software Reset.” 0 Do not perform a software reset. 1 Initiate a software reset and hold the module in the software-reset state.

9.3.2.4 Command Register (COMMAND)

The 1-Wire module can be configured to run in Search ROM Accelerator mode using the command register. See Section 9.4.1.4, “Search ROM Accelerator Mode.”

Figure 9-5 shows the register. Table 9-6 describes the register fields.

Offset 0x0006 (COMMAND) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SRA	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-5. Command Register

Table 9-6. Command Register Field Descriptions

Field	Description
15–2	Reserved
1 SRA	Search ROM Accelerator. This bit is cleared when the reset-presence-pulse bit CONTROL[RPP] is set. 0 Deactivate the search ROM accelerator. 1 Switch to search ROM accelerator mode.
0	Reserved

9.3.2.5 Transmit/Receive Register (TX/RX)

Data sent and received from the 1-Wire module passes through the Transmit/Receive (TX/RX) register location. The 1-Wire module is double-buffered with separate transmit and receive buffers connected to the TX/RX register. See Section 9.4.1.3, “Byte Transfers.”

Figure 9-6 shows the register. Table 9-7 describes the register fields.

Offset 0x0008 (TX/RX) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DATA							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-6. Transmit/Receive Register

Table 9-7. Transmit/Receive Register Field Descriptions

Field	Description
15–8	Reserved
7-0 DATA	Data byte. When writing: The data byte is written to the Transmit buffer. When reading: A data byte is read from the Receive buffer. The data is valid only when INTERRUPT[RBF] is set.

9.3.2.6 Interrupt Register (INTERRUPT)

Flags for the reset/presence-detect sequence and byte transfer operations are located in the Interrupt Register. These flags can generate an interrupt if the corresponding enable bit is set in the Interrupt Enable Register.

If interrupts are enabled, reading the Interrupt Register clears the interrupt even if all the current flags are not cleared; the interrupt service routine should clear all pending flags during each routine call.

NOTE

When a byte is written to the Transmit/Receive Register, software then waits for a Transmit Shift Register Empty (TSRE) interrupt to occur. When the TSRE flag is set, the Receive Buffer Full (RBF) flag is also set. The RBF flag does not trigger an interrupt, assuming it is disabled. However, software should read the Transmit/Receive Register to clear the RBF flag in order to give a proper status of the pending interrupts.

Figure 9-7 shows the register. Table 9-8 describes the register fields.

Offset 0x000A (INTERRUPT) Access: User read

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	RSRF	RBF	TSRE	TBE	PDR	PD
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Figure 9-7. Interrupt Register

Table 9-8. Interrupt Register Field Descriptions

Field	Description
15–6	Reserved
5 RSRF	Receive shift register full. Hardware automatically clears this flag when data in the receive shift register is transferred to the receive buffer. 0 The receive shift register is empty or currently receiving data. 1 A byte is waiting in the receive shift register to be transferred to the receive buffer.

Table 9-8. Interrupt Register Field Descriptions (Continued)

Field	Description
4 RBF	Receive Buffer Full. This flag is cleared when software reads the byte from the TX/RX register. This flag prevents new data from being shifted into the receive buffer from the receive shift register. 0 No new data 1 A byte is waiting to be read from the TX/RX register.
3 TSRE	Transmit Shift Register Empty. Hardware automatically clears this flag when data in the transmit buffer is transferred to the transmit shift register. 0 Sending data 1 The transmit shift register is empty and is ready to receive the next byte from the Transmit buffer.
2 TBE	Transmit Buffer Empty. This flag is cleared when software writes a byte to the TX/RX register. 0 The Transmit buffer is currently sending data to the transmit shift register. 1 Nothing to transmit
1 PDR	Presence Detect Result. When a presence-detect (PD) interrupt occurs, this bit reflects the result of the presence-detect sequence. Note that this bit does not generate an interrupt. 0 Device found 1 Device not found
0 PD	Presence Detect. After an 1-Wire reset has been issued, this flag is set after the appropriate amount of time for a presence detect pulse to have occurred. This flag is cleared when software reads the interrupt register. 0 A reset/presence-detect sequence has not been issued. 1 Reset/presence-detect sequence has completed. The result is provided in the PDR bit.

9.3.2.7 Interrupt Enable Register (INTERRUPT_EN)

The Interrupt Enable Register allows the system programmer to specify the source of interrupts. During a reset (hardware or software), all bits in this register are cleared, disabling all interrupt sources.

Figure 9-8 shows the register. Table 9-9 describes the register fields.

Offset 0x000C (INTERRUPT_EN) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ERSF	ERBF	ETSE	ETBE	IAS	EPD
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9-8. Interrupt Enable Register

Table 9-9. Interrupt Enable Register Field Descriptions

Field	Description
15–6	Reserved
5 ERSF	Enable receive shift register full interrupt. 0 Disable interrupt. 1 Enable interrupt.

Table 9-9. Interrupt Enable Register Field Descriptions (Continued)

Field	Description
4 ERBF	Enable Receive Buffer Full Interrupt. 0 Disable interrupt. 1 Enable interrupt.
3 ETSE	Enable Transmit Shift Register Empty Interrupt. 0 Disable interrupt. 1 Enable interrupt.
2 ETBE	Enable Transmit Buffer Empty Interrupt. 0 Disable interrupt. 1 Enable interrupt.
1 IAS	Interrupt Trigger Active State. This bit determines the polarity for all interrupts. Note that this bit is not an interrupt-enable bit. 0 Active high interrupt 1 Active low interrupt
0 EPD	Enable Presence Detect. 0 Disable interrupt. 1 Enable interrupt.

9.4 Functional Description

The 1-Wire module interfaces with a generic 1-Kbit add-only memory, through a simple 1-bit bus. Software uses the 1-Wire bus to program and read the 1-Kbyte memory.

The protocol involves first issuing one of four ROM function commands before the EPROM is accessible:

- Read ROM
- Match ROM
- Search ROM
- Skip ROM

Through the 1-Wire bus, the host software interfaces with the generic 1-Wire device and allows the required commands to be issued to control the EPROM of a generic 1-Wire device. The host (through the 1-Wire interface) is the bus master, and the generic 1-Wire device(s) are the slave(s)

9.4.1 Normal Operating Modes

The 1-Wire module supports the following 1-Wire bus protocol functions:

- Reset/Presence-detect pulse using the control register (See Section 9.4.1.1, “Reset/Presence-Detect Pulse”)
- Bit Transfers using the control register (See Section 9.4.1.2, “Bit Transfers”)
- Byte Transfers using the TX/RX register (See Section 9.4.1.3, “Byte Transfers”)
- Search ROM Accelerator Mode using the Command register and the TX/RX register (See Section 9.4.1.4, “Search ROM Accelerator Mode”)

9.4.1.1 Reset/Presence-Detect Pulse

The 1-Wire module provides for an automated initialization sequence for the 1-Wire bus. Software initiates the initialization sequence by setting CONTROL[RPP]. The automated initialization sequence is as follows:

1. Generate a reset pulse.
2. Listen for a response from an external device by sampling for the 1-Wire device presence bit.
3. After an amount of time determined by the 1-Wire standard, latch the presence bit (true or false) in CONTROL[PST].

If an external device is detected (PST = 1), software can begin communications on the 1-Wire bus.

The presence pulse is used by the 1-Wire to determine if at least one generic 1-Wire device is connected. Software determines if more than one generic 1-Wire device exists; see Section 9.4.1.1, "Reset/Presence-Detect Pulse."

9.4.1.2 Bit Transfers

After the initialization sequence (see Section 9.4.1.1, "Reset/Presence-Detect Pulse"), software can write and read one bit at a time using the control register.

9.4.1.2.1 Write-0 Sequence

The Write-0 sequence writes a zero bit to the generic 1-Wire device. Setting the CONTROL[WR0] bit initiates the Write-0 pulse sequence. Once the write is complete, the WR0 bit is automatically cleared.

9.4.1.2.2 Write-1/Read Sequence

The Write-1 sequence writes a one bit to the generic 1-Wire device. Setting the CONTROL[WR1] bit initiates the Write-1 pulse sequence. Once the write is complete, the WR1 bit is automatically cleared.

Because the Write-1 and Read timings are identical, this sequence also reads a bit from the bus. The sampled value is stored in the read status bit CONTROL[RDST] and is valid after the WR1 bit is self-cleared.

9.4.1.3 Byte Transfers

After the initialization sequence (see Section 9.4.1.1, "Reset/Presence-Detect Pulse"), software can transfer a byte at a time using the TX/RX register. Writing to the register connects to the Transmit buffer; reading to the register connects to the Receive buffer. See Figure 9-9.

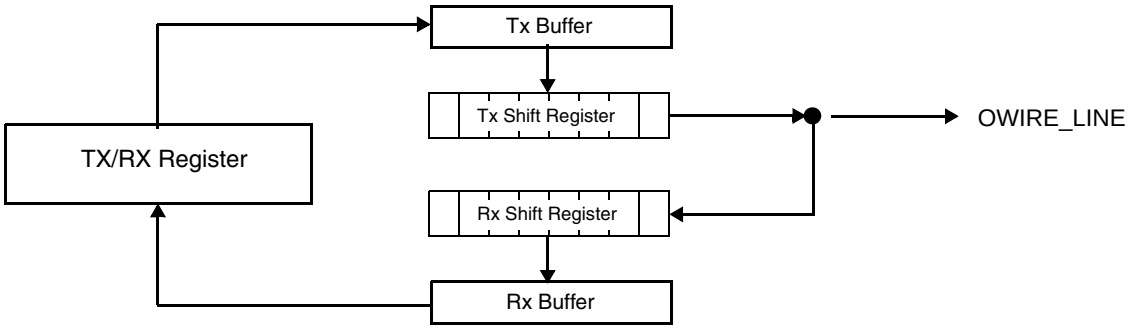


Figure 9-9. Byte Transfers

The Transmit buffer connects to an internal Transmit Shift Register where data is shifted serially onto the bus LSB first. Similarly, the Receive buffer connects to an internal Receive Shift Register where data is sampled serially from the bus.

Software can read a byte from the generic 1-Wire device as follows:

1. Write 0xFF to the Transmit/Receive register location (connected to the Transmit buffer).
2. Wait for the receive-buffer-full (INTERRUPT[RBF]) interrupt (or poll the flag bit directly if the interrupt is disabled). During this time, the hardware is writing ones on the bus while sampling the wired-AND of the data from the device. The read data is shifted into the Receive Shift Register. When a byte is collected in the Receive Shift Register, the data is transferred to the Receive buffer, and the RBF flag is set.
3. Read from the Transmit/Receive register location (connected to the Receive buffer) upon receiving the RBF interrupt.

If the Receive buffer is full, new data is not shifted from the Receive Shift Register until the current data is read. To prevent the loss of data, software must read the TX/RX register to clear the receive-buffer-full flag (INTERRUPT[RBF]). This allows the Receive Shift Register to shift new data into the Receive buffer.

9.4.1.4 Search ROM Accelerator Mode

In search ROM accelerator mode, the 1-Wire module relieves software from having to perform single-bit operations on the bus and helps to determine whether more than one generic 1-Wire device exists.

The host transmits the 16-byte search value based on the last ROM value found. These 16 bytes are 0x00 for the first run. The 16 bytes returned contain the new ROM code and are also used to generate the next 16 bytes to transmit. This process is repeated until serial numbers duplicate to find all devices.

The 1-Wire module enters search ROM accelerator mode when COMMAND[SRA] is set. This protocol specifies that the bus master read two bits (a bit and its complement), then writes a bit to specify which devices should remain on the bus for further processing. This mode requires that a reset followed by the search ROM command (0xF0) has already been issued on the 1-Wire bus.

The 1-Wire module automatically exits search ROM accelerator mode if the 1-Wire bus is re-initialized; see Section 9.4.1.1, “Reset/Presence-Detect Pulse.”

9.4.2 Low Power Mode

The 1-Wire module automatically goes into low-power mode whenever it is not communicating with a generic 1-Wire device. The main clock is gated off in low-power mode.

As soon as software writes to any register, the 1-Wire module exits low-power mode.

9.4.3 Clocks

The 1-Wire module takes a main clock as a module input and passes it through a clock divider. (See the block diagram in Figure 9-1.) Software must program the divider factor to generate a 1-MHz clock that is used as an internal time base for the module, as given by Equation 9-1.

$$\text{time\_base} = \text{main\_clock} \div (\text{TIME\_DIVIDER[DVDR]} + 1) \qquad \textit{Eqn. 9-1}$$

For example, if the main clock frequency is 30 MHz, the value to write to the divider register is 29. If the main clock input frequency is not an integer, the programmer must ensure the time base frequency is within the range given by Equation 9-2.

$$0.98\ \text{MHz} \le \text{time\_base} \le 1.02\ \text{MHz} \qquad \textit{Eqn. 9-2}$$

NOTE

A main clock frequency below 10 MHz causes improper function of the module.

9.4.4 Reset

The 1-Wire module supports two levels of reset: hardware and software.

9.4.4.1 Hardware Reset

Whenever a device reset occurs, a hard reset is performed on the 1-Wire module, clearing all values written to all registers.

9.4.4.2 Software Reset

Software initiates a software reset by setting the reset bit RESET[RST]. A software reset clears all data written to the registers except for the command and interrupt registers (COMMAND, INTERRUPT).

Note that the reset register (RESET) itself is not cleared during a software reset. Software must clear the RST bit to release the software reset.

9.4.5 Interrupts

The 1-Wire generates interrupts through the programming of the interrupt enable register; see Section 9.3.2.7, “Interrupt Enable Register (INTERRUPT_EN).” The 1-Wire can generate interrupts under the following conditions:

- Receive shift register or buffer full

- Transmit shift register or buffer empty
- Presence detect

Once any of these conditions are met, the Interrupt register (see Section 9.3.2.6, “Interrupt Register (INTERRUPT)”) sets the corresponding bit and generates an interrupt if enabled in the interrupt enable register. The IAS bit within the interrupt enable register determines if the interrupt generated is active low, or active high. By default all interrupts are active high, and software should not modify IAS.

Chapter 10
ARM9 Platform Overview

10.1 Introduction

The ARM9 Platform consists of the ARM926EJ-S processor, ETM9, ETB9, a 5x5 Multi-Layer AHB crossbar switch (MAX), and two internal AHB complexes. The instruction bus of the ARM926EJ-S processor (I-AHB) is connected directly to MAX Master Port 0. The data bus of the ARM926EJ-S processor (D-AHB) is connected directly to MAX Master Port 1. Three alternate bus master interfaces are connected to MAX Master Ports 2, 3 and 4. The five slave ports of the MAX are AHB-Lite-compatible buses. Slave Ports 0, 1 and 2 are designated for external platform accesses. Slave ports 3 and 4 are internal to the platform. Slave port 3 has three slave connected to it: an AIPS(A) peripheral interface gasket, the ROM controller and the ASIC interrupt module. Slave port 4 has two slaves connected to it: an AIPS(B) peripheral interface gasket and the ROMPATCH configuration registers.

The ROMPATCH module supports external boot mechanism in addition to patching of ROM.

Internal MAXMUX modules reside on MAX slave ports 3 and 4 and provide address decoding, read data muxing, bus watchdog timers and other miscellaneous functions for these AHB systems within the platform. A clock control module (CLKCTL) is provided to support a power conscious design methodology as well as implementation of several clock synchronization circuits and registers for use both inside and outside the platform.

An ARM Abort Processing Engine (AAPE) sits on the ARM926EJ-S Instruction AHB and Data AHB to ensure that aborted accesses on those busses are not filtered out by the instruction cache or data cache but are in fact recognized by the core.

A block diagram of the ARM9 Platform is shown in Figure 10-1.

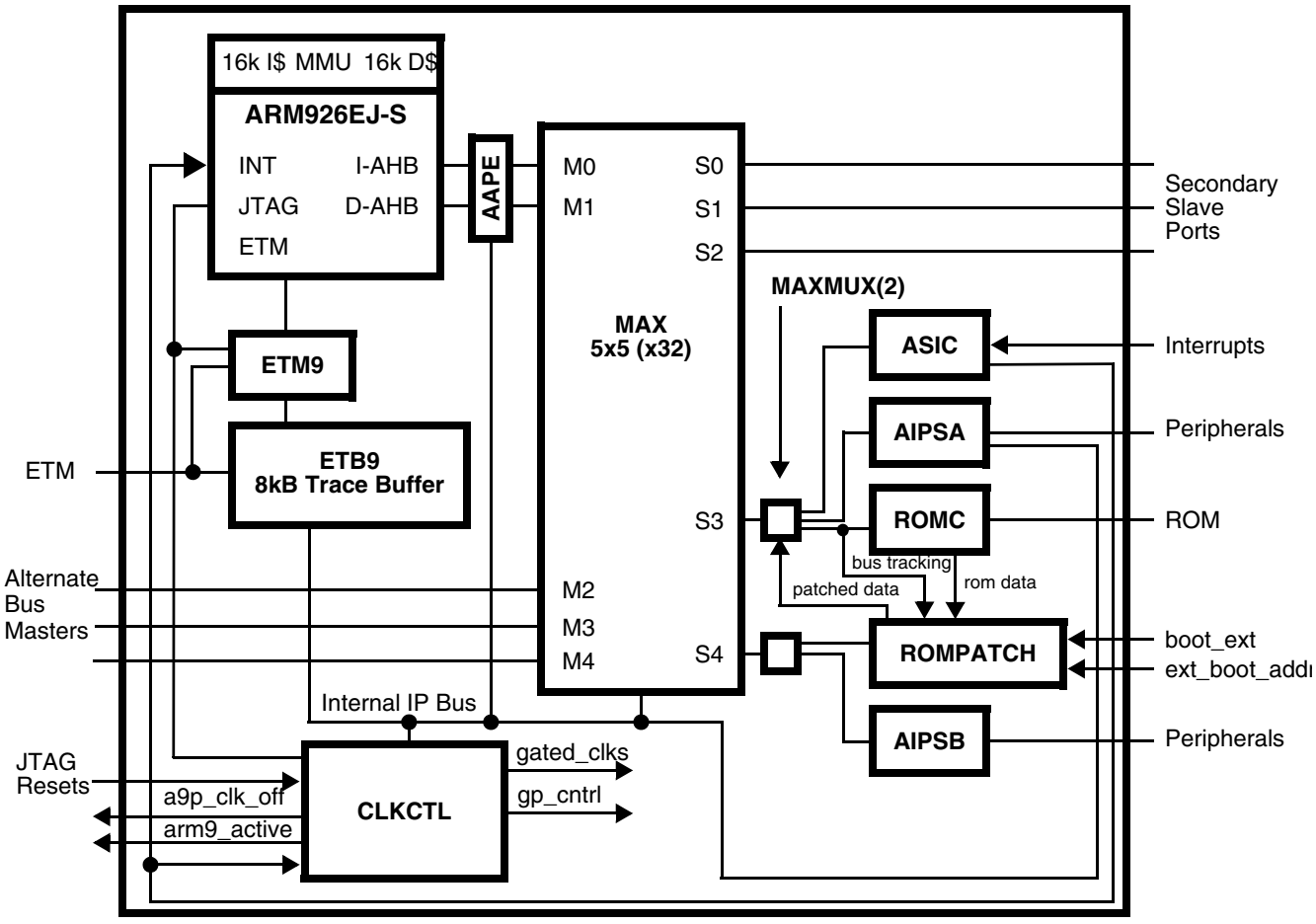


Figure 10-1. ARM9 Platform Block Diagram

10.2 ARM9 Platform Submodules

The submodules of the platform are listed below along with short functional descriptions.

10.2.1 ARM926EJ-S Processor

The ARM926EJ-S (ARM926) is a member of the ARM9 family of general-purpose microprocessors targeted at multi-tasking applications. The ARM926 supports the 32-bit ARM and 16-bit THUMB instructions sets. The ARM926 includes features for efficient execution of Java byte codes. A JTAG port is provided to support the ARM Debug Architecture, along with associated signals to support the ETM9 real-time trace module. The ARM926EJ-S is a Harvard cached architecture including an ARM9EJ-S integer core, a Memory Management Unit (MMU), separate instruction and data AMBA AHB interfaces, and separate instruction and data caches.

The ARM926EJ-S processor is a fully synthesizable macrocell, with a configurable memory system. Both instruction and data caches are 16 Kbyte. The cache is virtually accessed and virtually tagged. The data cache has physical tags as well. The MMU provides virtual memory facilities which are required to support various platform operating systems such as Symbian OS, Windows CE and Linux. The MMU contains eight fully associative TLB entries for lockdown and 64 set associative entries. See the ARM926EJ-S Technical Reference Manual for more information.

10.2.2 ARM9 Embedded Trace Macrocell & Embedded Trace Buffer

The ARM9 platform includes an ARM9 Embedded Trace Macrocell (ETM9) and Embedded Trace Buffer (ETB) supporting real-time instruction and data tracing. The ETM9/ETB external interface may run at the ARM926EJ-S clock frequency or at half the ARM926EJ-S clock frequency. The Embedded Trace Buffer is sized at 2048x32 and can be used as general scratch pad memory when not being used for real-time tracing. This scratch pad memory is accessible using the AIPSA, on platform slot 4. The ETB registers can be accessed using the AIPSA, on platform slot 3. See the ETM9 and ETB technical reference manuals for more information.

10.2.3 5x5 Multi-Layer AHB Crossbar Switch (MAX)

The ARM926EJ-S processor instruction and data buses and all alternate bus master interfaces arbitrate for resources using a 5x5 Multi-Layer AHB Crossbar Switch (MAX). There are five (M0 - M4) fully functional master ports and five (S0 - S4) fully functional slave ports. The MAX is unidirectional. All master and slave ports are AHB-Lite-compatible. See Section 10.9.1, “Definition of AHB-Lite,” for an explanation of AHB-Lite.

The design of the crossbar switch allows for concurrent transactions to proceed from any master port to any slave port. That is, it is possible for five slave ports to be active at the same time as a result of five independent master requests. If a particular slave port is simultaneously requested by more than one master port, arbitration logic exists inside the crossbar to allow the higher priority master port to be granted the bus, while stalling the other requestor(s) until that transaction has completed. The slave port arbitration schemes supported are fixed, programmable fixed, programmable default input port parking, and a round robin arbitration scheme.

The Crossbar Switch also monitors the **max_halt_request** input which request a bus grant from all five slave ports. The priority of the **max_halt_request** is programmable and defaults to the highest. Upon receiving bus grants for all five output ports, the **max_halted** output asserts. At this point, the clock control module can turn off **hclk** and be assured there are no outstanding AHB transactions in progress. Once the CCM is granted a port, no other master receives a grant on that port until the CCM bus request (**max_halt_request**) negates.

Brief descriptions below provide more detail on the MAX. For complete functionality, see the ARM9 Platform “Multi-Layer AHB Crossbar Switch” Module (MAX) specification.

10.2.3.1 MAX Configuration Registers

The Crossbar Switch has configuration and control registers accessible using the IPBus (on platform slot 1 of the AIPSA). Programmable registers exist to control arbitration schemes, bus parking, as well as other

crossbar bus switch functionality. Alternate master priority registers exists within the MAX module for each slave output port. The alternate priority register can be selected for use by the internal arbitration logic.

A write-block sticky bit is implemented for those applications where it is desirable to prevent changes to the MAX registers after boot. See the MAX module design specification for more details.

10.2.3.2 Master Ports

Master Port 0 of the MAX is connected directly to the ARM926EJ-S I-AHB. Master Port 1 of the MAX is connected directly to the ARM926EJ-S D-AHB. The other three master ports exit the platform and are connected to external alternate bus masters. Multiple external masters may be attached to a single alternate bus master port using use of an external arbiter.

Master Port priorities are determined by the MAX priority register bit settings. See the MAX module design specification for more details.

10.2.3.3 Slave Ports

Slave ports 0 through 4 are identical AHB-Lite buses. Slave ports 0, 1 and 2 are secondary AHB slave ports, are accessible off platform, and have no on platform slaves connected to them. Slave ports 3 and 4 are internal to the platform only.

10.2.3.4 Debug Support

In addition to the JTAG, ETM9 and ETB9 interfaces, several internal ARM926EJ-S signals have been brought out of the platform. These signals, along with alternate bus master and secondary AHB signals already available on the top-level of the platform, enable the user to gain insight into the operation of the processor and the MAX.

10.2.4 ARM Abort Processing Engine (AAPE)

The ARM Abort Processing Engine (AAPE) ensures that cacheable or bufferable accesses that are aborted (receive an AHB ERROR response) on the ARM926EJ-S I-AHB or D-AHB are recognized by the ARM926EJ-S and not lost in the instruction cache or data cache. The registers of the AAPE can be accessed using AIPS A on-platform slot 5.

Cacheable or bufferable accesses on the ARM926EJ-S D-AHB that are aborted result in the AAPE zeroing out the data on the D-AHB and sending an abort signal directly to the ARM926EJ-S. Cacheable or bufferable accesses on the ARM926EJ-S I-AHB that are aborted result in the AAPE overriding the I-AHB data with a software interrupt appropriate for the current mode of operation (ARM, THUMB or JAVA).

See the AAPE specification for more details.

10.2.5 ARM Simple Interrupt Controller (ASIC)

NOTE

The ARM simple interrupt controller (ASIC) module is the same one used on the ARM9 platform (AVIC) with the vectoring feature removed for use with an ARM9. This is Freescale IP.

The ARM9 platform interrupt controller is called the ASIC and is accessible to the ARM926EJ-S only using MAX slave port 3. It generates normal and fast interrupts to the ARM926EJ-S processor.

See the ASIC chapter for more details on ASIC operation.

10.2.6 ROM Controller and BIST Engine (ROMC)

The **rom_connect** input on the ARM9 Platform must be tied high if ROM exists on the ROMC interface. The ROMC module supports a minimum of 1 Kbyte of ROM and a maximum of 4 Mbyte. Non power-of-two sizes between 1 Kbyte and 4 Mbyte are supported by strapping the **rom_size[11:0]** inputs, which correspond to slave port 3 AHB **haddr[21:10]**. The **rom_wait** input should be tied high at integration time if a wait state is required to make read data timing on ROM accesses. A configurable BIST engine is provided.

10.2.6.1 ROM Addressing

The first 16 Kbyte of ROM is always mapped starting at **haddr[31:0]**=0x0000_0000. Any ROM larger than 16 Kbyte has the remainder of its space mapped starting at **haddr[31:0]**=0x0040_4000. Any ROM size smaller than 16 Kbyte, is aliased throughout the 16 Kbyte region. Accesses to the “hole” between these two regions are terminated with an ERROR response by the ROMC.

10.2.7 AHB <-> IP-Bus Interface (AIPS)

There are two AIPS modules instantiated in the ARM9 platform (AIPS A and AIPS B). The AIPS module design supports many configuration options so that it may be reused across multiple systems. This section begins with a short description of the generic AIPS module, followed by specific implementation details of the two AIPS instantiations used in the ARM9 platform.

10.2.7.1 Brief Description of the Generic AIPS Module

The AIPS module interfaces an AHB-Lite 2.0 bus to the IP-Bus in order to foster reuse of SRS-compatible peripherals. Each AIPS module requires a minimum of 2 **hclk** clocks for a read, and 3 **hclk** clocks for a write. An AIPS module supports a maximum of 32 on-platform peripherals, 32 off-platform peripherals, and two global external module enables. Each AIPS module occupies a total of 64 Mbytes of address space. Each standard peripheral connected to an AIPS module consumes 16 Kbyte of the memory map (**ipsa_module_en[31:0]**). The two global external IPS module enables are provided to support a 63-Mbyte address space (**ips_module_en_glbl[1]** consumes 32 Mbytes; **ips_module_en_glbl[0]** consumes 31 Mbyte) for peripherals needing a larger slice of the memory map. Each AIPS module contains programmable security access control registers for all downstream peripherals.

10.2.7.2 AIPS Configuration on the ARM9 Platform

This section discusses the particular configuration of both AIPS modules (AIPS A and AIPS B) used within the ARM9 platform. Some of the configuration options are controlled during synthesis, and some are handled with tie-offs in an AIPS wrapper.

The attributes common to both AIPS modules used in the ARM9 platform are as follows:

- 32-bit data path.
- Write buffering is disabled.
- Memory map option #1 is used.
- The AIPS_DLY_CYCLE parameter is disabled.
- The **aips_byte_config[0]** input is tied high and the **aips_byte_config[1]** input is tied to the **cfg_bigend** output of the ARM926EJ-S so both Big Endian and Little Endian modes are supported.
- The only “sideband” signal implemented in each AIPS module is the **ips*_cacheable** output.
- Only **hresp[0]** is driven: **hresp[2:1]** are tied off within the wrapper.
- The AIPS modules do not support 64-bit data accesses.
- Although the AIPS modules on the ARM9 platform are designed for 32-bit operation on the IP-Bus side, they support connection of 8-bit and 16-bit peripherals. However, the AIPS does not support accesses of a larger size than the peripheral maximum width. This means a 32-bit access to a 16-bit or 8-bit peripheral as well as a 16-bit access to an 8-bit peripheral are not supported and the AIPS responds with **hresp** = ERROR to any such access attempt.
- The **aips_rstcfg[63:0]** inputs control the reset configuration of the Master Privilege Register (MPR) within the AIPS module. These inputs have been tied-off in the wrapper to configure the AIPS to allow the ARM926EJ-S processor to be the default “trusted” master out of reset. All other masters are non-trusted. Note the ARM926EJ-S can reprogram the MPR at any time after reset.

Table 10-1 shows the default (out of reset) privilege configuration of the ARM926EJ-S processor (trusted). Table 10-2 shows the default (out of reset) privilege configuration for all bus masters other than the ARM926EJ-S processor (non-trusted).

Table 10-1. AIPS MPR Reset Configuration for Trusted Master (ARM9)

Default (out of reset) ARM926EJ-S Trusted Master Privileges (hmaster = 0x1)
Master Buffered Writes: No buffered Writes
Master Trusted for Reads: Yes - Trusted
Master Trusted for Writes: Yes - Trusted
Master Privilege Level: Supervisor accesses are NOT downgraded to User accesses

Table 10-2. AIPS MPR Configuration for Non-Trusted Masters

Default (out of reset) Non-Trusted Master Privileges (hmaster != 0x1)
Master Buffered Writes: No buffered Writes
Master Trusted for Reads: No - Not Trusted

Table 10-2. AIPS MPR Configuration for Non-Trusted Masters (continued)

Default (out of reset) Non-Trusted Master Privileges (hmaster != 0x1)
Master Trusted for Writes: No - Not Trusted
Master Privilege Level: Supervisor accesses ARE downgraded to User accesses

10.2.7.3 AIPS A Peripheral Support

AIPS A shares MAX slave port 3 with the ROM controller (ROMC) and the ARM simple interrupt controller (ASIC). AIPS A is configured to support 16 standard (16 Kbyte) external peripherals, as well as the two global external module enables (31 Mbyte and 32 Mbyte). AIPS A also supports 5 on-platform peripherals as shown in Table 10-3.

Table 10-3. On-platform IP-Bus Peripherals (AIPS "A")

On-platform Peripheral	AIPS "A" On-platform Slot
MAX Configuration Registers	1
CLKCTL (General Purpose Registers)	2
ETB Registers	3
ETB Memory	4
AAPE Registers	5

10.2.7.4 AIPS B Peripheral Support

AIPS B shares MAX slave port 4 with the ROMPATCH. AIPS B is configured to support 32 standard (16 Kbyte) external peripherals, as well as the two global external module enables (31 Mbyte and 32 Mbyte). AIPS B does not support any on-platform peripherals.

10.2.7.5 Peripheral Summary

Table 10-4 is a summary of the total ARM9 platform internal (on-platform) and external (off-platform) peripherals which are supported.

Table 10-4. Summary of ARM9 Platform Peripheral Support

AIPS Module	Number of On-Platform Peripherals	Number of Off-Platform Peripherals	Number of Off-platform Global Module Enables (31MB, 32MB)
AIPS A	5	16	1, 1
AIPS B	None	32	1, 1
Total	5	48	2, 2

10.2.7.6 Register Reset Summary

Table 10-5 is a summary of the reset state of the registers for AIPSA.

Table 10-5. Summary of ARM9 Platform AIPSA Register Reset State

Register	Reset Value
Master Privilege Register (0–7)	0x07000000
Master Privilege Register (8–15)	0x00000000
Peripheral Access Control Register (0–7)	0x54444400
Peripheral Access Control Register (8–15)	0x00000000
Peripheral Access Control Register (16–23)	0x00000000
Peripheral Access Control Register (24–31)	0x00000000
Off Platform Peripheral Access Control Register (0–7)	0x44444444
Off Platform Peripheral Access Control Register (8–15)	0x44444444
Off Platform Peripheral Access Control Register (16–23)	0x00000000
Off Platform Peripheral Access Control Register (24–31)	0x00000000
Off Platform Peripheral Access Control Register (32–33)	0x44000000

Table 10-6 is a summary of the reset state of the registers for AIPSB.

Table 10-6. Summary of ARM9 Platform AIPSB Register Reset State

Register	Reset Value
Master Privilege Register (0–7)	0x07000000
Master Privilege Register (8–15)	0x00000000
Peripheral Access Control Register (0–7)	0x50000000
Peripheral Access Control Register (8–15)	0x00000000
Peripheral Access Control Register (16–23)	0x00000000
Peripheral Access Control Register (24–31)	0x00000000
Off Platform Peripheral Access Control Register (0–7)	0x44444444
Off Platform Peripheral Access Control Register (8–15)	0x44444444
Off Platform Peripheral Access Control Register (16–23)	0x44444444
Off Platform Peripheral Access Control Register (24–31)	0x44444444
Off Platform Peripheral Access Control Register (32–33)	0x44000000

10.2.8 MAX Internal Slave Port Muxes (MAXMUX)

There are two instantiations of the MAXMUX module in the ARM9 platform. The first instantiation resides on MAX slave port 3 and handles the AHB infrastructure for the AIPS A, ASIC and ROMC

modules. The second instantiation resides on MAX slave port 4 and handles the AHB infrastructure for the AIPS B and ROMPATCH modules.

10.2.8.1 Peripheral Bus Timeout Monitors

The ARM9 platform MAX module returns an AHB **hresp** = ERROR termination status on attempted accesses to undefined regions of memory. Likewise, the AIPS module returns and AHB **hresp** = ERROR termination status on attempted accesses to unpopulated regions of the peripheral space. The memory controller (ROMC) is, by design, either zero or one wait state responders. The ASIC and ROMPATCH are also single wait state responders (CHECK). However, the danger still exists that peripherals residing on the AIPS module's (2) IP-Buses could hang and not properly terminate an access. Therefore, the ARM9 platform supports AHB bus timeout monitors on each internal MAX slave port. The timeout monitors are implemented in each instantiation of the MAXMUX module.

The timeout interval for both modules are set by the **bmon_timeout[1:0]** inputs which are statically tied off at integration time. The timeout monitors both share a single enable bit in the general purpose register of the CLKCTL module. When the timeout monitors are enabled, and AHB access to either internal MAX slave port that does not terminate within the time it takes for the bus monitor to reach zero are terminated by the timeout monitor in the MAXMUX module. The timeout monitor terminates a "hung" access by forcing an **hresp** = ERROR and **hready** termination. The bus timeout clock counts for the **bmon_timeout[1:0]** encoding is shown in Table 10-7.

Table 10-7. Bus Monitor Timeout Interval

bmon_timeout[1:0]	Timeout Interval
00	31 clocks
01	127 clocks
10	511 clocks
11	2047 clocks

NOTE

The ARM9 platform does not include timeout monitors for MAX slave ports 0, 1 and 2.

10.2.9 ROM Patch (ROMPATCH)

The ROM patch module (ROMPATCH) is used to patch errant ROM code. The registers of the ROMPATCH are programmed by the ARM926EJ-S using MAX slave port 4. The ROMPATCH only patches accesses to the ROMC. The ROMPATCH module can be used to patch source code or data tables. The module supports 16 patches.

10.2.9.1 External Boot

An external boot feature exists in the ROMPATCH module which allows patching of the reset vector fetch (address = 0x0000_0000) if the boot_int signal is negated. This mechanism causes the ARM926EJ-S to, in effect, fetch the reset vector from the address indicated by the **ext_boot_addr[31:2]** inputs.

10.2.10 Clock Control Module (CLKCTL)

The Clock Control Module (CLKCTL) performs module level clock gating and ARM926EJ-S JTAG synchronization. Additionally, the CLKCTL module is a 32-bit peripheral connected to AIPS A on-platform IP-Bus slot 2. The IP-Bus interface allows access to a general purpose control register and a read only TAPID register. Four of the general purpose control outputs, **a9p_gp_cntrl[3:0]**, are ported to the top-level of the ARM9 platform. The upper 12 bits are used internally and are not brought to the top-level. Additionally a TAPID version, **tapid_ver[3:0]**, is brought in from the top-level.

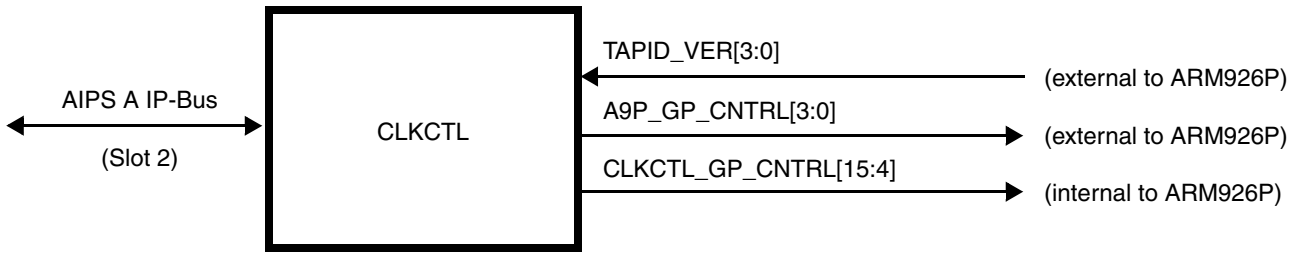


Figure 10-2. CLKCTL General Purpose Register Interface

10.2.10.1 CLKCTL Registers

The CLKCTL module is connected to AIPS A IP-Bus on-platform slot 2 as a 32-bit peripheral. The registers residing in the CLKCTL module are shown in Table 10-8.

Table 10-8. CLKCTL Registers

AIPSA_ADDR[4:2]	Register Name	Register Definition	Width
3'b000	GP_CNTRL	General Purpose Control Register	[15:0]
3'b001	GP_SER	General Purpose Set Enable Register	[15:0]
3'b010	GP_CER	General Purpose Clear Enable Register	[15:0]
3'b100	TAPID	TAPID Register (Read Only)	[31:0]

10.2.10.1.1 GP_CNTRL, GP_SER and GP_CER Registers

The GP_CNTRL register is write/read. All bits in the CP_CNTRL register are cleared on reset. All bits of the GP_CNTRL register may be written concurrently by writing directly to the GP_CNTRL register. To prevent the need to perform a read-modify write operation when the user wishes to set or clear a single bit in the GP_CNTRL register the GP_SER and GP_CER registers have been added. A single bit may be set in the GP_CNRL register by writing a one to the relative bit location in the Set Enable Register (GP_SER). Similarly, a single bit may be cleared by writing a one to the relative bit location in the Clear Enable Register (GP_CER). Reading the GP_SER or GP_CER registers returns all zeros.

Table 10-9 details the functions of all the bits in the CLKCTL module General Purpose Control Register.

Table 10-9. CLKCTL General Purpose Control Bit (GP_CNTRL) Usage

GP_CNTRL Register Bit	Reset Value	Description
[3:0]	4'b0000	Driven off-platform for use at the SoC level.
[4]	1'b0	When set, enables the peripheral bus timeout monitors. See Section 10.2.8.1, "Peripheral Bus Timeout Monitors," for more information.
[5]	1'b0	Reserved for future use.
[6]	1'b0	When set, this bit enables clocks to the ETB. Clocks to the ETB are automatically issued when a debugger is connected (dbgen asserted). However, if the ETB memory is to be used as general purpose memory, this bit must be set. This bit should not be set in a non-debug mode environment if the ETB memory is not going to be used.
[7]	1'b0	When set this bit allows a9p_clk_off to assert even when debug activity is present.
[10:8]	3'b000	Reserved for future use.
[11]	1'b0	When set, this bit enables the MAX Slave Port Sharing Widget (SPSW).
[15:12]	4'b0000	Reserved for future use.

10.2.10.1.2 TAPID Register

The TAPID register is provided for software to determine the version of the platform. These bits correspond to the static state of the **tapid[31:0]** signals which include the **tapid_ver[3:0]** platform inputs. See Section 10.4, "JTAG ID Register," for more details.

10.2.10.2 JTAG Synchronization

The CLKCTL module synchronizes the external JTAG interface to the ARM926EJ-S clock (clk). The inputs and outputs of the synchronization circuit are connected to the JTAG interface on the ARM926EJ-S, ETM9 and ETB9 modules.

10.2.11 Just Another Module (JAM)

The JAM (Just Another Module) implements miscellaneous logic with the platform. Functionality within the JAM includes muxing of BIST outputs from the instruction cache memory, data cache memory, MMU memory, ETB memory and ROMC memory.

10.2.12 Test Wrapper

The ARM9 Platform test architecture is composed primarily of two functions: scan and BIST. The test module (ARM926P_TEST) includes a test control unit which decodes primary test mode input signals and places the platform into various test modes including scan, ac path testing, BIST, and safe state. These test modes support the ability to test a deeply embedded platform.

10.3 ARM9 Platform Hierarchy

The first two levels of the ARM9 Platform design hierarchy are shown in Figure 10-3.

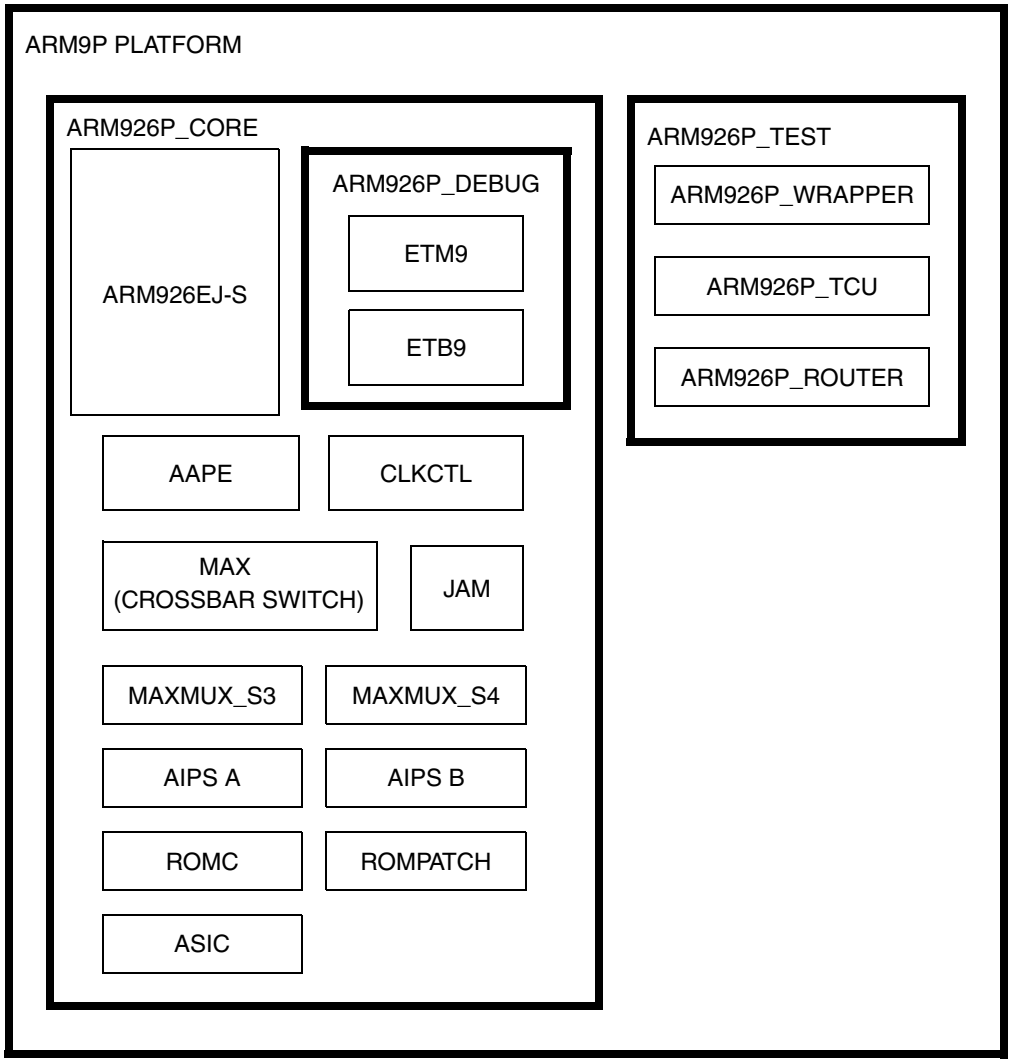


Figure 10-3. ARM9 Platform Hierarchy

10.4 JTAG ID Register

The ARM926EJ-S processor has a 32-bit input bus which corresponds to the JTAG ID register. This 32-bit register is defined as shown in Table 10-10. ARM requires bits [31:12] be set in accordance with their general rules such that Multi-ICE can auto-detect the device type.

Table 10-10. ARM926EJ-S JTAG ID Register Definition

tapid[31:28]	tapid[27:12]	tapid[11:8]	tapid[7:1]	tapid[0]
Version	Part Number	Manufacturing ID		1
0x0	0x7926	tapid_ver[3:0]	0b010_0000	1

Normally, the Freescale Manufacturing ID is 0x00E. However, since ARM uses all of the available bits that are usually used to track platform level revisions, the top 4 bits of the manufacturing ID are brought out for use at the system level (**tapid_ver[3:0]**), and the rest of the manufacturing ID is used to indicate platform revisions.

JTAG ID register bits [11:8] (**tapid_ver[3:0]** on the platform I/O) are incremented by the system integration team as appropriate throughout the life of the system.

JTAG ID register bits [7:1] are incremented by the ARM9 platform team as appropriate to indicate platform revisions.

10.5 System Memory Map

The memory map decode is based on **haddr[31:27]** on the MAX master ports, which is decoded to determine which slave port has been selected. Five bits are used, which is more than optimal; however, this is necessary to try and keep the ARM9 platform memory map consistent with that of the ARM9 platform. Table 10-11 shows a simplified breakdown of the regions decoded within the 4GB address space.

Table 10-11. ARM9 Platform High Level Address and Location Map

haddr[31:27]	Size	Usage	Location
0000x	256 Mbyte	ROM	MAX Slave Port 3
0001x	256 Mbyte	Reserved	—
001xx	512 Mbyte	Reserved	—
0100x	256 Mbyte	AIPS A	MAX Slave Port 3
0101x	256 Mbyte	AIPS B	MAX Slave Port 4
01100	128 Mbyte	ROMPATCH	MAX Slave Port 4
01101	128 Mbyte	ASIC	MAX Slave Port 3
0111x	256 Mbyte	Secondary AHB	MAX Slave Port 2
10xxx	1 Gbyte	Secondary AHB	MAX Slave Port 1[1]
11xxx	1 Gbyte	Secondary AHB	MAX Slave Port 0[2]

1 If MAX slave port sharing is disabled accesses to haddr[31:30] = 2'b10 are steered to MAX slave port 1 only. If MAX slave port sharing is enabled accesses to haddr[31:30] = 2'b10 can potentially be steered to MAX slave port 0 in addition to MAX slave port 1.

2 If MAX slave port sharing is disabled accesses to haddr[31:30] = 2'b11 are steered to MAX slave port 0 only. If MAX slave port sharing is enabled accesses to haddr[31:30] = 2'b11 can potentially be steered to MAX slave port 1 in addition to MAX slave port 0.

10.5.1 ARM9 Platform Memory Map

Table 10-12 shows the complete ARM9 Platform memory map.

Table 10-12. ARM9 Platform Detailed Memory Map

Address Range	Size	Use
0000_0000 - 0000_3FFF	16 Kbyte	ROM: First 16 Kbyte
0000_4000 - 0040_3FFF	4 Mbyte	Reserved
0040_4000 - 007F_FFFF	4080 Kbyte	ROM: Exceeding 16 Kbyte
0080_0000 - 0FFF_FFFF	248 Mbyte	Reserved
1000_0000 - 3FFF_FFFF	768 Mbyte	Reserved
4000_0000 - 41FF_FFFF	32 Mbyte	AIPS A Off platform global module enable 0
4200_0000 - 43EF_FFFF	31 Mbyte	AIPS A Off platform global module enable 1
43F0_0000 - 43F0_3FFF	16 Kbyte	AIPS A Control Registers (on platform slot 0)
43F0_4000 - 43F0_7FFF	16 Kbyte	AIPS A - MAX Registers (on platform slot 1)
43F0_8000 - 43F0_BFFF	16 Kbyte	AIPS A - CLKCTL (on platform slot 2)
43F0_C000 - 43F0_FFFF	16 Kbyte	AIPS A - ETB Registers (on platform slot 3)
43F1_0000 - 43F1_3FFF	16 Kbyte	AIPS A - ETB Memory (on platform slot 4)
43F1_4000 - 43F1_7FFF	16 Kbyte	AIPS A - AAPE Registers (on platform slot 5)
43F1_8000 - 43F7_FFFF	416 Kbyte	**Reserved** (AIPS A Unused on platform slots [31:6])
43F8_0000 - 43FB_FFFF	256 Kbyte	AIPS A Off platform slots [15:0]
43FC_0000 - 43FF_FFFF	256 Kbyte	**Reserved** (AIPS A Unused off platform slots [31:16])
4400_0000 - 4FFF_FFFF	192 Mbyte	**Reserved** (Aliased AIPS A Space)
5000_0000 - 51FF_FFFF	32 Mbyte	AIPS B Off platform global module enable 0
5200_0000 - 53EF_FFFF	31 Mbyte	AIPS B Off platform global module enable 1
53F0_0000 - 53F0_3FFF	16 Kbyte	AIPS B Control Registers (on platform slot 0)
53F0_4000 - 53F7_FFFF	496 Kbyte	**Reserved** (AIPS B Unused on platform slots [31:1])
53F8_0000 - 53FF_FFFF	512 Kbyte	AIPS B Off platform slots [31:0]
5400_0000 - 5FFF_FFFF	192 Mbyte	**Reserved** (Aliased AIPS B Space)
6000_0000 - 67FF_FFFF	128 Mbyte	ROMPATCH (Aliased throughout entire region)
6800_0000 - 6FFF_FFFF	128 Mbyte	ASIC (Aliased throughout entire region)
7000_0000 - 7FFF_FFFF	256 Mbyte	Secondary AHB (MAX Slave Port 2)
8000_0000 - BFFF_FFFF	1 Gbyte	Secondary AHB (MAX Slave Port 1)[1]
C000_0000 - FFFF_FFFF	1 Gbyte	Secondary AHB (MAX Slave Port 0)[2]

[1] If MAX slave port sharing is disabled accesses to haddr[31:30] = 2'b10 are steered to MAX slave port 1 only. If MAX slave port sharing is enabled accesses to haddr[31:30] = 2'b10 can potentially be steered to MAX slave port 0 in addition to MAX slave port 1.

[2] If MAX slave port sharing is disabled accesses to haddr[31:30] = 2'b11 are steered to MAX slave port 0 only. If MAX slave port sharing is enabled accesses to haddr[31:30] = 2'b11 can potentially be steered to MAX slave port 1 in addition to MAX slave port 0.

10.5.2 External Boot

When the **boot_int** input signal is asserted, the ARM926EJ-S boots internal from ROM on slave port 3 AHB. When **boot_int** is negated, the ARM926EJ-S reset vector fetch is routed to an address indicated by the **ext_boot_addr[31:2]** input pins. This vectoring is done by the ROMPATCH module, which monitors accesses to the ROMC and over-rides the reset vector fetch.

NOTE

When **boot_int** is negated and the ARM926EJ-S boots externally the ARM9 Platform is placed in an insecure state.

10.5.3 Memory Map Considerations

- Accesses to non-aliased "Reserved" locations in Table 10-12 result in an AHB error response.
- Accesses to unsupported address locations through the MAX result in an AHB error response and the access does not pass through the MAX.
- Accesses to address locations on either internal AHB bus (MAX slave ports 3 or 4) which do not map to a specific module time-out in 128 AHB clock cycles.
- Accesses to unimplemented locations within the ARM simple interrupt controller (ASIC) and ROMPATCH register space are terminated without a bus-error. Writes have no effect and reads return all zeros.
- Accesses to the ASIC and ROMPATCH are restricted to the ARM926EJ-S. Any master other than the ARM926EJ-S attempting to access the ASIC or the ROMPATCH receive an immediate AHB error response.

10.6 Platform Clocking

This section describes some of the clocking considerations within the ARM9 Platform. The circuits contained in the ARM9 Platform to address most of these issues are implemented within the Clock Control Module (CLKCTL).

10.6.1 ARM926EJ-S Clock Considerations

The ARM926EJ-S processor design uses a single clock, **clk**. In many systems, it is desirable for the ARM926EJ-S processor to run at a higher frequency than the AHB system bus (which runs on **hclk**). To support this, ARM926EJ-S requires a separate AHB clock enable for each of the two bus masters. **dhclken** is used to signify the rising edge of **hclk** for the system in which the data BIU is the bus master. **ihclken** is used to signify the rising edge of **hclk** for the system in which the instruction BIU is the bus master.

Figure 10-4 shows the relationship between **clk**, **hclk** and **dhclken**/**ihclken**. The ARM9 Platform provides a single **hclken_early** input pin that is fed to CLKCTL module, wherein it gets synchronized with respect

to **clk** (**hclken**). **hclken** is fed to both the **dhclken** and **ihclken** inputs on the ARM926EJ-S. If **hclk** and **clk** are the same frequency, the **hclken_early** input to the platform must be tied high.

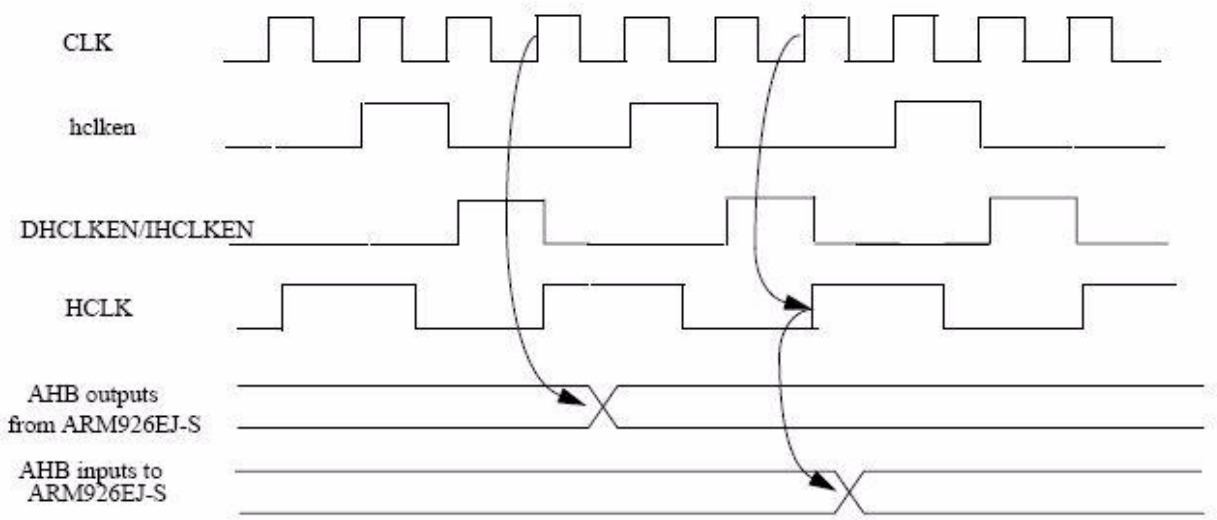


Figure 10-4. AHB Clock Relationship

clk and **hclk** must be synchronous and the skew between **clk** and **hclk** to the ARM9 Platform should be minimized. This requires some synchronization inside the chip clock control module.

10.6.2 ARM926EJ-S JTAG Port Clocking Considerations

The ARM926EJ-S does not support direct connection to the JTAG interface. The JTAG interface must be synchronized to the **clk** domain. This synchronization takes place within the platform CLKCTL module. See the ARM9 Platform CLKCTL design specification for more detail.

10.6.2.1 JTAG_TCK

The **jtag_tck** clock must be less than 1/8 the frequency of the **clk** input in order for the JTAG port and synchronizer to function properly. Note that the frequency of **clk** can vary when executing low-power code. Therefore, care must be taken such that **jtag_tck** is less than 1/8 the lowest possible frequency of **clk**.

10.6.3 External Alternate Bus Master Interfaces

Both alternate bus master ports on the ARM9 Platform must have the AHB synchronized to **hclk** external to the platform. All alternate bus master AHB inputs and outputs to/from the ARM9 Platform are synchronous to **hclk**. The Baseband processor, for instance, may be running at 66 MHz. Synchronization off platform must be implemented such that the alternate bus master interface to the ARM9 Platform would run at the **hclk** frequency.

10.6.4 External Secondary AHB Ports

All three secondary AHB ports inputs and outputs to and from the ARM9 Platform must be synchronous to the **hclk** and runs at the **hclk** frequency.

10.7 Platform Resets

This section describes the various ARM9 Platform reset inputs. Figure 10-5 shows the reset paths within the ARM9 Platform.

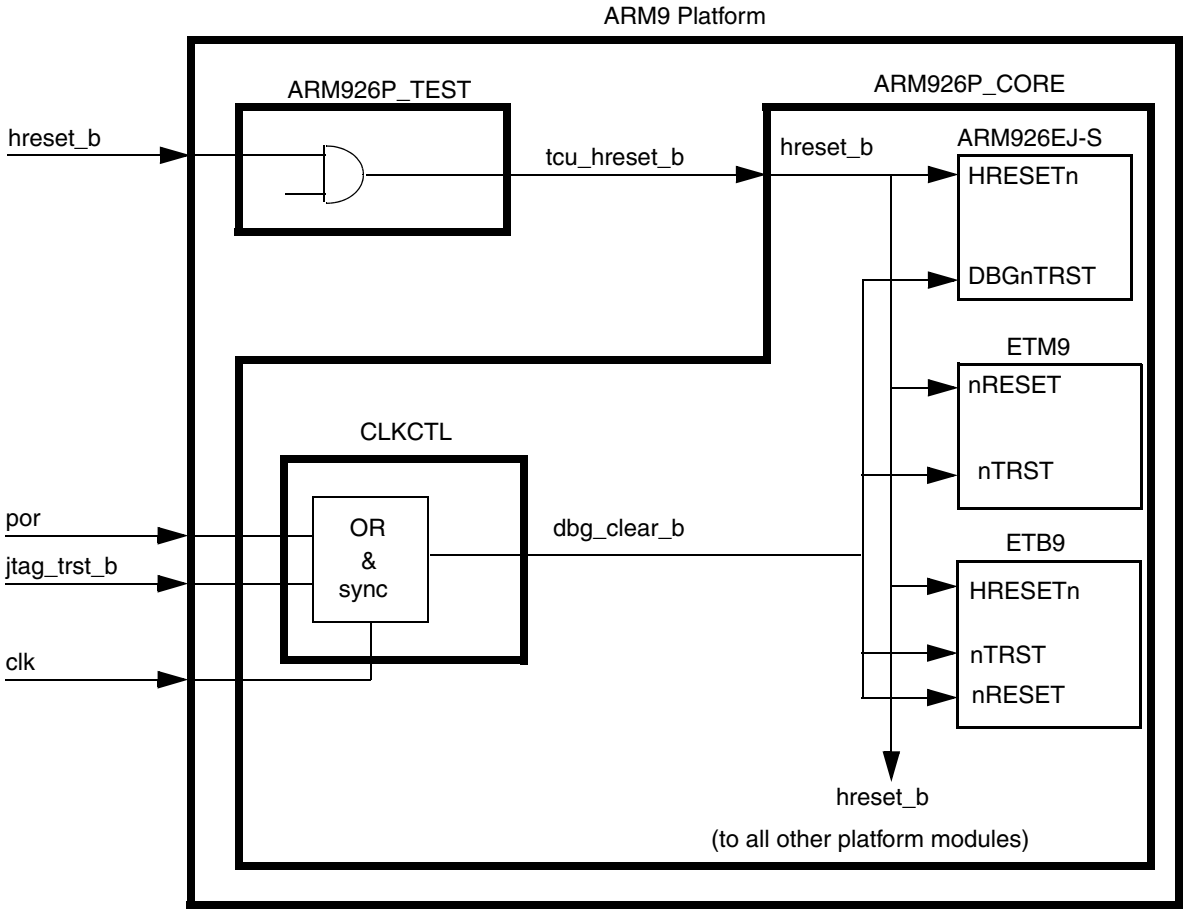


Figure 10-5. ARM9 Platform Resets

10.7.1 hreset_b

The **hreset_b** input is the asynchronous system reset for both the **clk** and **hclk** domains. It is gated with a test mode signal in the scan wrapper, and is then buffered for distribution throughout the platform.

10.7.2 POR and JTAG_TRST_B

The power-on-reset (**por**) and the JTAG reset (**jtag_trst_b**) are combined in the CLKCTL module to drive the **dbg_clear_b** signal to the ARM926EJ-S and ETM9 modules. The **dbg_clear_b** output of the

CLKCTL module can be considered as the JTAG or debug reset of the platform. The **dbg_clear_b** signal asserts asynchronously when either **por** or **jtag_trst_b** asserts, and negates synchronously to **clk** (through a synchronizer).

10.8 Power Management

10.8.1 Register Level Clock Gating

Under normal operating conditions, clocks internal to the platform are only issued to registers or banks of flops that need a rising edge for proper functionality. Otherwise, the clocks are held low.

10.8.2 Block Level Clock Gating

Clocks to individual modules within the platform are enabled only when necessary. On the internal AHBs for example, the CLKCTL module only enables **hclk** to a slave module when the current AHB access is addressed to that module. Slaves can also drive a signal to the CLKCTL if it requires its **hclk** to run for any other reason. See the CLKCTL module design specification for more detail.

10.8.3 External Clock Gating

The ARM926EJ-S processor may be put into a low-power state by the wait-for-interrupt instruction. This instruction switches the ARM926EJ-S into a low-power state until either an interrupt (nIRQ/nFIQ) or a debug request occurs. The switch into the low-power state is indicated by the assertion of the **arm_standbywfi** output signal. If **arm_standbywfi** is asserted then it is guaranteed that all ARM926EJ-S external interfaces are in an IDLE state. The **arm_standbywfi** signal is intended to be used to shut down clocks to the other parts of the system, such as external coprocessors, which do not need to be clocked if the ARM926EJ-S is idle. The ARM926EJ-S **clk** must not be stopped during wait-for-interrupt mode if an external debugger is connected to the JTAG port. An active **clk** is required to be able to write values into the ARM9EJ-S debug control register, which is required for a debugger to be able to force wait-for-interrupt mode to be exited. It should also be noted that the ARM926EJ-S needs **clk** to run in order for an interrupt to cause the negation of **arm_standbywfi**.

The JTAG synchronizer in the CLKCTL module needs to have an “always” clock running to it in order to, at any time, detect JTAG activity and thereby determine that a debugger is connected to the JTAG port. The presence of an active JTAG debugger is detected by monitoring the JTAG TMS signal. After **por** (or trst_b) assertion, a low state on TMS coincident with a rising-edge on **tck** transitions the JTAG tap-controller from the test-logic-reset state to the run-test-idle state. The **dbgen** signal is asserted, and held asserted, whenever the tap-controller is not in the test-logic-reset state. Once **dbgen** asserts, an active **trst_b** or **por** is required to clear it (that is, once a debugger is detected to be connected, it is assumed to stay connected).

When asserted, the **a9p_clk_off** output of the platform indicates to an external clock control module that **clk** and **hclken_early** should be turned off at the earliest opportunity. However, in order to assure that no alternate bus masters are in the middle of a transaction, the external clock control module must assert the **max_halt_request** input of the crossbar switch. This requests ownership of all AHB Output Ports. Once

max_halted is asserted, the external clock control module is then free to gate off **hclk** as all transactions on both the internal and secondary AHB.

Low power entry and exit sequences are described in the ARM simple interrupt controller (ASIC) chapter.

10.8.4 Well Biasing

A well bias clamp enable input, **wt_en**, is driven by an external clock control module to the ARM9 Platform. When asserted, V_{BB+} is shorted to VDD and V_{BB-} is shorted to GND.

10.9 Platform AHB Interfaces

This sections describes the major bus interfaces of the ARM9 Platform and the crossbar switch. A simple block diagram of the bus connections to the platform is shown in Figure 10-1. A definition of AHB-Lite, a functional description of the alternate bus master ports, and finally a description of the multi-layer crossbar switch slave ports follows.

10.9.1 Definition of AHB-Lite

All master and slave ports of the Multi-Layer AHB Crossbar switch are AHB-Lite-compatible. Therefore all AHBs connected externally to the ARM9 Platform must be AHB-Lite-compatible.

The definition of “AHB-Lite” for the ARM9 Platform is as follows:

- AHB split and retry protocols are not supported within the ARM9 Platform. This means that all slaves connected to AHB-Lite ports (input or output) are prohibited from requesting a split or a retry. This also means there is only one response signal, **hresp0**.
- AMBA bus request and bus grant are not supported on the AHB-Lite interfaces.
- Bursts are supported. The default configuration of the Crossbar Switch (MAX) insures no early fixed length burst terminations due to the switch arbiter.

10.9.2 Alternate Bus Master Ports

There are three alternate bus master ports (ABM) on the ARM9 Platform which are connected directly to the Multi-Layer AHB Crossbar Switch. These three ABM interfaces are AHB-Lite-compatible.

10.9.2.1 hmaster

Alternate bus masters external to the ARM9 Platform should be aware that two values of the **hmaster** field are used by bus masters internal to the platform. The reserved and available **hmaster** encoding is shown in Table 10-13.

Table 10-13. hmaster Encodings

hmaster	Use
0x0	Reserved: MAX default
0x1	Reserved: ARM926EJ-S I-AHB and D-AHB

Table 10-13. hmaster Encodings (continued)

hmaster	Use
0x9	RTIC
0xb	eSDHC2
0xc	SDMA
0xd	USB OTG

10.9.2.2 Bus Error

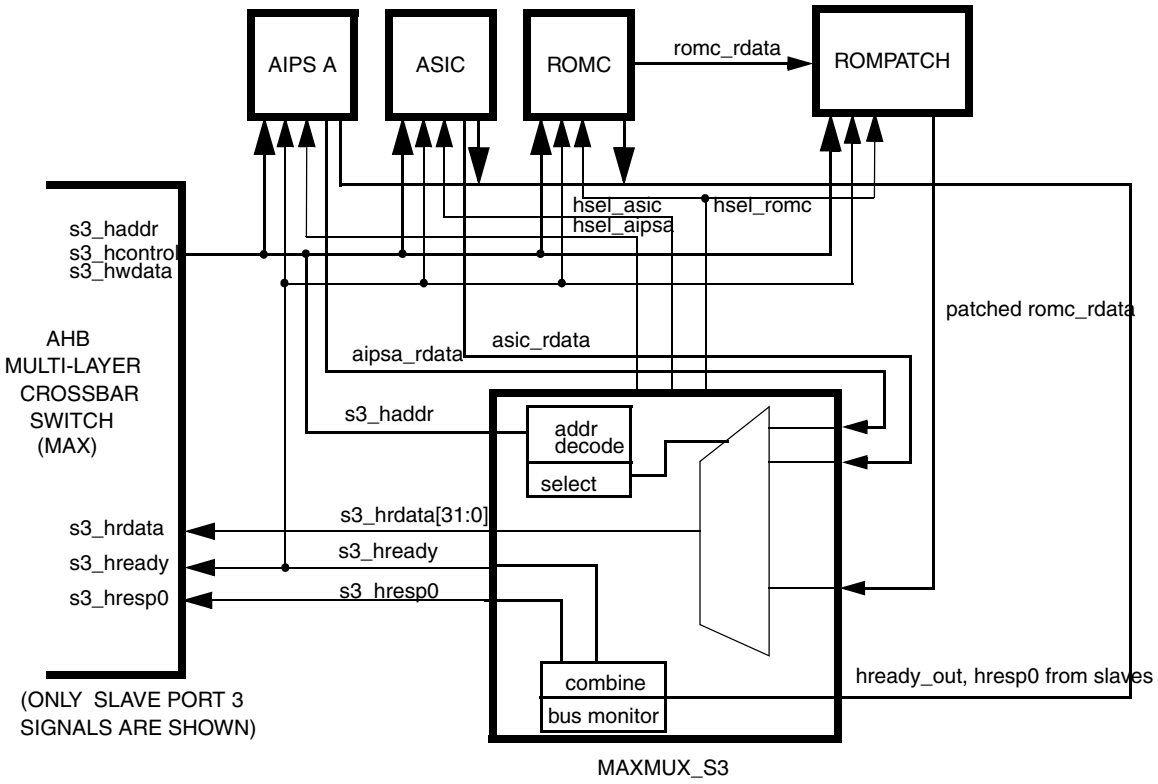
A slave two cycle ERROR response (**hresp0** = HIGH) allows for a bus master to cancel the remaining transfers in a burst. However, this is not an AHB requirement, and it is acceptable for the master to continue the remaining transfers of the burst.

AHB error responses generated on accesses to cacheable or bufferable memory address on the I-AHB and D-AHB interfaces of the ARM926EJ-S are normally ignored by the processor. In the ROMPATCH module, a feature can be enabled which, on the above described accesses, gates 0's onto **hrdata[31:0]** on data reads, and SWI opcodes onto **hrdata[31:0]** for instruction prefetches. At the same time, the ROMPATCH module generates an abort which guarantees entry into the ARM926EJ-S abort exception handler.

10.9.2.3 Halt Request (max_halt_request)

Care must be taken to ensure that the Halt Low Priority bit is not changing as the Clock Control Module Halt request is asserted. This results in unpredictable behavior. This can be avoided by not modifying this bit in the Slave General Purpose Control Register or in the Alternate Slave General Purpose Control register in software where Halt could be requested. Also, the Halt Low Priority bit should be programmed the same in both the Slave General Purpose Control Register and the Alternate Slave General Purpose Control Register, if it is likely the MAX can change between the General Purpose and Alternate registers during the time Halt could be requested.

Care should also be taken to ensure that the Clock Control Module Halt request is not asserted until at least two clock cycles after the last locked access performed by any master connected to the MAX.


AIPS A
ASIC
ROMC
romc_rdata
ROMPATCH
hsel_asic
hsel_aipsa
hsel_romc
s3_haddr
s3_hcontrol
s3_hwdata
patched romc_rdata
AHB
MULTI-LAYER
CROSSBAR
SWITCH
(MAX)
aipsa_rdata
asic_rdata
s3_haddr
addr
decode
select
s3_hrdata
s3_hready
s3_hresp0
s3_hrdata[31:0]
s3_hready
s3_hresp0
combine
bus monitor
hready_out, hresp0 from slaves
(ONLY SLAVE PORT 3
SIGNALS ARE SHOWN)
MAXMUX_S3

Chapter 11
ARM9 Platform AAPE

11.1 Introduction

The ARM926EJ-S ARM Abort Processing Engine (AAPE) is a 32-bit IP Bus peripheral which monitors the Instruction AHB (I-AHB) and the Data AHB (D-AHB) to ensure that when bufferable or cacheable accesses receive AHB ERROR responses, the cache controller does not mask them off, thereby allowing the acknowledgement and processing of the ERROR response.

11.1.1 Overview

The AAPE module provides the following features:

- Concurrent and separate monitoring of the I-AHB (instruction bus) and the D-AHB (data bus) of ARM926EJ-S core
- D-AHB abort generation and address tracking
- D-AHB data override for aborted READs separately enabled from abort generation
- I-AHB data override for aborted instruction prefetches
- IP Bus Register Interface

The block diagram of the AAPE is shown in Figure 11-1.

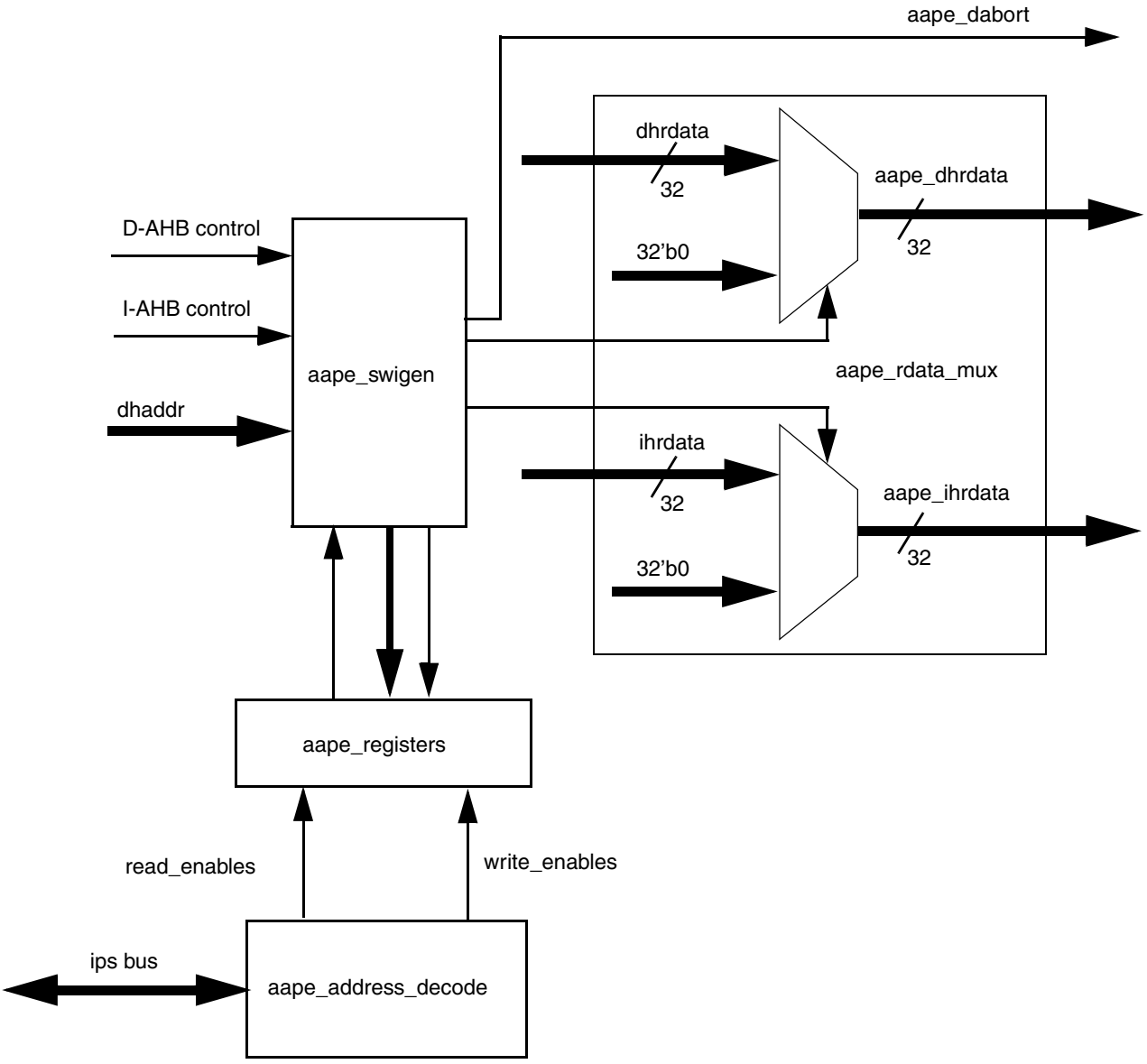


Figure 11-1. AAPE Block Diagram

11.1.2 Features

When enabled, the AAPE tracks cacheable/bufferable accesses on the D-AHB bus that terminate in an ERROR response, and forces data aborts in the ARM926EJ-S. The address of the aborted accesses are latched to allow subsequent processing by software. The AAPE can also be enabled to override the data of the aborted reads with zeros. The abort generation and data override features can be enabled separately. One scenario which makes use of the separate enablements is one in which illegal accesses to secure memory should yield zeroed out data, but not necessarily cause the ARM926EJ-S to enter the ABORT mode.

Abort tracking can also be enabled separately on the I-AHB. When enabled, The AAPE overrides aborted instruction fetches with the SWI (Software Interrupt) instruction or the Undefined Java byte code. However, no abort forcing is done by the AAPE for instruction fetches that terminate in AHB error responses. There is no latching of the address that has the AHB ERROR response.

11.2 Memory Map and Register Definition

The AAPE module has three registers: the control register, the status register, and the abort address register (read-only). The registers are IP bus compliant. Read and write transfers both require one IP bus clock cycle. The registers can only be accessed in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses. User mode, byte and half-word accesses result in a transfer error.

The registers are fully decoded. Write accesses to unimplemented locations within the AAPE result in a transfer error. Read accesses to unimplemented locations within the AAPE do not return an error response but return all 0s as read data.

This section includes the module memory map and detailed descriptions of all registers. For the base address of a particular module instantiation, see the system memory map.

11.2.1 Memory Map

The memory map for the AAPE program-visible registers is show in Table 11-1.

Table 11-1. AAPE Memory Map

Base Address Offset	Register	Access	Reset Value	Section/Page
0x000	Abort Control Register (ABCNTL)	R/W	0x0000_0000	11.2.3.1/11-5
0x004	Abort Status Register (ABSR)	R/W	0x0000_0000	11.2.3.2/11-6
0x008	Abort Address Register (ABDADR)	R	0x0000_0000	11.2.3.3/11-7

11.2.2 Register Summary

Table 11-2 is the register summary table.

Table 11-2. AAPE Register Summary

Name		31					26	25	24						17	16	15					10	9	8	7					1	0
ABCNTL ($BASE + 0x000)	R	0	0	0	0	0	0	IAOEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAOEN	DAGEN	0	0	0	0	0	0	0
	W																														
ABSR ($BASE + 0x004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IAB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAB
	W																														
ABDADR ($BASE + 0x008)	R	DABORTADDR																													
	W																														

11.2.3 Register Descriptions

This section provides detailed descriptions of the AAPE registers.

Register conventions: Figure 11-2 and Table 11-3 explain conventions used in register diagrams and tables.

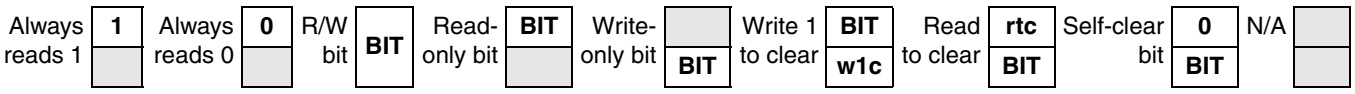


Figure 11-2. Register Field Conventions

Table 11-3. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)

Table 11-3. General Register Conventions (continued)

Convention	Description
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

11.2.3.1 Abort Control Register (ABCNTL)

The AAPE abort control register (ABCNTL) enables the abort tracking logic and data override for the I-AHB and D-AHB busses. Figure 11-3 shows the register. Table 11-4 describes the register fields.

Offset 0x0000 Access: Supervisor read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	IAOEN	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DAOEN	DAGEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-3. ABCNTL Register Diagram

Table 11-4. ABCNTL Register Field Description

Field	Description
31–26	Reserved
25 IAOEN	I-AHB Aborted RDATA Override Enable. This bit is cleared by Reset or by writing a 0 to the bit. Setting this bit enables the I-AHB abort tracking engine and the override of the RDATA bus when a I-AHB abort of cacheable/bufferable data occurs. For an aborted cacheable/bufferable fetch the value sent to the I-Cache depends on the operating mode of the ARM926EJ-S core. In THUMB mode, the value sent is a THUMB SWI #16 (0xDF10) on both halves of the aape_ihrdata bus. In ARM mode, the value sent is an ARM SWI #16 (0xEF000010). In Java mode, the value sent is the byte code 0xFF on both halves of the aape_ihrdata bus 0 I-AHB abort tracking and aborted RDATA Override disabled. 1 I-AHB abort tracking and aborted RDATA Override enabled.
24–10	Reserved

Table 11-4. ABCNTL Register Field Description (continued)

Field	Description
9 DAOEN	D-AHB Aborted RDATA Override Enable. This bit is cleared by Reset or by writing a 0 to the bit. Setting this bit enables the D-AHB abort tracking engine, and drive 0s on the aape_dhrdata bus when a D-AHB abort of cacheable/bufferable data occurs, but does not enable the assertion of aape_dabort. For an aborted cacheable/bufferable access the value sent to the D-Cache is all 0's. 0 D-AHB abort tracking and aborted RDATA Override disabled. 1 D-AHB abort tracking and aborted RDATA Override enabled.
8 DAGEN	D-AHB Abort Cache Override Enable. This bit is cleared by Reset or by writing a 0 to the bit. Setting this bit enables the D-AHB abort tracking engine. The AAPE generates assert aape_dabort which is ORed with the abort from the D-Cache when a D-AHB abort occurs on a cacheable/bufferable access. 0 D-AHB abort tracking disabled. 1 D-AHB abort tracking and abort forcing enabled.
7–0	Reserved

11.2.3.2 Abort Status Register (ABSR)

The AAPE abort status register (ABSR) has two bits to flag the occurrence of ERROR responses on the I-AHB and D-AHB busses. The IAB status bit is set when an ERROR response occurs on the I-AHB bus for a bufferrable/cacheable read access, provided that bit IAOEN of the ABCNTL register is set. Once set, IAB stays set until cleared by a Reset condition or by writing a 1 to the bit.

Similarly, the DAB status bit is set when an ERROR response occurs on the D-AHB bus for a bufferable/cacheable read access, provided that either DAOEN or DAGEN bit or both in the ABCNTL register is set. Once set DAB stays set until cleared by a Reset condition or by writing a 1 to the bit. Figure 11-4 shows the register. Table 11-5 describes the register fields.

Offset 0x0004 — Access: Supervisor read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IAB
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAB
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-4. ABSR Register Diagram

Table 11-5. ABSR Field Descriptions

Field	Description
31–16	Reserved
16 IAB	I-AHB Abort indicator. Indicates that a cacheable or bufferable abort occurred on the I-AHB. Reset and writing a 1 to this bit clears it. 0 Normal 1 I-AHB cacheable or bufferable access abort occurred
15–1	Reserved
0 DAB	D-AHB Abort indicator. Indicates that a cacheable or bufferable abort occurred on the D-AHB. Reset and writing a 1 to this bit clears it. 0 Normal 1 D-AHB cacheable or bufferable access abort occurred

11.2.3.3 Abort Address Register (ABDADR)

The D-AHB Abort Address Register (ABDADR) records the address of D-AHB cacheable or bufferable accesses that terminate with an AHB ERROR response when either or both DAOEN or DAGEN bits of the ABCNTL register is set. It can only be updated with an address that generated the abort if the DAB flag in the ABSR register is in the cleared state when the aborted access occurs. Once the address is captured (the DAB flag in ABSR is also set then), it remains until the DAB flag in the ABSR register is cleared by software writing a 1 to it. After the DAB flag is cleared this register is then allowed to latch the address of the next aborted access. This register reads 0s when the DAB bit is cleared. Writing to this register has no effect. This read-only register is cleared on reset. Figure 11-5 shows the register. Table 11-6 describes the register fields.

Offset 0x0008 Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DABORTADDR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DABORTADDR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-5. ABDADR Register Diagram

Table 11-6. ABDADR Field Description

Name	Description
31–0 DABORTADDR	D-AHB Abort Address. Address of the first D-AHB aborted access before the DAB bit in the ABSR register is set. The address can only be latched if either or both DAOEN or DAGEN bits in ABCNTL is set and the DAB flag in ABSR register is cleared. This value remains until the DAB bit is cleared. When DAB is cleared, these bits read as 0s until the next occurrence of an abort data access, when the corresponding address is latched.

11.3 AAPE Functional Description

11.3.1 D-AHB Abort Tracking, Abort Forcing, and Data Override

Abort tracking on the D-AHB is enabled by setting either the DAOEN or DAGEN bit in the ABCNTL register. Abort forcing is enabled by setting the DAGEN bit. Setting the DAOEN bit enables the data override function on the D-AHB.

11.3.1.1 D-AHB Abort Tracking

The AAPE performs abort tracking on the D-AHB by:

- Identifying “taken” cacheable and/or bufferable accesses on D-AHB through the decoding of **dhprot[3:2]** (either or both signals asserted), **dhtrans[1]** (asserted), and **dhready** (asserted),
- Identifying whether the access terminates with AHB ERROR monitoring the assertion of **dhresp0**,
- Setting the DAB abort flag DAB in the ABSR if an AHB ERROR occurs,
- Updating the ABDADR register with the address of the “taken” and aborted data access. A new address can only be latched if the DAB bit has been cleared previously.

11.3.1.2 Abort Forcing On Aborted D-AHB Accesses

If abort forcing is enabled and abort tracking indicates a D-AHB ERROR response, the AAPE asserts the **aape_dabort** signal until the ARM926EJS recognizes the abort and enters the abort mode. The abort handler software can subsequently read the ABDADR register to obtain the address of the aborted data access. The handler should also check the setting of DAB status flag, then clear it by writing a 1 to this bit. Clearing the DAB bit allows a subsequent aborted address to be latched in the ABDADR register. Reading the ABDADR register has no effect on the DAB status bit.

11.3.1.3 D-AHB Abort Data Override

If the data override function is enabled and abort tracking indicates a D-AHB ERROR response, the AAPE drives zeros on the **aape_dhrdata** bus, that is, returning zeros to the aborted read access. This data conforms to the normal set up and hold timing of the D-AHB bus. For reads that terminate normally, the AAPE allows the value on the dhrdata data bus to flow through to the **aape_dhrdata** bus.

11.3.2 I-AHB Abort Tracking and Data Override

The abort tracking and data override functions on the I-AHB is enabled by setting the IAOEN in the ABCNTL register.

11.3.2.1 I-AHB Abort Tracking

The AAPE performs abort tracking on the I-AHB by the following:

- Identifying cacheable and/or bufferable accesses on I-AHB through the decoding of **ihprot[3:2]** (either or both signals asserted), **ihtrans[1]** (asserted), and **ihready** (asserted),
- Identifying whether the access terminates with AHB ERROR monitoring the assertion of **ihresp0**,
- Setting the IAB abort flag in the ABSR register.

11.3.2.2 I-AHB Abort Data Override

If abort tracking indicates an I-AHB ERROR response, the AAPE drives an SWI instruction on the **aape_ihrdata** bus. Depending on the current mode of the ARM926EJ-S, a 32- bit SWI (ARM mode), two 16-bit SWI instructions for the upper and lower half words (THUMB), or four undefined instructions (0xFF) are generated for each byte lane.

The ARM SWI has the opcode value 0xEF occupying bits [31:24], and a value of 16 in the comment field (bits [23:0]. The THUMB SWI has the opcode value 0xDF occupying bits [15:8], and a value of 16 in the comment field (bits [7:0]). When the SWI is eventually executed, the handler software loads the SWI instruction and extract the comment value to identify an instruction fetch abort and branch to the proper handler for further processing.

The following two code snippets illustrate how the SWI might be processed. The difference between the two examples is due to the presence or absence of the ROMPATCH module which also makes use of the SWI mechanism to perform opcode patching operations. The comment field values used by the ROMPATCH range from 0 to 15.

11.3.2.2.1 Software Response to I-AHB Abort (No ROMPATCH Present)

```
stmfd    sp!, {r0-r1,lr}@ push register onto SWI stack
mrs      r0, spsr @ get saved status register
tst      r0, #0x20@ check if call was in THUMB mode
ldrneh   r0, [lr,#-2]@ yes: load opcode half-word and
bicne    r0, r0, #0xff00@ yes: extract THUMB comment
ldreq    r0, [lr,#-4]@ no: load opcode word and
biceq    r0, r0, #0xff000000@ no: extract ARM comment
                 @ now r0 has comment field
cmp      r0, #16  @ compare to 16
ldreq    r1, =iahb_abort_hdler@ = 16: pointer to I-AHB abort handler
ldrne    r1, =swi_hdler@ != 16: pointer to standard SWI
                 @ handler
mov      lr, pc   @ != 16: set link register
bx       r1       @ != 16: jump to standard SWI
                 @ handler
ldmfd    sp!, {r0-r1,pc}^@ != 16: pop registers from stack
```

11.3.2.2.2 Software Response to I-AHB Abort (ROMPATCH Present)

```
stmfd    sp!, {r0-r1,lr}@ push register onto SWI stack
mrs      r0, spsr @ get saved status register
tst      r0, #0x20@ check if call was in THUMB mode
ldrneh   r0, [lr,#-2]@ yes: load opcode half-word and
bicne    r0, r0, #0xff00@ yes: extract THUMB comment
```

```
ldreq    r0, [lr,#-4]@ no: load opcode word and
biceq    r0, r0, #0xff000000@ no: extract ARM comment
                 @ now r0 has comment field
cmp      r0, #16  @ compare to 16
ldrlt    lr, =rompatch_tbl_prt@ < 16: get top of current ROMPATCH
                 @ table; global variable which is
                 @ changeable per context
ldrlt    r1, [lr, r0 lsl #2]@ < 16: read function pointer from
                 @ table assumed an array of pointers
                 @ patch functions
strlt    r1, [sp, #8]@ < 16: store function pointer onto
                 @ stack in position of link register
ldmltfd  sp!, {r0-r1,pc}^@ < 16: "fake" return from SWI, will
                 @ vector core to appropriate patch
                 @ function and set core back to previous
                 @ mode of operating
ldreq    r1, =iahb_abort_hdler@ = 16: pointer to I-AHB abort handler
ldrgt    r1, =swi_hdler@ > 16: pointer to standard SWI
                 @ handler
mov      lr, pc   @ > 16: set link register
bx       r1       @ > 16: jump to standard SWI
                 @ handler
ldmfd    sp!, {r0-r1,pc}^@ > 16: pop registers from stack
```

Chapter 12
Advanced Technology Attachment (ATA)

12.1 Overview

The ATA block is an AT attachment host interface, which is compatible with the ATA-6 standard. Its main use is to interface with IDE hard disc drives and ATAPI optical disc drives. It interfaces with the ATA device over a number of ATA signals.

See Figure 12-1 for the block diagram of the ATA interface.

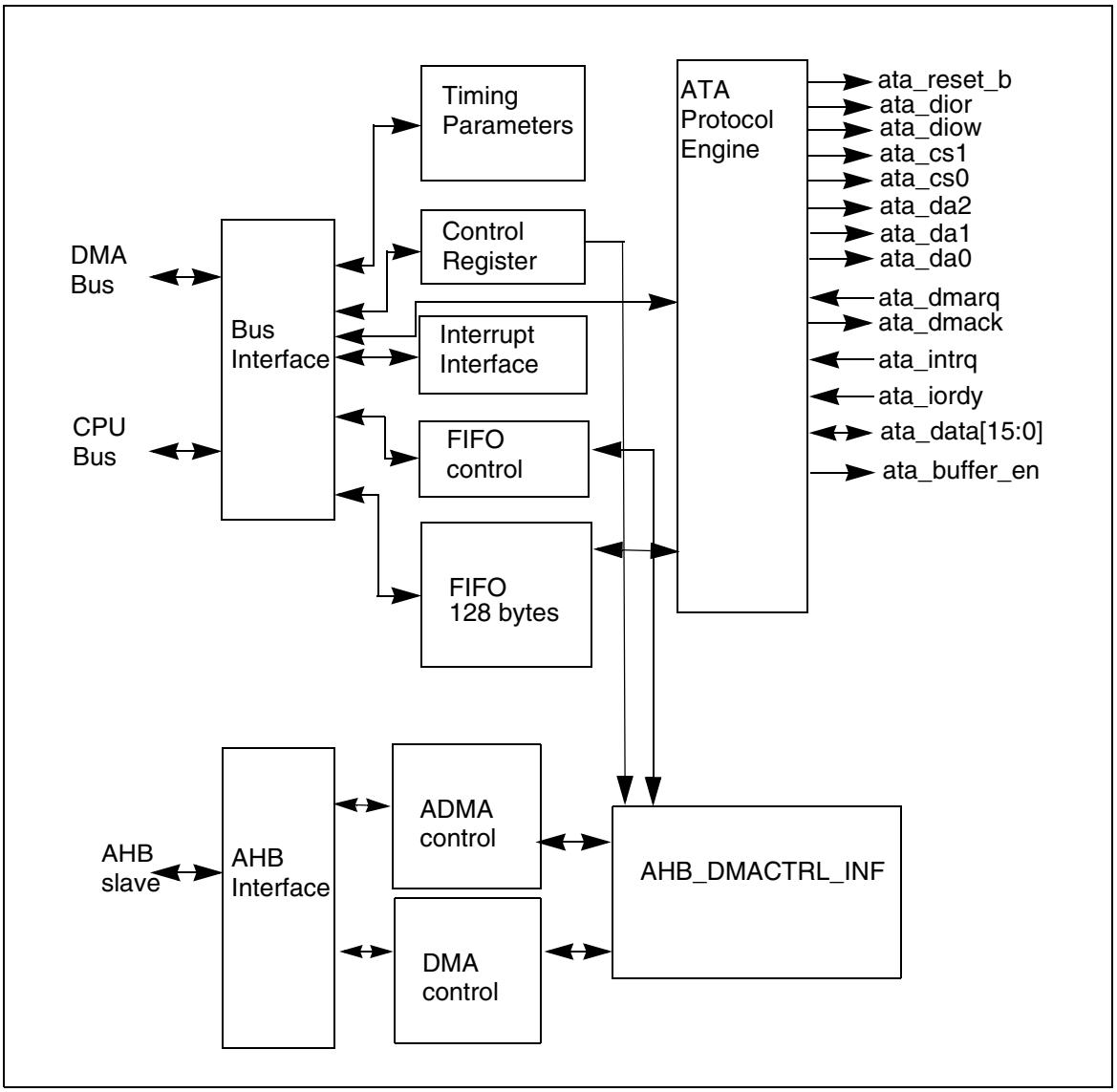


Figure 12-1. ATA Interface Block Diagram

In Figure 12-1, the CPU and DMA buses communicate with the host processor and host DMA unit, respectively. The AHB bus enables direct transfers to and from external memory. All internal registers are visible from both the CPU and DMA buses, allowing smart DMA access to program the interface.

Two access modes are possible over the ATA bus: PIO mode and DMA mode. There are also two types of DMA mode: DMA slave mode (controlled by the host DMA) and DMA master mode (controlled by the internal DMA master). See Section 12.1.2, “Modes of Operation,” for more information about these access modes.

12.1.1 Features

The ATA interface includes the following features:

- Programmable timing on the ATA bus. Works with a wide range of bus frequencies.
- Compatible with ATA-6 specification
  - Supports PIO modes 0–4
  - Supports multiword DMA (MDMA) modes 0, 1, and 2
  - Supports ultra DMA (UDMA) modes 0–4 with a bus clock of at least 50 MHz
  - Supports ultra DMA (UDMA) mode 5 with a bus clock of at least 80 MHz
- 128-byte FIFO included within the interface
- FIFO receive, transmit, and end-of-transmission alarms to DMA unit
- Zero-wait cycles transfer between DMA bus and FIFO (enables fast FIFO reads/writes)
- Supports AMBA 2.0 specification
- Supports 32-bit system memory addressing
- Supports DMA reads/writes by IPS bus (DMA slave mode)
- Supports single DMA reads and writes by the AHB bus (DMA master mode)
- Supports scatter-gather (ADMA) DMA reads and writes by the AHB bus (DMA master mode)
- Supports 4-words burst length set by software

12.1.2 Modes of Operation

The interface offers two alternative modes of operation, PIO mode and DMA mode (which includes DMA slave and DMA master modes).

12.1.2.1 PIO Mode

An access to the ATA bus in PIO mode occurs whenever an ATA PIO register is read or written to by the host CPU or the host (smart) DMA unit. During a PIO transfer, the incoming IP bus cycle is translated to an ATA PIO bus cycle by the ATA protocol engine. No buffering of data occurs, so the host CPU or host DMA cycle is stalled until the ATA bus read data is available on read, or is stalled until the IP bus data can be put on the ATA bus during a write.

PIO accesses can be made at any time, even during a running ATA DMA transfer. In this case, the DMA transfer is paused, the PIO cycle is completed, and the DMA transfer is resumed.

12.1.2.2 DMA Mode

In DMA mode, data is transferred between the ATA bus and the FIFO. Two different DMA protocols are supported on the ATA bus: ultra DMA (UDMA) mode and multiword DMA (MDMA) mode. These are described in more detail in the sections indicated below.

In DMA mode transfers, data is transferred between the ATA bus and the FIFO. Either the host smart DMA unit or the internal DMA master is responsible to write/read data to/from the FIFO to keep the transfer

going. The case where the smart DMA unit is responsible is designated as DMA slave mode; while the case where the internal DMA master is responsible is designated as DMA master mode.

In DMA slave mode, the fifo_rcv_alarm and fifo_tx_alarm signals are sent to the host DMA unit to avoid FIFO overflow and underflow, respectively. An additional alarm signal (fifo_txfer_end_alarm) signals the end of the transfer to the smart DMA, which then completes the transfer and informs the CPU. Further details on these alarm signals are given in Section 12.4.3.6, "FIFO Alarm Register (FIFO_ALARM)."

In DMA master mode, prior to transfer, the DMA master must be programmed with the burst length, DMA system start base address or ADMA descriptor table start address (if a descriptor table has been prepared), and DMA mode. DMA transfer to the AHB bus is initiated when the DMA master receives the sys_dma_req signal. Further details on DMA master mode operation are given in Section 12.5.6, "Using DMA Master Mode to Receive Data From the ATA Bus," and Section 12.5.7, "Using DMA Master Mode To Transmit Data to the ATA bus."

The typical packet size is 32 bytes (8 32-bit words), but other packet sizes can also be handled.

Further details on DMA mode may be found in the following sections:

- Section 12.5.4, "Receiving Data from ATA Bus in DMA Slave Mode"
- Section 12.5.5, "Transmitting Data to ATA Bus in DMA Slave Mode"
- Section 12.5.6, "Using DMA Master Mode to Receive Data From the ATA Bus"
- Section 12.5.7, "Using DMA Master Mode To Transmit Data to the ATA bus"

12.2 External Signal Description

Table 12-1 lists the signals between this module and peripherals within the chip.

Table 12-1. Signal Properties

Name	Function	Reset State	Type
ata_reset_b	ATA bus reset signal. Active low. If active, the ATA device is reset[1].	0	out
ata_dior	ATA bus read strobe	1	out
ata_diow	ATA bus write strobe	1	out
ata_cs1	ATA bus chip select 1	1	out
ata_cs0	ATA bus chip select 0	1	out
ata_da2	ATA bus address line 2	0	out
ata_da1	ATA bus address line 1	0	out
ata_da0	ATA bus address line 0	0	out
ata_dmarq	ATA bus DMA request	—	in
ata_dmack	ATA bus DMA acknowledge	1	out
ata_intrq	ATA bus interrupt request	—	in
ata_iordy	ATA bus iordy	—	out
ata_data[15:0]	ATA data bus (little-endian)	Hi-z	tristate in-out

[1]This signal is a standard ATA bus signal. It conforms with the ATA specification.

12.2.1 Signal Descriptions

For a detailed description of the ATA bus signals, see the ATA-6 specification.

12.2.1.1 ata_reset_b (out)

When negated, the ATA reset signal indicates the ATA bus is in reset state. The ATA bus is in reset whenever the appropriate bit in the control register is cleared. After system reset, the ATA bus is reset.

12.2.1.2 ata_dior (out)

During PIO and MDMA transfers, the DIOR ATA signal functions as a read strobe. During UDMA data in bursts, it functions as HDMARDY. During UDMA data out burst mode, it functions as host strobe.

12.2.1.3 ata_diow (out)

During PIO and MDMA transfers, the DIOW ATA signal functions as a write strobe. During UDMA burst mode, it is used by the host terminate running UDMA transfers.

12.2.1.4 ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0 (out)

The address group of the ATA bus consists of the chip selects ata_cs0 and ata_cs1, and the address lines ata_da2, ata_da1, and ata_da0. All five lines follow the same timing.

12.2.1.5 ata_dmarq (in)

The ATA bus device DMA request is pulled high by the device if it wants to transfer data using MDMA or UDMA mode.

12.2.1.6 ata_dmack (out)

The ATA bus host DMA acknowledge is pulled low by the host when it grants the DMA request.

12.2.1.7 ata_intrq (in)

The ATA bus interrupt request is pulled high by the device whenever it wants to interrupt the host CPU.

12.2.1.8 ata_iordy (in)

The ATA bus IORDY line has three functions:

- IORDY—Active low, wait during PIO cycles
- DDMARDY—Active low, device ready during UDMA out-transfers
- DSTROBE—Device strobe during UDMA in-transfers

12.2.1.9 ata_data[15:0] (in/out—tristate)

The ATA data bus signal carries data to/from the ATA device.

12.2.1.10 ata_buffer_en

When the buffer direction control signal is asserted, data is driven outward to the device; when negated, data is driven inward to the host.

12.2.2 ATA Bus Timing

This section describes the ATA bus timing and explains how to ensure that the ATA interface meets timing requirements. Timing diagrams and timing relation equations are provided.

12.2.2.1 Timing Parameters

Table 12-2 shows the ATA timing parameters and their determining factors. Determining factors include the system design, the bus buffer used, the cable delay and the cable skew.

Table 12-2. Timing Parameters

Name	Meaning	Controlled by
T	Bus clock period	clock generator
ti_ds	Set-up time ata_data to ata_iordy edge (UDMA-in only)	top level design
ti_dh	hold time ata_iordy edge to ata_data (UDMA-in only)	top level design
tco	propagation delay bus clock L-to-H to ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data, ata_buffer_en	top level design
tsu	set-up time ata_data to bus clock L-to-H	top level design
tsui	set-up time ata_iordy to bus clock H-to-L	top level design
thi	hold time ata_iordy to bus clock H to L	top level design
tskew1	Max difference in propagation delay bus clock L-to-H to any of following signals ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data (write), ata_buffer_en	top level design
tskew2	Max difference in buffer propagation delay for any of following signals ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_dior, ata_diow, ata_dmack, ata_data (write), ata_buffer_en	transceiver
tskew3	Max difference in buffer propagation delay for any of following signals ata_iordy, ata_data (read)	transceiver
tbuf	Max buffer propagation delay	transceiver
tcable1	cable propagation delay for ata_data	cable
tcable2	cable propagation delay for control signals ata_dior, ata_diow, ata_iordy, ata_dmack	cable
tskew4	Max difference in cable propagation delay between ata_iordy and ata_data (read)	cable

Table 12-2. Timing Parameters (continued)

Name	Meaning	Controlled by
tskew5	Max difference in cable propagation delay between (ata_dior, ata_diow, ata_dmack) and ata_cs0, ata_cs1, ata_da2, ata_da1, ata_da0, ata_data (write)	cable
tskew6	Max difference in cable propagation delay without accounting for ground bounce	cable

12.2.2.2 PIO Mode Timing

12.2.2.2.1 PIO Read Mode Timing

A timing diagram for PIO read mode is given in Figure 12-2.

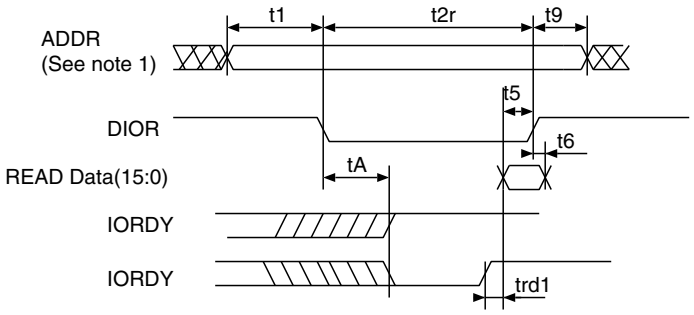


Figure 12-2. PIO Read Mode Timing

To meet timing requirements, a number of timing parameters must be controlled. Table 12-2 shows relations between different timing parameters, and identifies parameters in the ATA timing registers which may be programmed by the user to satisfy timing constraints (see Section 12.4.3.2, “Timing Registers”).

In Table 12-2 (and Table 12-3 below), the first column lists parameters from the ATA specification; the second column refers to timing parameters shown in Figure 12-2; the third column shows the relation between these timing parameters and programmable values; and the fourth column identifies the registers that may be programmed to meet the timing relations.

Table 12-3. Timing Parameter Relations for PIO Read

ATA Parameter	PIO Read Mode Timing Parameter[1]	Relation	Programmable Register
t1	t1	t1(min) = time_1 * T - (tskew1 + tskew2 + tskew5)	time_1
t2 (read)	t2r	t2(min) = time_2r * T - (tskew1 + tskew2 + tskew5)	time_2r
t9	t9	t9(min) = time_9 * T - (tskew1 + tskew2 + tskew6)	time_9
t5	t5	t5(min) = tco + tsu + tbuf + tbuf+ tcable1 + tcable2	time_2 (affects tsu and tco)
tA	tA	tA(min) = (1.5 + time_ax) * T - (tco + tsui + tcable2 + tcable2 + 2*tbuf)	time_ax

Table 12-3. Timing Parameter Relations for PIO Read (continued)

ATA Parameter	PIO Read Mode Timing Parameter[1]	Relation	Programmable Register
trd	trd1	trd1(max) = (-trd)+ (tskew3 + tskew4) trd1(min) = (time_pio_rdx - 0.5)*T - (tsu + thi) (time_pio_rdx - 0.5) * T > tsu + thi + tskew3 + tskew4	time_pio_rdx
t0	—	t0(min) = (time_1 + time_2r+ time_9) * T	time_1, time_2r, time_9

1 See Figure 12-2.

12.2.2.2.2 PIO Write Mode Timing

A timing diagram for PIO write mode is given in Figure 12-3.

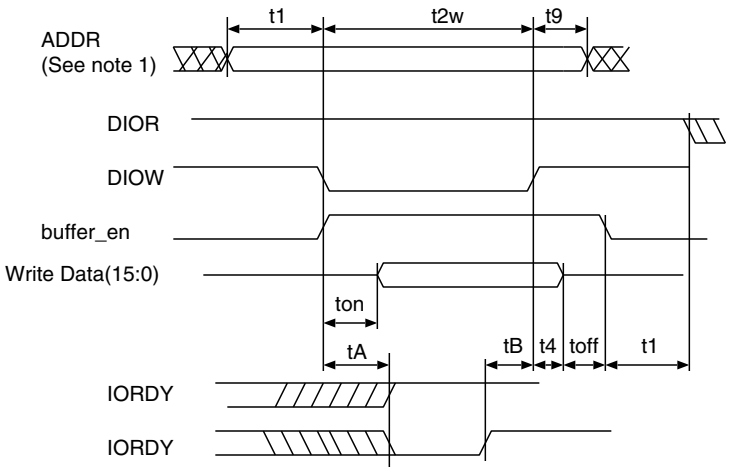


Figure 12-3. PIO Write Mode Timing

Table 12-4 shows relations between timing parameters, and identifies parameters in the ATA timing registers which may be programmed by the user to meet timing constraints (see Section 12.4.3.2, “Timing Registers”).

Table 12-4. Timing Parameters Relations for PIO Write

ATA Parameter	PIO Write Mode Timing Parameter[1]	Relation	Programmable Value
t1	t1	t1(min) = time_1 * T - (tskew1 + tskew2 + tskew5)	time_1
t2 (write)	t2w	t2(min) = time_2w * T - (tskew1 + tskew2 + tskew5)	time_2w
t9	t9	t9(min) = time_9 * T - (tskew1 + tskew2 + tskew6)	time_9
t3	—	t3(min) = (time_2w - time_on)* T - (tskew1 + tskew2 +tskew5)	if not met, increase time_2w
t4	t4	t4(min) = time_4 * T - tskew1	time_4
tA	tA	tA = (1.5 + time_ax) * T - (tco + tsui + tcable2 + tcable2 + 2*tbuf)	time_ax

Table 12-4. Timing Parameters Relations for PIO Write (continued)

ATA Parameter	PIO Write Mode Timing Parameter[1]	Relation	Programmable Value
t0	—	t0(min) = (time_1 + time_2 + time_9) * T	time_1, time_2r, time_9
—	—	Avoid bus contention when switching buffer on by making ton long enough	—
—	—	Avoid bus contention when switching buffer off by making toff long enough	—

1 See Figure 12-3.

12.2.2.3 Timing in Multiword DMA (MDMA) Mode

Figure 12-4 and Figure 12-5 show read and write timings respectively for MDMA mode.

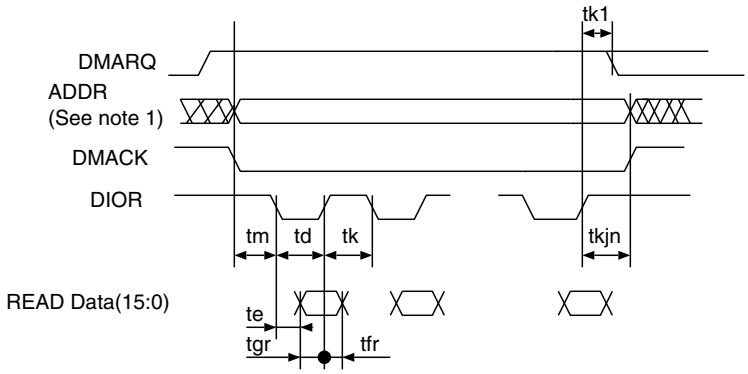


Figure 12-4. MDMA Read Timing

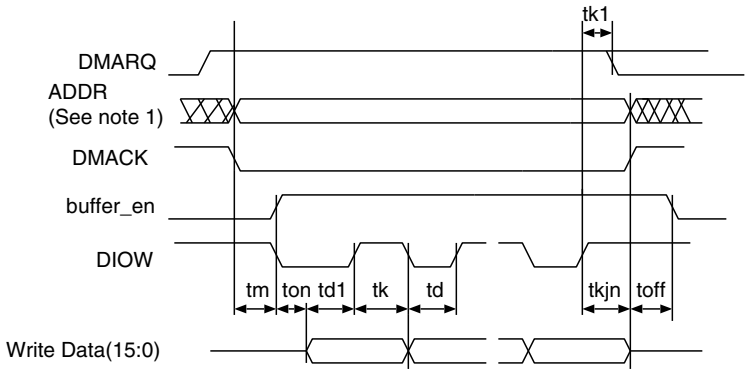


Figure 12-5. MDMA Write Timing

To meet timing requirements for MDMA mode, a number of timing parameters must be controlled. Table 12-5 shows the relations between different timing parameters, and identifies parameters in the ATA timing registers which may be programmed by the user to satisfy timing constraints (see Section 12.4.3.2, “Timing Registers”).

In Table 12-5, the first column lists parameters from the ATA specification; the second column refers to timing parameters shown in Figure 12-4 and Figure 12-5; the third column shows the relation between these timing parameters and programmable values; and the fourth column identifies the registers which may be programmed to meet timing constraints.

Table 12-5. Timing Parameter Relations for MDMA Read and Write

ATA Parameter	MDMA Read Timing[1] and MDMA Write Timing[2]	Relation	Programmable Value
tm, ti	tm	tm(min) = ti(min) = time_m * T - (tskew1 + tskew2 + tskew5)	time_m
td	td, td1	td1(min) = td(min) = time_d * T - (tskew1 + tskew2 + tskew6)	time_d
tk	tk[3]	tk(min) = time_k * T - (tskew1 + tskew2 + tskew6)	time_k
t0	—	t0(min) = (time_d + time_k) * T	time_d, time_k
tg (read)	tgr	tgr(min-read) = tco + tsu + tbuf + tbuf + tcable1 + tcable2 tgr(min-drive) = td - te(drive)	time_d
tf (read)	tfr	tfr(min) = 5 ns	(met by design)
tg(write)	—	tg(min-write) = time_d * T -(tskew1 + tskew2 + tskew5)	time_d
tf(write)	—	tf(min-write) = time_k * T - (tskew1 + tskew2 + tskew6)	time_k
tL	—	tL(max) = (time_d + time_k-2)*T - (tsu + tco + 2*tbuf + 2*tcable2)	time_d, time_k[4]
tn, tj	tkjn	tn= tj= tkjn = time_jn * T - (tskew1 + tskew2 + tskew6)	time_jn
—	ton toff	ton = time_on * T - tskew1 toff = time_off * T - tskew1	—

1 See Figure 12-4.

2 See Figure 12-5.

3 tk1 in the MDMA figures (Figure 12-4 and Figure 12-5) equals (tk -2*T)

4 tk1 in the MDMA figures equals (tk -2*T)

12.2.2.4 Timing for Ultra DMA (UDMA) Data In-Transfers

Figure 12-6 shows timing for the start of a UDMA data in-transfer.

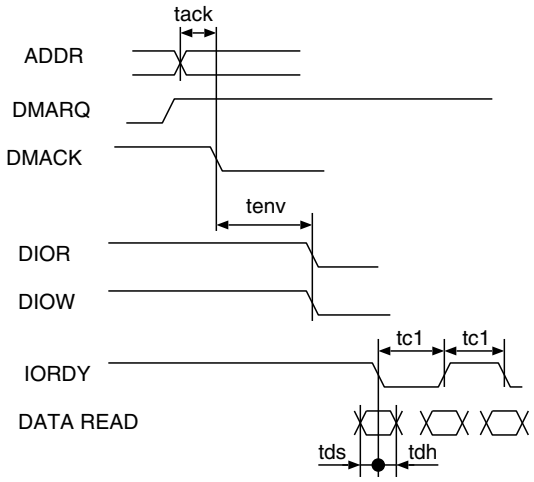


Figure 12-6. Timing for Start of UDMA Data In-transfer

Figure 12-7 shows timing for host termination of a UDMA data in-transfer.

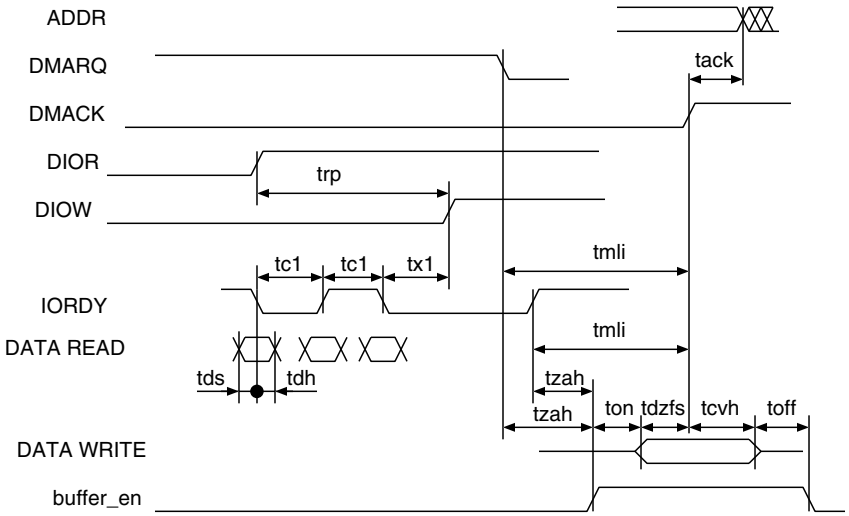


Figure 12-7. Timing for Host Termination of UDMA Transfer

Figure 12-8 shows timing for device termination of a UDMA data in-transfer.

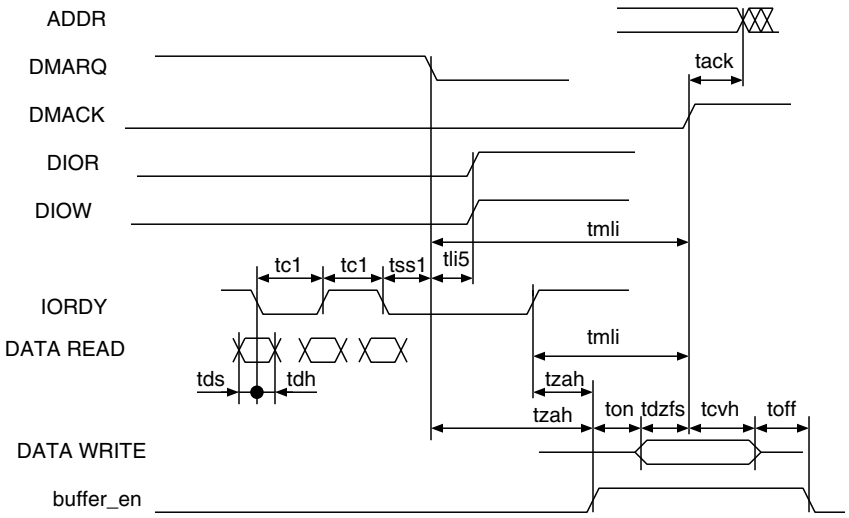


Figure 12-8. Timing for Device Termination of a UDMA Transfer

Timing parameters for UDMA data in-bursts are listed in Table 12-6. The UDMA in-burst timing parameters listed in the second column refer to Figure 12-6 through Figure 12-8.

Table 12-6. Timing Parameter Relations for UDMA Data In-Bursts

ATA Parameter	UDMA In-burst Timing Parameter	Relation	Programmable Value
tack	tack	tack(min) = (time_ack * T) - (tskew1 + tskew2)	time_ack
tenv	tenv	tenv(min) = (time_env * T) - (tskew1 + tskew2) tenv(max) = (time_env * T) + (tskew1 + tskew2)	time_env
trp	trp	trp(min) = time_rp * T - (tskew1 + tskew2 + tskew6)	time_rp
—	tx1[1]	(time_rp * T) - (tco + tsu + 3T + 2 *tbuf + 2*tcable2) > trfs (drive)	time_rp
tmli	tmli1	tmli1(min) = (time_mlix + 0.4) * T	time_mlix
tzah	tzah	tzah(min) = (time_zah + 0.4) * T	time_zah
tdzfs	tdzfs	tdzfs = (time_dzfs * T) - (tskew1 + tskew2)	time_dzfs
tcvh	tcvh	tcvh = (time_cvh *T) - (tskew1 + tskew2)	time_cvh
—	ton toff[2]	ton = time_on * T - tskew1 toff = time_off * T - tskew1	—

1 There is a special timing requirement in the ATA host that requires the internal DIOW to go only high three clocks after the last active edge on the DSTROBE signal. The equation given on this line tries to capture this constraint.

2 Make ton and toff big enough to avoid bus contention.

12.2.2.5 Timing for UDMA Data Out-Transfers

Figure 12-9 shows the timing for the start of a UDMA data out-transfer.

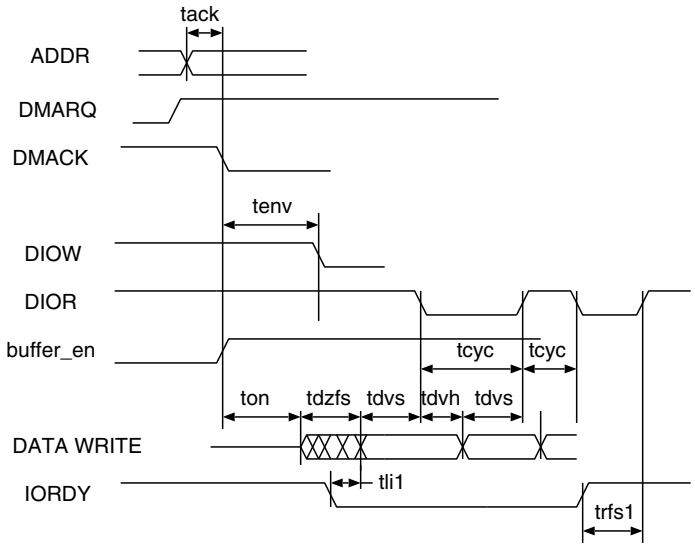


Figure 12-9. Timing of Start of UDMA Data Out-Transfer

Figure 12-10 shows the timing for host termination of a UDMA data out-transfer.

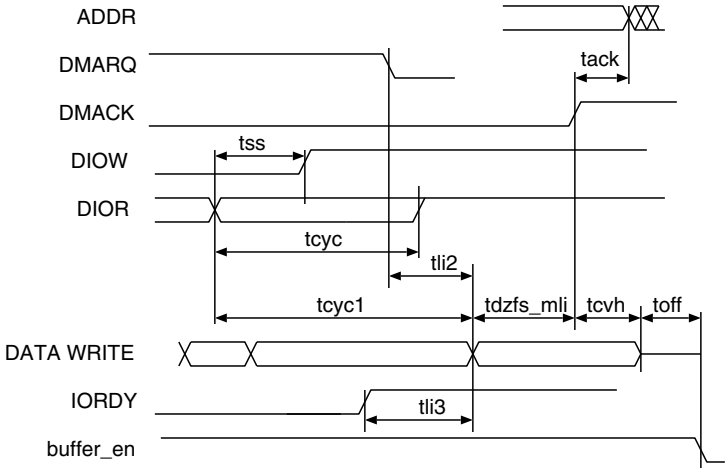


Figure 12-10. Timing of Host Termination of UDMA Data Out-Transfer

Figure 12-11 shows timing for device termination of a UDMA data out-transfer.

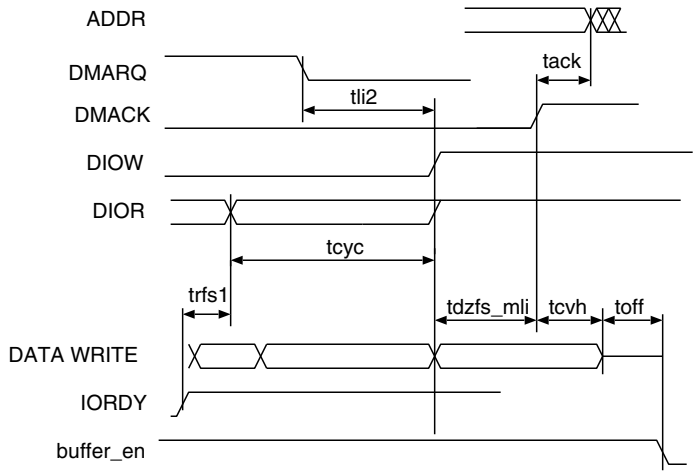


Figure 12-11. Timing for Device Termination of UDMA Data Out-Transfer

Timing parameters and relations for UDMA data out-bursts are listed in Table 12-7. The UDMA out-burst timing parameters listed in the second column refer to Figure 12-9, Figure 12-10, and Figure 12-11.

Table 12-7. Timing Parameters and Relations for UDMA Out-Burst

ATA Parameter	UDMA Out-burst Timing Parameter	Timing Relation	Programmable Value
tack	tack	tack(min) = (time_ack * T) - (tskew1 + tskew2)	time_ack
tenv	tenv	tenv(min) = (time_env * T) - (tskew1 + tskew2) tenv(max) = (time_env * T) + (tskew1 + tskew2)	time_env
tdvs	tdvs	tdvs = (time_dvs * T) - (tskew1 + tskew2)	time_dvs
tdvh	tdvh	tdvs = (time_dvh * T) - (tskew1 + tskew2)	time_dvh
tcyc	tcyc	tcyc = time_cyc * T - (tskew1 + tskew2)	time_cyc
t2cyc	—	t2cyc = time_cyc * 2 * T	time_cyc
trfs1	trfs	trfs = 1.6 * T + tsui + tco + tbuf + tbuf	—
—	tdzfs	tdzfs = time_dzfs * T - (tskew1)	time_dzfs
tss	tss	tss = time_ss * T - (tskew1 + tskew2)	time_ss
tmli	tdzfs_mli	tdzfs_mli =max(time_dzfs, time_mli) * T - (tskew1 + tskew2)	—
tli	tli1	tli1 > 0	—
tli	tli2	tli2 > 0	—
tli	tli3	tli3 > 0	—
tcvh	tcvh	tcvh = (time_cvh *T) - (tskew1 + tskew2)	time_cvh
—	ton toff	ton = time_on * T - tskew1 toff = time_off * T - tskew1	—

12.3 Advanced DMA in DMA Master Mode

Two optional DMA algorithms are supported for the Host Controller in DMA master mode:

- Single DMA (SDMA): After an SDMA transfer command is completed, a DMA interrupt is generated and the new system address is programmed by the Host Driver. The 32-bit System Address Register is used as a data pointer and limited to 32-bit system memory addressing.
- Advanced DMA (ADMA): ADMA defines a programmable descriptor table in the system memory. The Host Driver can calculate the system address at the sector boundary and programs the descriptor table before executing ADMA. This reduces the frequency of interrupts to the host system, and higher speed DMA transfer is realized because a Host CPU intervention is not needed during a long DMA-based data transfer. Furthermore, ADMA can support 32-bit or 64-bit system memory addressing. In ADMA, the 64-bit ADMA System Address Register is used as a Descriptor pointer instead of the 32-bit System Address Register.

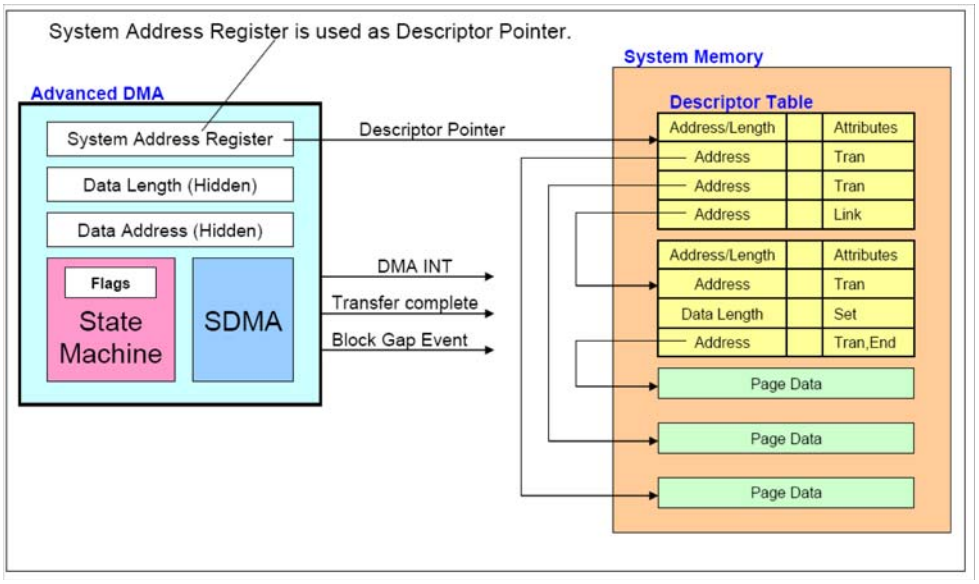


Figure 12-12. Host Controller Block Diagram

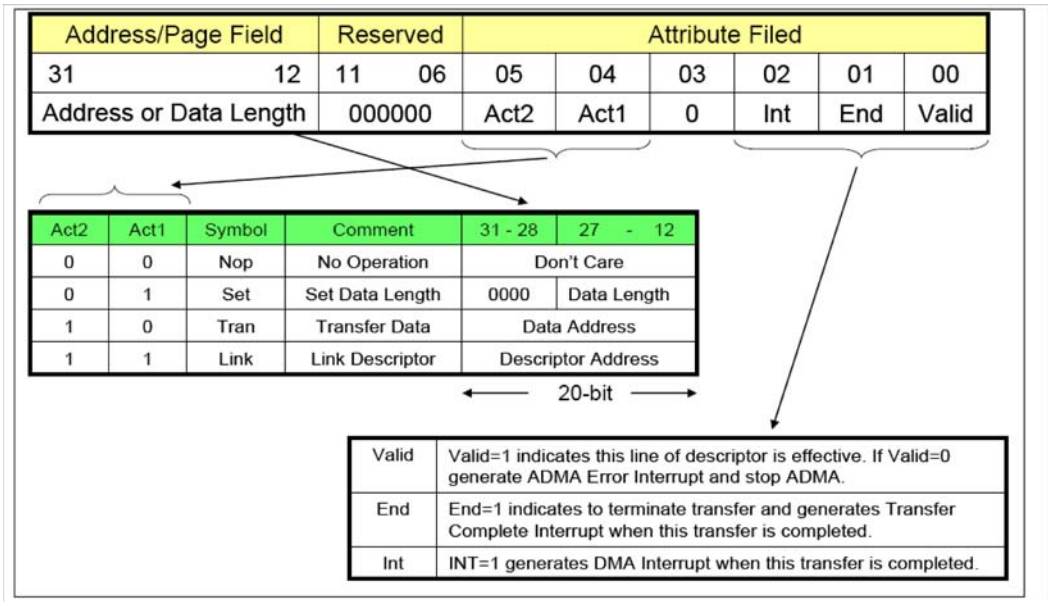


Figure 12-13. 32-Bit Descriptor Table Definition

NOTE

Data length is specified in the same way as the transfer Sector Size of the Sector Size Register. For example, 0x0001 indicates 1 byte, 0x0200 indicates 512 bytes, and so on. The data length should be a multiple of the sector size.

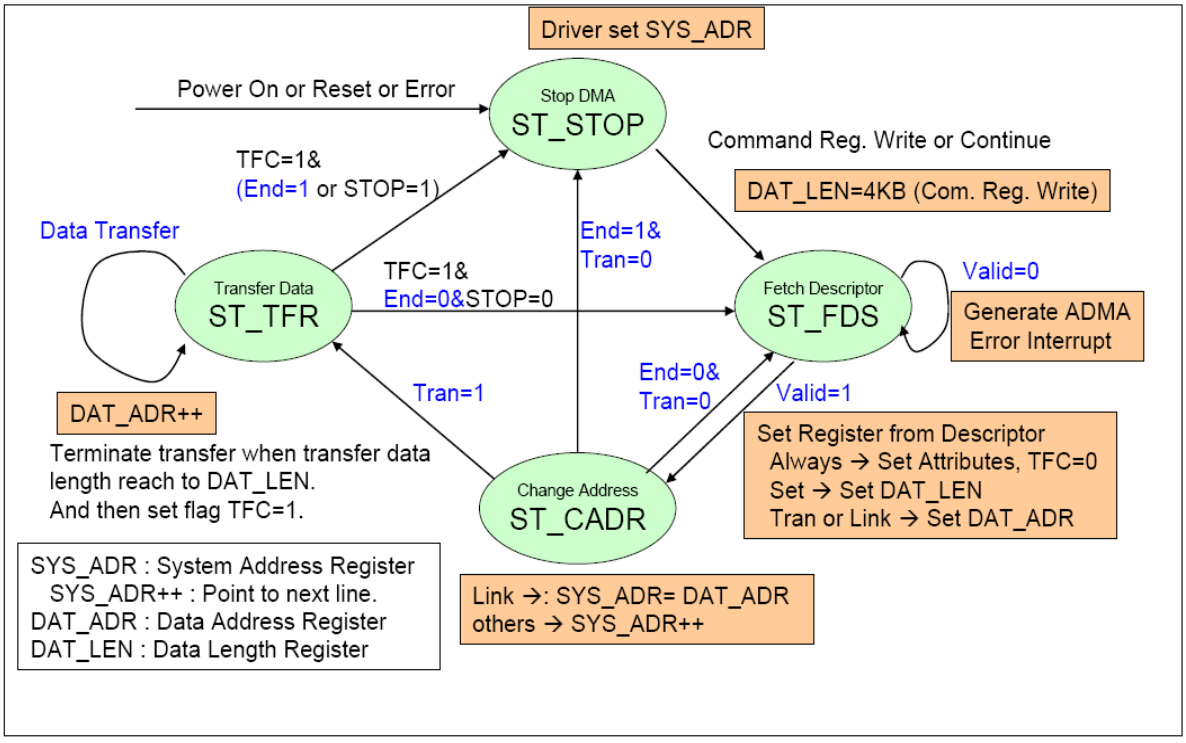


Figure 12-14. State Diagram for ADMA

12.4 Memory Map and Register Definitions

12.4.1 Memory Map

Table 12-8 shows the ATA memory map.

Table 12-8. ATA Memory Map

Address	Register	Description	Access	Reset Value	Section/Page
Offset: 00 (TIME_OFF)	TIME_OFF	transceiver timing parameter. Controls toff	R/W	0x01	12.4.3.2.1/12-24
Offset: 01 (TIME_ON)	TIME_ON	transceiver timing parameter. Controls ton	R/W	0x01	12.4.3.2.2/12-24
Offset: 02 (TIME_1)	TIME_1	PIO timing parameter. Controls t1	R/W	0x01	12.4.3.2.3/12-25
Offset: 03 (TIME_2W)	TIME_2W	PIO timing parameter. Controls t2 during write cycles	R/W	0x01	12.4.3.2.4/12-25
Offset: 04 (TIME_2R)	TIME_2R	PIO timing parameter. Controls t2 during read cycles	R/W	0x01	12.4.3.2.5/12-25
Offset: 05 (TIME_AX)	TIME_AX	PIO timing parameter. Controls tA	R/W	0x01	12.4.3.2.6/12-25
Offset: 06 (TIME_PIO_RDX)	TIME_PIO_RDX	PIO timing parameter. Controls trd	R/W	0x01	12.4.3.2.7/12-26
Offset: 07 (TIME_4)	TIME_4	PIO timing parameter. Controls t4	R/W	0x01	12.4.3.2.8/12-26
Offset: 08 (TIME_9)	TIME_9	PIO timing parameter. Controls t9	R/W	0x01	12.4.3.2.9/12-26
Offset: 09 (TIME_M)	TIME_M	MDMA timing parameter. Controls tm	R/W	0x01	12.4.3.2.10/12-26
Offset: 0A (TIME_JN)	TIME_JN	MDMA timing parameter. Controls tn and tj	R/W	0x01	12.4.3.2.11/12-27
Offset: 0B (TIME_D)	TIME_D	MDMA timing parameter. Controls td	R/W	0x01	12.4.3.2.12/12-27
Offset: 0C (TIME_K)	TIME_K	MDMA timing parameter. Controls tk	R/W	0x01	12.4.3.2.13/12-27
Offset: 0D (TIME_ACK)	TIME_ACK	UDMA timing parameter. Controls tack	R/W	0x01	12.4.3.2.14/12-27
Offset: 0E (TIME_ENV)	TIME_ENV	UDMA timing parameter. Controls tenv	R/W	0x01	12.4.3.2.15/12-28
Offset: 0F (TIME_RPX)	TIME_RPX	UDMA timing parameter. Controls trp	R/W	0x01	12.4.3.2.16/12-28
Offset: 10 (TIME_ZAH)	TIME_ZAH	UDMA timing parameter. Controls tzah	R/W	0x01	12.4.3.2.17/12-28
Offset: 11 (TIME_MLIX)	TIME_MLIX	UDMA timing parameter. Controls tmli	R/W	0x01	12.4.3.2.18/12-28
Offset: 12 (TIME_DVH)	TIME_DVH	UDMA timing parameter. Controls tdvh	R/W	0x01	12.4.3.2.19/12-29
Offset: 13 (TIME_DZFS)	TIME_DZFS	UDMA timing parameter. Controls tdzfs	R/W	0x01	12.4.3.2.20/12-29
Offset: 14 (TIME_DVS)	TIME_DVS	UDMA timing parameter. Controls tdvs	R/W	0x01	12.4.3.2.21/12-29
Offset: 15 (TIME_CVH)	TIME_CVH	UDMA timing parameter. Controls tcvh	R/W	0x01	12.4.3.2.22/12-29
Offset: 16 (TIME_SS)	TIME_SS	UDMA timing parameter. Controls tss	R/W	0x01	12.4.3.2.23/12-30
Offset: 17 (TIME_CYC)	TIME_CYC	UDMA timing parameter. Controls tcyc and t2cyc	R/W	0x01	12.4.3.2.24/12-30

Table 12-8. ATA Memory Map (continued)

Address	Register	Description	Access	Reset Value	Section/Page
Offset: 1C (FIFO_DATA_16)	FIFO_DATA_16	16 bit wide data port to/from FIFO	R/W	0x00	12.4.3.3.1/12-30
Offset: 18 (FIFO_DATA_32)	FIFO_DATA_32	32 bit wide data port to/from FIFO	R/W	0x0000	12.4.3.3.1/12-30
Offset: 20 (FIFO_FILL)	FIFO_FILL	FIFO filling in halfwords	Read only	0x00	12.4.3.3.2/12-31
Offset: 24 (ATA_CONTROL)	ATA_CONTROL	ATA interface control register	R/W	0x00	12.4.3.4/12-31
Offset: 28 (INTERRUPT_PENDING)	INTERRUPT_ PENDING	Interrupt pending register	Read only	0x1–[1]	12.4.3.5.1/12-33
Offset: 2C (INTERRUPT_ENABLE)	INTERRUPT_ ENABLE	Interrupt enable register	R/W	0x0–[1]	12.4.3.5.2/12-34
Offset: 30 (INTERRUPT_CLEAR)	INTERRUPT_CLEAR	Interrupt clear register	Write only	0x––[1]	12.4.3.5.3/12-35
Offset: 34 (FIFO_ALARM)	FIFO_ALARM	FIFO alarm threshold	R/W	0x00	12.4.3.6/12-36
Offset: 38 (ADMA_ERR_STATUS)	ADMA_ERR_STATUS	ADMA error status register	Read only	0x00	12.4.3.7/12-36
Offset: 3C (SYS_DMA_BADDR)	SYS_DMA_BADDR	single DMA transfer start base address	R/W	0x00	12.4.3.8/12-37
Offset: 40 (ADMA_SYS_ADDR)	ADMA_SYS_ADDR	ADMA descriptor table start address	R/W	0x00	12.4.3.9/12-38
Offset: 48 (BLOCK_CNT)	BLOCK_CNT	sector number for DMA transfer	R/W	0x00	12.4.3.10/12-39
Offset: 4C (BURST_LENGTH)	BURST_LENGTH	burst length for a burst transfer on AHB	R/W	0x10	12.4.3.11/12-39
Offset: 50 (SECTOR_SIZE)	SECTOR_SIZE	sector size for device	R/W	0x200	12.4.3.12/12-39
Offset: A0[2] (DRIVE_DATA)	DRIVE_DATA	drive data register	16-bit RW	—	12.4.3.13/12-40
Offset: A4 (DRIVE_FEATURES)	DRIVE_FEATURES	drive features register	R/W	—	12.4.3.13/12-40
Offset: A8 (DRIVE_SECTOR_COUNT)	DRIVE_SECTOR_COUNT	drive sector count register	R/W	—	12.4.3.13/12-40
Offset: AC (DRIVE_SECTOR_NUM)	DRIVE_SECTOR_NUM	drive sector number register	R/W	—	12.4.3.13/12-40
Offset: B0 (DRIVE_CYL_LOW)	DRIVE_CYL_LOW	drive cylinder low register	R/W	—	12.4.3.13/12-40
Offset: B4 (DRIVE_CYL_HIGH)	DRIVE_CYL_HIGH	drive cylinder high register	R/W	—	12.4.3.13/12-40
Offset: B8 (DRIVE_DEV_HEAD)	DRIVE_DEV_HEAD	drive device head register	R/W	—	12.4.3.13/12-40
Offset: BC[3] (DRIVE_COMMAND)	DRIVE_COMMAND	drive command register	Write only	—	12.4.3.13/12-40

Table 12-8. ATA Memory Map (continued)

Address	Register	Description	Access	Reset Value	Section/Page
Offset: BC (DRIVE_STATUS)	DRIVE_STATUS	drive status register	Read only	—	12.4.3.13/12-40
Offset: D8[4] (DRIVE_ALT_STATUS)	DRIVE_ALT_STATUS	Drive alternate status register	Read only	—	12.4.3.13/12-40
Offset: D8 (DRIVE_CONTROL)	DRIVE_CONTROL	Drive control register	Write only	—	12.4.3.13/12-40

1 Dashes in “Reset Value” column indicate that some bits in the register have no reset value.

2 An access at offset A0 reads or writes the 16-bit drive data register.

3 The drive command and drive status registers are both at offset BC. A write at offset BC accesses the drive command register, while a read at the same offset accesses the drive status register.

4 The drive alternate status and drive control registers are both at offset D8. A write at offset D8 accesses the drive control register, while a read at the same offset accesses the drive alternate status register.

12.4.2 Register Summary

Figure 12-15 shows the key to the register fields and Table 12-9 shows the register figure conventions.

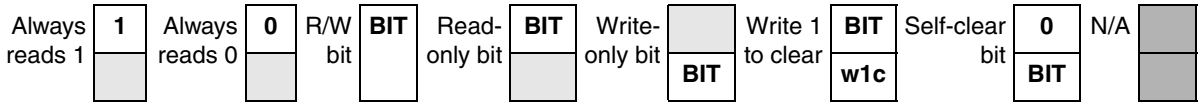


Figure 12-15. Key to Register Fields

Table 12-9. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit’s value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 12-9. Register Figure Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 12-10 summarizes the ATA registers. The registers at offsets A0 through D8 are addressable, but not present in the ATA interface module. These registers are not included in Table 12-10—they are described in Section 12.4.3.13, "Drive Registers Connected to ATA Bus."

Table 12-10. ATA Register Summary

Name		7	6	5	4	3	2	1	0
Offset: 00 (TIME_OFF)	R	TIME_OFF[7:0]							
	W								
Offset: 01 (TIME_ON)	R	TIME_ON[7:0]							
	W								
Offset: 02 (TIME_1)	R	TIME_1[7:0]							
	W								
Offset: 03 (TIME_2W)	R	TIME_2W[7:0]							
	W								
Offset: 04 (TIME_2R)	R	TIME_2R[7:0]							
	W								
Offset: 05 (TIME_AX)	R	TIME_AX[7:0]							
	W								
Offset: 06 (TIME_PIO_RDX)	R	TIME_RDX[7:0]							
	W								
Offset: 07 (TIME_4)	R	TIME_4[7:0]							
	W								
Offset: 08 (TIME_9)	R	TIME_9[7:0]							
	W								
Offset: 09 (TIME_M)	R	TIME_M[7:0]							
	W								
Offset: 0A (TIME_JN)	R	TIME_JN[7:0]							
	W								
Offset: 0B (TIME_D)	R	TIME_D[7:0]							
	W								

Table 12-10. ATA Register Summary (continued)

Name		7	6	5	4	3	2	1	0
Offset: 0C (TIME_K)	R	TIME_K[7:0]							
	W								
Offset: 0D (TIME_ACK)	R	TIME_ACK[7:0]							
	W								
Offset: 0E (TIME_ENV)	R	TIME_ENV[7:0]							
	W								
Offset: 0F (TIME_RPX)	R	TIME_RPX[7:0]							
	W								
Offset: 10 (TIME_ZAH)	R	TIME_ZAH[7:0]							
	W								
Offset: 11 (TIME_MLIX)	R	TIME_MLIX[7:0]							
	W								
Offset: 12 (TIME_DVH)	R	TIME_DVH[7:0]							
	W								
Offset: 13 (TIME_DZFS)	R	TIME_DZFS[7:0]							
	W								
Offset: 14 (TIME_DVS)	R	TIME_DVS[7:0]							
	W								
Offset: 15 (TIME_CVH)	R	TIME_CVH[7:0]							
	W								
Offset: 16 (TIME_SS)	R	TIME_SS[7:0]							
	W								
Offset: 17 (TIME_CYC)	R	TIME_CYC[7:0]fifo_data[15:0]							
	W								

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset: 1C (FIFO_DATA_16)	R	fifo_data[15:0]															
	W																

Table 12-10. ATA Register Summary (continued)

Name		7	6	5	4	3	2	1	0

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset: 18 (FIFO_DATA_32)	R	fifo_data[23:16]															
	W																
	R	fifo_data[15:0]															
	W																

Name		7	6	5	4	3	2	1	0
Offset: 20 (FIFO_FILL)	R	FIFO_FILL[7:0]							
	W								
Name		15	14	13	12	11	10	9	8
Offset: 24 (ATA_CONTROL)	R				dma_srst	dma_sel		dma_start_stop	dma_enable
	W								
		7	6	5	4	3	2	1	0
	R	fifo_rst_b	ata_rst_b	fifo_tx_en	fifo_rcv_en	dma_pending	dma_ultra_selected	dma_write	iordy_en
	W								
Name		7	6	5	4	3	2	1	0
Offset: 28 (INTERRUPT_PENDING)	R	ata_intrq1	fifo_underflow	fifo_overflow	controller_idle	ata_irtrq2	dma_err	dma_trans_over	
	W								
Offset: 2C (INTERRUPT_ENABLE)	R	ata_intrq1	fifo_underflow	fifo_overflow	controller_idle	ata_irtrq2	dma_err	dma_trans_over	
	W								
Offset: 30 (INTERRUPT_CLEAR)	R								
	W		fifo_underflow	fifo_overflow			dma_err	dma_trans_over	
Offset: 34 (FIFO_ALARM)	R	FIFO_ALARM[7:0]							
	W								

Table 12-10. ATA Register Summary (continued)

Name		7	6	5	4	3	2	1	0
Offset: 38 (ADMA_ERR_STATUS)	R						adma_len_mismatch	adma_err_state	
	W								

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset: 3C (SYS_DMA_BADDR)	R	sys_dma_baddr[15:0]															
	W																
	R	sys_dma_baddr[31:16]															
	W																
Offset: 40 (ADMA_SYS_ADDR)	R	adma_sys_addr[15:0]															
	W																
	R	adma_sys_addr[31:16]															
	W																

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset: 48 (BLOCK_CNT)	R	block_cnt															
	W																
Offset: 4C (BURST_LENGTH)	R											burst_length[5:0]					
	W																
Offset: 50 (SECTOR_SIZE)	R	sector_size															
	W																

12.4.3 Register Descriptions

This section contains the detailed register descriptions for the ATA registers.

12.4.3.1 Byte Order

The ATA interface works both in little-endian or big-endian mode. The addresses of all registers are the same in either mode. The 16-bit and 32-bit registers represent strings of 2 or 4 bytes. The byte order in the 16-bit or 32-bit register is dependent on the mode selection.

- Little-endian mode, 16- or 32-bit register:
  - bits [7:0]: byte 0
  - bits [15:8]: byte 1

  - bits [23:8]: byte 2
  - bits [31:24]: byte 3
- Big-endian mode, 32-bit register
  - bits [31:24]: byte 0
  - bits [23:16]: byte 1
  - bits [15:8]: byte 2
  - bits [7:0]: byte 3
- Big-endian, 16-bit register
  - bits [15:8]: byte 0
  - bits [7:0]: byte 1

12.4.3.2 Timing Registers

The registers at offsets 0x00 through 0x17 contain timing parameters. These timing parameters control the timing on the ATA bus.

Every timing parameter is 8 bits wide and can assume valid values between 0x01 and 0xFF. The reset value for all registers is 0x01.

12.4.3.2.1 TIME_OFF Register

Figure 12-16 shows the valid bits in the TIME_OFF register, and Table 12-8 describes the bit fields.

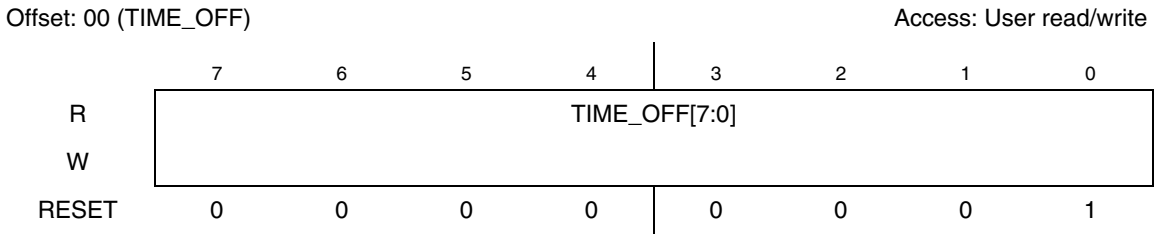


Figure 12-16. TIME_OFF Register

12.4.3.2.2 TIME_ON Register

Figure 12-17 shows the valid bits in the TIME_ON register, and Table 12-8 describes the bit fields.

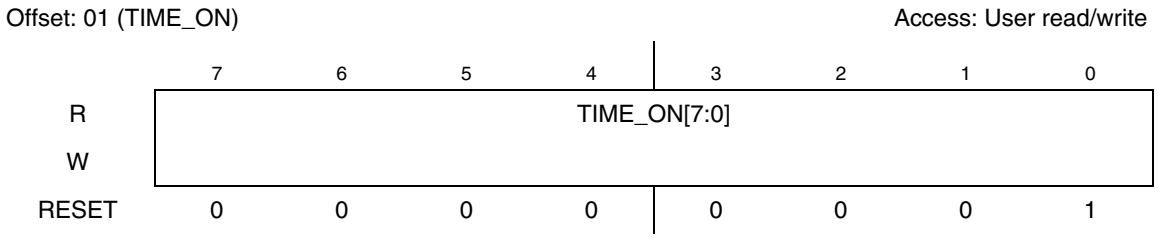


Figure 12-17. TIME_ON Register

12.4.3.2.3 TIME_1 Register

Figure 12-18 shows the valid bits in the TIME_1 register, and Table 12-8 describes the bit fields.

Offset: 02 (TIME_1) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_1[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-18. TIME_1 Register

12.4.3.2.4 TIME_2W Register

Figure 12-19 shows the valid bits in the TIME_2W register, and Table 12-8 describes the bit fields.

Offset: 03 (TIME_2W) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_2W[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-19. TIME_2W Register

12.4.3.2.5 TIME_2R Register

Figure 12-20 shows the valid bits in the TIME_2R register, and Table 12-8 describes the bit fields.

Offset: 04 (TIME_2R) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_2R[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-20. TIME_2R Register

12.4.3.2.6 TIME_AX Register

Figure 12-21 shows the valid bits in the TIME_AX register, and Table 12-8 describes the bit fields.

Offset: 05 (TIME_AX) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_AX[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-21. TIME_AX Register

12.4.3.2.7 TIME_PIO_RDX Register

Figure 12-22 shows the valid bits in the TIME_PIO_RDX register, and Table 12-8 describes the bit fields.

Offset: 06 (TIME_PIO_RDX) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_RDX[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-22. TIME_PIO_RDX Register

12.4.3.2.8 TIME_4 Register

Figure 12-23 shows the valid bits in the TIME_4 register, and Table 12-8 describes the bit fields.

Offset: 07 (TIME_4) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_4[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-23. TIME_4 Register

12.4.3.2.9 TIME_9 Register

Figure 12-24 shows the valid bits in the TIME_9 register, and Table 12-8 describes the bit fields.

Offset: 08 (TIME_9) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_9[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-24. TIME_9 Register

12.4.3.2.10 TIME_M Register

Figure 12-25 shows the valid bits in the TIME_M register, and Table 12-8 describes the bit fields.

Offset: 09 (TIME_M) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_M[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-25. TIME_M

12.4.3.2.11 TIME_JN Register

Figure 12-26 shows the valid bits in the TIME_JN register, and Table 12-8 describes the bit fields.

Offset: 0A (TIME_JN) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_JN[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-26. TIME_JN Register

12.4.3.2.12 TIME_D Register

Figure 12-27 shows the valid bits in the TIME_D register, and Table 12-8 describes the bit fields.

Offset: 0B (TIME_D) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_D[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-27. TIME_D Register

12.4.3.2.13 TIME_K Register

Figure 12-28 shows the valid bits in the TIME_K register, and Table 12-8 describes the bit fields.

Offset: 0C (TIME_K) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_K[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-28. TIME_K Register

12.4.3.2.14 TIME_ACK Register

Figure 12-29 shows the valid bits in the TIME_ACK register, and Table 12-8 describes the bit fields.

Offset: 0D (TIME_ACK) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_ACK[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-29. TIME_ACK Register

12.4.3.2.15 TIME_ENV Register

Figure 12-30 shows the valid bits in the TIME_ENV register, and Table 12-8 describes the bit fields.

Offset: 0E (TIME_ENV) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_ENV[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-30. TIME_ENV Register

12.4.3.2.16 TIME_RPX Register

Figure 12-31 shows the valid bits in the TIME_RPX register, and Table 12-8 describes the bit fields.

Offset: 0F (TIME_RPX) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_RPX[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-31. TIME_RPX Register

12.4.3.2.17 TIME_ZAH Register

Figure 12-32 shows the valid bits in the TIME_ZAH register, and Table 12-8 describes the bit fields.

Offset: 10 (TIME_ZAH) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_ZAH[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-32. TIME_ZAH Register

12.4.3.2.18 TIME_MLIX Register

Figure 12-33 shows the valid bits in the TIME_MLIX register, and Table 12-8 describes the bit fields.

Offset: 11 (TIME_MLIX) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_MLIX[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-33. TIME_MLIX

12.4.3.2.19 TIME_DVH Register

Figure 12-34 shows the valid bits in the TIME_DVH register, and Table 12-8 describes the bit fields.

Offset: 12 (TIME_DVH) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_DVH[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-34. TIME_DVH Register

12.4.3.2.20 TIME_DZFS Register

Figure 12-35 shows the valid bits in the TIME_DZFS register, and Table 12-8 describes the bit fields.

Offset: 13 (TIME_DZFS) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_DZFS[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-35. TIME_DZFS Register

12.4.3.2.21 TIME_DVS Register

Figure 12-36 shows the valid bits in the TIME_DVS register, and Table 12-8 describes the bit fields.

Offset: 14 (TIME_DVS) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_D[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-36. TIME_DVS

12.4.3.2.22 TIME_CVH Register

Figure 12-37 shows the valid bits in the TIME_CVH register, and Table 12-8 describes the bit fields.

Offset: 15 (TIME_CVH) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_CVH[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-37. TIME_CVH Register

12.4.3.2.23 TIME_SS Register

Figure 12-38 shows the valid bits in the TIME_SS register, and Table 12-8 describes the bit fields.

Offset: 16 (TIME_SS) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_SS[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-38. TIME_SS Register

12.4.3.2.24 TIME_CYC Register

Figure 12-39 shows the valid bits in the TIME_CYC register, and Table 12-8 describes the bit fields.

Offset: 17 (TIME_CYC) Access: User read/write

	7	6	5	4	3	2	1	0
R	TIME_CYC[7:0]							
W								
RESET	0	0	0	0	0	0	0	1

Figure 12-39. TIME_CYC

12.4.3.3 FIFO Data Registers

12.4.3.3.1 FIFO_Data Register in 16-bit and 32-bit Modes (FIFO_DATA_16, FIFO_DATA_32)

The FIFO_DATA register is used to read or write data to the internal FIFO. It can be accessed as a 16-bit register or as a 32-bit register. Word writes (respectively long writes) put two bytes (respectively four bytes) into the FIFO. Word reads (respectively long reads) read two bytes (respectively four bytes) from the FIFO.

Figure 12-40 shows the valid bits in the FIFO_Data Register in 16-bit mode.

Offset: 1C (FIFO_DATA_16) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIFO_DATA[15:0]															
W																
RESET	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 12-40. FIFO_Data Register In 16-bit Mode

Figure 12-41 shows the valid bits in the FIFO_Data Register in 32-bit mode.

Offset: 18 (FIFO_DATA_32) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIFO_DATA[32:16]															
RESET	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIFO_DATA[15:0]															
RESET	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 12-41. FIFO_Data Register in 32-bit Mode

12.4.3.3.2 FIFO Fill Register (FIFO_FILL)

FIFO_FILL is a read-only register. Any read returns the current number of halfwords present in the FIFO.

Figure 12-42 shows the valid bits in the FIFO_FILL register.

Offset: 20 (FIFO_FILL) Access: User read-only

	7	6	5	4	3	2	1	0
R	FIFO_FILL[7:0]							
W								
RESET	0	0	0	0	0	0	0	0

Figure 12-42. FIFO_FILL Register

12.4.3.4 ATA Control Register (ATA_CONTROL)

Figure 12-43 shows the valid bits in the ATA control register, and Table 12-11 describes the bit fields.

Offset: 24 (ATA_CONTROL) Access: User read/write

	15	14	13	12	11	10	9	8
R W				dma_srst	dma_select		dma_start_stop	dma_enable
RESET:	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R W	fifo_rst_b	ata_rst_b	fifo_tx_en	fifo_rcv_en	dma_pend-ing	dma_ultra_selected	dma_write	iordy_en
RESET:	0	0	0	0	0	0	0	0

Figure 12-43. ATA Control Register

Table 12-11. ATA Control Register Field Descriptions

Field	Description
15-13 Reserved	N/A
12 dma_srst	This field controls if internal DMA controller is in reset or enabled 0 > internal DMA controller normal operation 1 > internal DMA controller reset
11-10 dma_select	This field controls DMA mode selected 00 > Single DMA select 01 > 32-bit ADMA select 10 > 64-bit ADMA select 11 > reserved
9 dma_start_stop	This field controls DMA master start or stop 0 > stop DMA master 1 > start DMA master
8 dma_enable	This field controls DMA master enable 0 > DMA master is disable 1 > DMA master is enable
7 fifo_rst_b	This field controls the internal FIFO reset 0 FIFO reset 1 FIFO normal operation
6 ata_rst_b	This bit controls the level on the ata_reset_b pin, which controls the reset of the internal ATA protocol engine. 0 ata_reset_b = 0, ATA drive is reset, and internal protocol engine reset. 1 ata_reset_b = 1, ATA drive is not reset; internal protocol engine normal operation.
5 fifo_tx_en	FIFO transmit enable. This bit controls whether the FIFO makes transmit data requests to the DMA. If enabled, the FIFO requests the DMA to refill it whenever the FIFO level drops below the alarm threshold. 0 FIFO refill by DMA disabled 1 FIFO refill by DMA enabled
4 fifo_rcv_en	FIFO receive enable. This bit controls whether the FIFO makes receive data requests to the DMA. If enabled, the FIFO requests the DMA to empty it whenever the FIFO level meets or exceeds the alarm threshold. 0 FIFO empty by DMA disabled 1 FIFO empty by DMA enabled
3 dma_pending	DMA pending bit. This bit controls whether the ATA interface responds to a DMA request originating in the drive. If this bit is set, the ATA interface starts a MDMA or UDMA burst whenever the drive asserts the ata_dmarq signal. 0 ATA interface does not start DMA burst 1 ATA interface starts MDMA or UDMA burst whenever drive asserts dmarq
2 dma_ultra_selected	This bit determines the protocol (UDMA or MDMA) for any new DMA burst 1=UDMA protocol is used 0=MDMA protocol is used

Table 12-11. ATA Control Register Field Descriptions (continued)

Field	Description
1 dma_write	This bit determines the data direction on any new DMA burst 1=DMA out burst, ATA interface writes to drive 0=DMA in burst, ATA interface reads from drive
0 iordy_en	This bit determines whether ata_iordy handshake is used during PIO mode 1=IORDY handshake is used 0=IORDY is disregarded

12.4.3.5 Interrupt Registers

The three interrupt registers control the interrupt interface between the ATA module and the CPU/DMA. Two interrupts (ipbus_int and fifo_txfe_end_alarm) are controlled by these registers, as follows:

- The ipbus_int interrupt is controlled by bits 1,2,3,4,5 and 6 of the interrupt registers. It is asserted if one of the 6 bits is set in the interrupt_pending register, while the same bit is set in the interrupt_enable register. This interrupt goes to the CPU.
- The fifo_txfer_end_alarm interrupt is controlled by bit 7 of the interrupt registers. If ata_intrq1 is set in both the interrupt enable and interrupt pending register, then fifo_txfer_end_alarm is asserted. The purpose of this interrupt is to inform the DMA that the running data transfer has ended. This interrupt goes to the smart DMA.

12.4.3.5.1 Interrupt Pending Register (INTERRUPT_PENDING)

Figure 12-44 shows the valid bits in the interrupt pending register, and Table 12-12 describes the bit fields.

Offset: 28 (INTERRUPT_PENDING) Access: User read-only

	7	6	5	4	3	2	1	0
R	ata_intrq1	fifo_ underflow	fifo_ overflow	controller_ idle	ata_irtrq2	dma_err	dma_trans _over	
W								
RESET	0[1]	0	0	1	0	0	0	—

Figure 12-44. Interrupt_Pending Register

[1] Interrupts ata_intrq1 and ata_intrq2 only reset to 0 if during reset the interrupt input is low.

Table 12-12. Interrupt Pending Register Field Description

Field	Description
7 ata_intrq1	ATA interrupt request 1. This bit reflects the value of the ata_intrq interrupt input. It is set in the when the drive interrupt is pending, and cleared otherwise. The interrupt clear register has no influence on this bit.
6 fifo_underflow	FIFO underflow. This bit reports FIFO underflow. Sticky bit. It is set when there is a FIFO underflow condition. It is cleared by writing a ‘1’ to this bit in the interrupt clear register.
5 fifo_overflow	FIFO overflow. This bit reports FIFO overflow. Sticky bit. It is set when there is a FIFO overflow condition. It is cleared by writing a ‘1’ to this bit in the interrupt clear register.

Table 12-12. Interrupt Pending Register Field Description (continued)

Field	Description
4 controller_idle	Controller Idle. This bit reports controller idle. It is set when the ATA protocol engine is idle, there is no activity on the ATA bus. It is cleared when there is activity on the ATA bus. The interrupt clear register has no influence on this bit.
3 ata_intrq2	ATA interrupt request 2. This bit reflects the value of the ata_intrq interrupt input. It is set when the drive interrupt is pending, and cleared otherwise. It has exactly same functioning as ata_intrq1, but this bit affects ipbus_int, while the other affects interrupt to the DMA.
2 dma_err	DMA error. This bit reflects Single DMA error or ADMA error interrupt.It is set in the interrupt pending register when AHB bus response error, ADMA read or write error and ADMA length mismatch error happen. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signaling interrupt to the CPU. It is cleared by writing a '1' to this bit in the interrupt clear register
1 dma_trans_over	DMA transfer over. This bit reflects Single DMA or ADMA read or write transfer over. It is set in the interrupt pending register when Single DMA or ADMA transfer is over without error. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signaling interrupt to the CPU. It is cleared by writing a '1' to this bit in the interrupt clear register.
0 Reserved	N/A

12.4.3.5.2 Interrupt Enable Register (INTERRUPT_ENABLE)

Figure 12-45 shows the valid bits in the interrupt enable register, and Table 12-13 describes the bit fields.

Offset: 2C (INTERRUPT_ENABLE) Access: User read/write

	7	6	5	4	3	2	1	0
R W	ata_intrq1	fifo_ underflow	fifo_ overflow	controller_ idle	ata_irtrq2	dma_err	dma_trans _over	
RESET	0	0	0	0	0	0	0	—

Figure 12-45. Interrupt_Enable Register

Table 12-13. Interrupt Enable Register Field Description

Field	Description
7 ata_intrq1	ATA interrupt request 1. If this bit is set, then fifo_txfer_end_alarm is driven to the DMA when the corresponding bit is set in the interrupt enable register. This informs the DMA that the current transfer is finished. The interrupt clear register has no influence on this bit.
6 fifo_underflow	FIFO underflow. If this bit is set, then ipbus_int is driven to the CPU when the corresponding bit is set in the interrupt enable register.
5 fifo_overflow	FIFO overflow. If this bit is set, then ipbus_int is driven to the CPU when the same bit is set in the interrupt enable register.
4 controller_idle	Controller Idle. If this bit is set, then ipbus_int is driven to the CPU when the same bit is set in the interrupt enable register. The interrupt clear register has no influence on this bit.

Table 12-13. Interrupt Enable Register Field Description

Field	Description
3 ata_intrq2	ATA interrupt request 2. This bit reflects the value of the ata_intrq interrupt input. If this bit is set, then ipbus_int is driven to the CPU when the same bit is set in the interrupt enable register. This informs the CPU that the drive is requesting attention. The interrupt clear register has no influence on this bit.
2 dma_err	DMA error. This bit reflects Single DMA error or ADMA error interrupt.It is set in the interrupt pending register when AHB bus response error, ADMA read or write error and ADMA length mismatch error happen. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signaling interrupt to the CPU. It is cleared by writing a ‘1’ to this bit in the interrupt clear register
1 dma_trans_over	DMA transfer over. This bit reflects Single DMA or ADMA read or write transfer over. It is set in the interrupt pending register when Single DMA or ADMA transfer is over without error. When the bit is set in the interrupt pending register, and the same bit is set in the interrupt enable register, ipbus_int is active, signaling interrupt to the CPU. It is cleared by writing a ‘1’ to this bit in the interrupt clear register.
0 Reserved	N/A

12.4.3.5.3 Interrupt Clear Register (INTERRUPT_CLEAR)

Figure 12-46 shows the valid bits in the interrupt clear register, and Table 12-14 describes the bit fields.

Offset: 30 (INTERRUPT_CLEAR) Access: User write-only

	7	6	5	4	3	2	1	0
R								
W		fifo_ underflow	fifo_ overflow			dma_err	dma_trans _overf	
RESET	—	—	—	—	—	—	—	—

Figure 12-46. Interrupt_Clear Register

Table 12-14. Interrupt Clear Register Field Description

Field	Description
7 Reserved	N/A
6 fifo_underflow	FIFO underflow. Writing ‘1’ to this bit clears the corresponding bit in the interrupt pending register.
5 fifo_overflow	FIFO overflow. Writing ‘1’ to this bit clears the corresponding bit in the interrupt pending register.
4–3 Reserved	N/A
2 dma_err	DMA error. Writing ‘1’ to this bit clears the corresponding bit in the interrupt pending register.

Table 12-14. Interrupt Clear Register Field Description (continued)

Field	Description
1 dma_trans_over	DMA transfer over. Writing ‘1’ to this bit clears the corresponding bit in the interrupt pending register.
0 Reserved	N/A

12.4.3.6 FIFO Alarm Register (FIFO_ALARM)

This register contains the threshold (in 16-bit halfword units) which triggers fifo_rcv_alarm or fifo_tx_alarm signals to the DMA interface.

- If the fifo_rcv_en bit is set in the ATA control register and fifo_fill ≥ fifo_alarm, then the fifo_rcv_alarm signal is asserted to request the DMA to empty the FIFO.
- If the fifo_tx_en bit is set in the ATA control register and fifo_fill < fifo_alarm), then the fifo_tx_alarm signal is asserted to request the DMA to refill the FIFO.

For single DMA and ADMA transfers, the recommended value for FIFO_ALARM is 0x20.

Figure 12-47 shows the valid bits in the FIFO alarm register.

Offset: 34 (FIFO_ALARM) Access: User read/write

	7	6	5	4	3	2	1	0
R W	FIFO_ALARM[7:0]							
RESET	0	0	0	0	0	0	0	0

Figure 12-47. FIFO Alarm Register

12.4.3.7 ADMA_ERR_STATUS Register

Figure 12-48 shows the valid bits in the ADMA_ERR_STATUS register.

Offset: 38 (ADMA_ERR_STATUS)

	7	6	5	4	3	2	1	0
R W	reserved					adma_len_mismatch	adma_err_state	
RESET:	0	0	0	0	0	0	0	0

Figure 12-48. ADMA_ERR_STATUS Register

This register indicates what kind of error has happened. When an ADMA error interrupt occurs, the ADMA error states field in this register holds the ADMA state, and the ADMA system address register

holds the address around the error descriptor. For recovering the error, the host driver requires the ADMA state to identify the error descriptor address as given in Table 12-15.

Table 12-15. ADMA Error States Register Field Descriptor

Fields	Descriptor
7-3 uncommitted	N/A
2 adma_len_mismatch	adma_len_mismatch. Total data length in descriptor table does not match the total sector data set by command
1-0 adma_err_state	adma_err_state. ADMA state when error occurs. 00 ST_STOP—Previous location set in the ADMA system address register is the error descriptor address. 01 ST_FDS—Current location set in the ADMA system address register is the error descriptor address. 10 ST_CARD—This state is never set because it only increments the descriptor pointer and does not generate an ADMA error. 11 ST_TFR—Previous location set in the ADMA system address register is the error descriptor address.

12.4.3.8 SYS_DMA_BADDR Register

See Figure 12-49 for illustration of valid bits in the SYS_DMA_BADDR register.

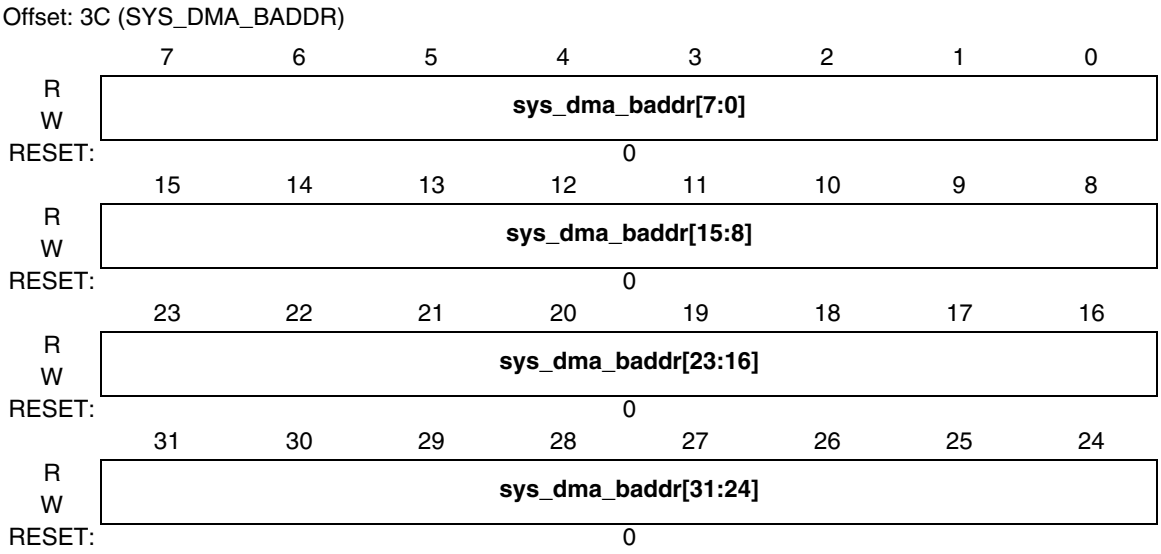


Figure 12-49. SYS_DMA_BADDR Register

This register contains the system memory address for a DMA transfer. When the host controller stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only if a transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register.

12.4.3.9 ADMA_SYS_ADDR Register

See Figure 12-50 for an illustration of valid bits in the ADMA_SYS_ADDR register.

Offset: 40 (ADMA_SYS_ADDR)

	7	6	5	4	3	2	1	0
R W	adma_sys_addr[7:0]							
RESET:	0							
	15	14	13	12	11	10	9	8
R W	adma_sys_addr[15:8]							
RESET:	0							
	23	22	21	20	19	18	17	16
R W	adma_sys_addr[23:16]							
RESET:	0							
	31	30	29	28	27	26	25	24
R W	adma_sys_addr[31:24]							
RESET:	0							
	39	8	37	36	35	34	33	32
R W	adma_sys_addr[39:32]							
RESET:	0							
	47	46	45	44	43	42	41	40
R W	adma_sys_addr[47:40]							
RESET:	0							
	55	54	53	52	51	50	49	48
R W	adma_sys_addr[55:48]							
RESET:	0							
	63	62	61	60	59	58	57	56
R W	adma_sys_addr[63:56]							
RESET:	0							

Figure 12-50. ADMA_SYS_ADDR Register

This register holds the byte address of the executing command of the descriptor table. The 32-bit descriptor uses the lower 32 bits of this register (only the 32-bit descriptor is supported). At the start of ADMA, the host driver sets the starting address of the descriptor table. The ADMA engine increments this register address on every fetch of a descriptor command. When the ADMA is stopped at the block gap, this register indicates the address of the next executable descriptor command. When the ADMA error interrupt is generated, this register holds the valid descriptor address depending on the ADMA state.

12.4.3.10 BLOCK_CNT Register

See Figure 12-51 for an illustration of valid bits in the BLOCK_CNT register.

Offset: 48 (BLOCK_CNT)

	7	6	5	4	3	2	1	0
R W	block_cnt[7:0]							
RESET:	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	block_cnt[15:8]							
RESET:	0	0	0	0	0	0	0	0

Figure 12-51. BLOCK_CNT Register

This register is block number set for a DMA transfer command. It can support 1 to 65535 blocks.

12.4.3.11 BURST_LENGTH Register

This register controls the burst length and may be set by software. Only burst lengths of 4 words are supported, so this register must be set to 4.

Figure 12-52 shows the valid bits in the BURST_LENGTH register.

Offset: 4C (BURST_LENGTH)

	7	6	5	4	3	2	1	0
R W			burst_length[5:0]					
RESET:	0	0	0	1	0	0	0	0

Figure 12-52. BURST_LENGTH Register

Table 12-16. BURST_LENGTH Register Field Descriptor

Field	Descriptor
7-6 Uncommitted	N/A
5-0 burst_length	Burst_length: 000001 = 1 words 00001x = 2 words 0001xx = 4 words 001xxx = 8 words 01xxxx = 16 words 1xxxxx = 32 words

12.4.3.12 SECTOR_SIZE Register

This register is for sector size set. The default value is 512 bytes.

Figure 12-53 shows the valid bits in the SECTOR_SIZE register.

Offset: 50 (SECTOR_SIZE)

	7	6	5	4	3	2	1	0
R W	sector_size							
RESET:	0	0	1	0	0	0	0	1

Figure 12-53. SECTOR_SIZE Register

12.4.3.13 Drive Registers Connected to ATA Bus

Some drive registers are addressable but are not present in the ATA interface module. These are listed in Table 12-17. If a read or write access is made to one of these registers, the read or write is mapped to a PIO read or write cycle on the ATA bus, and the corresponding register in the device attached to the ATA bus is accessed. No description of these registers is given here; consult the ATA specification for information on these registers.

If the drive_data register is accessed while the ATA interface operates in big-endian mode, the bytes to/from the ATA bus are swapped. No swaps occur in little-endian mode or for the other registers.

Table 12-17. Drive Registers connected to ATA Bus

Address	Name	Description	Access
Offset: A0 (DRIVE_DATA)	drive_data	Drive data register	R/W
Offset: A4 (DRIVE_FEATURES)	drive_features	Drive features register	R/W
Offset: A8 (DRIVE_SECTOR_COUNT)	drive_sector_count	Drive sector count register	R/W
Offset: AC (DRIVE_SECTOR_NUM)	drive_sector_num	Drive sector number register	R/W
Offset: B0 (DRIVE_CYL_LOW)	drive_cyl_low	Drive cylinder low register	R/W
Offset: B4 (DRIVE_CYL_HIGH)	drive_cyl_high	Drive cylinder high register	R/W
Offset: B8 (DRIVE_DEV_HEAD)	drive_dev_head	Drive device head register	R/W
Offset: BC (DRIVE_COMMAND)	drive_command	Drive command register	Write-only
Offset: BC (DRIVE_STATUS)	drive_status	Drive status register	Read-only
Offset: D8 (DRIVE_ALT_STATUS)	drive_alt_status	Drive alternate status register	Read-only
Offset: D8 (DRIVE_CONTROL)	drive_control	Drive control register	Write-only

12.5 Functional Description

The ATA interface provides two alternative modes for communication with the ATA peripherals connected to the ATA bus:

- PIO mode read/write operations through the ATA bus
- DMA transfers through the ATA bus, including:
  - DMA slave mode transfers between host and FIFO, which utilize the IPS bus
  - DMA master mode transfers between external memory and FIFO, which utilize the AHB bus

The operation of the peripheral under these two modes is described in detail in subsequent sections.

12.5.1 Resetting the ATA Bus

The ATA bus reset ata_reset_b is asserted whenever ata_rst_b (bit #6) of the ATA control register is cleared (see Section 12.4.3.4, “ATA Control Register (ATA_CONTROL)”). Clearing the bit also resets the ATA protocol engine. When the bit is set, the reset is released.

12.5.2 Programming ATA Bus Timing and iordy_en

The timing of the ATA interface is programmable using the 24 timing registers described in Section 12.4, “Memory Map and Register Definitions.” Programming the registers requires that the ATA bus be idle, so before reprogramming the user should ensure the following:

a) The dma_pending bit in the ATA control register is cleared (see Section 12.4.3.4, “ATA Control Register (ATA_CONTROL)”).
b) The controller_idle bit in the interrupt pending register is set (see Section 12.4.3.5.1, “Interrupt Pending Register (INTERRUPT_PENDING)”).

These two conditions can be met by first clearing dma_pending and then waiting until controller_idle is set before reprogramming the timing parameters. If dma_pending was set before the reprogramming started, it should be set again after the new timing is in effect to allow the drive to finish the current DMA transfer.

The bus timing should only be reprogrammed during an ongoing DMA transfer when the operating system requires a change in the bus clock (for example, in dynamic voltage frequency scaling).

The controller_idle bit must be set before reprogramming the bus timing: otherwise, the new timing values may affect a bus cycle that is still running and cause an error. PIO reads or writes to the ATA bus terminate after the bus cycle with the CPU has been terminated.

The iordy_en bit in the ATA control register determines whether the ATA interface responds to the drive’s IORDY signal. Just as with timing registers, it should only be reprogrammed when dma_pending is cleared and controller_idle is set.

12.5.3 Access to ATA Bus in PIO Mode

Before accessing the ATA bus in PIO mode, the user must do the following:

a) Set the ata_rst_b bit in the ATA control register
b) Program the timing parameters

The drive is accessed in PIO mode simply by reading or writing to the correct drive register. The bus cycle is translated to an ATA cycle, and the drive is accessed.

Reads and writes to the drive are not possible while the ATA bus is in reset; the attempted operation fails and is discarded.

12.5.4 Receiving Data from ATA Bus in DMA Slave Mode

In DMA receive slave mode, the protocol engine transfers data from the drive to the FIFO using the multiword DMA (MDMA) or ultra DMA (UMDMA) protocol. The transfer pauses when any of the following conditions occur:

- The FIFO is full.
- The drive negates its DMA request signal ata_dmarq.
- The dma_pending bit in the ata_control register is cleared.

When the condition is removed, the transfer restarts. At the end of the transfer, the drive signals the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register, which also indicates when the transfer has ended.

The transfer of data from FIFO to memory is handled by the host system DMA. DMA data transfers from device to host are set up as follows:

1. Make sure the ATA bus is not in reset and all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty or by resetting it.
3. Initialize the DMA channel connected to fifo_rcv_alarm. Every time the fifo_rcv_alarm is high, the DMA should read <packetsize> 32-bit words from the FIFO, and store them in main memory (here “packetsize” is defined as the number of 32-bit words in a packet; typically, packetsize = 8). This keeps the transfer going while avoiding FIFO overrun.
4. Write 2*<packetsize> to the FIFO alarm register (note that this setting corresponds to one packet of data, because the alarm threshold is specified in units of 16-bit halfwords). Thus the FIFO notifies the DMA when there is at least one packet ready for transfer.
5. Ensure the FIFO is out of reset by setting the fifo_rst_b bit in the ATA control register (see Section 12.4.3.4, “ATA Control Register (ATA_CONTROL)”).
6. Set the fifo_rcv_en bit in the ATA control register. This enables the DMA to empty the FIFO.
7. Set the dma_pending bit, and clear the dma_write bit. Also, program ultra_mode_selected=0 (for MDMA) or ultra_mode_selected=1 (for UDMA).

Steps (1–7) have prepared the host side of the DMA.

8. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus (consult the ATA specification for the specifics of this operation).
9. At this point, when the drive requests a DMA transfer by pulling ata_dmarq high, the ATA interface acknowledges with ata_dmack, and the transfer starts. Data is transferred automatically to the FIFO, and from there to the host memory.
10. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads cause the running DMA to pause; after the read is complete, the DMA resumes. Alternatively, the host can wait until the drive asserts ata_intrq, which also indicates end of transfer.
11. When the transfer ends, the host or host DMA should wait until the controller_idle bit is set in the ATA control register, then read the remaining halfwords from the FIFO and transfer these to memory. Note that there may be less than <packetsize> remaining 32-bit words, in which case the transfer is not performed automatically by the DMA.

12.5.5 Transmitting Data to ATA Bus in DMA Slave Mode

In DMA transmit mode, the protocol engine transfers data from the FIFO to the drive using the multiword DMA (MDMA) or ultra DMA (UDMA) protocol. The transfer pauses when one of following conditions occurs:

- The FIFO is empty
- The drive negates its DMA request signal ata_dmarq
- The dma_pending bit in the ata_control register is cleared

When the condition is removed, the transfer restarts. At the end of the transfer, the drive signals the host by asserting the ata_intrq signal. Alternatively, the host can read the device status register, which also indicates when the transfer has ended.

The transfer of data from FIFO to memory is handled by the host system DMA. DMA data transfers from host to device are set up as follows:

1. Make sure the ATA bus is not in reset, and all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty or by resetting it.
3. Initialize the DMA channel connected to fifo_tx_alarm. Every time the fifo_tx_alarm signal is high, the DMA should read <packetsize> 32-bit words from the main memory and write them to the FIFO (typical packetsize is 8 32-bit words). Program the DMA so that it does not transfer more than <sectorsize> 32-bit words total.
4. Write FIFO_SIZE – 2 × <packetsize> to the FIFO alarm register. In this way, FIFO notifies the DMA when there is room for at least one additional packet. FIFO_SIZE is given in 16-bit halfwords: the typical value is 64 halfwords, or 128 bytes.
5. Prepare the ATA for a DMA transfer from host to device as follows:
   a) Make sure the FIFO is out of reset by setting bit fifo_rst_b in the ATA control register (see Section 12.4.3.4, “ATA Control Register (ATA_CONTROL)”).
   b) Program fifo_tx_en=1 in the ATA control register. This enables the FIFO to be filled by DMA.
   c) Program dma_pending =1 and dma_write=1. Also, program ultra_mode_selected =0 (for MDMA) or ultra_mode_selected=1 (for UDMA).

Steps (1–5) have prepared the host side of the DMA.

6. Send commands to the drive in PIO mode that cause it to request a DMA transfer on the ATA bus (consult the ATA specification for the specifics of this operation).
7. At this point, when the drive requests a DMA transfer by pulling ata_dmarq high, the ATA interface acknowledges with ata_dmack, and the transfer starts. Data is transferred automatically from the FIFO, and also from host memory to FIFO.
8. During the transfer, the host can monitor for end of transfer by reading some device ATA registers. These reads cause the running DMA to pause; after the read is completed, the DMA resumes. Alternatively, the host can wait unit the drive asserts ata_intrq, which also indicates end of transfer.

When the transfer ends, no additional FIFO manipulations are needed.

12.5.6 Using DMA Master Mode to Receive Data From the ATA Bus

In DMA master mode for receiving data from the ATA bus, the internal DMA engine initializes a burst data write transfer on the AHB bus when the internal signal *sys_dma_req* is asserted.

Sys_dma_req is asserted when these conditions occur:

- *fifo_rcv_alarm* in DMA read
- In a DMA read, the data left in the FIFO is not enough to trigger the fifo_alarm, but the sector word counter has not reached a sector boundary

Sys_dma_req is negated when these conditions occur:

- A burst transfer ends
- DMA start is set
- The host is not active

For single DMA, burst length is the minimum set by software, FIFO alarm, and left data number in the last sector word counter.

For ADMA, burst length is the minimum set by software, FIFO alarm, left data number in the last sector word counter, and left data number indicated in a descriptor pair.

The following list describes the steps in setting up a DMA data transfer from the device to external memory:

1. Make sure that the ATA bus is not in reset, and that all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty, or by resetting it.
3. To make the ATA ready for a DMA transfer from device to host, do the following:
   a) Make sure the FIFO is out of reset by setting the bit fifo_rst_b to 1 in the ata_control register.
   b) Program fifo_rcv_en=1 in the ata_control register. This enables the FIFO to by emptied by the DMA.
   c) Program dma_pending =1, dma_write=0, ultra_mode_selected=0/1 in the ata_control register. ultra_mode_selected should be 1; it should be 0 if you want to transfer data using MDMA mode.
4. To make the internal DMA controller ready for a DMA transfer, do the following:
   a) Program fifo_alarm, burst_length, block_cnt, DMA system start base address (data buffer start address) for single DMA mode, or ADMA system address (descriptor table start address) for ADMA mode.
   b) Program dma_en = 1, dma_select=01 (if you select ADMA mode), dma_start_stop=1. If you select ADMA mode, make sure that the descriptor table is ready before starting a DMA transfer.
5. Now the host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request a DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. See the ATA specification for information on communicating with the drive.

6. When the drive requests a DMA transfer by pulling *ata_dmarq* high, the ATA interface acknowledges with *ata_dmack*, and the transfer starts. Data is transferred automatically to the FIFO, and then to the external memory.
7. When *sys_dma_req* is asserted, the DMA engine will write burst length data from FIFO to external memory.
8. During the transfer, the host can detect end of transfer by waiting for a dma_trans_over interrupt. After the end of transfer is detected, the host must reset dma_start_stop to 0.

12.5.7 Using DMA Master Mode To Transmit Data to the ATA bus

In DMA master mode for transmitting data to the ATA bus, the internal DMA engine initializes a burst data read transfer on the AHB bus when the internal signal *sys_dma_req* is asserted.

Sys_dma_req is asserted when these conditions occur:

- *fifo_tx_alarm* in DMA write

Sys_dma_req is negated when these conditions occur:

- A burst transfer ends
- DMA start is set
- The host is not active

For Single DMA, burst length is the minimum set by software and the FIFO alarm.

For ADMA, burst length is the minimum set by software, the FIFO alarm, and the left data number indicated in a descriptor pair.

The following list describes the steps in setting up a DMA data transfer from external memory to the device:

1. Make sure the ATA bus is not in reset, and all timing registers are programmed.
2. Make sure the FIFO is empty by reading it until empty, or by resetting it.
3. To make the ATA ready for a DMA transfer from host to device, do following:
   a) Make sure the FIFO is out of reset by setting bit *fifo_rst_b* to 1 in the ATA control register.
   b) Program *fifo_tx_en*=1 in *ata_control* register. This enables the FIFO to be filled by DMA.
   c) Program *dma_pending* =1, *dma_write*=1, *ultra_mode_selected*=0/1 in *ata_control* register. ultra_mod_selected should be 1 if you want to transfer data using UDMA mode, it should be 0 if you want to transfer data using MDMA mode.
4. To make the internal DMA controller ready for a DMA transfer, do the following:
   a) Program fifo_alarm, burst_length, block_cnt, and the DMA system start base address (data buffer start address) for single DMA mode, or ADMA system address (descriptor table start address) for ADMA mode.
   b) Program dma_en = 1, dma_select=01(if select ADMA mode), dma_start_stop=1. If select ADMA mode, it is necessary to make sure descriptor table is ready before start DMA transfer.

5. When *sys_dma_req* is asserted, DMA engine will read a burst length data from external memory to FIFO.
6. Now, the host side of the DMA is ready. Send commands to the drive in PIO mode that cause it to request DMA transfer on the ATA bus. The nature of these commands is beyond the scope of this document. You should consult the ATA specification to know how to communicate with the drive.
7. When the drive now requests DMA transfer by pulling *ata_dmarq* high, the ATA interface acknowledges with *ata_dmack*, and the transfer starts. Data is transferred automatically from the FIFO if FIFO is not empty.
8. During the transfer, the host can detect end of transfer by waiting for ata_intrq2 interrupt. After the end of transfer is detected, the host must reset dma_start_stop to 0.

12.6 Initialization and Application of ATA

If the host asserts RESET, the ATA device executes the hardware reset protocol, regardless of power management mode. For more details about the power-on and hardware reset protocol, see the ATA specification).

The host issues an IDENTIFY DEVICE command after the power-on reset or hardware reset protocol is complete, in order to determine the current status of features implemented by the device.

Chapter 13
Digital Audio Multiplexer (AUDMUX)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

Table 13-1 defines terms used in this chapter.

Table 13-1. Definition Of Terms

Term	Definition
PTCR	Port Timing Control Register
PDCR	Port Data Control Register
CNMCR	CE Bus Network Mode Control Register
FSPOL	Frame sync polarity
CLKPOL	Clock polarity
TFSDIR	Transmit Frame Sync Direction Control
TCLKDIR	Transmit Clock Direction Control
RFSDIR	Receive Frame Sync Direction Control
RCLKDIR	Receive Clock Direction Control
TFSEL	Transmit Frame Sync Port Select
RFSEL	Receive Frame Sync Port Select
TCSEL	Transmit Clock Port Select
RCSEL	Receive Clock Port Select
RXDSEL	Receive Data Port Select
INMMASK	Internal Network Mode Masking
CE Bus	Consumer Electronic Bus
CEN	CE Bus Enable
CNTHI	CE Bus disable signal High Period Count
CNTLO	CE Bus disable signal Low Period Count

13.1 Overview

The Digital Audio Mux (AUDMUX) provides a programmable interconnect device for voice, audio, and synchronous data routing between host serial interfaces (such as SSI) and peripheral serial interfaces (that is, audio and voice CODECs, also known as coder-decoders). The AUDMUX interconnections allow multiple, simultaneous, audio/voice/data flows between the ports in point-to-point or point-to-multipoint configurations. This section includes a top level diagram that shows the functional organization of the module, including all off-chip signals.

The AUDMUX allows the audio system connectivity to be modified through programming (as opposed to altering the PCB schematics of the system). Figure 13-1 shows the block diagram. The full description of the module is in Section 13.4, “Functional Description.”

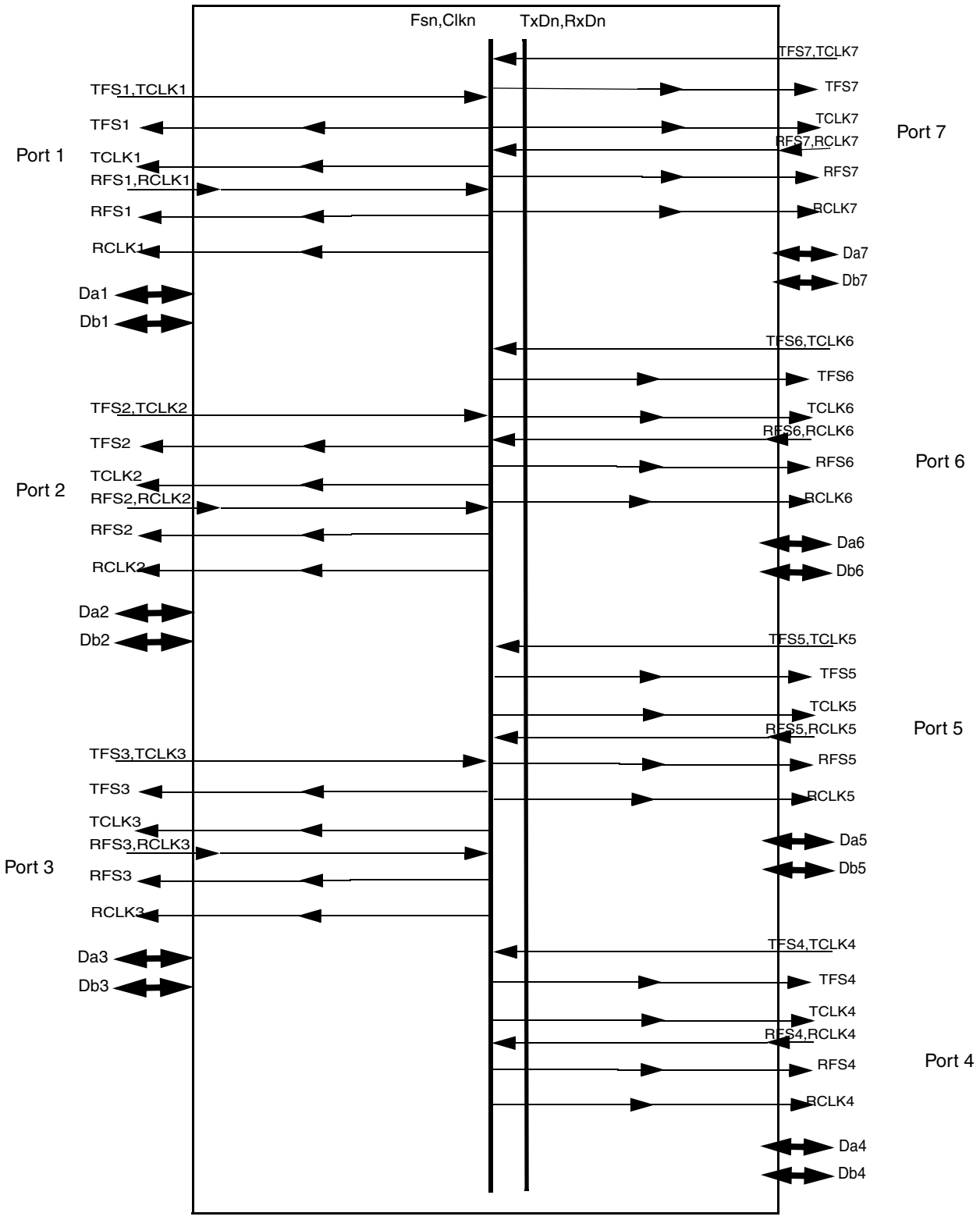


Figure 13-1. AUDMUX Block Diagram

13.1.1 Features

Key features of the module include the following:

- Three internal ports
- Four external ports
- Full 6-wire SSI interfaces for asynchronous receive and transmit
- Configurable 4-wire (synchronous) or 6-wire (asynchronous) peripheral interfaces
- Independent Tx/Rx Frame sync and clock direction selection for host or peripheral
- Each host interface's capability to connect to any other host or peripheral interface in a point-to-point or point-to-multipoint (network mode)
- Transmit and Receive Data switching to support external network mode
- CE Bus network mode to provide synchronous switching on RxD

13.1.2 Modes of Operation

Each AUDMUX port can be configured to operate in the following modes:

- Normal mode—the port is connected point-to-point to a single other port
- Internal network mode—the port is connected point-to-multipoint to multiple other ports
- CE bus network mode—the port receives data from the CE bus port (port 7) and one other port as selected by software.

Additionally, each port can operate in synchronous or asynchronous timing modes.

Modes of operation are described in more detail in Section 13.1.2, "Modes of Operation".

13.2 External Signal Description

Table 13-2 lists AUDMUX pin-level signals for the external ports, where:

- P*n* is P4, through P7
- $m = n - 3$.

The port is configured as an external port by a static system-level signal input p*n*_int_ext_select.

Table 13-2. Off-Chip Module Signals

Name	Module Port	I/O	Function	Reset State	Pull-up
AUDm_TXD	P*n*	I/O	Transmit Data from Pn	1	Active
AUDm_RXD	P*n*	I/O	Receive Data at Pn	1	Active
AUDm_TXC	P*n*	I/O	Transmit Clock input/output at Pn	1	—
AUDm_RXC	P*n*	I/O	Receive Clock input/output at Pn	1	—
AUDm_TXFS	P*n*	I/O	Transmit Frame sync input/output at Pn	1	—
AUDm_RXFS	P*n*	I/O	Receive Frame sync input/output at Pn	1	—

13.3 Memory Map and Register Definitions

This section includes the module memory map and detailed descriptions of all registers. For the base address of a particular module instantiation, see the system memory map.

13.3.1 Memory Map

The AUDMUX memory map is shown in Table 13-3.

Table 13-3. AUDMUX Memory Map

Address	Register	Access	Reset Value	Section/Page
0x0000 (PTCR1)	Port Timing Control Register 1	RW	0xAD40_0800	13.3.3.4/13-8
0x0004 (PDCR1)	Port Data Control Register 1	RW	0x0000_A000	13.3.3.5/13-10
0x0008 (PTCR2)	Port Timing Control Register 2	RW	0xA500_0800	13.3.3.4/13-8
0x000C (PDCR2)	Port Data Control Register 2	RW	0x0000_8000	13.3.3.5/13-10
0x0010 (PTCR3)	Port Timing Control Register 3	RW	0x9CC0_0800	13.3.3.4/13-8
0x0014 (PDCR3)	Port Data Control Register 3	RW	0x0000_6000	13.3.3.5/13-10
0x0018 (PTCR4)	Port Timing Control Register 4	RW	0x0000_0800	13.3.3.4/13-8
0x001C (PDCR4)	Port Data Control Register 4	RW	0x0000_4000	13.3.3.5/13-10
0x0020 (PTCR5)	Port Timing Control Register 5	RW	0x0000_0800	13.3.3.4/13-8
0x0024 (PDCR5)	Port Data Control Register 5	RW	0x0000_2000	13.3.3.5/13-10
0x0028 (PTCR6)	Port Timing Control Register 6	RW	0x0000_0800	13.3.3.4/13-8
0x002C (PDCR6)	Port Data Control Register 6	RW	0x0000_0000	13.3.3.5/13-10
0x0030 (PTCR7)	Port Timing Control Register 7	RW	0x0000_0800	13.3.3.4/13-8
0x0034 (PDCR7)	Port Data Control Register 7	RW	0x0000_C000	13.3.3.5/13-10
0x0038 (CNMCR)	CE Bus Network Mode Control Register (CNMCR)	RW	0x0003_1010	13.3.3.6/13-12

13.3.2 Register Summary

Table 13-4 shows the control register and address mapping for the AUDMUX.

Table 13-4. Register Summary

Address[1]		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
[(*n* - 1) x 8] + 0x0000 PTCR*n*	R	TFSDIR	TFSEL[3:0]				TCLKDIR	TCSEL[3:0]				RFSDIR	RFSEL[3:0]				RCLKDIR
	W																
	R	RCSEL[3:0]				SYN	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 13-4. Register Summary (continued)

Address[1]		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
[(*n* - 1) x 8] + 0x0004 PDCR*n*	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RXDSEL[2:0]			TX RX EN	0	0	MODE[1:0]		INMMASK[7:0]							
	W																
0x0038 (CNMCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN	FS POL	CLK POL
	W																
	R	CNTHI[7:0]								CNTLOW[7:0]							
	W																

1 *n* ranges from 1 through 7.

13.3.3 Register Descriptions

There are two configuration registers for each port. There is also a separate register for CE Bus Network mode control. Each pair of configuration registers is identical for each port; however, the default values following a reset differ as shown in Table 13-3.

- Section 13.4.3.1, "Default Port Configuration," describes the default configuration of the ports.
- Section 13.4.3.2, "Default CE Bus Configuration," describes the default configuration of the CE Bus.

13.3.3.1 Default Port Configuration

After a reset, each port defaults to normal mode (PDCR*n*[MODE] = 00) with synchronous timing mode (PTCR*n*[SYN] = 1) enabled.

The default port-to-port connections are as follows:

- Port 1 to Port 6
  - Port 6 provides the clock and frame sync.
- Port 2 to Port 5
  - Port 5 provides the clock and frame sync.
- Port 3 to Port 4
  - Port 4 provides the clock and frame sync.
- Port 7 to Port 7 (in data loopback mode)
  - Clock and frame syncs are inputs.

13.3.3.2 Default CE Bus Configuration

The default configuration of all the ports is to set the MODE field to normal mode (that is, no port selects CE Bus network mode). Correspondingly, the default configuration of the CEN field in the CNMCR is clear. To minimize AUDMUX setup for audio testing, the rest of the CNMCR fields default to the following configuration:

- FSPOL is set.
- CLKPOL is set.
- CNTHI is set to 16.
- CNTLOW is set to 16.

This configuration uses a frame sync that is logic high when asserted.

The clock is driven by the transmitter on the rising edge and is sampled by the receiver on the falling edge. The AUDMUX switches between devices on the rising edge; this provides a buffer of one-half clock period between the moment data is sampled and when the AUDMUX switches between devices. Refer to Figure 13-16 for a timing diagram where FSPOL and CLKPOL are both set.

CNTHI and CNTLOW are set to 16 to allow CE Bus to drive data during timeslot 1 and the other device to drive data during timeslots 0, 2,..., N-1 where N is the number of timeslots used and each timeslot has 16 bits.

13.3.3.3 Register Conventions

Figure 13-2 and Table 13-5 explain conventions used in register diagrams and tables.

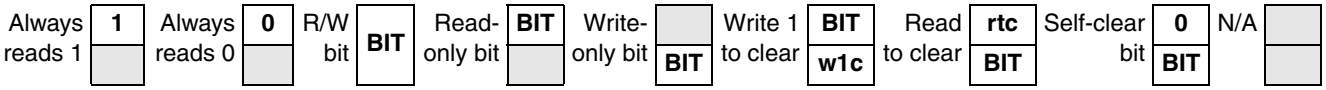


Figure 13-2. Register Field Conventions

Table 13-5. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)

Table 13-5. General Register Conventions (continued)

Convention	Description
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

13.3.3.4 Port Timing Control Register *n* (PTCR*n*)

PTCR*n* is the Port Timing Control Register for Port *n*, where *n* ranges from 1 through 7.

Offset 0x0000 (PTCR1) Access: User read/write
0x0008 (PTCR2)
0x0010 (PTCR3)
0x0018 (PTCR4)
0x0020 (PTCR5)
0x0028 (PTCR6)
0x0030 (PTCR7)

	31	30	27	26	25	22	21	20	17	16
R W	TFS DIR	TFSEL[3:0]		TCLK DIR	TCSEL[3:0]		RFS DIR	RFSEL[3:0]		RCLK DIR
Reset	(See Table 13-3 for reset values.)									

	15	12	11	10	0
R W	RCSEL[3:0]		SYN	—	
Reset	(See Table 13-3 for reset values.)				

Figure 13-3. Port Timing Control Register for Port *n*

Table 13-6. Port Timing Control Register Field Descriptions

Field	Description
31 TFS DIR	Transmit Frame Sync Direction Control. This bit sets the direction of the TxFS pin of the interface as an output or input. When set as an input, the TFSEL settings are ignored. When set as an output, the TFSEL settings determine the source port of the frame sync. 0 TxFS is an input. 1 TxFS is an output.
30–27 TFSEL[3:0]	Transmit Frame Sync Select. Selects the source port from which TxFS is sourced. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 … x110 Port 7 x111 Reserved
26 TCLKDIR	Transmit Clock Direction Control. This bit sets the direction of the TxClk pin of the interface as an output or input. When set as an input, the TCSEL settings are ignored. When set as an output, the TCSEL settings determine the source port of the clock. 0 TxClk is an input. 1 TxClk is an output.
25–22 TCSEL[3:0]	Transmit Clock Select. Selects the source port from which TxClk is sourced. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 … x110 Port 7 x111 Reserved
21 RFS DIR	Receive Frame Sync Direction Control. This bit sets the direction of the RxFS pin of the interface as an output or input. When set as an input, the RFSEL settings are ignored. When set as an output, the RFSEL settings determine the source port of the frame sync. 0 RxFS is an input. 1 RxFS is an output.
20–17 RFSEL[3:0]	Receive Frame Sync Select. Selects the source port from which RxFS is sourced. RxFS can be sourced from TxFS and RxFS from other ports. 0xxx Selects TxFS from port. 1xxx Selects RxFS from port. x000 Port 1 … x110 Port 7 x111 Reserved
16 RCLKDIR	Receive Clock Direction Control. This bit sets the direction of the RxClk pin of the interface as an output or input. When set as an input, the RCSEL settings are ignored. When set as an output, the RCSEL settings determine the source port of the clock. 0 RxClk is an input 1 RxClk is an output

Table 13-6. Port Timing Control Register Field Descriptions (continued)

Field	Description
15–12 RCSEL[3:0]	Receive Clock Select. Selects the source port from which RxClk is sourced. RxClk can be sourced from TxClk and RxClk from other ports. 0xxx Selects TxClk from port. 1xxx Selects RxClk from port. x000 Port 1 … x110 Port 7 x111 Reserved
11 SYN	Synchronous/Asynchronous Select. When SYN is set, synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals (that is, the port is a 4-wire interface). When SYN is cleared, asynchronous mode is chosen and separate clock and frame sync signals are used for the transmit and receive sections (that is, the port is a 6-wire interface). 0 Asynchronous mode 1 Synchronous mode (default)
10–0	Reserved

13.3.3.5 Port Data Control Register *n* (PDCR*n*)

Figure 13-4 PDCR*n* is the Port Data Control Register for Port *n*, where *n* ranges from 1 through 7.

Offset 0x0004 (PDCR1) Access: User Read/Write
0x000C (PDCR2)
0x0014 (PDCR3)
0x001C (PDCR4)
0x0024 (PDCR5)
0x002C (PDCR6)
0x0034 (PDCR7)

	31			16
R W	—			
Reset	(See Table 13-3 for reset values.)			

	15 13	12	11 10	9 8	7 0
R W	RXDSEL[2:0]	TXRX EN	—	MODE[1:0]	INMMASK[7:0]
Reset	(See Table 13-3 for reset values.)				

Figure 13-4. Port Data Control Register for Port *n*

Table 13-7 lists the PDCR*n* register field descriptions.

Table 13-7. PDCR (Port *n*) Field Descriptions

Field	Description
31–16	Reserved
15–13 RXDSEL[2:0]	Receive Data Select. Selects the source port for the RxD data. RXDSEL is ignored if MODE = 01 (that is, Internal Network Mode is enabled). xxx Port number for RxD 000 Port 1 … 110 Port 7 111 Reserved
12 TXRXEN	Transmit/Receive Switch Enable. Swaps the transmit and receive signals. 0 No switch (Transmit Pin = Transmit, Receive Pin = Receive) 1 Switch (Transmit Pin = Receive, Receive Pin = Transmit)
11–10	Reserved
9–8 MODE[1:0]	Mode Select. This field selects the mode in which the port is to operate. The modes of operation include the following: • Normal mode, in which the RxD from the port selected by RXDSEL is routed to the port. • Internal Network mode in which RxD from other ports are ANDed together. RXDSEL is ignored. INMMASK determines which RxD signals are ANDed together. • CE Bus Network mode, in which the port receives data from CE Bus port (Port 7) and any other port as selected by RXDSEL[3:0] in different time slots within a frame. The ce_bus_dis signal routes the data from the CE Bus port if low; otherwise, data from the other port (as selected by RXDSEL) is routed to the port. 00 Normal mode 01 Internal Network mode 10 CE Bus Network mode 11 Reserved
7–0 INMMASK[7:0]	Internal Network Mode Mask. Bit mask that selects the ports from which the RxD signals are to be ANDed together for internal network mode. Bit 6 represents RxD from Port 7 and bit0 represents RxD from Port 1. 0 Includes RxDn for ANDing 1 Excludes RxDn from ANDing

13.3.3.6 CE Bus Network Mode Control Register (CNMCR)

Figure 13-5 shows the CE bus network mode control register.

Offset 0x0038 (CNMCR) Access: User Read/Write

	31												19	18	17	16
R W	—													CEN	FSPOL	CLKPOL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	15							8	7							0
R W	CNTHI[7:0]								CNTLOW[7:0]							
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

Figure 13-5. CE Bus Network Mode Control Register (CNMCR)

Table 13-8 shows the CNCMR field descriptions.

Table 13-8. CNMCR Field Descriptions

Field	Description
31–19	Reserved
18 CEN	CE Bus enable. This signal controls the generation of ce_bus_dis signal. If set, the ce_bus_dis signal is generated as per the CNTHI and CNTLOW settings. 0 CE Bus disable signal is held low. 1 CE Bus disable signal is generated.
17 FSPOL	Frame Sync Polarity Select. This field selects the frame sync polarity of the CE Bus port (Port 7) used for the generation of ce_bus_dis signal. 0 Polarity 0 1 Polarity 1
16 CLKPOL	Clock Polarity Select. This field selects the bit clock polarity of the CE Bus port (Port 7) used for generation of ce_bus_dis signal. 0 Polarity 0 1 Polarity 1
15–8 CNTHI[7:0]	CE Bus disable signal high period count. This field selects the number of bit clocks for which the CE Bus disable signal ce_bus_dis is to remain high following the detection of the frame sync, that is, when CE Bus is not transferring data. This allows the user to specify the time until the CE Bus starts transmitting (with respect to the beginning of timeslot 0). 00000000 0 bit clock period 00000001 1 bit clock period 00000010 2 bit clock period … 11111111= 255 bit clock period • If operating in 4-wire mode (SYN is set), the clock used for ce_bus_dis generation is selected by the TCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for ce_bus_dis generation is selected by the TFSEL[3:0] field of Port 7's control register PTCR7. If operating in 6-wire mode (SYN is clear), the clock used for ce_bus_dis generation is selected by the RCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for ce_bus_dis generation is selected by the RFSEL[3:0] field of Port 7's control register PTCR7.

Table 13-8. CNMCR Field Descriptions (continued)

Field	Description
7–0 CNTLOW[7:0]	CNTLOW - CE Bus disable signal low period count. This field selects the number of bit clocks for which the CE Bus disable signal ce_bus_dis is to remain low, that is, when CE Bus is transferring data. 00000000 = 0 bit clock period 00000001 = 1 bit clock period 00000010 = 2 bit clock period . . . . 11111111 = 255 bit clock period • If operating in 4-wire mode (SYN is set), the clock used for ce_bus_dis generation is selected by the TCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for ce_bus_dis generation is selected by the TFSEL[3:0] field of Port 7's control register PTCR7. If operating in 6-wire mode (SYN is clear), the clock used for ce_bus_dis generation is selected by the RCSEL[3:0] field of Port 7's control register PTCR7. The frame sync used for ce_bus_dis generation is selected by the RFSEL[3:0] field of Port 7's control register PTCR7.

13.3.3.6.1 CE_Bus_dis Signal Generation Limitations

Certain restrictions are in place for the utilization of CE Bus network mode. They are as follows:

- Only early frame syncs are supported.
- CE Bus data transfers only take place in contiguous time slots.
- Only one port can be connected to the CE Bus port at a time.
- CE Bus Network mode can be used for only two ports (Port 7 and one other port).

Transmission of data by CE bus and frame sync generation by the master (CE bus/port) occurs on the same clock edge. This assures a 1/2-bit delay between the moment the RxD lines are switched inside the AUDMUX (as driven by ce_bus_dis) and the moment the receive data is sampled by the serial interface.

13.4 Functional Description

This section provides a complete functional description of the AUDMUX module. Figure 13-1 shows the AUDMUX block diagram.

13.4.1 AUDMUX Ports Overview

There is no functional difference among Ports 1 through 7. The main difference is whether a port is connected to an on-chip serial interface (for example, SSI) or connected to the chip's pads to connect to off-chip serial devices (that is, any 4-wire or 6-wire external SSI, voice, I2S, or AC97 CODEC).

Port 7 can be physically connected to any of the peripherals previously mentioned, as well as the CE Bus. The limitation of connecting CE Bus only at Port 7 is because of data selection in CE bus network mode at Ports 1 to 7. Ports 1 to 6 communicate with CE Bus devices by being configured to connect to Port 7.

All ports can be configured as four- or six-wire interfaces. When configured as a six-wire interface, the additional RFS and RCLK signals of the interface enable the serial interface to be used in asynchronous mode with separate receive and transmit clocks.

All ports have a Tx/Rx switch to provide flexibility in supporting network mode configurations. The Tx/Rx switch enables the transmit and receive data lines to be swapped so that mastership of the serial bus can be passed among multiple external devices connected to a single port.

In addition to supporting the default (point-to-point) normal mode, all ports also support two special types of network mode—internal network mode and CE bus network mode. With internal network mode, a point-to-multipoint network configuration with an arbitrary number of slaves can be supported if the external slaves are put into the high-impedance state (as defined in the SSI network mode protocol) and have pull-up resistors on their TxD contacts. (Alternatively, this can be viewed as requiring a pull-up resistor on the corresponding AUDMUX RxD contact.) With CE bus network mode, a point-to-multipoint network with two slaves can be supported; the slaves do not have to put the TxD line into high-impedance state, and they do not require any pull-up resistors.

Bit clock direction selection enables each port to be configured as a master or slave in the flow.

Possible scenarios include:

- SSI (internal port) drives a voice CODEC and a BT (Blue tooth) CODEC (both on external Port 6) and the Bottom Connector (on external Port 7) simultaneously using network mode. SSI is the master.
- An external processor (external port - Port 5) drives a voice CODEC and a BT CODEC (both on external Port 6) and the Bottom Connector (on Port 7) simultaneously using network mode. The external processor is the master.

13.4.2 Operating Modes

The following terms are used in the descriptions of AUDMUX operating modes:

- Network mode—Time-division multiplexed protocol for sending unique data to multiple devices on a serial bus.
- Internal network mode—Physical bus configuration where multiple serial buses are effectively connected within the AUDMUX using digital logic to create point-to-multipoint connectivity. An arbitrary number of devices are supported. Devices must be put into the high-impedance state as specified by the network mode protocol. TxDATA lines of devices must be pulled high.
- External network mode—Physical bus configuration where multiple serial buses are electrically connected together on a printed circuit board (that is, external to the AUDMUX). Devices must put their TxDATA lines into the high-impedance state as specified by the network mode protocol.
- CE bus network mode—Physical bus configuration where multiple serial buses are effectively connected within the AUDMUX using digital logic in order to create point-to-multipoint connectivity. This mode can only utilize three AUDMUX ports simultaneously. One of the ports must be Port 7. Devices do not have to be put into the high-impedance state, nor do they require pull-up resistors on their TxDATA contacts.

13.4.2.1 Port Receive Data Modes

Each port has logic to select which data lines are used to create the RxD line for the corresponding host interface. Figure 13-6 shows the logic used to create the RxD line for Port 1. This logic has the following modes of operation (as determined by MODE[1:0]):

- Normal mode—see Section 13.4.2.1.1, "Normal Mode," for more information
- CE bus network mode—see Section 13.4.2.1.3, "CE Bus Network Mode," for more information
- Internal network mode—see Section 13.4.2.1.2, "Internal Network Mode," for more information

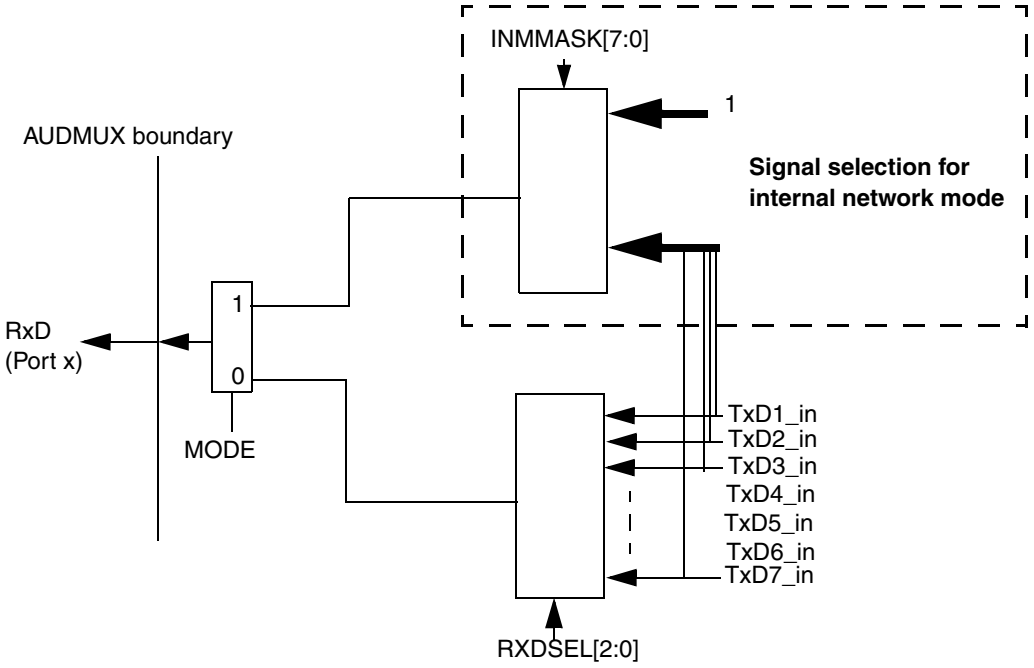


Figure 13-6. Receive Data Logic for Port *n*

13.4.2.1.1 Normal Mode

In normal mode (MODE = 00), the port is connected in a point-to-point configuration (as a master or a slave) and the RXDSEL[2:0] setting selects the transmit signal from any port. In normal mode, any data format can be used (that is, SSI normal mode, SSI network mode, AC-97, and others.

13.4.2.1.2 Internal Network Mode

In internal network mode (MODE = 01), the output of the AND gate is routed (using the output of the port) to the RxD signal of the corresponding host interface. The INMMASK bit vector selects the transmit signals of the ports that are to be connected in network mode. The transmit signals received at the AUDMUX ports (TxDn_in) are ANDed together to form the output. In internal network mode, only one device can be transmitting in its predesignated timeslot and all other transmit signals must remain high (be in high-impedance state and pulled-up). Therefore, non-active signals in the selection will be high and do not influence the output of the AND gate.

Network mode is a protocol where a master SSI is connected to more than one slave SSI device and communication occurs on a time-slotted frame. Though network mode can allow master-slave and slave-slave communication, internal network mode supports only master-slave communication.

There are two scenarios where internal network mode can be used with external network mode:

1. Slave-only devices are attached to an external port.
2. A master device is attached to an external port and all slave devices connected to the same external port are disabled.

NOTE

When internal network mode is enabled at an external port, RXDSEL[3:0] for RxDn_obe selection is ignored and RxD_obe is always driven high (that is, asserted for all timeslots). All slave devices connected to the same port must be disabled.

Internal Network Mode Example 1

SSI_m and SSI_n are used with Port 4 in internal network mode as shown in Figure 13-7. No pull-up resistors are required because the interfaces combined in internal network mode are on-chip interfaces.

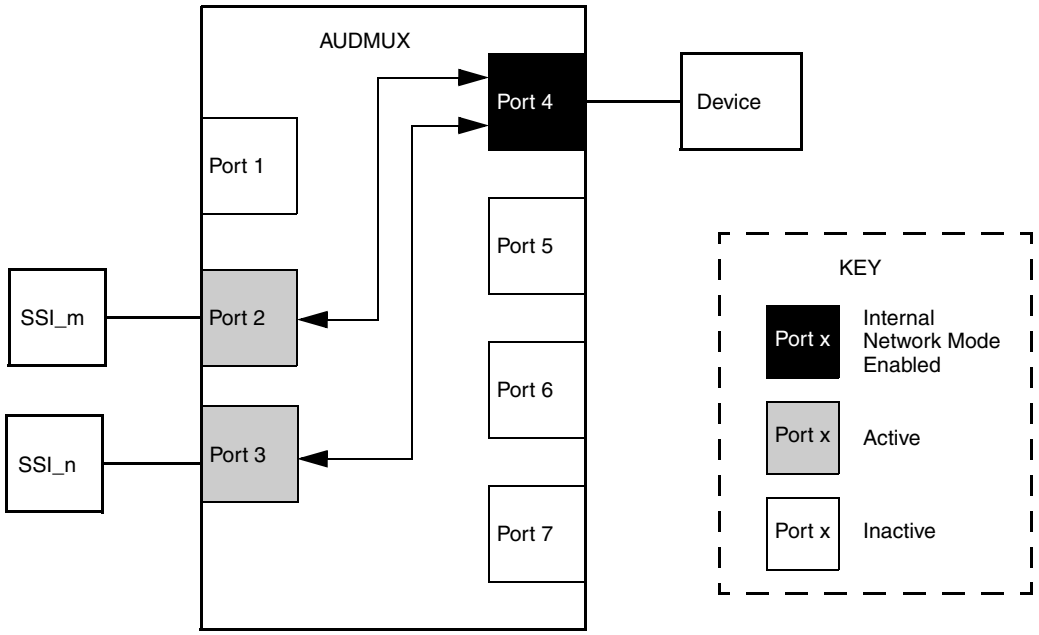


Figure 13-7. Block Diagram For Example 1

See Figure 13-8 for the timing diagram of Example 1. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots.

The data lines for SSI_m and SSI_n (as well as their output enables) are shown. The on-chip interfaces drive a logic ‘1’ when their output enables are logic ‘0’. The combined TxDATA line, which is the logical AND of the individual TxDATA lines, is used for Port 4’s TxDATA line.

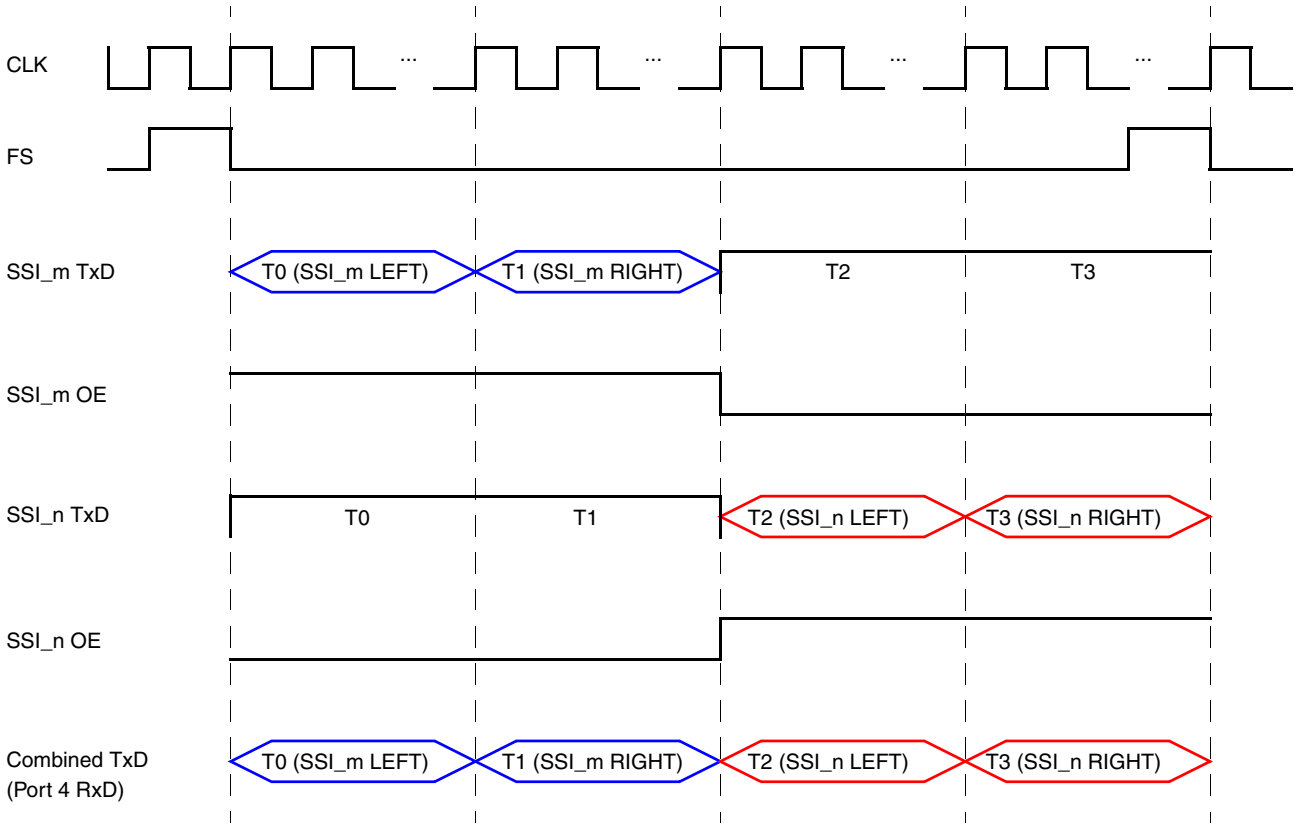


Figure 13-8. Example Using Internal Ports for Transmit Data

Internal Network Mode Example 2

Figure 13-9 shows the case where SSI, port 4, and port 5 are used with port 6 in internal network mode. Port 4 and port 5 are external ports so pull-up resistors are required on the port 4 RxDATA and port 5 RxDATA contacts. This example shows the timing associated with using adjacent timeslots for the SSI, Port 4, and Port 5.

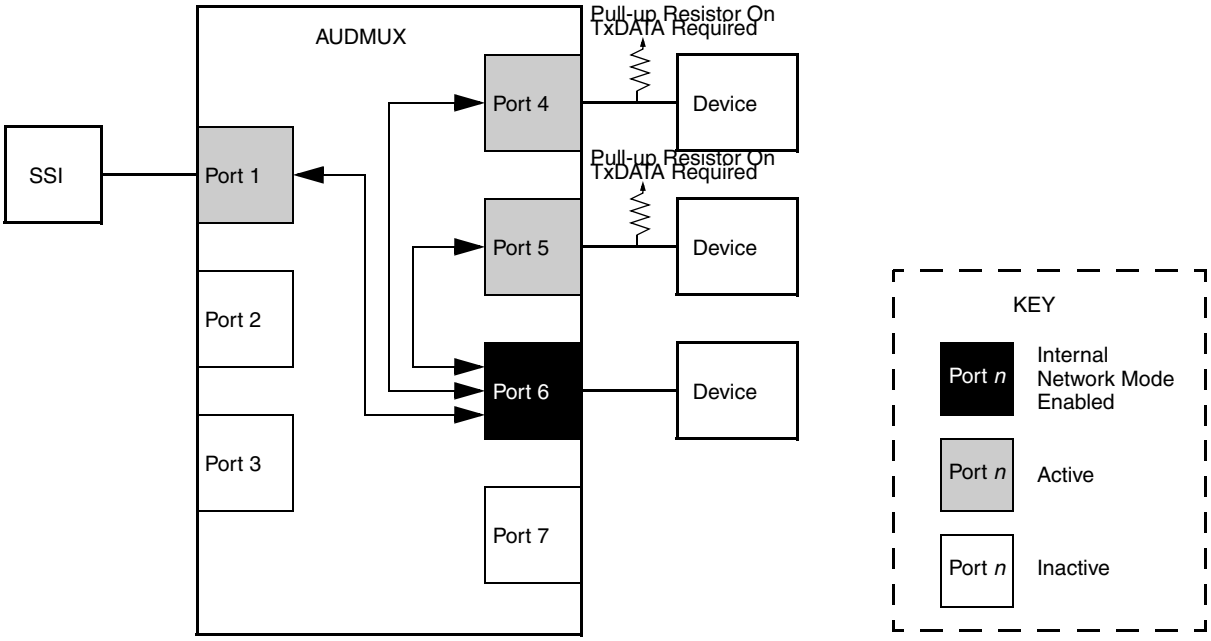


Figure 13-9. Block Diagram for Example 2

The resistance value of the pull-up resistors must be sufficiently high such that a value of 0 can be pulled up to logic 1 within half of a period of the bit clock. The required resistance must be no larger than:

$$R_{max} = 1 / (2 \times f_{bc} \times C)$$

where:

f_{bc} is the frequency of the bit clock

C is the total system capacitance (including capacitance of ICs, board traces, and so on)

Figure 13-10 shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots. Adjacent timeslots are allocated to SSI, port 4, port 5, and port 4, respectively.

The data lines for the SSI, port 4, and port 5 are shown. The SSI transmits a logic 1 when its corresponding output enable is a logic 0. The data lines from port 4 and port 5 at the contact are pulled high by pull-up resistors when they are in the high-impedance state. The data lines from port 4 and port 5 at the AUDMUX are pure digital signals and are constantly driven. The combined TxDATA line, which is the logical AND of the SSI, port 4, and port 5's TxDATA lines, is used for port 6's TxDATA line.

The highlighted areas in Figure 13-10show the transition time that occurs while a TxDATA line is being pulled high. In this example, this transition time is a maximum of half the period of the serial bit clock.

This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bit clock frequency and system capacitance.

Hysteresis must be enabled at port 4's RxDATA contact and port 5's RxDATA contact to prevent the digital signals created by the contact from toggling rapidly during the pull-up period. The contacts typically require a transition within 25 ns unless hysteresis is enabled. Instead of using hysteresis, one could select a pull-up resistor sufficiently high to pull-up the signal at the contact within 25 ns; however, that would result in a higher resistance value and higher current drain.

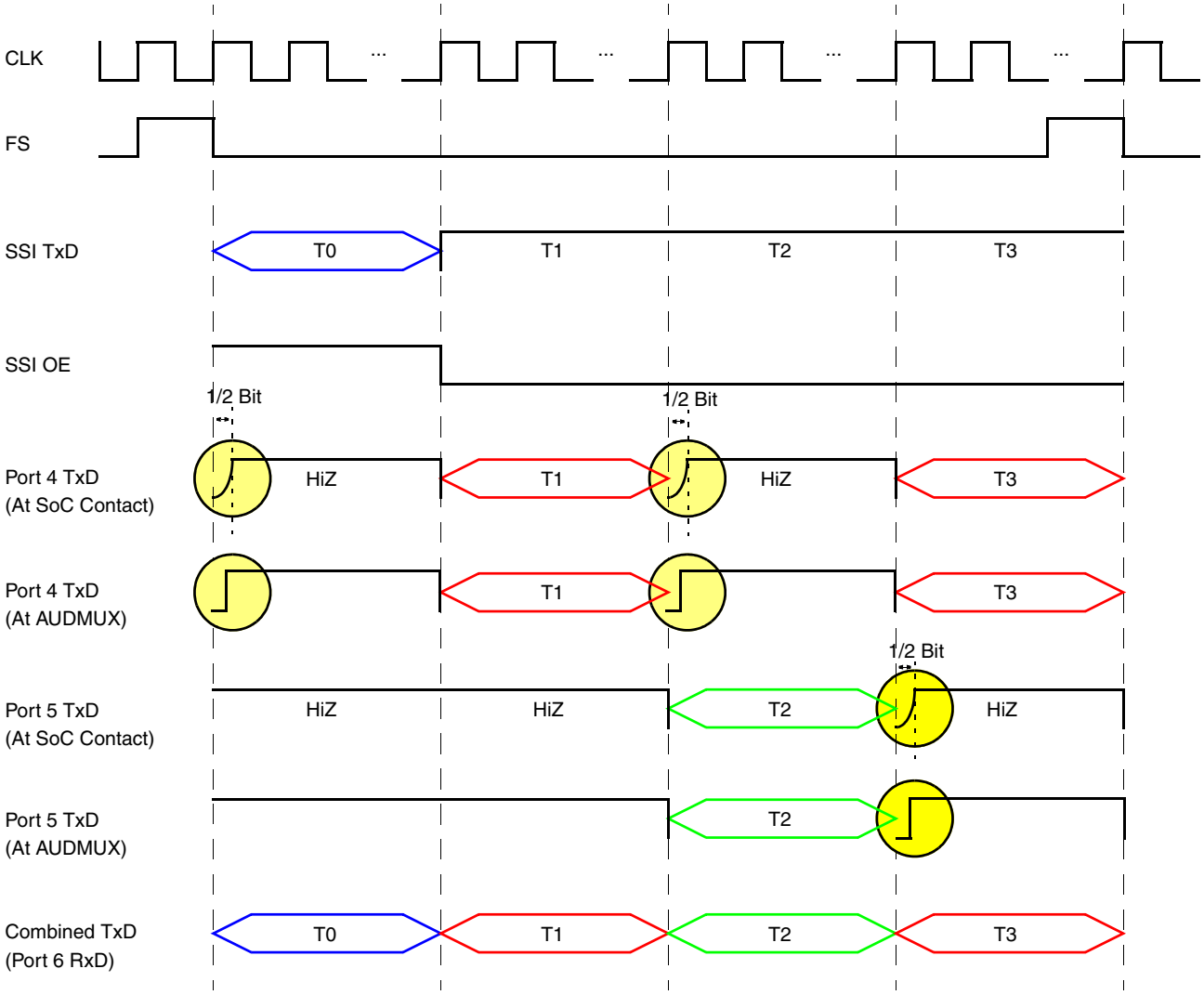


Figure 13-10. Example Using External Ports for Transmit Data in Consecutive Timeslots

Internal Network Mode Example 3

Figure 13-11 shows the case where SSI and port 4 are used with port 6 in internal network mode. Since port 4 is an external port, a pull-up resistor is required on the port 4 TxDATA contact. This example shows the timing associated with inserting empty timeslots after the timeslots have been used by external ports.

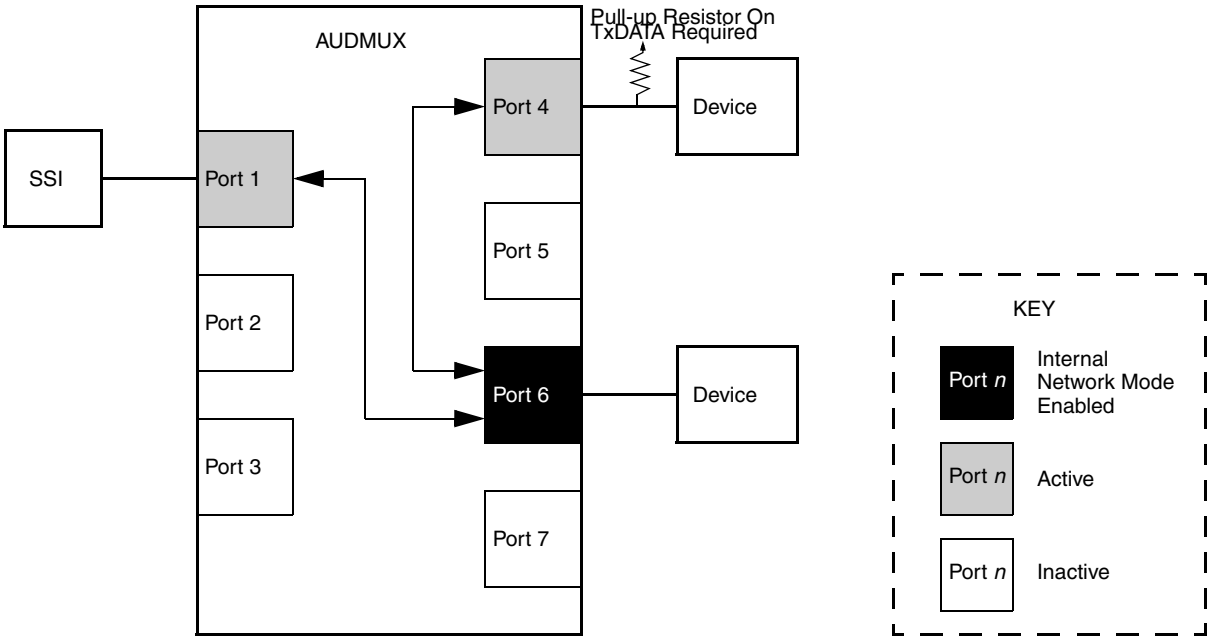


Figure 13-11. Block Diagram for Example 3

The resistance value of the pull-up resistor must be sufficiently high such that a logic 0 can be pulled up to logic 1 by the time that the next occupied timeslot occurs. This allows a much weaker pull-up to be used as compared to Example 2. The required resistance must be no larger than:

$$R_{max} = (4 \times n + 1) / (2 \times f_{bc} \times C)$$

where:

n is the number of bits per timeslot

f_{bc} is the frequency of the bit clock

C is the total system capacitance (ICs, board traces, and so on)

Figure 13-12 shows the timing diagram for this example. The clock and frame sync signals show the bit and frame timing for the serial bus. The vertical dashed lines divide the frame into four timeslots. In this example, empty timeslots are inserted after the timeslots have been used by external ports.

The data lines for the SSI and Port 4 are shown. The SSI transmits a logic 1 when its corresponding output enable is a logic 0. The data line from Port 4 at the contact is pulled high by a pull-up resistor when they are in the high-impedance state. The data line from Port 4 at the AUDMUX is a pure digital signal and is constantly driven. The combined TxDATA line, which is the logical AND of the SSI and Port 4's TxDATA lines, is used for Port 6's RxDATA line.

The highlighted areas in Figure 13-12show the transition time that occurs while port 4's TxDATA line is being pulled high. In this example, this transition time is a maximum of two timeslots plus 1/2 the period of the serial bit clock. This prevents corruption of the first data bit of the next timeslot. It is critical that the pull-up resistance is sufficient for the given bit clock frequency and system capacitance.

Hysteresis must be enabled at port 4's RxDATA contact to prevent the digital signal created by the contact from toggling rapidly during the extended pull-up period. The contacts typically require a transition within 25 ns unless hysteresis is enabled.

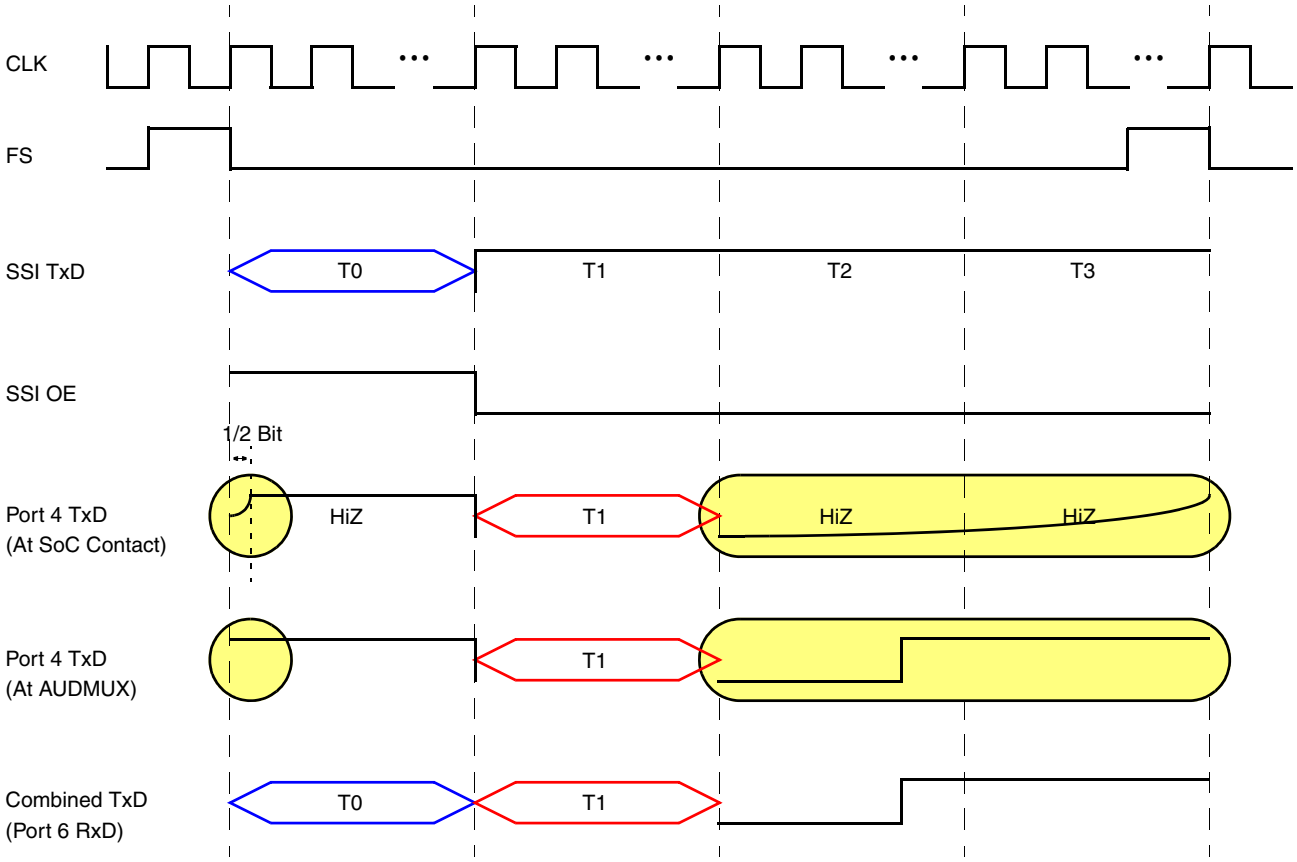


Figure 13-12. Example Using External Ports for Transmit Data in Nonconsecutive Timeslots

13.4.2.1.3 CE Bus Network Mode

To support network mode with devices connected to CE bus, a special network mode has been added. In CE bus network mode (MODE[1:0] = 10), a 2 × 1 multiplexer is used to synchronously switch between two ports (that is, network slaves) as determined by a signal called ce_bus_dis, which is generated by the AUDMUX. CE bus network mode supports switching between two ports. One of the ports is the CE bus port (Port 7) and the other port is selected by RXDSEL[2:0]. In contrast to internal network mode, external ports do not require pull-up resistors, as the CE bus network mode logic switches data lines according to the timeslot timing.

ce_bus_dis Signal Generation

The AUDMUX uses clock and frame sync timing to generate the ce_bus_dis signal (and thus drive the synchronous multiplexer used in CE bus network mode). The CE bus controls a contiguous block of timeslots. In addition, each timeslot is controlled by one of the two ports. For example, Port 7 (CE bus) could transmit in timeslots 1-2 and Port 6 could transmit in timeslots 0 and 3.

If the CEN bit is set, the ce_bus_dis signal generation logic is enabled. Otherwise, the signal generation logic is disabled and the ce_bus_dis signal remains low.

The CNTLOW[7:0] field controls the number of bit clock periods for which the ce_bus_dis signal is held low. This establishes the length of the CE Bus port's timeslot(s).

The CNTHI[7:0] field controls the number of bit clock periods for which the ce_bus_dis signal is held high after frame sync detection. This establishes the number of bits between the frame sync detection and the start of the CE bus port's timeslot(s). The internal counter used to determine the deassertion of the ce_bus_dis signal is reset on every frame sync. This assures that the number of bit clock periods during which the ce_bus_dis signal is held high after the assertion of frame sync is always correct.

The Port 7 control register PTCR7 selects the frame sync used to generate the ce_bus_dis signal. If the SYN bit is set, then frame sync is determined by the TFSDIR and TFSEL[3:0] fields of PTCR7. If the SYN bit is clear (that is, Port 7 is 6-wire), then TxFS is determined by the TFSDIR and TFSEL[3:0] and the RxFS is determined by the RFSDIR and RFSEL[3:0] fields.

Similarly, the Port 7 control register PTCR7 determines the bit clock used for counting the low and high periods of the ce_bus_dis signal. If the SYN bit is set, then the bit clock is determined by the TCLKDIR and TCSEL[3:0] fields of PTCR7. If the SYN bit is clear (that is, Port 7 is 6-wire), then TxCLK is determined by the TCLKDIR and TCSEL[3:0] and the RxCLK is determined by RCLKDIR and RCSEL[3:0] fields of PTCR7.

The FSPOL and CLKPOL fields of the CNMCR determine the frame sync and bit clock polarities used for frame sync detection. Different scenarios with combinations of frame sync and bit clock polarities are shown in Figure 13-13 through Figure 13-17. For all these scenarios, CEN is set to 1. In RxD*n*_in, *n* can be any number from 1 and 6.

The CE bus network mode control register CNMCR is used to control the generation of the ce_bus_dis signal. See Section 13.3.3.6, "CE Bus Network Mode Control Register (CNMCR)," for complete details on the CNMCR.

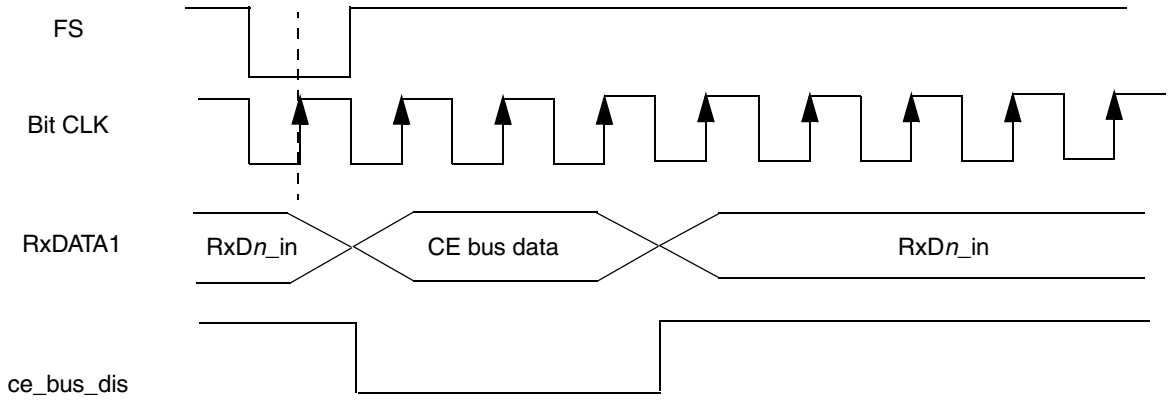


Figure 13-13. ce_bus_dis Signal Timing for FSPOL=0, CLKPOL=0, CNTLOW=3, CNTHI=0

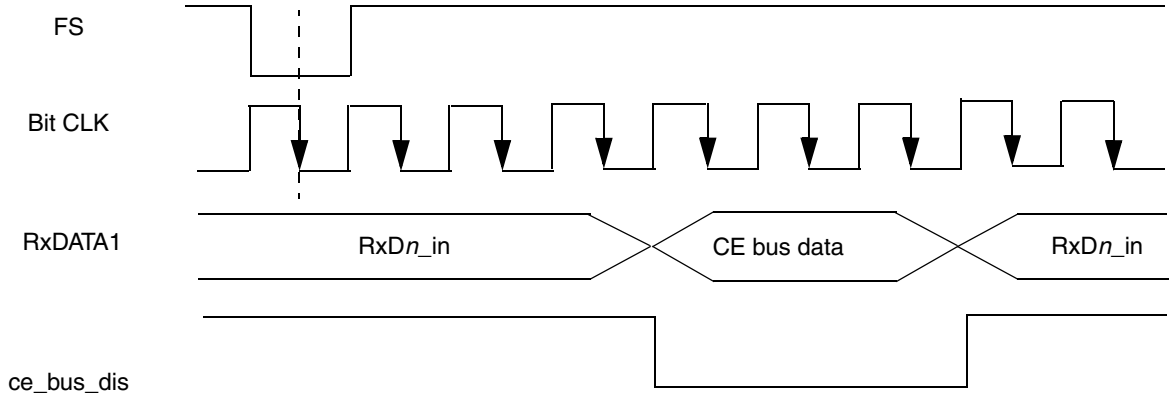


Figure 13-14. ce_bus_dis Signal Timing with FSPOL=0, CLKPOL=1, CNTLOW=0b11, CNTHI=0b11

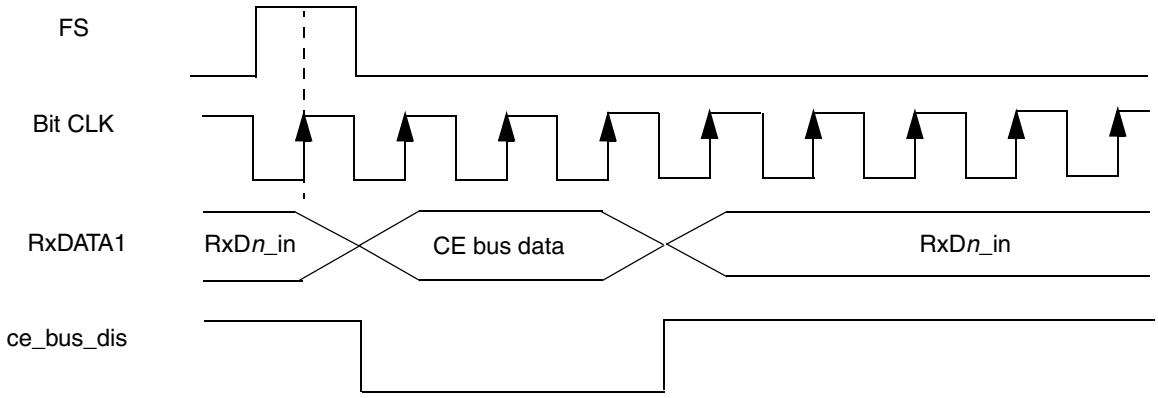


Figure 13-15. ce_bus_dis Signal Timing with FSPOL=1, CLKPOL=0, CNTLOW=0b11, CNTHI=0

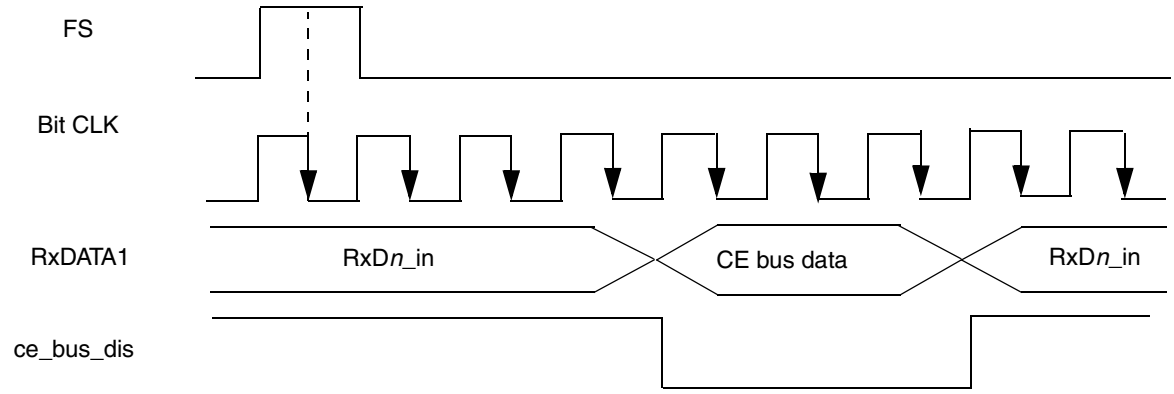


Figure 13-16. ce_bus_dis Signal Timing with FSPOL=1, CLKPOL=1, CNTLOW=0b11, CNTHI=0b11

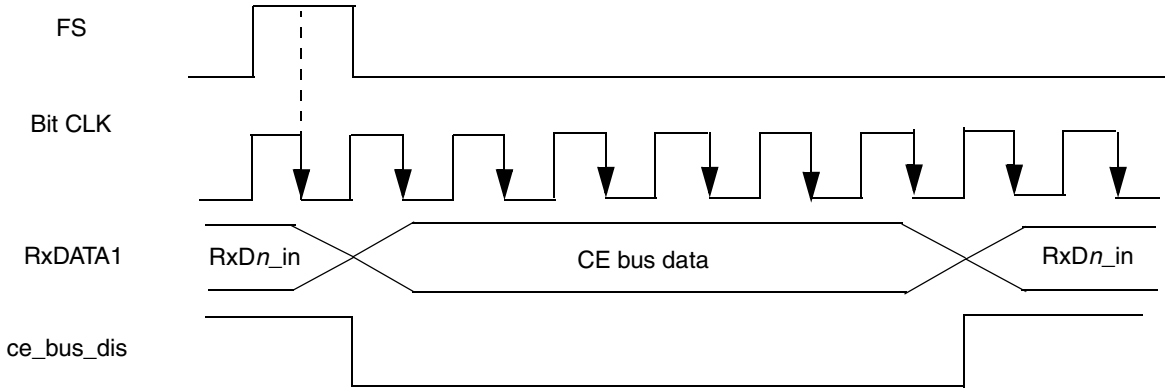


Figure 13-17. ce_bus_dis Signal Timing with FSPOL=1, CLKPOL=1, CNTLOW=0b110, CNTHI=0

13.4.2.2 Transmit Data Output Enable Assertion

The TxDATA line from the internal network mode master (connected at any internal port) is put into the high-impedance state at the contact depending upon the assertion or negation of TxD_obe. Its corresponding output enable is generated by the network mode master.

In the case of an external network mode master (connected at an external port), the corresponding TxD_obe is always asserted after the port data register configuration.

13.4.2.3 Tx/Rx Switch and External Network Mode

External network mode is the traditional network mode connection. It is called external network mode to differentiate from the internal network mode. In external network mode, devices are connected to a single external port in a star or multi-drop configuration.

In network mode, there can be only one master (driving the frame sync and clock source) with the other devices configured in normal slave mode or network slave mode. Unlike internal network mode, both master-slave and slave-slave communication can take place in external network mode. CODEC devices transmit on a single timeslot while processor serial interfaces (that is, SSI) can process more than one timeslot of data while in network master or slave mode.

Figure 13-18 shows the Tx/Rx data switch. RxD_obe is the output buffer enable signal and RxD_out is the data transmit signal from the serial interface. The TxD_in signal is the receive data signal going towards the RXDSEL multiplexers of all ports.

D_TxRx is the data contact that serves as the chip-level transmit data contact when the TxRx switch is not enabled. D_RxTx is the data contact that serves as the chip-level receive data contact when the TxRx switch is not enabled. The roles of these contacts are reversed when the TxRx switch is enabled.

When TXRXEN is disabled (TXRXEN=0), RxD_out is routed to D_TxRx and D_RxTx is routed to TxD_in. The output buffer enable, selected by RXDSEL[2:0], is routed to Db_obe.

When the Tx/Rx switch is enabled (TXRXEN=1), RxD_out is routed to D_RxTx and D_TxRx is routed to TxD_in. The output buffer enable, selected by RXDSEL[2:0], is routed to Da_obe.

If the RXDSEL*n*[2:0] field for any port *n* is configured to select data from an internal port, the output buffer enable is selected by RXDSEL*n*[2:0] and is routed to Da*n*_obe / Db*n*_obe. In the case when the RXDSEL*n*[2:0] field for port *n* is configured to select data from an external port, the output buffer enable is always high and routed to Da*n*_obe / Db*n*_obe, depending on the TXRXEN*n* switch configuration.

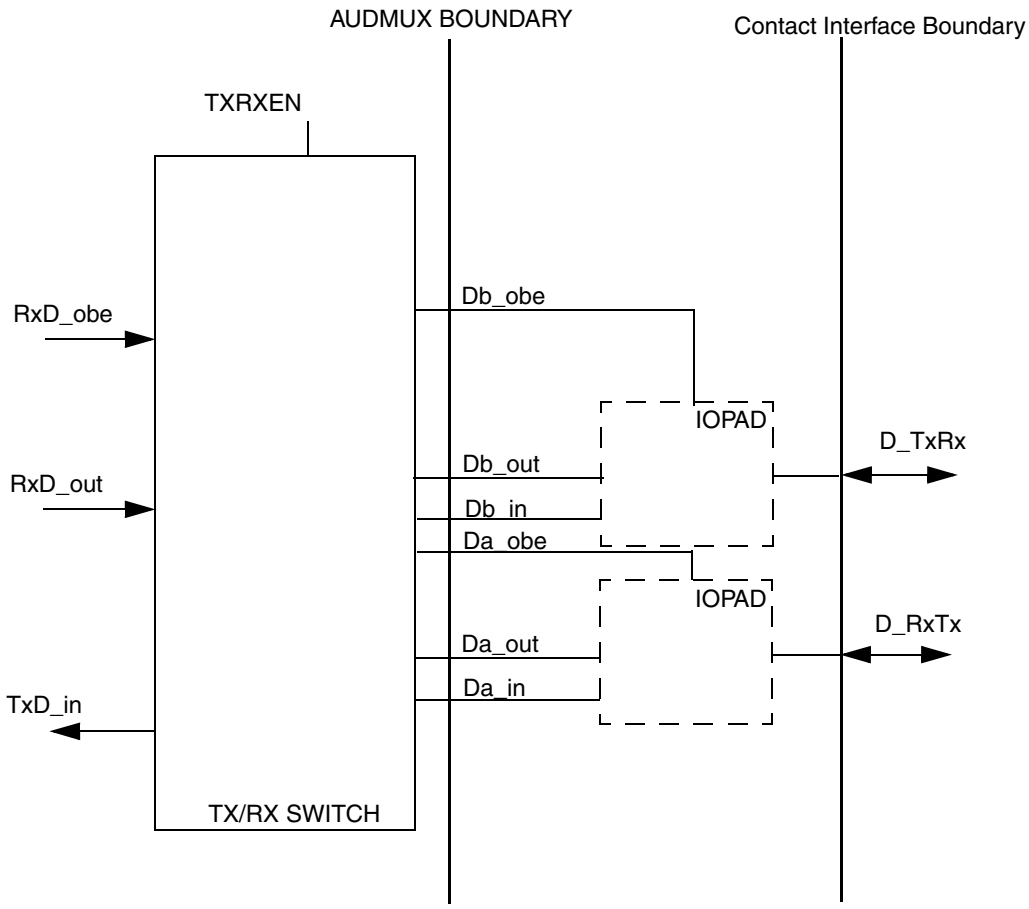


Figure 13-18. Tx/Rx Switch

13.4.2.4 Timing Modes

The AUDMUX ports are constructed as 6-wire interfaces. However, they can be used either in synchronous or asynchronous modes as determined by the SYN bit.

13.4.2.4.1 Synchronous Mode (4-Wire Interface)

In synchronous mode, the port has a 4-wire interface (that is, RxD, TxD, TxCLK, TxFS). The receive data timing is determined by TxCLK and TxFS.

As shown in Figure 13-19, port *n* signals can be routed to port *m*, producing 6-wire to 4-wire port connectivity.

TFS_in, RFS_in, TCLK_in, and RCLK_in are the input frame sync and bit clocks from the serial interface (port *n*) with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports.

The TFS_out and TCLK_out are selected at port *n* by the TFSEL and TCSEL multiplexer settings, respectively. RFS_out and RCLK_out are selected at port *n* by the RFSEL and RCSEL multiplexer settings, respectively. Similarly, in the external direction, port *m* is configured as a 4-wire port; TFSEL selects the FS_obe and FS_out signals. In this mode, the configuration of RFSEL and RCSEL is not used, since the RFS_out and RCLK_out contacts at port *m* are not available.

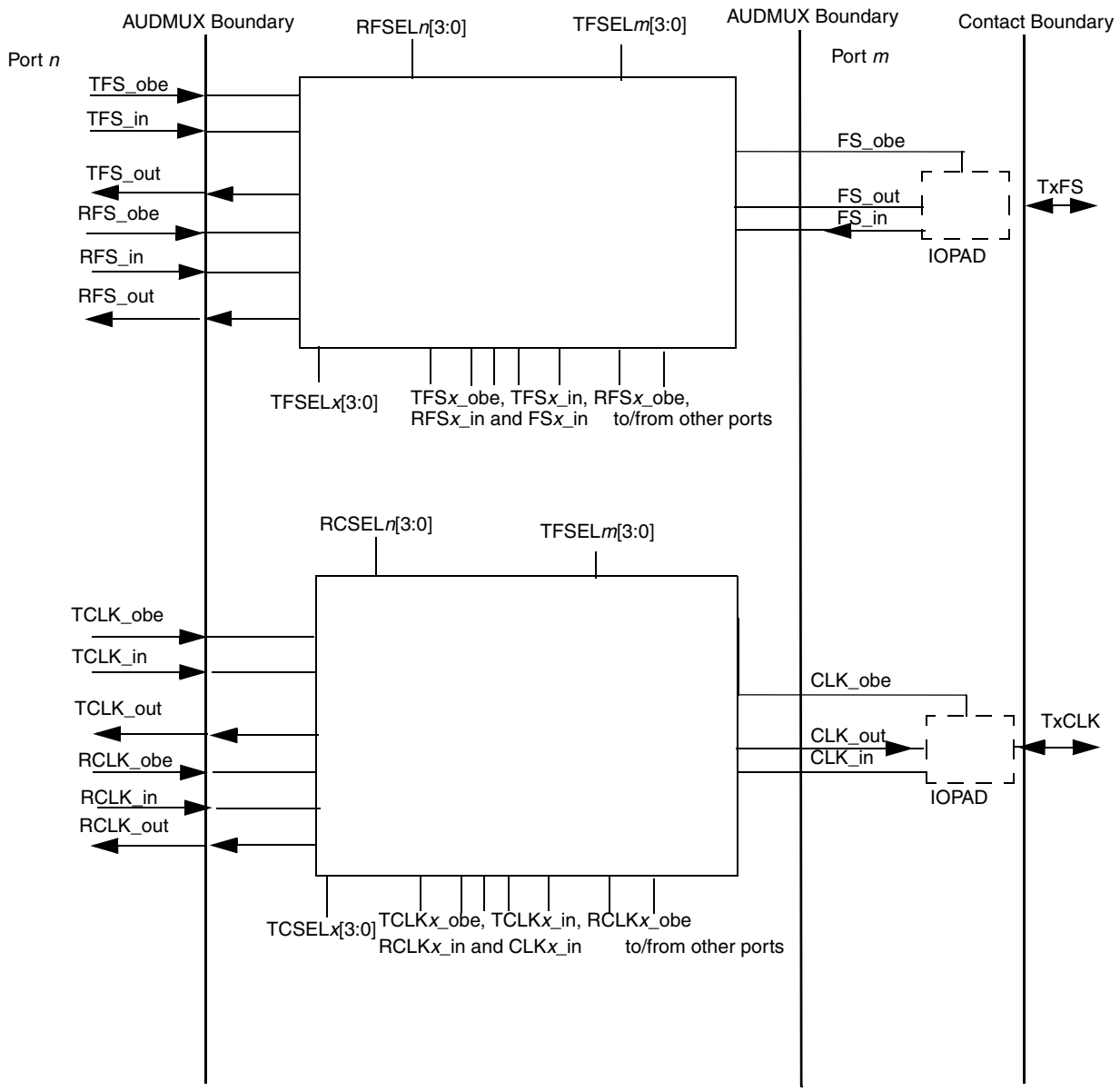


Figure 13-19. Frame Sync and Clock Routing When External Port Is 4-Wire

13.4.2.4.2 Asynchronous Mode (6-Wire Interface)

In asynchronous mode, the port has a 6-wire interface (including RxD, TxD, TxCLK, TxFS, RxCLK, and RxFS). This mode has additional receive clock (RxCLK) and frame sync (RxFS) signals as compared to the synchronous or 4-wire interface.

As shown in Figure 13-20 and Figure 13-21, port *n* signals can be routed to port *m*, producing 6-wire to 6-wire port connectivity.

TFS_in, RFS_in, TCLK_in, and RCLK_in are input frame sync and bit clocks from the serial interface (port *n*) with their corresponding output buffer enable signals (_obe). TFS_out, RFS_out, TCLK_out, and RCLK_out are the frame sync and bit clocks that are transmitted to the serial interface from the other ports.

TFS_out and TCLK_out are selected by the TFSEL and TCSEL multiplexer settings, respectively. RFS_out and RCLK_out are selected by the RFSEL and RCSEL multiplexer settings, respectively. Similarly, in the external direction, the TFSEL selects the TxFS_obe and TxFS_out signals and TCSEL selects the TxCLK_obe and TxClk_out signals. The RFSEL selects the RxFS_obe and RxFS_out signals and RCSEL selects the RxCLK_obe and RxCLK_out signals.

NOTE

Since FS_in and CLK_in from external interfaces are also routed to the TFSEL and TCSEL multiplexers of the external ports, respectively, these signals do not have corresponding buffer enable signals. Consequently, their corresponding inputs to the TFSEL and TCSEL multiplexer of the external ports have to be tied high.

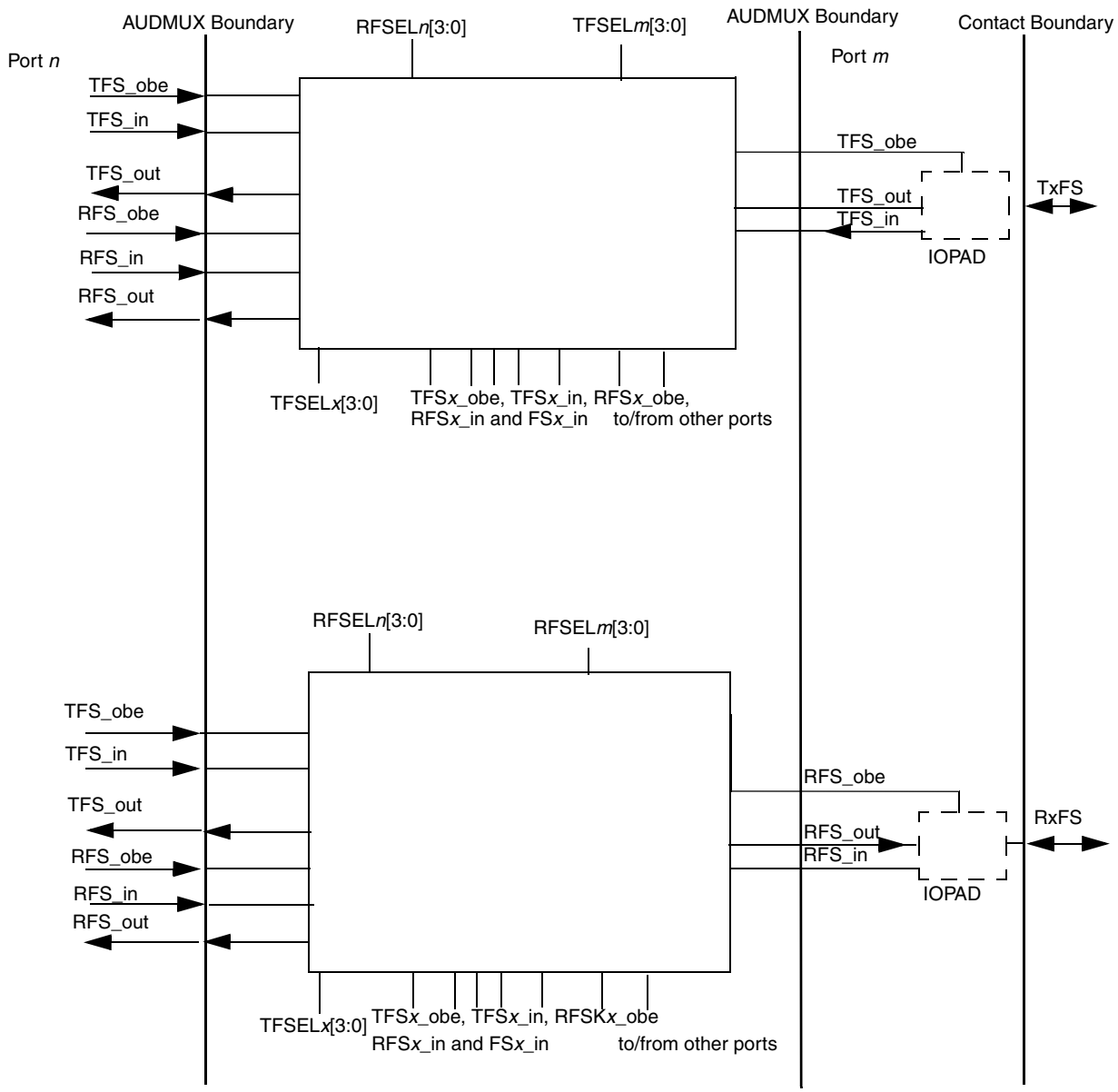


Figure 13-20. Frame Sync Routing When External Port is 6-Wire

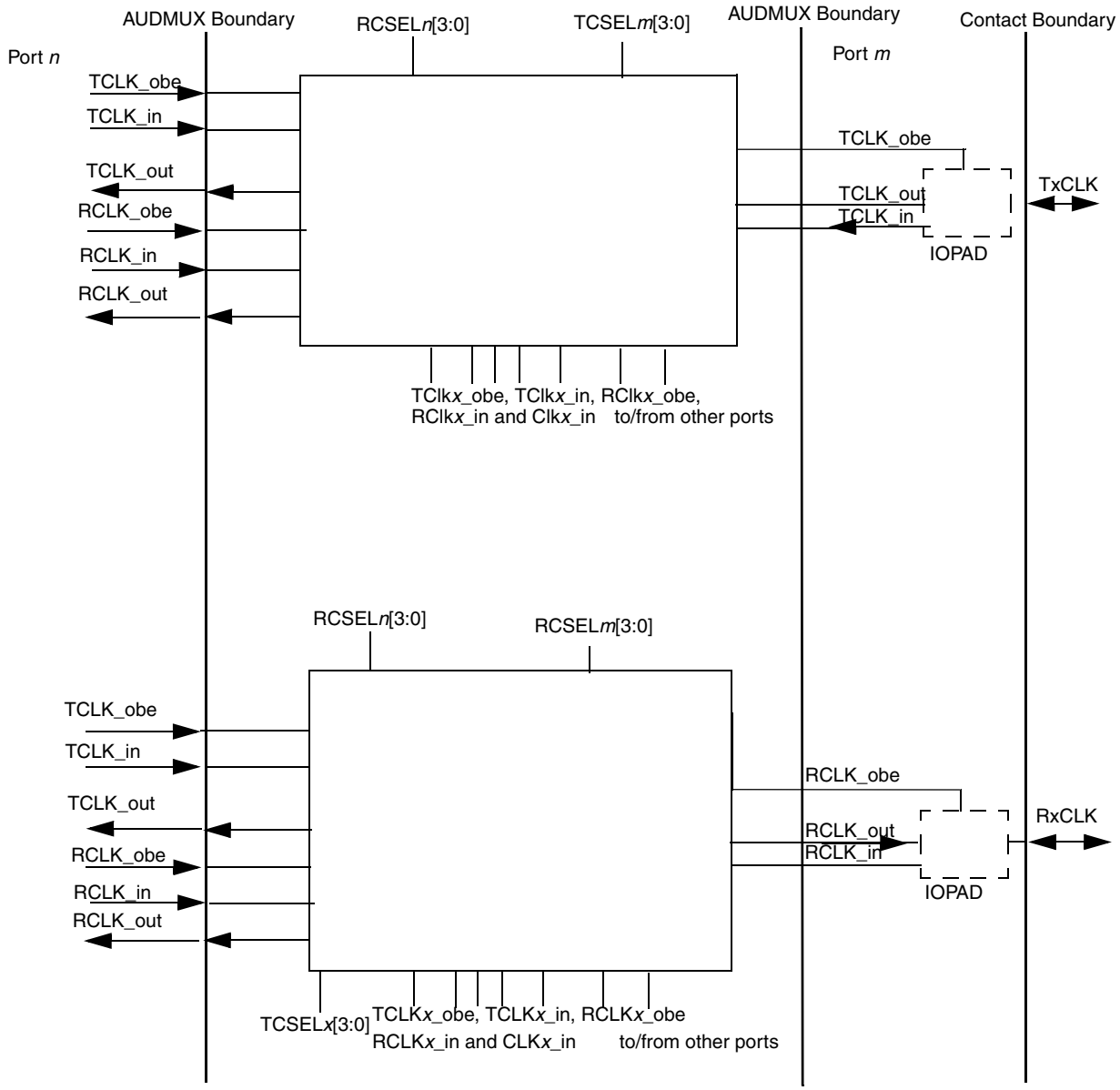


Figure 13-21. Clock Routing When External Port is 6-Wire

13.4.3 AUDMUX Default Configuration

The AUDMUX reverts back to its default settings following a reset. Section 13.4.3.1, "Default Port Configuration," and Section 13.4.3.2, "Default CE Bus Configuration," describes the default configuration of the ports and CE Bus, respectively.

13.4.3.1 Default Port Configuration

After a reset, each port defaults to normal mode (PDCR*n*[MODE] = 00) with synchronous timing mode (PTCR*n*[SYN] = 1) enabled.

The default port-to-port connections are as follows:

- Port 1 to Port 6
  - Port 6 provides the clock and frame sync.
- Port 2 to Port 5
  - Port 5 provides the clock and frame sync.
- Port 3 to Port 4
  - Port 4 provides the clock and frame sync.
- Port 7 to Port 7 (in data loopback mode)
  - Clock and frame syncs are inputs.

13.4.3.2 Default CE Bus Configuration

The default configuration of all the ports is to set the MODE field to normal mode (that is, no port selects CE Bus network mode). Correspondingly, the default configuration of the CEN field in the CNMCR is clear. To minimize AUDMUX setup for audio testing, the rest of the CNMCR fields default to the following configuration:

- FSPOL is set.
- CLKPOL is set.
- CNTHI is set to 16.
- CNTLOW is set to 16.

This configuration uses a frame sync that is logic high when asserted.

The clock is driven by the transmitter on the rising edge and is sampled by the receiver on the falling edge. The AUDMUX switches between devices on the rising edge; this provides a buffer of one-half clock period between the moment data is sampled and when the AUDMUX switches between devices. See Figure 13-16 for a timing diagram where FSPOL and CLKPOL are both set.

CNTHI and CNTLOW are set to 16 to allow CE Bus to drive data during timeslot 1 and the other device to drive data during timeslots 0, 2, …, N-1 where N is the number of timeslots used and each timeslot has 16 bits.

13.4.4 Connectivity Between Ports

Four basic types of connections are provided by the AUDMUX:

- Internal port to external port
- External port to external port
- Internal port to internal port
- Loopback

13.4.4.1 Internal Port to External Port Connectivity

The internal port is connected to a processor's serial interface. TxD_obe is the buffer enable signal from the serial interface, TxD_in is the input transmit data from the serial interface to the AUDMUX, and RxD_out is the receive data output from the AUDMUX to the serial interface.

RXDSEL[2:0] of the external port selects the buffer enable signal (TxD_obe) and transmit data output (TxD_out) signal from the TxD_obe and RxD_in signals. RXDSEL[2:0] is a common signal to both selection MUXes.

NOTE

Since buffer TxD_in signals from external interfaces do not have corresponding buffer enable signals, their buffer enable signals into the selection multiplexer are tied high. This ensures that selection of TxD_in, as RxD_out also drives the RxD_obe output high.

Transmit Data from the serial interface goes into the RXDSEL data multiplexer and comes out as RxD_out. RxD_out is routed to Da_TxRx when TXRXEN is disabled and to D_RxTx when TXRXEN is enabled. Similarly, D_RxTx is routed to TxD_in when TXRXEN is disabled and D_TxRx is routed to TxD_in when TXRXEN is enabled. The routing of frame syncs is shown in Figure 13-20 and the routing of interface clocks is shown in Figure 13-21.

If internal network mode is disabled, then RXDSEL selects the TxD_in, which is sent from the AUDMUX to the serial interface connected at port *n*. When the internal network mode is selected, RxD_out is constructed by ANDing selected TxD_in signals from the ports (as determined by INMMASK).

If there is more than one device attached to the external port at D_TxRx and D_RxTx and one of the devices is a network master, then following conditions must be observed:

1. When the external master is enabled in network mode, then the serial interface at port *n* must be configured as a slave (normal or network mode). No Tx/Rx switching is required.
2. When the external master is disabled and the serial interface at port *n* and other slave devices must communicate, then the serial interface at port *n* must be configured as a network mode master and the Tx/Rx switch at port *m* must be enabled (TXRXEN=1). This ensures that the transmit and receive paths are connected appropriately.

To communicate with more than one port, internal network mode can be enabled at port *n*. In internal network mode, it is possible to communicate with any device attached to the other ports. Internal network mode shall be enabled at the port that is the SSI network mode master.

It is also possible for a port to communicate with two (and only two) ports by enabling CE Bus network mode at Port *n*. It is then possible to communicate with any device attached to two enabled ports; one of which is Port 7 and the other is selected by RXDSEL*n*[2:0]. CE Bus network mode is enabled at the port that is the SSI network mode master.

13.4.4.2 External Port to External Port Connectivity

External ports can communicate with external ports directly. External ports can communicate together in the following ways:

1. Each port's receive logic is configured in normal mode (MODE = 00). Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE = 01). All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. Since an external port is being used as the internal network mode master, all other devices on the same AUDMUX port as the internal network mode master must be disabled. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.
3. One port is configured in CE Bus network mode (MODE[0:1] = 10) to receive and/or transmit data from and to the CE Bus connected to Port 7 and one other port. The ce_bus_dis signal is generated by the AUDMUX depending upon the CNMCR register configuration. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any of the three ports can be the master.

13.4.4.3 Internal Port to Internal Port Connectivity

Internal ports can communicate with other internal ports directly, thereby providing a means for synchronous interprocessor communication. Internal ports can communicate together in three ways:

1. Each port's receive logic is configured in normal mode (MODE = 00). Each port's RXDSEL[2:0] field is configured to select the other port's transmit data. Bit fields associated with clock/frame sync selection and direction are configured for each port. Either port can be the master.
2. One port is configured in internal network mode (MODE = 01). All desired data lines are combined by the AND gate as determined by INMMASK[7:0]. This configuration can be used with a combination of internal and external ports. All external ports must have a pull-up resistor on its RxDATA pin. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any port can be the master.
3. One port is configured in CE Bus network mode (MODE[0:1] = 10) to receive and/or transmit data from and to the CE Bus connected at Port 7 and one other port. The ce_bus_dis signal is generated by the AUDMUX depending upon the CNMCR register configuration. Bit fields associated with clock/frame sync selection and direction are configured for each port. Any of the three ports can be the master.

13.4.4.4 Loopback Connectivity

AUDMUX ports can communicate with themselves in order to provide loopback functionality. Port *n* can route its TxDATA signal to its own RxD_out signal by setting RXDSEL*n*[2:0] to its own port number. This is supported by all ports in the AUDMUX.

In addition, ports can provide loopback support in internal network mode as well as CE Bus network mode. With internal network mode, the internal network mode master can loop its TxDATA signal (combined with those of other ports, if desired) back into its RxD_out signal. Port *n*'s INMMASK must be set such that bit (*n* - 1) is clear in order to enable the loopback.

With CE Bus network mode, the CE Bus network mode master can loop its TxDATA signal (combined with CE Bus's TxDATA) back into its RxD_in signal. RXDSEL*n*[2:0] must be set to select port *n*.

13.4.5 AUDMUX Clocking

This section provides information about AUDMUX clocking including clock inputs and the clock diagram.

13.4.5.1 AUDMUX Clock Inputs

The IP Bus read/write clock— peripheral clock—is an input to the AUDMUX. It is used for all AUDMUX register accesses. It is driven only when there is an AUDMUX access on the IP bus.

The AUDMUX uses the selected Tx/Rx bit clock to drive the synchronous switching of the CE Bus Network mode. Specifically, this clock is used to drive the ce_bus_dis signal generation logic. The bit clock used for this function can come from either internal serial interfaces or external codecs. The clock used for CE bus network mode is determined by PTCR7. See "ce_bus_dis Signal Generation" for more details of the clock selection for CE bus network mode.

13.4.5.2 AUDMUX Clock Diagram

Figure 13-22 shows the clocking used in the AUDMUX.

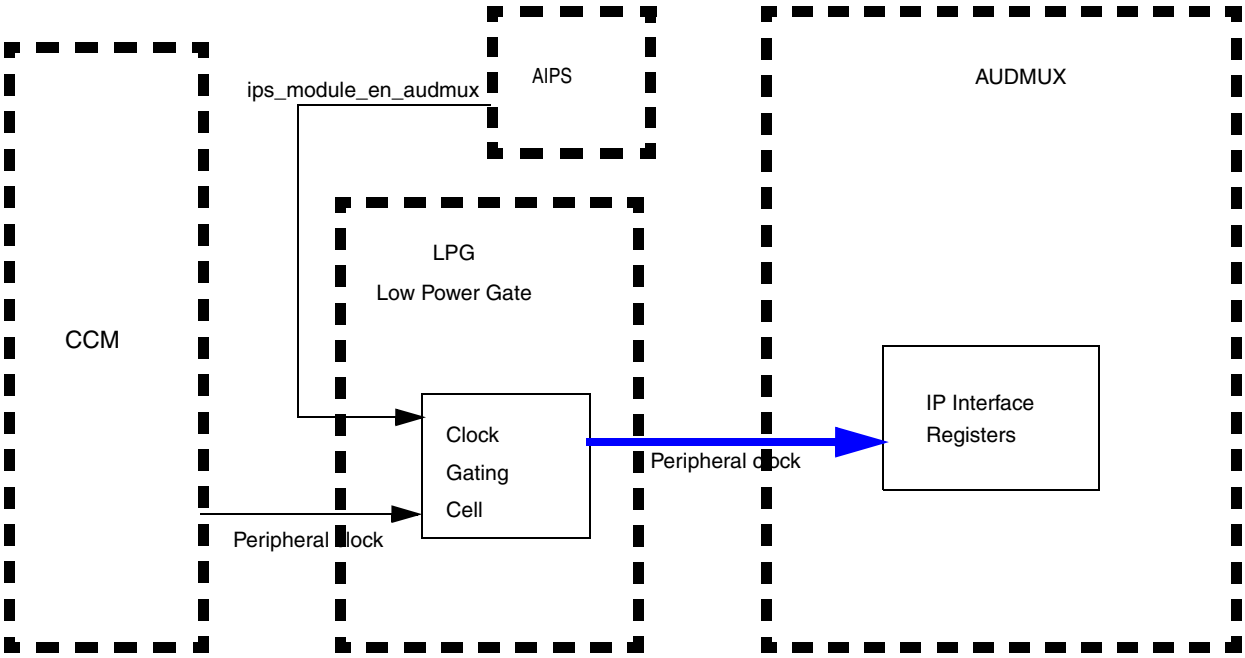


Figure 13-22. AUDMUX Clocking Scheme

13.4.5.3 Clocking Restrictions

Since the AUDMUX requires only peripheral clock, the AUDMUX places no restrictions on the bus frequency. All registers in the AUDMUX are control registers so their values do not change frequently. Their values are programmed when changing between use cases (not during use cases).

Chapter 14
ARM926EJ-S Simple Interrupt Controller (ASIC)

14.1 Introduction

The ARM926EJ-S Simple Interrupt Controller (ASIC) is a 32-bit peripheral which collects interrupt requests from up to 64 sources and provides an interface to the ARM926EJ-S core. The ASIC includes:

- Hardware acceleration of normal and fast interrupts
- Software-controlled priority levels for normal interrupts.

Figure 14-1 provides a block diagram of the ASIC.

14.1.1 Features

The ASIC module includes the following features:

- Supports up to 64 interrupt sources
- Supports fast and normal interrupts
- Selects normal or fast interrupt request for any interrupt source
- Indicates pending interrupt sources using a register for normal and fast interrupts
- Indicates highest priority interrupt number using register (can be used a table index)
- Independently enable or disable any interrupt source
- Provides a mechanism for software to schedule an interrupt
- Supports up to 16 software controlled priority levels for normal interrupts and priority masking
- Single bit disabling of all normal interrupts and of all fast interrupts, used in enabling of secure operations

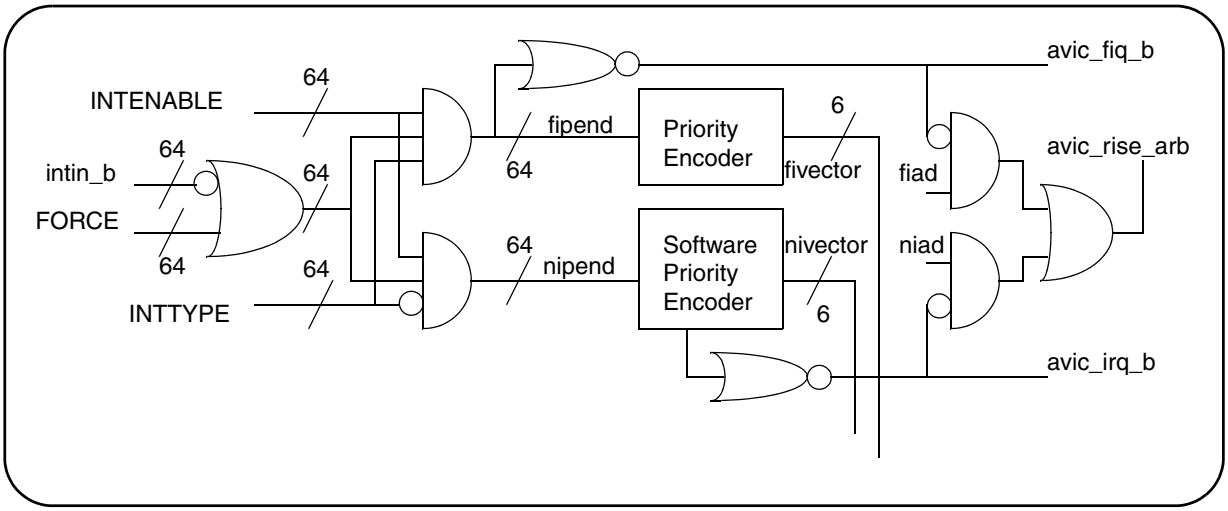


Figure 14-1. ASIC Block Diagram

14.2 Overview

The interrupt controller consists of a set of control registers and associated logic to perform interrupt masking, priority support, and hardware acceleration of normal interrupts.

The control registers are summarized as follows:

- The interrupt source registers (INTSRCH and INTSRCL) reflect the status of up to 64 interrupt sources.
- The interrupt force registers (INTFRCH and INTFRCL) are used to assert interrupt requests corresponding to the different interrupt sources.
- The interrupt enable registers (INTENABLEH and INTENABLEL) allow individual bit masking of the interrupt source registers.
- The interrupt type registers (INTTYPEH and INTTYPEL) determine the interrupt type (normal or fast) associated with each interrupt source.
- The normal interrupt pending registers (NIPNDH and NIPNDL) indicate pending normal interrupt requests. These registers are equivalent to the logical AND of the interrupt source and enable registers (INTENABLEH / INTENABLEL), together with the NOT of the interrupt type registers. (corresponding to the “nipend” signal in Figure 14-1) The NIPNDH and NIPNDL registers are bitwise NORed together to form the nIRQ signal (avic_irq_b in Figure 14-1), which is routed to the ARM926EJ-S core and is maskable by the normal interrupt disable bit in the processor status register (CPSR). The normal interrupt vector register (NIVECSR) indicates the vector index of the highest-priority pending normal interrupt.
- The fast interrupt pending registers (FIPNDH and FIPNDL) play the same role with fast interrupts as the normal interrupt pending registers do with normal interrupts. The bitwise nor of FIPNDH and FIPNDL forms the nFIQ signal (avic_fiq_b in Figure 14-1), which is maskable by the fast interrupt disable bit in the CPSR. The fast interrupt vector register (FIVECSR) indicates the vector index of highest priority pending fast interrupt.

All interrupt controller registers are readable and writable in privileged mode only. Attempted writes to read-only registers are ignored. These registers are all 32 bits wide, and can be only modified using 32-bit writes.

Interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest number
2. Normal interrupt requests, in order of highest priority level, then highest source number with the same priority

The ASIC provides 16 software-controlled priority levels for normal interrupts. Any interrupt can be placed in any priority level. The ASIC also provides a normal interrupt priority level mask (NIMASK), which disables any interrupt with a priority level less than or equal to the mask. If a level-0 normal interrupt and a level-1 normal interrupt are asserted at the same time, the level-1 normal interrupt is selected assuming that NIMASK has not disabled level-1 normal interrupts. If two level-1 normal interrupts are asserted at the same time the level-1 normal interrupt with the highest source number is selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

14.3 Interrupt Controller Programming Model

14.3.1 Register Summary

The ASIC module has 26 registers. All of these registers are single cycle access, as the ASIC sits on the ARM926EJ-S platform slave port 3 AHB.

Table 14-1 summarizes the ASIC registers—for the module base address, see the system memory map. Figure 14-2 is a key for the notations used in the table.

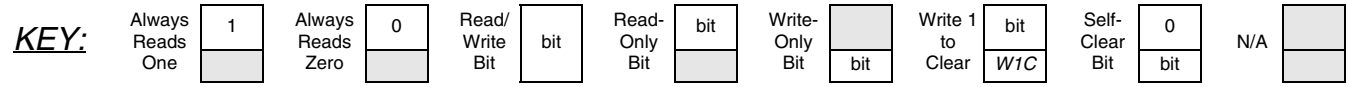


Figure 14-2. Key to Register Summary

Table 14-1. ASIC Register Summary

Name (Base Address Offset)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTCNTL (0x0000)	R	0	0	0	0	0	0	0	0	BYP_EN	NIDIS	FIDIS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																																
NIMASK (0x0004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
	W																																
INTENNUM (0x0008)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											ENNUM				
INTDISNUM (0x000C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																											DISNUM				
INTENABLEH (0x0010)	R	INTENABLE[63:32]																															
	W																																
INTENABLEL (0x0014)	R	INTENABLE[31:0]																															
	W																																

Table 14-1. ASIC Register Summary

Name (Base Address Offset)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTTYPEH (0x0018)	R	INTTYPE[63:32]																															
	W																																
INTTYPEL (0x001C)	R	INTTYPE[31:0]																															
	W																																
NIPRIORITY7 (0x0020)	R	NIPR63				NIPR62				NIPR61				NIPR60				NIPR59				NIPR58				NIPR57				NIPR56			
	W																																
NIPRIORITY6 (0x0024)	R	NIPR55				NIPR54				NIPR53				NIPR52				NIPR51				NIPR50				NIPR49				NIPR48			
	W																																
NIPRIORITY5 (0x0028)	R	NIPR47				NIPR46				NIPR45				NIPR44				NIPR43				NIPR42				NIPR41				NIPR40			
	W																																
NIPRIORITY4 (0x002C)	R	NIPR39				NIPR38				NIPR37				NIPR36				NIPR35				NIPR34				NIPR33				NIPR32			
	W																																
NIPRIORITY3 (0x0030)	R	NIPR31				NIPR30				NIPR29				NIPR28				NIPR27				NIPR26				NIPR25				NIPR24			
	W																																
NIPRIORITY2 (0x0034)	R	NIPR23				NIPR22				NIPR21				NIPR20				NIPR19				NIPR18				NIPR17				NIPR16			
	W																																
NIPRIORITY1 (0x0038)	R	NIPR15				NIPR14				NIPR13				NIPR12				NIPR11				NIPR10				NIPR9				NIPR8			
	W																																
NIPRIORITY0 (0x003C)	R	NIPR7				NIPR6				NIPR5				NIPR4				NIPR3				NIPR2				NIPR1				NIPR0			
	W																																
NIVECSR (0x0040)	R	NIVECTOR																NIPRILVL															
	W																																
FIVECSR (0x0044)	R	FIVECTOR																															
	W																																
INTSRCH (0x0048)	R	INTIN[63:32]																															
	W																																
INTSRCL (0x004C)	R	INTIN[31:0]																															
	W																																
INTFRCH (0x0050)	R	FORCE[63:32]																															
	W																																
INTFRCL (0x0054)	R	FORCE[31:0]																															
	W																																
NIPNDH (0x0058)	R	NIPEND[63:32]																															
	W																																
NIPNDL (0x005C)	R	NIPEND[31:0]																															
	W																																
FIPNDH (0x0060)	R	FIPEND[63:32]																															
	W																																
FIPNDL (0x0064)	R	FIPEND[31:0]																															
	W																																

14.3.2 Detailed Register Descriptions

14.3.2.1 Interrupt Control Register

The interrupt control register (INTCNTL) controls the hardware acceleration done by the ASIC. Both normal interrupts and fast interrupts can be enabled to jump directly to the interrupt service routine.

This register is located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. This register can only be modified using 32-bit writes.

Offset 0x0000 (INTCNTL) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	BYP_EN	NIDIS	FIDIS	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-3. Interrupt Control Register

Table 14-2. Interrupt Control Register Field Descriptions

Field	Description
31–24	Reserved.
23 BYPASS_EN	Bypass enable. This bit, when reset, disables propagation of asynchronous interrupts. 0 Synchronized interrupts (reset value) 1 Asynchronous interrupts.(Backward compatibility.)
22 NIDIS	Normal Interrupt Disable. This bit, when set, disables the generation of the normal interrupt signal. This bit is similar to the I bit of the ARM926EJ-S core. This bit along with the FIDIS bit is used to enable secure operations. 0 Does not affect the normal interrupt generation 1 Disable all normal interrupts
21 FIDIS	Fast Interrupt Disable. This bit, when set, disables the generation of the fast interrupt signal. This bit is similar to the F bit of the ARM926EJ-S core. This bit along with the NIDIS bit is used to enable secure operations. 0 Does not affect the fast interrupt generation 1 Disable all fast interrupts
20–0	Reserved.

14.3.2.2 Normal Interrupt Mask Register (NIMASK)

The normal interrupt mask register (NIMASK) controls the normal interrupt mask level. All normal interrupts with a priority level less than or equal to the value of the NIMASK field (in 2's complement notation) are disabled. The priority level of normal interrupts are determined by the normal interrupt priority level registers NIPRIORITY*n*, *n* = 0…7.

Any setting with NIMASK[4] = 1 corresponds to a negative number in 2's complement notation, and does not disable any normal interrupt priority levels. The reset value of NIMASK is 0b1_1111 (corresponding to –1, in 2's complement notation), which does not disable any normal interrupts.

This hardware mechanism can be used to create reentrant normal interrupt routines by disabling lower priority normal interrupts. Refer to Section 14.4.5, “Writing Reentrant Normal Interrupt Routines,” for more details on the use of the NIMASK register.

This register is located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. This register can be only modified using 32-bit writes.

Offset 0x0004 (NIMASK) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	NIMASK				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-4. Normal Interrupt Mask Register

Table 14-3. NIMASK Descriptions

Field	Description
31–5	Reserved.
4–0 NIMASK	Normal Interrupt Mask. Controls normal interrupt mask level. All normal interrupts of priority level lower than or equal to the NIMASK are disabled. 00000 Disable priority level 0 normal interrupts 00001 Disable priority level 1 and lower normal interrupts ... 011111 Disable all normal interrupts 1*nnnn* Do not disable any normal interrupts.

14.3.2.3 Interrupt Enable Number Register (INTENNUM)

The interrupt enable number register (INTENNUM) provides a hardware-accelerated enabling of interrupts. Any write to this register enables one interrupt source. If the ENNUM field is equal to 0b00_0000, then interrupt source 0 is enabled; if the ENNUM field equal 0b00_0001, then interrupt source 1 is enabled; and so forth. This register is decoded into a “one-hot mask is logically ORed with the INTENABLEH / INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to enable an interrupt source. For example, to enable interrupts 10 and 20 the software can perform two writes to the ASIC: first write 10 to INTENNUM register, then write 20 to INTENNUM register (the order of the writes is irrelevant).

This register is located on the ARM926EJ-S platform slave port 3 AHB. It is accessible in 1 cycle, and can only be accessed in privileged mode. This register can be only modified using 32-bit writes. This register always reads back all 0s.

Offset 0x0008 (INTENNUM) Access: Privileged write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W											ENNUM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-5. Interrupt Enable Number Register

Table 14-4. INTENNUM Register Field Descriptions

Field	Description
31–6	Reserved.
5–0 ENNUM	Interrupt Enable Number. Writing to this register enables the interrupt source associated with this value. ENNUM bits are self-clearing. 0 Enable interrupt source 0 1 Enable interrupt source 1 ... 63 Enable interrupt source 63

14.3.2.4 Interrupt Disable Number Register (INTDISNUM)

The interrupt disable number register (INTDISNUM) provides a hardware-accelerated disabling of interrupts. Any write to this register disables one interrupt source. If the DISNUM field is equal 0b000000, then interrupt source 0 is disabled; if the DISNUM = 0b000001, then interrupt source 1 is disabled; and so forth. This register is decoded into a “one-hot” mask that is inverted and logically ANDed with the INTENABLEH / INTENABLEL register.

This hardware mechanism alleviates the need for an atomic read/modify/write sequence to disable an interrupt source. For instance, to disable interrupts 10 and 20, the software can write 10 to INTDISNUM register, then write 20 to INTDISNUM register (the order of the writes is irrelevant).

This register is located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. This register can be only modified using 32-bit writes. This register always reads back all 0s.

Offset 0x000C (INTDISNUM) Access: Privileged write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W											DISNUM					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-6. Interrupt Disable Number Register

Table 14-5. INTDISNUM Register Field Descriptions

Field	Description
31 — 6	Reserved.
5 – 0 DISNUM	Interrupt Disable Number. Writing to this register disables the interrupt source associated with this value. The bits are self-clearing after they are written. 0 Disable interrupt source 0 1 Disable interrupt source 1 ... 63 Disable interrupt source 63

14.3.2.5 Interrupt Enable Registers (INTENABLEH and INTENABLEL)

The interrupt enable register high (INTENABLEH) and the interrupt enable register low (INTENABLEL) are used to enable pending interrupt requests to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers are all interrupts masked.

This register can be updated by the following methods:

- Writing directly to the INTENABLEH / INTENABLEL registers,
- Setting bits with the INTENNUM register,
- Clearing bits with the INTDISNUM register.

These registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. These registers can only be modified using 32-bit writes.

Offset 0x0010 (INTENABLEH) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTENABLE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTENABLE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-7. Interrupt Enable Register High

Offset 0x0014 (INTENABLEL)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTENABLE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTENABLE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-8. Interrupt Enable Register Low

Table 14-6. INTENABLEH / INTENABLEL Descriptions

Field	Description
63–0 INTENABLE	Interrupt Enable. This bit enables the corresponding interrupt source to request a normal interrupt or a fast interrupt. A reset operation clears this bit. If an enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal or a fast interrupt request depending on the associated INTTYPEH / INTTYPEL setting. 0 Interrupt disabled 1 Interrupt enabled and generates a normal or fast interrupt upon assertion

14.3.2.6 Interrupt Type Registers (INTTYPEH and INTTYPEL)

The interrupt type register high (INTTYPEH) and the interrupt type register low (INTTYPEL) are used to select whether a pending interrupt source, when enabled with the INTENABLEH / INTENABLEL, will cause a normal interrupt or a fast interrupt to the core. Each bit in this register corresponds to an interrupt source available in the system. The reset state of these registers cause all enabled interrupt sources to generate a normal interrupt.

These registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. These registers can be only modified using 32-bit writes.

Offset 0x0018 (INTTYPEH) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTTYPE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTTYPE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-9. Interrupt Type Register High

Offset 0x001C (INTTYPEL) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTTYPE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTTYPE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-10. Interrupt Type Register Low

Table 14-7. INTTYPEH / INTTYPEL Descriptions

Field	Description
63–0 INTTYPE	Interrupt type. This bit controls whether the corresponding interrupt source requests a normal interrupt or a fast interrupt. If a INTTYPE bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request. 0 Interrupt source generates a normal interrupt (nIRQ) 1 Interrupt source generates a fast interrupt (nFIQ)

14.3.2.7 Normal Interrupt Priority Level Registers (NIPRIORITY*n*, *n* = 0…7)

The normal interrupt priority level registers NIPRIORITY*n*, (*n* = 0…7) provide a software-controllable prioritization of normal interrupts. Normal interrupts with a higher priority level preempt normal interrupts with a lower priority. The reset state of these registers forces all normal interrupts to the lowest priority level.

If a level 0 normal interrupt and a level 1 normal interrupt are asserted at the same time, the level 1 normal interrupt is selected assuming that NIMASK has not disabled level 1 normal interrupts. If two level 1 normal interrupts are asserted at the same time, the level 1 normal interrupt with the highest source number is selected, also assuming that NIMASK has not disabled level 1 normal interrupts.

These registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. These registers can be only modified using 32-bit writes.

Offset 0x0020 (NIPRIORITY7) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR63				NIPR62				NIPR61				NIPR60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR59				NIPR58				NIPR57				NIPR56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-11. Normal Interrupt Priority Level Register 7

Offset 0x0024 (NIPRIORITY6) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR55				NIPR54				NIPR53				NIPR52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR51				NIPR50				NIPR49				NIPR48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-12. Normal Interrupt Priority Level Register 6

Offset 0x0028 (NIPRIORITY5) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR47				NIPR46				NIPR45				NIPR44			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR43				NIPR42				NIPR41				NIPR40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-13. Normal Interrupt Priority Level Register 5

Offset 0x002C (NIPRIORITY4) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR39				NIPR38				NIPR37				NIPR36			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR35				NIPR34				NIPR33				NIPR32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-14. Normal Interrupt Priority Level Register 4

Offset 0x0030 (NIPRIORITY3) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR31				NIPR30				NIPR29				NIPR28			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR27				NIPR26				NIPR25				NIPR24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-15. Normal Interrupt Priority Level Register 3

Offset 0x0034 (NIPRIORITY2) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR23				NIPR22				NIPR21				NIPR20			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR19				NIPR18				NIPR17				NIPR16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-16. Normal Interrupt Priority Level Register 2

Offset 0x0038 (NIPRIORITY1) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR15				NIPR14				NIPR13				NIPR12			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR11				NIPR10				NIPR9				NIPR9			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-17. Normal Interrupt Priority Level Register 1

Offset 0x003C (NIPRIORITY0) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPR7				NIPR6				NIPR5				NIPR4			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPR3				NIPR2				NIPR1				NIPR0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-18. Normal Interrupt Priority Level Register 1

Table 14-8. Normal Interrupt Priority Level Register 1 Field Descriptions

Field	Description
31–0 NIPR*n* (*n* = 63…0)	Normal Interrupt Priority Level — Selects the software controlled priority level for the associated normal interrupt source. These registers do not affect the prioritization of fast interrupt priorities. 00 Lowest-priority normal interrupt (reset value) … 15 Highest-priority normal interrupt

14.3.2.8 Normal Interrupt Vector and Status Register

The normal interrupt vector and status register (NIVECSR) provides the priority of the highest pending normal interrupt and provides the vector index of the interrupt's service routine. This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending normal interrupt source.

This read-only register is located on the ARM926EJ-S platform slave port 3 AHB. It is accessible in 1 cycle, and can only be accessed in privileged mode.

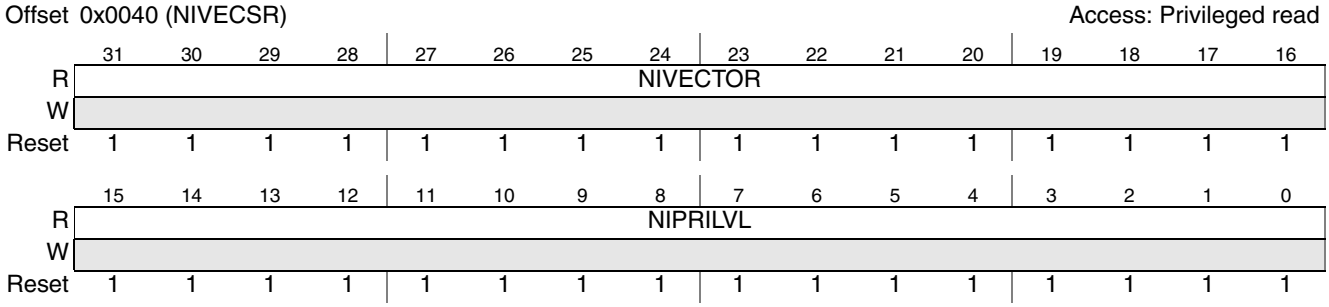


Figure 14-19. Normal Interrupt Vector and Status Register

Table 14-9. NIVECSR Register Field Descriptions

Field	Description
31–16 NIVECTOR	Normal Interrupt Vector — Indicates vector index for the highest pending normal interrupt. … –1 No normal interrupt request pending (reset value) 00 Interrupt 0 highest priority pending normal interrupt 01 Interrupt 1 highest priority pending normal interrupt ... 63 Interrupt 63 highest priority pending normal interrupt 64+ (not –1) = unused, does not occur
15–0 NIPRILVL	Normal Interrupt Priority Level — Indicates the priority level of the highest priority normal interrupt. This number can be written to the NIMASK to disable the current priority normal interrupts to build a reentrant normal interrupt system. –1 No normal interrupt request pending (reset value) 00 Highest priority normal interrupt is level 0 01 Highest priority normal interrupt is level 1 ... 15 Highest priority normal interrupt is level 15 16 (not –1) = unused, does not occur

14.3.2.9 Fast Interrupt Vector and Status Register

The fast interrupt vector and status register (FIVECSR) provides the vector index for the highest priority active fast interrupt's service routine (the higher the source number of the fast interrupt, the higher the priority level). This hardware mechanism replaces the previous necessity for core support of the FF1 command. This number can be directly used as an index into a vector table to select the highest pending fast interrupt source.

This read-only register is located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode.

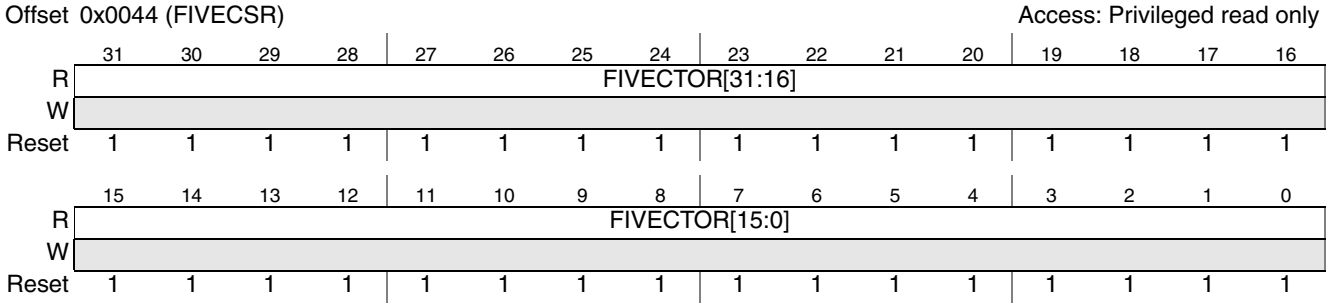


Figure 14-20. Fast Interrupt Vector and Status Register

Table 14-10. FIVECSR Register Field Descriptions

Field	Description
31–0 FIVECTOR	Fast Interrupt Vector. Indicates vector index for the highest pending fast interrupt. -01 No fast interrupt request pending 00 Interrupt 0 highest pending fast interrupt 01 Interrupt 1 highest pending fast interrupt ... 63 Interrupt 63 highest pending fast interrupt 64+ (not -1) = unused, does not occur

14.3.2.10 Interrupt Source Registers

The interrupt source register high (INTSRCH) and the interrupt source register low (INTSRCL) are each 32 bits wide. INTSRCH and INTSRCL reflect the status of all interrupt request inputs into the interrupt controller. Unused bit positions always read zero (no request pending). The state of this register out of reset is determined by the peripheral circuits generating the requests; normally, the requests would be inactive.

These read-only registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. This read-only register should be accessed with 32 bit reads only.

NOTE

The state of these registers out of reset is determined by the peripheral circuits generating the requests. Normally, the requests would be inactive.

Offset 0x0048 (INTSRCH) Access: Privileged read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTIN[63:48]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTIN[47:32]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-21. Interrupt Source Register High

Offset 0x004C (INTSRCL) Access: Privileged read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTIN[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTIN[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-22. Interrupt Source Register Low

Table 14-11. INTSRCH / INTSRCL Field Descriptions

Field	Description
63 –0 INTIN	Interrupt Source. Indicates the state of the corresponding hardware interrupt source. 0 Interrupt source negated 1 Interrupt source asserted

14.3.2.11 Interrupt Force Registers

The interrupt force register high (INTFRCH) and the interrupt force register low (INTFRCL) are each 32 bits wide. The interrupt force registers allow for software generation of interrupts for each of the possible interrupt sources for functional or debug purposes. The system level design may reserve one or more sources for software purposes to allow software to self-schedule interrupts by forcing one or more of these “sources” in the appropriate interrupt force register(s).

These registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode. These registers can be only modified using 32-bit writes.

Offset 0x0050 (INTFRCH) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FORCE[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FORCE[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-23. Interrupt Force Register High

Offset 0x0054 (INTFRCL) Access: Privileged read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FORCE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FORCE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-24. Interrupt Force Register Low

Table 14-12. INTFRCH / INTFRCL Descriptions

Field	Description
63–0 FORCE	Interrupt Source Force Request. Used to force a request for the corresponding interrupt source. 0 Standard interrupt operation 1 Interrupt force asserted

14.3.2.12 Normal Interrupt Pending Registers

The normal interrupt pending register high (NIPNDH) and the normal interrupt pending register low (NIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers; therefore, the reset state of these registers are determined by the normal interrupt enable registers, the interrupt mask register, and the interrupt source registers. The value reflected in these registers is unaffected by the value of the NIMASK register.

These read-only registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode.

Offset 0x0058 (NIPNDH) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPEND[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPEND[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-25. Normal Interrupt Pending Register High

Offset 0x005C (NIPNDH) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	NIPEND[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NIPEND[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-26. Normal Interrupt Pending Register High

NIPNDH 0x0058

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPEND[63:48]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPEND[47:32]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-27. Normal Interrupt Pending Register High

NIPNDL 0x005C

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	NIPEND[31:16]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	NIPEND[15:0]															
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-28. Normal Interrupt Pending Register Low

Table 14-13. NIPNDH / NIPNDL Descriptions

Field	Description
63–0 NIPEND	Normal Interrupt Pending Bit. If a normal interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a normal interrupt request. The normal interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a normal interrupt. 0 No normal interrupt request 1 Normal interrupt request pending

14.3.2.13 Fast Interrupt Pending Registers (FIPNDH)

The fast interrupt pending register high (FIPNDH) and the fast interrupt pending register low (FIPNDL) are 32-bit wide registers used to monitor the outputs of the enable and masking operations. These registers are actually only a set of buffers; therefore, the reset state of these registers are determined by the fast interrupt enable registers, the interrupt mask register, and the interrupt source registers.

These read-only registers are located on the ARM926EJ-S platform slave port 3 AHB, accessible in 1 cycle, and can only be accessed in privileged mode.

Offset 0x0060 (FIPNDH) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIPEND[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIPEND[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-29. Fast Interrupt Pending Register High

Offset 0x0064 (FIPNDL) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FIPEND[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIPEND[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-30. Fast Interrupt Pending Register Low

Table 14-14. FIPNDH / FIPNDL Descriptions

Field	Description
63–0 FIPEND	Fast Interrupt Pending Bit. If a fast interrupt enable bit is set and the corresponding interrupt source is asserted, the interrupt controller asserts a fast interrupt request. The fast interrupt pending bits reflect the interrupt input lines which are asserted and are currently enabled to generate a fast interrupt. 0 No fast interrupt request 1 Fast interrupt request pending

14.4 ARM926EJ-S Interrupt Controller Operation

14.4.1 ARM926EJ-S Prioritization of Exception Sources

The ARM926EJ-S core imposes the following priority among the various exceptions:

- Reset (highest priority)
- Data Abort
- Fast Interrupt
- Normal Interrupt
- Prefetch Abort
- Undefined Instruction and SWI (lowest priority)

14.4.2 ASIC Prioritization of Interrupt Sources

The ASIC module prioritizes the various interrupt sources by source number where higher source numbers have higher priority. Fast interrupt always have higher priority over normal interrupts.

The interrupt requests are prioritized in the following sequence:

1. Fast interrupt requests, in order of highest source number
2. Normal interrupt requests, in order of highest priority level, then in order of highest source number with the same priority level

14.4.3 Assigning and Enabling Interrupt Sources

The interrupt controller provides for flexible assignment of any interrupt source to either of the two core interrupt request inputs. This is done by setting the appropriate bits in the INTENABLEH / INTENABLEL registers and the INTTYPEH / INTTYPEL registers. Usually, interrupt assignment is done once during system initialization and does not affect interrupt latency.

Interrupt assignment is the first of three steps required to enable an interrupt source, and this is done at chip integration. The second step is to program the source to generate interrupt requests. The final step is to

enable the interrupt inputs in the core by clearing the normal interrupt disable (I) and/or the fast interrupt disable (F) bits in the program status register (CPSR).

14.4.4 Enabling Interrupts Sources

There are two methods of enabling or disabling interrupts in the ASIC:

- One method is to read the INTENABLEH / INTENABLEL registers, logically OR or BIT CLEAR the read values with generated masks, then write back to the INTENABLEH / INTENABLEL registers.
- A second method is to perform an atomic write to the source number in the INTENNUM register. The ASIC decodes this 6-bit register, and enable one of the 64 interrupt sources. The ASIC automatically generates a “one-hot” enable mask and logically OR this mask to the correct INTENABLEH or INTENABLEL register. Interrupts are disabled in exactly the same way, except the source number is written to the INTDISNUM register.

14.4.5 Writing Reentrant Normal Interrupt Routines

The ASIC can be used to create a reentrant normal interrupt system, as described below. This enables preempting of lower priority level interrupts by higher priority level interrupts. This requires a small amount of software support and overhead.

1. Push the link register (LR_irq) on to the stack (SP_irq)
2. Push the saved status register (SPSR_irq) on to the stack
3. Read the current value of NIMASK and push this value on to the stack
4. Read current priority level using NIVECSR
5. Interrupts of the equal or lesser priority than the current priority level should be masked using the NIMASK register by writing value from NIVECSR
6. Clear the I bit in the ARM926EJ-S core using a MSR / MRS command sequence (now a higher priority normal interrupt can preempt a lower priority one)
   Also change the operating mode of the core to System Mode from IRQ mode
7. Push System Mode link register (LR) on to the stack (SP_user)
8. The traditional interrupt service routine is now included
9. Pop System Mode link register (LR) from the stack (SP_user)
10. Set I bit in the ARM926EJ-S core using a MSR / MRS command sequence (thus disabling all normal interrupts)
    Also change the operating mode of the core to IRQ Mode from System mode
11. Pop the original value of normal interrupt mask and write to the NIMASK register
12. The saved status register should be popped from the stack (SP_irq)

13. The link register should be popped from the stack into the PC
14. Return from nIRQ

NOTE

Steps 1, 2, 13, and 14 are performed automatically by most C compilers, and are included for completeness.

14.4.6 Low Power Entry Sequence

The following sequence is recommended to enter low-power mode, with respect to the ARM9 platform.

1. Mask interrupts in ARM core using core instructions
2. Program other low-power mode entry steps including programming the low power mode (LP CTL) Bits in CCM
3. Program ASIC and external wake-up controller for same wake-up interrupts. Mask the non-wakeable interrupts.
4. ARM platform does internal housekeeping tasks such as draining the L1 and L2 buffers, flushing the caches, and so on.
5. Read the status of pending interrupts, if any wake-up interrupt is pending, service the pending interrupt (after unmasking the interrupts in the core) and then restart the low power sequence (if required)
6. If there is no pending wake-up interrupt, then execute WFI Instruction.
7. Core performs internal house keeping and asserts StandbyWFI. Based on StandbyWFI, the platform asserts a9p_clk_off.
8. CCM sends out the MAX_HALT_REQUEST.
9. MAX_HALTED is received, CCM requests other modules (if programmed) to enter low-power mode.

The following sequence is recommended for low-power exit:

1. Whenever a wake-up interrupt is detected by external wake-up controller or by the platform's ASICn (until the platform clocks are alive), CCM's A926P_CLK_OFF is negated.
2. The system takes steps to exit low power mode. CCM restarts the core clocks.
3. The platform synchronize the pending interrupt and exit standby WFI mode.
4. Unmask/Enable the interrupts in the core by core instructions.
5. Core services the interrupt, and restarts execution from the previous stage.

NOTE

In case the platform is programmed to bypass the interrupt synchronizers (ASIC INTCNTL[23] = 1) in low power mode, then after step 3 of the low-power exit sequence, the core has to wait for few hclk cycles till the standby WFI signal has propagated through all the synchronizations flops stages. So software needs to have enough nops or dummy instructions to avoid these hclk cycles. By default, interrupt synchronizers are not bypassed.

14.4.7 AHB Interface of ASIC

The ASIC is AHB-compatible. This means that IDLE or BUSY cycles that are presented to the ASIC receive an avic_hready (as required by specification).

The ASIC reports a transfer error if the HPROT[3:1] bits are not equal to 001. This means that only privileged accesses are allowed to the ASIC.

The ASIC reports a transfer error if non-32 bit writes are attempted. Any size read is allowed.

The ASIC reports a transfer error if address of the access is not within the range of one of the ASIC registers.

The ASIC does not report a transfer error if writes are performed to read-only registers. These writes are ignored and have no affect on the ASIC.

Chapter 15
Clock Controller Module (CCM)

15.1 Introduction

The CCM performs the following functions:

- Controls the system frequency
- Distributes clocks to various parts of the chip
- Controls the reset mechanism of the chip
- Provides advanced low-power management

15.1.1 Overview

The CCM contains four functional parts:

- Clock control and gating logic
- Reset control logic
- Boot control logic
- Low-power control logic

Figure 15-1 shows a top-level block diagram of the CCM.

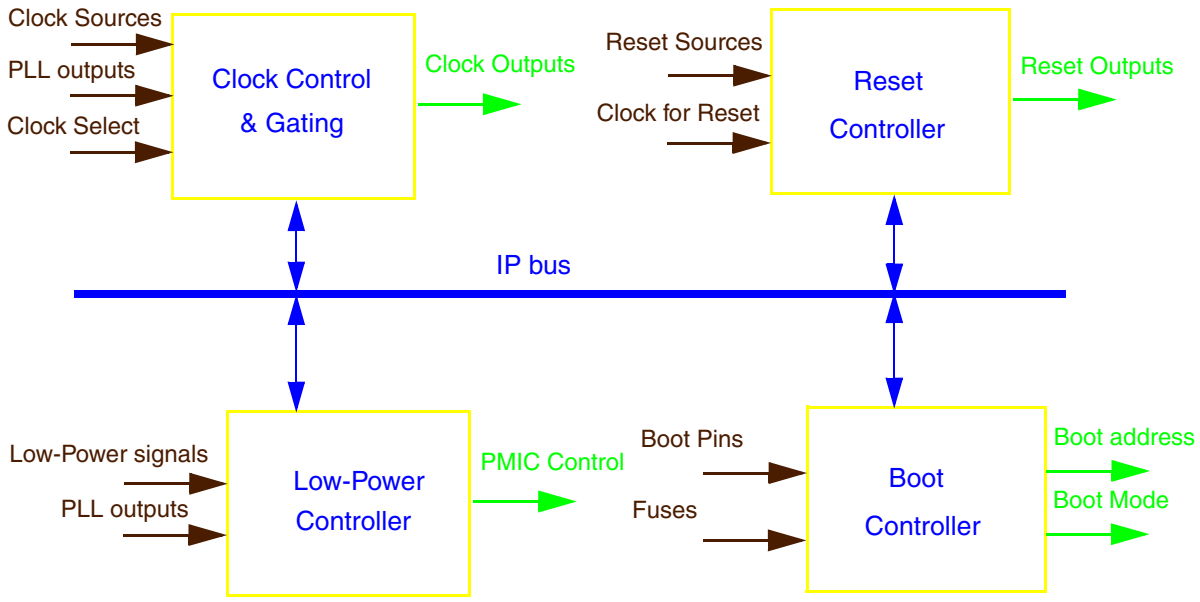


Figure 15-1. CCM Block Diagram

15.1.2 Features

The CCM includes these distinctive features:

- Core PLL—supports seamless switching of the ARM Clock among 532/399/266/133 MHz
- Clock distributions with clock gating on each clock output—minimizes power consumption in the clock tree
- Reset distributions with security check—helps with security of the chip
- Boot controller—provides different boot information to the chip based on BOOT pins
- Power manager—controls power modes and minimize power consumption using techniques such as AWB, power gating, DPTC, and DVFS
- Registers—accessible using IP bus

15.1.3 Modes of Operation

There are three modes of operation:

- Normal operating mode
- Low-power mode
- Debug mode

15.1.3.1 Normal Operating Mode

The normal operating mode is run mode. The clocks are generated by PLLs.

15.1.3.2 Low-Power Modes

The low-power modes are actually system low-power modes. There are three low-power modes:

- Wait
- Doze
- Stop

NOTE

The DryIce module (which handles volatile key storage and supplies a trusted time source) and its 32 KHz oscillator can be powered independently during these low-power modes. See the DryIce chapter for more details.

15.2 External Signal Description

Table 15-1 describes all signals that connect off-chip.

Table 15-1. External Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
OSC24M_EXTAL	osc24m_clk	OSC24M external input	I	—	—
CLK_SEL	ipp_clk_sel	Clock mode select input	I	—	—

Table 15-1. External Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
BOOTIN[1:0]	ipp_boot_in	Boot mode pins	I	—	—
POR_B	ipp_por_reset_in_b	Power-on-reset signal from external pad	I	—	Active
RESET_IN_B	ipp_reset_in_b	External reset signal. All modules are reset but the PLL, fuse and CCM.	I	—	Active
OCSC32K_EXTAL	ext32k_clk	OSC32K external input	I	—	—
CLKO	ipp_clko	CKO clockout pin	O	0	—

15.3 Memory Map and Register Definition

This section provides memory maps and detailed descriptions of all registers.

15.3.1 Memory Map

Table 15-2 shows the CCM memory map. For the base address of a particular module instantiation, see the system memory map.

Table 15-2. CCM Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
General Registers				
0x0000 (MPCTL)	Core PLL Control Register (MPCTL)	R/W	0x800B_2C01	15.3.3.1/15-9
0x0004 (UPCTL)	USB PLL Control Register (UPCTL)	R/W	0x8400_2800	15.3.3.2/15-11
0x0008 (CCTL)	Clock Control Register (CCTL)	R/W	0x4003 0000	15.3.3.3/15-13
0x000C (CGCR0)	Clock Gating Control Register 0 (CGCR0)	R/W	0x028A_0100	15.3.3.4/15-15
0x0010 (CGCR1)	Clock Gating Control Register 1 (CGCR1)	R/W	0x0400_8100	15.3.3.5/15-17
0x0014 (CGCR2)	Clock Gating Control Register 2 (CGCR2)	R/W	0x0000_0438	15.3.3.6/15-17
0x0018 (PCDR0)	PER Clock Divider Register 0 (PCDR0)	R/W	0x0101_0101	15.3.3.7/15-19
0x001C (PCDR1)	PER Clock Divider Register 1 (PCDR1)	R/W	0x0101_0101	15.3.3.8/15-22
0x0020 (PCDR2)	PER Clock Divider Register 2 (PCDR2)	R/W	0x0101_0101	15.3.3.9/15-23
0x0024 (PCDR3)	PER Clock Divider Register 3 (PCDR3)	R/W	0x0101_0101	15.3.3.10/15-24
0x0028 (RCSR)	CCM Status Register (RCSR)	R/W	0x0000_0000	15.3.3.11/15-26
0x002C (CRDR)	CCM Reset and Debug Register (CRDR)	R/W	0x0000_0000	15.3.3.12/15-29
0x0030 (DCVR0)	DPTC Comparator Value Registers (DCVR0–DCVR3)	R/W	0x0000_0000	15.3.3.13/15-30
0x0034 (DCVR1)	DPTC Comparator Value Registers (DCVR0–DCVR3)	R/W	0x0000_0000	15.3.3.13/15-30
0x0038 (DCVR2)	DPTC Comparator Value Registers (DCVR0–DCVR3)	R/W	0x0000_0000	15.3.3.13/15-30

Table 15-2. CCM Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x003C (DCVR3)	DPTC Comparator Value Registers (DCVR0–DCVR3)	R/W	0x0000_0000	15.3.3.13/15-30
0x0040 (LTR0)	Load Tracking Register 0 (LTR0)	R/W	0x0000_0000	15.3.3.14/15-31
0x0044 (LTR1)	Load Tracking Register 1 (LTR1)	R/W	0x0000_0000	15.3.3.15/15-32
0x0048 (LTR2)	Load Tracking Register 2 (LTR2)	R/W	0x0000_0000	15.3.3.16/15-33
0x004C (LTR3)	Load Tracking Register 3 (LTR3)	R/W	0x0000_0000	15.3.3.17/15-34
0x0050 (LTBR0)	Load Tracking Buffer Register 0 (LTBR0)	R/W	0x0000_0000	15.3.3.18/15-35
0x0054 (LTBR1)	Load Tracking Buffer Register 1 (LTBR1)	R/W	0x0000_0000	15.3.3.19/15-36
0x0058 (PMCR0)	Power Management Control Register 0 (PMCR0)	R/W	0x002C_9828	15.3.3.20/15-37
0x005C (PMCR1)	Power Management Control Register 1 (PMCR1)	R/W	0x00A0_0000	15.3.3.21/15-39
0x0060 (PMCR2)	Power Management Control Register 2 (PMCR2)	R/W	0x0000_A030	15.3.3.22/15-41
0x0060 (PMCR2)	Power Management Control Register 2 (PMCR2)	R/W	0x0000_A030	15.3.3.22/15-41
0x0064 (MCR)	Miscellaneous Control Register (MCR)	R/W	0x4300_0000	15.3.3.23/15-42
0x0068 (LPIMR0)	Low Power Interrupt Mask Registers (LPIMR0)	R/W	0x0000_0000	15.3.3.24/15-44
0x006C (LPIMR1)	Low Power Interrupt Mask Registers (LPIMR1)	R/W	0x0000_0000	15.3.3.25/15-44

15.3.2 Register Summary

The conventions in Figure 15-2 and Table 15-3 serve as a key for the register summary and individual register diagrams.

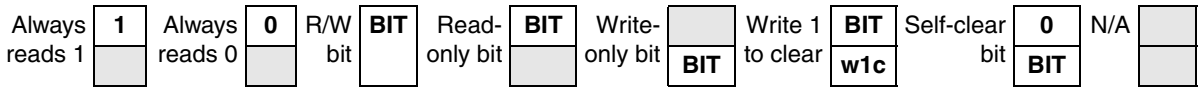


Figure 15-2. Key to Register Fields

Table 15-3 provides a key for register figures and tables and the register summary.

Table 15-3. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.

Table 15-3. Register Conventions (continued)

Convention	Description
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 15-4 shows the CCM register summary table.

Table 15-4. CCM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (MPCTL)	R	BRMO	0	PD				MFD									
	W																
	R	LOCK	0	MFI				MFN									
	W																
0x0004 (UPCTL)	R	BRMO	0	PD				MFD									
	W																
	R	LOCK	0	MFI				MFN									
	W																
0x0008 (CCTL)	R	CLK MUX		POST DIV		MPLL RST	UPLL RST	LP CTL		UPLL DIS		USB DIV					
	W																
	R	CG CTL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x000C (CGCR0)	R	0	0	0	AHB Clock Gating												
	W																
	R	PER Clock Gating															
	W																

Table 15-4. CCM Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x0010 (CGCR1)	R	IPG Clock Gating[31:16]															
	W																
	R	IPG Clock Gating[15:0]															
	W																
0x0014 (CGCR2)	R	0	0		0	0		0	0		0	0		IPG Clock Gating[51:48]			
	W																
	R	IPG Clock Gating[47:32]															
	W																
0x0018 (PCDR0)	R	0	0	PER3 DIV						0	0	PER2 DIV					
	W																
	R	0	0	PER1 DIV						0	0	PER0 DIV					
	W																
0x001C (PCDR1)	R	0	0	PER7 DIV						0	0	PER6 DIV					
	W																
	R	0	0	PER5 DIV						0	0	PER4 DIV					
	W																
0x0020 (PCDR2)	R	0	0	PER11 DIV						0	0	PER10 DIV					
	W																
	R	0	0	PER9 DIV						0	0	PER8 DIV					
	W																
0x0024 (PCDR3)	R	0	0	PER15 DIV						0	0	PER14 DIV					
	W																
	R	0	0	PER13 DIV						0	0	PER12 DIV					
	W																
0x0028 (RCSR)	R	MEM CTRL		MEM TYPE		PAGE SIZE		BUS WIDTH		USB SRC		BT SRC		BT RES			
	W																
	R	SOFT_RESET	NFC_16bit_SEL	0	CLK SEL	BOOT REG		NFC_4K	NFC_FMS	SPARE SIZE	BOOT INT	EEPROM CFG	MLC SEL	RESTS			
	W																

Table 15-4. CCM Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x002C (CRDR)	R	BT UART SRC			0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x0030 (DCVR0)	R	ULV										LLV					
	W																
	R	LLV						ELV								0	0
	W																
0x0034 (DCVR1)	R	ULV										LLV					
	W																
	R	LLV						ELV								0	0
	W																
0x0038 (DCVR2)	R	ULV										LLV					
	W																
	R	LLV						ELV								0	0
	W																
0x003C (DCVR3)	R	ULV										LLV					
	W																
	R	LLV						ELV								0	0
	W																
0x0040 (LTR0)	R	0	0	DIV3CK			UPTHR					DNTHR					
	W																
	R	SIG D15	SIG D14	SIG D13	SIG D12	SIG D11	SIG D10	SIG D9	SIG D8	SIG D7	SIG D6	SIG D5	SIG D4	SIG D3	SIG D2	SIG D1	SIG D0
	W																
0x0044 (LTR1)	R	0	0	0	0	0	0	0	0	LT_ BRSH	LT_ BRSR	DNCNT					
	W																
	R	DNCNT		UPCNT								PNCTHR					
	W																
0x0048 (LTR2)	R	WSW15			WSW14			WSW13			WSW12			WSW11			WS W10
	W																
	R	WSW10		WSW9			0	0	EMAC								
	W																

Table 15-4. CCM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004C (LTR3)	R	WSW8				WSW7				WSW6				WSW5			
	W																WSW3
	R	WSW3				WSW2				WSW1				WSW0			
	W												0	0	0	0	0
0x0050 (LTBR0)	R	LTS7				LTS6				LTS5				LTS4			
	W																
	R	LTS3				LTS2				LTS1				LTS0			
	W																
0x0054 (LTBR1)	R	LTS15				LTS14				LTS13				LTS12			
	W																
	R	LTS11				LTS10				LTS9				LTS8			
	W																
0x0058 (PMCR0)	R	0	0	DVSUP		0	0	0	DVFS_UPD_FINISH	DVFEV	DVFIS	LBMI	LBFL	LBCF		PTVIS	DVFS_START
	W																
	R	FSVAIM	FSVAI		DPVCR	DPVV	WFIM	DRCE3	DRCE2	DRCE1	DRCE0	SCR	DVFEN	PTVAIM	PTVAI		DPTEN
	W																
0x005C (PMCR1)	R	0	0	CPEN_EMI	CPFA_EMI	CPSPA_EMI				WBCN							
	W																
	R	0	0	CPEN	CPFA	CPSPA				0	0	0	0	DVGP			
	W																
0x0060 (PMCR2)	R	0	0			0	0			0	0	0	0	0	ARMMEMDWN	VSTBY	OSC24M_DOWN
	W																
	R	ARM MEMON CNT								ARM CLKON CNT				0	0	DVFS_REQ	DVFS_ACK
	W																

Table 15-4. CCM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0064 (MCR)	R	USB XTA LMUX	CLKO EN	CLKO DIV						CLKO SEL				ESAI CLK MUX	SSI2 CLK MUX	SSI1 CLK MUX	USB CLK MUX
	W																
	R	PER CLK MUX															
	W																
0x0068 (LPIMR0)	R	LPIM[31:16]															
	W																
	R	LPIM[15:0]															
	W																
0x006C (LPIMR1)	R	LPIM[63:48]															
	W																
	R	LPIM[47:32]															
	W																

15.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number.

15.3.3.1 Core PLL (MPLL) Control Register (MPCTL)

This register contains parameters which determine the output frequency of the core PLL (MPLL), which is given by F_{VCO} in Equation 15-1.

$$2 \times F_{ref} \times \frac{MF_I + \frac{MF_N}{MF_D}}{PD} = Fvco$$

MPLL Output Frequency ***Eqn. 15-1***

F_{ref}: MPLL reference clock (input frequency)
MFx: Multiplication factors defined by MPCTL register fields (see Table 15-5)
PD: Predivider factor, whose value is also set in the MPCTL register

Figure 15-3 shows the MPCTL fields that are described in Table 15-5.

NOTE

The absolute value of MFN/MFD must be smaller than 1. Changes in the PD, MFD, MFI, and MFN fields take effect (and the PLL is relocked) only after the MPLL_RST bit in the CCTL register is set.

Offset 0x0000 (MPCTL) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BRMO	0	PD				MFD									
W																
Reset	1	0	0	0	0	0*	0	0	0	0	0	0	1	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCK	0	MFI				MFN									
W																
Reset	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1

Figure 15-3. Core PLL Control Register (MPCTL)

Table 15-5. MPCTL Field Descriptions

Name	Description
31 BRMO	Binary rate multiplier (BRM) Order bit. Determines if the BRM is first or second order. The first-order BRM is used if the MF fractional part is between 1/10 and 9/10, or if MF_I > 13. Otherwise the second-order BRM is used. The BRMO bit is cleared by a hardware reset. 1 BRM is second order. 0 BRM is first order.
30	Reserved
29–26 PD	**Predivider factor.** Defines the predivider factor (PD) applied to the PLL input frequency. as specified in Equation 15-2. PD is an integer between 1and 16 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range. 0000 1 0001 2 ... 1111 16 **Note:** Changes in this field take effect (and the PLL is relocked) only after the MPLL_RST bit in the CCTL register is set. **Note:** The default value of PD is depends on the ipp_clk_sel boot up value. (PD=0x0 if ipp_clk_sel = 0; PD = 0x1 if ipp_clk_sel = 1).

Table 15-5. MPCTL Field Descriptions (continued)

Name	Description
25–16 MFD	**Multiplication factor denominator. Defines the denominator part of the BRM value for the MF, as specified by** Equation 15-2. 000000000 Denominator = 1 000000001 Denominator = 2 … 1111111111 Denominator = 1024 **Note:** Changes in this field take effect (and the PLL is relocked) only after the MPLL_RST bit in the CCTL register is set.
15 LOCK	This bit shows if the PLL lock flag is set. 0 PLL lock flag is negated. 1 PLL lock flag is set.
14	Reserved
13–10 MFI	**Multiplication factor integer part**. Defines the integer part of the BRM value for the MF, as specified in Equation 15-2. The MFI is encoded so that MFI < 5 results in MFI = 5. 0000–0101 $MF_I = 5$ 0110 $MF_I = 6$ ... 1111 $MF_I = 15$ **Note:** Changes in this field take effect (and the PLL is relocked) only after the MPLL_RST bit in the CCTL register is set.
9–0 MFN	**Multiplication factor numerator**. Defines the numerator of the BRM value for the MF, as specified in Equation 15-2.**This value is a 2's complements number.** 0000000000 0 0000000001 1 ... 0111111111 511 1000000000 –512 ... 1111111111 –1 **Note:** Changes in this field take effect (and the PLL is relocked) only after the MPLL_RST bit in the CCTL register is set.

15.3.3.2 USB PLL Control Register (UPCTL)

This register contains parameters which determine the output frequency of the USB PLL (UPLL), which is given by F_{VCO} in Equation 15-2.

$$2 \times F_{ref} \times \frac{MF_I + \frac{MF_N}{MF_D}}{PD} = Fvco$$

UPLL Output Frequency ***Eqn. 15-2***

F_{ref}: Reference clock (input frequency) of UPLL.
MFx: Various multiplication factors, whose value is set using the UPCTL register (see bit descriptions below).
PD: Predivider factor, whose value is also set in the UPCTL register.

Figure 15-4 shows the UPCTL fields, which are described in Table 15-6.

NOTE

The MFN and MFD fields in UPCTL must satisfy the condition that the absolute value of MFN/MFD be smaller than 1.

Offset 0x0004 (UPCTL) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BRMO	0	PD						MFD							
W																
Reset	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOCK	0	MFI						MFN							
W																
Reset	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 15-4. USB PLL Control Register (UPCTL)

Table 15-6. UPCTL Field Descriptions

Name	Description
31 BRMO	Binary rate multiplier order bit. Determines if the binary rate multiplier (BRM) is first or second order. The first order BRM is used if a MF fractional part is between 1/10 and 9/10. In other cases, the second order BRM is used. The BRMO bit is cleared by a hardware reset. 1 BRM is second order. 0 BRM is first order.
30	Reserved
29–26 PD	**Predivider factor bits. These bits** define the predivider factor (PD) applied to the PLL input frequency, as specified in Equation 15-2. PD is an integer between 1and 16 (inclusive). PD is chosen to ensure that the resulting output frequency remains within the specified range.The change in this field will not take effect, and the PLL will be relocked by setting UPLL RST bit in CCTL register only. 0000 1 0001 2 ... 1111 16
25–16 MFD	**Multiplication factor (denominator part). Defines the denominator part of the BRM value for the MF.** See Equation 15-2. The change in this field will not take effect, and the PLL will be relocked by setting UPLL RST bit in CCTL register only. 000000000 1 000000001 2 … 1111111111 1024

Table 15-6. UPCTL Field Descriptions (continued)

Name	Description
15 LOCK	This bit shows if the PLL lock flag is set. 0 PLL lock flag is negated. 1 PLL lock flag is set.
14	Reserved
13–10 MFI	**Multiplication factor (integer part)**. Defines the integer part of the BRM value for the MF. See Equation 15-2. The MFI is encoded so that MFI < 5 results in MFI = 5. The change in this field will not take effect, and the PLL will be relocked by setting UPLL RST bit in CCTL register only. 0000–0101 5 0110 6 ... 1111 15
9–0 MFN	**Multiplication factor (numerator part)**. Defines the numerator of the BRM value for the MF. See Equation 15-2. The change in this field will not take effect, and the PLL will be relocked by setting UPLL RST bit in CCTL register only. **This value is a 2's complements number.** 0000000000 0 0000000001 1 ... 0111111111 511 1000000000 –512 ... 1111111111 –1

15.3.3.3 Clock Control Register (CCTL)

Offset 0x0008 (CCTL) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ARM CLK DIV		AHB CLK DIV		MPLL RST	UPLL RST	LP CTL		UPLL DIS	MPLL BYPASS	USB DIV					
Reset	0*	0*	0	1	0	0	0	0	0	0	0	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CG CTL	ARM SRC	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-5. Clock Control Register (CCTL)

Table 15-7. CCTL Field Descriptions

Field	Description
31–30 ARM CLK DIV	These bits control the ARM CLK DIV in the i.MX25 clock generation scheme (see Figure 15-28). 00 Divide by 1 01 Divide by 2 10 Divide by 3 11 Divide by 4 **Note:** The ARM CLK DIV default value depends on the ipp_clk_sel boot up value. If ipp_clk_sel = 0, then ARM CLK DIV = 3; if ipp_clk_sel = 1, then ARM CLK DIV = 1)
29–28 AHB CLK DIV	These bits control the AHB CLK DIV in the i.MX25 clock generation scheme (see Figure 15-28). 00 Divide by 1 01 Divide by 2 10 Divide by 3 11 Divide by 4
27 MPLL RST	This bit controls the MPLL restart. 0 Don't restart the MPLL. 1 Restart the MPLL, this bit will be cleared automatically.
26 UPLL RST	This bit controls the UPLL restart. 0 Don't restart the UPLL. 1 Restart the UPLL, this bit will be cleared automatically.
25–24 LP CTL	These bits define the low power mode to be entered after a wait for interrupt (WFI) is executed. 00 Run mode—no clocks are shut down. 01 Wait mode—all clocks except for ARM clock are active, except for those disabled by CGCR registers' settings. 10 Doze mode—all clocks except ARM Platform clock and ARM clock are active, except for those disabled by CGCR register settings. 11 Stop mode—all PLLs and all clocks are shut down.
23 UPLL DIS	This bit disables/enables the UPLL. 0 UPLL is enabled. 1 UPLL is disabled.
22 MPLL BYPASS	This bit controls whether to bypass MPLL and use the OSC24M clock as chip's source clock. 0 No bypass 1 Bypass
21–16 USB DIV	These bits control the divider for USB clock (the source clock is from UPLL). 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
15 CG CTRL	This bit controls the clock gating mode of the CCM's peripheral (PER) and USB clocks 0 All clock gating will be controlled by CG CTRL REGs. 1 All clock gating will be controlled by CG CTRL REGs and module clock enable signals.
14 ARM SRC	This bit controls the clock source used by the ARM Divider. 0 The clock source for the ARM Divider is 532 MHz 1 The clock source for the ARM Divider is 399 MHz
13–0	Reserved

15.3.3.4 Clock Gating Control Register 0 (CGCR0)

Offset 0x000C (CGCR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	AHB Clock Gating												
W																
Reset	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PER Clock Gating															
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 15-6. Clock Gating Control Register 0 (CGCR0)

Table 15-8. Clock Gating Control Register 0 Field Descriptions

Field	Description
31–29	Reserved
28–16 AHB Clock Gating	These bits control the clock gating of AHB clocks which are generated by CCM. Table 15-9 shows the correspondence between the individual bits and the clocks 0 Disable the clock output. 1 Enable the clock output.
15–0 PER Clock Gating	These bits control the clock gating of peripheral (PER) clocks which are generated by CCM. 0 Disable the clock output. 1 Enable the clock output.

Table 15-9. AHB Clock Gating

Bit	Clock
AHB Clock Gating[12]	hclk_usbotg
AHB Clock Gating[11]	hclk_slcdc
AHB Clock Gating[10]	hclk_sdma
AHB Clock Gating[9]	hclk_rtic
AHB Clock Gating[8]	hclk_lcdc
AHB Clock Gating[7]	hclk_fec
AHB Clock Gating[6]	hclk_esdhc2
AHB Clock Gating[5]	hclk_esdhc1
AHB Clock Gating[4]	hclk_esai
AHB Clock Gating[3]	hclk_emi

Table 15-9. AHB Clock Gating (continued)

Bit	Clock
AHB Clock Gating[2]	hclk_csi
AHB Clock Gating[1]	Reserved
AHB Clock Gating[0]	hclk_ata

Table 15-10. PER Clock Gating

Bit	Clock
PER Clock Gating[15]	ipg_per_uart
PER Clock Gating[14]	ipg_per_ssi2
PER Clock Gating[13]	ipg_per_ssi1
PER Clock Gating[12]	ipg_per_sim2
PER Clock Gating[11]	ipg_per_sim1
PER Clock Gating[10]	ipg_per_pwm
PER Clock Gating[9]	ipg_per_owire
PER Clock Gating[8]	ipg_per_nfc
PER Clock Gating[7]	ipg_per_lcdc
PER Clock Gating[6]	ipg_per_i2c
PER Clock Gating[5]	ipg_per_gpt
PER Clock Gating[4]	ipg_per_esdhc2
PER Clock Gating[3]	ipg_per_esdhc1
PER Clock Gating[2]	ipg_per_esai
PER Clock Gating[1]	ipg_per_epit
PER Clock Gating[0]	ipg_per_csi

15.3.3.5 Clock Gating Control Register 1 (CGCR1)

Offset 0x0010 (CGCR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IPG Clock Gating[31:16]															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IPG Clock Gating[15:0]															
Reset	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 15-7. Clock Gating Control Register 1 (CGCR1)

Table 15-11. Clock Gating Control Register 1 Field Descriptions

Field	Description
31–0 IPG Clock Gating[31:0]	These bits control the clock gating of IPG clocks generated by CCM. Table 15-13 shows the correspondence between clocks and individual bits. 0 Disable the clock output. 1 Enable the clock output.

15.3.3.6 Clock Gating Control Register 2 (CGCR2)

Offset 0x0014 (CGCR2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	IPG Clock Gating[51:48]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IPG Clock Gating[47:32]															
Reset	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0

Figure 15-8. Clock Gating Control Register 2 (CGCR2)

Table 15-12. Clock Gating Control Register 1 Field Descriptions

Field	Description
31–20	Reserved
19–0 IPG Clock Gating[51:32]	These bits control the clock gating of IPG clocks generated by CCM. Table 15-13 shows the correspondence between clocks and individual bits. 0 Disable the clock output. 1 Enable the clock output.

Table 15-13. IPG Clock Gating

Bit	Clock
IPG Clock Gating[51]	Reserved
IPG Clock Gating[50]	ipg_clk_uart5
IPG Clock Gating[49]	ipg_clk_uart4
IPG Clock Gating[48]	ipg_clk_uart3
IPG Clock Gating[47]	ipg_clk_uart2
IPG Clock Gating[46]	ipg_clk_uart1
IPG Clock Gating[45]	ipg_clk_tsc
IPG Clock Gating[44]	ipg_clk_ssi2
IPG Clock Gating[43]	ipg_clk_ssi1
IPG Clock Gating[42]	ipg_clk_spba
IPG Clock Gating[41]	ipg_clk_slcdc
IPG Clock Gating[40]	ipg_clk_sim2
IPG Clock Gating[39]	ipg_clk_sim1
IPG Clock Gating[38]	ipg_clk_sdma
IPG Clock Gating[37]	ipg_clk_scc
IPG Clock Gating[36]	Reserved
IPG Clock Gating[35]	ipg_clk_rngb
IPG Clock Gating[34]	ipg_clk_pwm4
IPG Clock Gating[33]	ipg_clk_pwm3
IPG Clock Gating[32]	ipg_clk_pwm2
IPG Clock Gating[31]	ipg_clk_pwm1
IPG Clock Gating[30]	Reserved
IPG Clock Gating[29]	LCDC_EN
IPG Clock Gating[28]	Reserved
IPG Clock Gating[27]	Reserved
IPG Clock Gating[26]	ipg_clk_iim

Table 15-13. IPG Clock Gating (continued)

Bit	Clock
IPG Clock Gating[25]	Reserved
IPG Clock Gating[24]	Reserved
IPG Clock Gating[23]	Reserved
IPG Clock Gating[22]	ipg_clk_gpt4
IPG Clock Gating[21]	ipg_clk_gpt3
IPG Clock Gating[20]	ipg_clk_gpt2
IPG Clock Gating[19]	ipg_clk_gpt1
IPG Clock Gating[18]	Reserved
IPG Clock Gating[17]	Reserved
IPG Clock Gating[16]	Reserved
IPG Clock Gating[15]	ipg_clk_fec
IPG Clock Gating[14]	ipg_clk_esdhc2
IPG Clock Gating[13]	ipg_clk_esdhc1
IPG Clock Gating[12]	Reserved
IPG Clock Gating[11]	ipg_clk_epit2
IPG Clock Gating[10]	ipg_clk_epit1
IPG Clock Gating[9]	ipg_clk_ect
IPG Clock Gating[8]	ipg_clk_dryice
IPG Clock Gating[7]	ipg_clk_cspi3
IPG Clock Gating[6]	ipg_clk_cspi2
IPG Clock Gating[5]	ipg_clk_cspi1
IPG Clock Gating[4]	ipg_clk_csi
IPG Clock Gating[3]	ipg_clk_can2
IPG Clock Gating[2]	ipg_clk_can1
IPG Clock Gating[1]	ipg_clk_ata
IPG Clock Gating[0]	Reserved

15.3.3.7 PER Clock Divider Register 0 (PCDR0)

This register, along with PCDR1–3 controls the frequency division for the different peripheral (PER) clocks. Table 15-15 shows the correspondence between PER clocks and modules.

Figure 15-9 shows the PCDR0 register's fields, which are described in Table 15-14.

Offset 0x0018 (PCDR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	PER3 DIV						0	0	PER2 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PER1 DIV						0	0	PER0 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 15-9. PER Clock Divider Register 0 (PCDR0)

Table 15-14. PCDR0 Field Descriptions

Field	Description
31–30	Reserved
29–24 PER3 DIV	These bits control the divider for the PER3 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
23–22	Reserved
21–16 PER2 DIV	These bits control the divider for the PER2 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
15–14	Reserved
13–8 PER1 DIV	These bits control the divider for the PER1 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

Table 15-14. PCDR0 Field Descriptions (continued)

Field	Description
7–6	Reserved
5–0 PER0 DIV	These bits control the divider for the PER0 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

Table 15-15. PER Clock Distribution

PER Clock	Module
PER Clock 15	UART
PER Clock 14	SSI2
PER Clock 13	SSI1
PER Clock 12	SIM2
PER Clock 11	SIM1
PER Clock 10	PWM
PER Clock 9	OWIRE
PER Clock 8	NFC
PER Clock 7	LCDC
PER Clock 6	I2C
PER Clock 5	GPT
PER Clock 4	eSDHC2
PER Clock 3	eSDHC1
PER Clock 2	ESAI
PER Clock 1	EPIT
PER Clock 0	CSI

15.3.3.8 PER Clock Divider Register 1 (PCDR1)

Offset 0x001C (PCDR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	PER7 DIV						0	0	PER6 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PER5 DIV						0	0	PER4 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 15-10. PER Clock Divider Register 1 (PCDR1)

Table 15-16. PCDR1 Field Descriptions

Field	Description
31–30	Reserved
29–24 PER7 DIV	These bits control the divider for the PER7 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
23–22	Reserved
21–16 PER6 DIV	These bits control the divider for the PER6 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
15–14	Reserved
13–8 PER5 DIV	These bits control the divider for the PER5 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

Table 15-16. PCDR1 Field Descriptions (continued)

Field	Description
7–6	Reserved
5–0 PER4 DIV	These bits control the divider for the PER4 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

15.3.3.9 PER Clock Divider Register 2 (PCDR2)

Offset 0x0020 (PCDR2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	PER11 DIV						0	0	PER10 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PER9 DIV						0	0	PER8 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 15-11. PER Clock Divider Register 2 (PCDR2)

Table 15-17. PCDR2 Field Descriptions

Field	Description
31–30	Reserved
29–24 PER11 DIV	These bits control the divider for the PER11 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
23–22	Reserved

Table 15-17. PCDR2 Field Descriptions (continued)

Field	Description
21–16 PER10 DIV	These bits control the divider for the PER10 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
15–14	Reserved
13–8 PER9 DIV	These bits control the divider for the PER9 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
7–6	Reserved
5–0 PER8 DIV	These bits control the divider for the PER8 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

15.3.3.10 PER Clock Divider Register 3 (PCDR3)

Offset 0x0024 (PCDR3) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	PER15 DIV						0	0	PER14 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	PER13 DIV						0	0	PER12 DIV					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Figure 15-12. PER Clock Divider Register 3 (PCDR3)

Table 15-18. PCDR3 Field Descriptions

Field	Description
31–30	Reserved
29–24 PER15 DIV	These bits control the divider for the PER15 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
23–22	Reserved
21–16 PER14 DIV	These bits control the divider for the PER14 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
15–14	Reserved
13–8 PER13 DIV	These bits control the divider for the PER13 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64
7–6	Reserved
5–0 PER12 DIV	These bits control the divider for the PER12 clock. The PER clock is the source, and the default frequency is133 MHz. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

15.3.3.11 CCM Status Register (RCSR)

Offset 0x0028 (RCSR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MEM CTRL		MEM TYPE		PAGE SIZE		BUS WIDTH		USB SRC		BT SRC		BT RES			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SOFT_RESET	NFC_16bit_SEL	0	CLK SEL	BOOT REG		NFC_4K	NFC_FMS	SPARE SIZE	BOOT INT	EEPROM CFG	MLC SEL	RESTS			
W																
Reset	0	0*	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0

Note: All reset values with "*" are depended on fuse value or GPIO value at boot up.

Figure 15-13. CCM Status Register (RCSR)

Table 15-19. RCSR Field Descriptions

Field	Description
31–30 MEM CTRL	These two bits show the MEM CTRL fuse/GPIO value (read only). 00 WEIM 01 NAND Flash 10 Reserved 11 Expansion Device (SD/MMC/MoviNAND/CE-ATA, I2C, SPI) See the MEM TYPE field settings for details.
29–28 MEM TYPE	This bit shows the MEM TYPE fuse/GPIO value (read only) If MEM CTRL = WEIM then 00 NOR 01 Reserved 10 OneNAND 11 Reserved If MEM CTRL = NAND Flash 00 3 address cycles 01 4 address cycles 10 5 address cycles 11 Reserved If MEM CTRL = Expansion Card Device 00 SD/MMC/MoviNAND/CE-ATA HDD 01 Reserved 10 Serial ROM using I2C 11 Serial ROM using SPI
27–26 PAGE SIZE	These two bits show the NAND Flash PAGE SIZE fuse/GPIO value (read only) If MEM CTRL = NAND Flash then 00 512 bytes 01 2 Kbytes 10 4 Kbytes 11 Reserved

Table 15-19. RCSR Field Descriptions (continued)

Field	Description
25–24 BUS WIDTH	These two bits show the BUS WIDTH fuse/GPIO value.(read only) MEM CTRL[1:0] = NAND Flash 00 8-bit bus 01 16-bit bus 10 Reserved 11 Reserved MEM CTRL[1:0] = WEIM (NOR) 00 16-bit address/data multiplexed interface 01 16-bit address/data unmultiplexed interface 10 Reserved 11 Reserved MEM CTRL[1:0] = Expansion Device (SPI) 00 2-Address word SPI device (16-bit) 01 3-Address word SPI device (24-bit) 10 Reserved 11 Reserved
23–22 USB SRC	These bits show BOOT USB SRC fuse/GPIO value. (read only) 00 USB OTG Internal PHY 01 USB OTG External ULPI PHY 10 Reserved 11 Reserved

21–20
BT SRC
	These bits show BOOT SRC fuse/GPIO value. (read only)
	If MEM CTRL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=00 (SD/MMC/MoviNAND/CE-ATA) then
	00 eSDHC1
	01 eSDHC2
	10 Reserved
	11 Reserved
	If MEM CTRL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=10 (Serial ROM using I2C), then
	00 I2C1
	01 I2C2
	10 I2C3
	11 Reserved
	If MEM CTRL[1:0]==11 (Expansion card device) && BT_MEM_TYPE[1:0]=11 (Serial ROM using SPI), then
	00 CSPI1
	01 CSPI2
	10 CSPI3
	11 Reserved
	Otherwise Reserved

19–16 BT RES	These bits show BOOT RES fuse/GPIO value. (read only) They are reserved for boot options
15 SOFT RESET	When “1” is Written to this bit, peripherals are reset. This bit will be clear automatically. 1 Peripherals are in the process of being reset. 0 Peripherals are not in the process of being reset.
14 NFC_16bit_SEL	This bit is used to configure the NAND Flash width. It is used after boot from NAND.It can be active while nf8_boot_b and nf16_boot_b are both high (negated). 0 8-bit width 1 16-bit width

Table 15-19. RCSR Field Descriptions (continued)

Field	Description
13	Reserved
12 CLK SEL	This bit shows value on CLK SEL pad. 0 CLK SEL is zero when it is latched by POR. 1 CLK SEL is one when it is latched by POR.
11–10 BOOT_REG	These two read-only bits show the boot mode from boot mode pin. 0+ Internal Boot 01 Function Test (including DTE mode) 10 External Boot 11 Boot Strap
9 NFC_4K	This bit is used to configure the NAND flash page size. It is defined by PAGE_SIZE during boot-up, and can be configured by software after boot-up. 0 Not 4-Kbyte page 1 4-Kbyte page
8 NFC_FMS	This bit is used to configure the NAND flash page size. It is defined by PAGE_SIZE during boot-up, and can be configured by software after boot-up. 0 Not 2-Kbyte page 1 2-Kbyte page
7 SPARE SIZE	This read-only bit shows the SPARE SIZE fuse/GPIO value. 0 128 bytes spare (Samsung: 4-bit ECC) 1 218 bytes spare (Micron, Toshiba: 8-bit ECC)
6 BOOT INT	This read-only bit shows the BOOT INT value. 0 External Boot 1 Internal Boot
5 EEPROM CFG	This read-only bit shows the EEPROM CFG fuse/GPIO value. It selects whether EEPROM device is used for load of configuration DCD data, prior to boot from other devices (not applicable when using EEPROM as boot device) 0 Use EEPROM DCD 1 Don’t use EEPROM DCD
4 MLC SEL	This read-only bit shows the MLC SEL fuse/GPIO value. 0 SLC NAND device 1 MLC NAND device
3–0 REST	Reset status bits. Shows what caused the most recent reset to the system.Otherwise, the last signal that is released is honored. 0000 POR reset 0001 Reset In reset. xx10 WDOG reset x1x0 SOFT RESET 1xx0 JTAG SW RESET

15.3.3.12 CCM Reset and Debug Register (CRDR)

Offset 0x002C (CRDR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BT UART SRC			BT LPB FREQ			0	0	0	0	0	0	0	0	0	0
W																
Reset	0*	0*	0*	0*	0*	0*	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: All reset values with "*" depend on fuse values or GPIO values at boot-up.

Figure 15-14. CCM Debug Register (CDCR)

Table 15-20. CDCR Field Descriptions

Name	Description
31–29 BT UART SRC	These bits show BT UART SRC fuse/GPIO value. (read only) 000 UART-1 001 UART-2 010 UART-3 011 UART-4 100 UART-5 OthersReserved
28–26 BT LPB FREQ	These bits show LPB ARM core frequency fuse/GPIO value. (read only) 000 133 MHz (default) 001 24 MHz 010 55.33 MHz 011 66 MHz 100 83 MHz 101 166 MHz 110 266 MHz 111 Normal Boot frequency.
25–0	Reserved

15.3.3.13 DPTC Comparator Value Registers (DCVR0–DCVR3)

These registers contain relevant DPTC lookup table values.

Offset 0x0030 (DCVR0)
0x0034 (DCVR1)
0x0038 (DCVR2)
0x003C (DCVR3)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ULV										LLV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LLV				ELV										0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-15. DPTC Comparator Value Register (DCVR0–DCVR3)

Table 15-21. DCVR0–DCVR3 Field Descriptions

Name	Description
31–22 ULV	Upper limit value. Upper performance limit of the reference circuit clock counter.
21–12 LLV	Lower limit value. Lower performance limit of the reference circuit clock counter.
11–2 ELV	Emergency limit value. Lower performance limit of the reference circuit clock counter. This serves as an “emergency” lower limit, which indicates a critical value.
1–0	Reserved

15.3.3.14 Load Tracking Register 0 (LTR0)

Offset 0x0040 (LTR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SIGD 15	SIGD 14	SIGD 13	0	UPTHR						DNTHR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SIGD 12	SIGD 11	SIGD 10	SIGD 9	SIGD 8	SIGD 7	SIGD 6	SIGD 5	SIGD 4	SIGD 3	SIGD 2	SIGD 1	SIGD 0	DIV3CK		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-16. Load Tracking Register 0 (LTR0)

Table 15-22. LTR0 Field Descriptions

Name	Description
31–30 SIGD*n*	SIGD15–14These two bits always use reset value which is logic 0. 1 Edge detection. 0 Level detection.
29 SIGD*n*	SIGD13 These bits define wheather the dvfs_w_sig[12:0] signals are detected using level of edge detection. 1 Edge detection. 0 Level detection.
28	Reserved
27–22 UPTHR	Upper threshold for load tracking.
21–16 DNTHR	Lower threshold for load tracking.
15–3 SIGD*n*	SIGD12–0 These bits define whether the dvfs_w_sig[12:0] signals are detected using level or edge detection. 1 Edge detection. 0 Level detection.
2–1 DIV3CK	Defines the division value of div_3_clk.
0	Reserved

15.3.3.15 Load Tracking Register 1 (LTR1)

Offset 0x0044 (LTR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	LT BRSH	LT BRSR	DNCNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DNCNT		UPCNT								PNCTHR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-17. Load Tracking Register 1 (LTR1)

Table 15-23. LTR1 Field Descriptions

Name	Description
31–24	Reserved
23 LTBRSH	Load tracking buffer shift 0 takes the original MSB of ld_add (ld_add[5:2]) 1 takes a shift of ld_add (ld_add[4:1])
22 LTBRSR	Load tracking buffer source 0 pre_ld_add 1 ld_add
21–14 DNCNT	These bits define the number of consecutive times the lower frequency threshold is undershot (that is, the effective frequency is lower than this threshold) in order to generate a dvfs_fdw signal. This signal causes a decrease of the system frequency. 00000001 2 00000010 3 ... The value 00000000 is not allowed.
13–6 UPCNT	These bits define the number of consecutive times the upper frequency threshold must be exceeded (threshold overcomes) in order to generate a dvfs_fup signal. This signal causes an increase of the system frequency. 00000001 2 00000010 3 ... The value 00000000 is not allowed.
5–0 PNCTHR	These bits define panic mode level threshold for load tracking.

15.3.3.16 Load Tracking Register 2 (LTR2)

Offset 0x0048 (LTR2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WSW15				WSW14				WSW13				WSW12			WSW11				WSW10
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSW10			WSW9			0	0	EMAC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-18. Load Tracking Register 2 (LTR2)

Table 15-24. LTR2 Field Descriptions

Name	Description
31–29 WSW15	These bits define the general purpose load tracking signals dvfs_w_sig[15:9]
28–26 WSW14	
25–23 WSW13	
22–20 WSW12	
19–17 WSW11	
16–14 WSW10	
13–11 WSW9	
10–9	Reserved
8–0 EMAC	These bits define the EMA configuration.

15.3.3.17 Load Tracking Register 3 (LTR3)

Offset 0x004C (LTR3) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WSW8				WSW7				WSW6				WSW5			WSW4				WSW3
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSW3				WSW2				WSW1			WSW0				0	0	0	0	0
W																				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-19. Load Tracking Register 3 (LTR3)

Table 15-25. LTR3 Field Descriptions

Name	Description
31–29, WSW8 28–26, WSW7 25–23, WSW6 22–20, WSW5 19–17, WSW4 16–14, WSW3 13–11, WSW2 10–8, WSW1 7–5, WSW0	These bit fields (each one in turn) define the weight of each of the general purpose load tracking signals (dvfs_w_sig[8:0]). The total CPU load as a result of the dvfs_w_sig signal is defined as a sum of each of the individual signal's dvfs_w_sig[n] value multiplied by its weight, as defined by the corresponding WSWn bit field.
4–0	Reserved

15.3.3.18 Load Tracking Buffer Register 0 (LTBR0)

Offset 0x0050 (LTBR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LTS7				LTS6				LTS5				LTS4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS3				LTS2				LTS1				LTS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-20. Load Tracking Buffer Register 0 (LTBR0)

Table 15-26. LTBR0 Field Descriptions

Name	Description
31–28, LTS7	These eight fields contain data from the last eight samples of load tracking.
27–24, LTS6	
23–20, LTS5	
19–16, LTS4	
15–12, LTS3	
11–8, LTS2	
7–4, LTS1	
3–0, LTS0	

15.3.3.19 Load Tracking Buffer Register 1 (LTBR1)

Offset 0x0054 (LTBR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LTS15				LTS14				LTS13				LTS12			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LTS11				LTS10				LTS9				LTS8			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-21. Load Tracking Buffer Register 1 (LTBR1)

Table 15-27. LTBR1 Field Descriptions

Name	Description
31–28 LTS15	These eight fields contain data from the first eight samples of load tracking.
27–24 LTS14	
23–20 LTS13	
19–16 LTS12	
15–12 LTS11	
11–8 LTS10	
7–4 LTS9	
3–0 LTS8	

15.3.3.20 Power Management Control Register 0 (PMCR0)

Offset 0x0058 (PMCR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			DVSUP					DVFS_UPD_FINISH	DVFEV	DVFIS	LBMI	LBFL	LBCF		PTVIS	DVFS_START
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS VAIM	FSVAI		DPVCR	DPVV	WFIM	DRCE3	DRCE2	DRCE1	DRCE0	SCR	DVFEN	PTVAIM	PTVAI		DPTEN
W																
Reset	1	0	0	1	1	0	0	0	0	0	1	0	1	0	0	0

Figure 15-22. Power Management Control Register 0 (PMCR0)

Table 15-28. PMCR0 Field Descriptions

Name	Description
31–30	Reserved
29–28 DVSUP	These two bits define the voltage level. 00 DVS0=0 DVS1=0 (highest frequency/voltage level) 01 DVS0=0 DVS1=1 10 DVS0=1 DVS1=0 11 DVS0=1 DVS1=1 (lowest frequency/voltage level)
27–25	Reserved
24 DVFS_UPD_FINISH	This bit shows DVFS UPDATE finish by software 0 Not finished 1 Finished
23 DVFEV	Always give a DVFS event. 0 Do not give an event always. 1 Always give event.
22 DVFIS	DVFS Interrupt select. These bits define destination of DVFS interrupts. 1 Core interrupt will be generated for DVFS events. 0 SDMA interrupt will be generated for DVFS events.
21 LBMI	Load buffer full mask interrupt. This bit masks the generation of this interrupt. 1 Load buffer full interrupt is masked. 0 Load buffer full interrupt is enabled.
20 LBFL	Load buffer full status bit. This bit indicates that log buffer registers are full. An interrupt will be generated if the LBMI bit is cleared. Software can only write 0 to this bit. 1 Load buffer is full. 0 Load buffer is not full.

Table 15-28. PMCR0 Field Descriptions (continued)

Name	Description
19–18 LBCF	DVFS load buffer programmable size 00 Load buffer size is 4 01 Load buffer size is 8 10 Load buffer size is 12 11 Load buffer size is 16
17 PTVIS	DPTC Interrupt select. These bits define destination of DPTC interrupts. 1 Core interrupts will be generated for DPTC events 0 SDMA interrupts will be generated for DPTC events.
16 DVFS_START	DVFS update start bit, configured by software.Asserted when update is in progress. 0 DVFS update is not active 1 DVFS update is in progress
15 FSVAIM	DVFS Frequency adjustment interrupt mask. This bit masks the DVFS frequency adjustment interrupt. FSVAI status bits will be still asserted in relevant cases. 1 interrupt is masked. 0 interrupt is enabled.
14–13 FSVAI[1:0]	FSVAI DVFS Frequency adjustment interrupt. These status bits indicate that the system frequency should be changed. 00 No interrupt 01 Frequency should be increased. Low priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVLV = 00 (highest frequency). 10 Frequency should be decreased. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVLV = 11 (lowest frequency). 11 Frequency should be increased immediately. High priority interrupt. Interrupt is asserted, if FSVAIM=0. Interrupt is masked if DVLV = 00 (highest frequency).
12 DPVCR	DPTC voltage change request 0 Disabled 1 Enabled
11 DPVV	DPTC voltage valid edge detect. Can be updated by SW. If written by SW, it is possible to assert it only after DPVCR is asserted. 0 Voltage is not valid 1 Received a voltage valid acknowledge
10 WFIM	DVFS Wait for Interrupt mask bit 0 Wait for interrupt not masked 1 Wait for interrupt masked.
9 DRCE3	DPTC reference circuit3 enable bit. This bit defines if reference circuit3 is enabled during DPTC operation. 1 DPTC reference circuit3 enabled. 0 DPTC reference circuit3 disabled.
8 DRCE2	DPTC reference circuit2 enable bit. This bit defines if reference circuit2 is enabled during DPTC operation. 1 DPTC reference circuit2 enabled. 0 DPTC reference circuit2 disabled.
7 DRCE1	DPTC reference circuit1 enable bit. This bit defines if reference circuit1 is enabled during DPTC operation. 1 DPTC reference circuit1 enabled. 0 DPTC reference circuit1 disabled.
6 DRCE0	DPTC reference circuit0 enable bits. This bit defines if reference circuit0 is enabled during DPTC operation. 1 DPTC reference circuit0 enabled. 0 DPTC reference circuit0 disabled.

Table 15-28. PMCR0 Field Descriptions (continued)

Name	Description
5 SCR	DPTC counting range. This bit sets how many times the system clock may increment and the reference circuits remain active (and their output signals are counted). 1 512 system clock count 0 256 system clock count
4 DVFEN	DVFS enable. This bit enables the DVFS block. Between disable and enable there has to be at least 3 cycles of div_3_clk. 1 DVFS enabled. 0 DVFS disabled.
3 PTVAIM	DPTC Voltage adjustment interrupt mask. This bit masks the DPTC voltage adjustment interrupt. PTVAI status bits will be still asserted in relevant case. 1 interrupt is masked. 0 interrupt is enabled.
2–1 PTVAI[1:0]	DPTC Voltage adjustment interrupt. These status bits indicate that the supply voltage should be changed. 00 No interrupt 01 Voltage should be decreased. Interrupt is asserted, if PTVAIM=0 10 Voltage should be increased. Low priority interrupt. Interrupt is asserted, if PTVAIM=0 11 Voltage should be increased immediately. High priority interrupt. Interrupt is asserted, if PTVAIM=0
0 DPTEN	DPTC enable. This bit enables the DPTC block and starts the reference circuit clock counting and compares this to look-up table values. 1 DPTC enabled 0 DPTC disabled

15.3.3.21 Power Management Control Register 1 (PMCR1)

Offset 0x005C (PMCR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	CPEN_EMI	CPFA_EMI	CPSPA_EMI				WBCN							
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	CPEN	CPSPA				0	0	CPFA	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-23. Power Management Control Register 1 (PMCR1)

Table 15-29. PMCR1 Field Descriptions

Name	Description
31–30	Reserved
29 CPEN_EMI	EMI well bias enable. 0 Disabled 1 Enabled
28 CPFA_EMI	EMI well bias frequency adjust. 0 The free-running oscillator is in reduced mode.(frequency is approximately 0.75 times the full frequency (default) 1 The free-running oscillator is set to full mode.
27–24 CPSPA_EMI	This field controls the EMI back-bias level of n-wells. (CSPA_EM[1:0]) and p-wells (CSPA_EM[3:2]). CSPA_EM[1:0]: 00 Increased back bias applied to the n-wells (appropriate for system voltage of 1.0 V or less). 01 Moderate back bias applied to the n-wells (appropriate for system voltage of 1.1 V or less) 10 Decreased back bias applied to the n-wells (appropriate for system voltage of 1.2 V or less). 11 Minimum back bias applied to the n-wells (appropriate for system voltage of 1.2 V or less). CSPA_EM[3:2]: 00 Increased back bias applied to the p-wells (appropriate for system voltage of 1.0 V or less). 01 Moderate back bias applied to the p-wells.(appropriate for system voltage of 1.1 V or less). 10 Decreased back bias applied to the p-wells (appropriate for system voltage of 1.2 V or less). 11 Minimum back bias applied to the p-wells (appropriate for system voltage of 1.2 V or less).
23–16 WBCN	Well bias counter. This field specifies the waiting period after exit from well bias mode, measured in CKIH (24 MHz) clock cycles. After this period, the signals emi_wbcp_fbd_b and mcu_wbcp_fbd_b are negated (preventing re-entry to well bias mode) and run mode is resumed. According to well bias specifications, the waiting period should be at least 3 μs, which implies that the WBCN value should be at least 72 (0b100_1000).
15–14	Reserved
13 CPEN	This bit enables/disables the ARM platform well bias. 0 ARM platform well bias is disabled. 1 ARM platform well bias is enabled.
12–9 CPSPA	This field controls the ARM platform back-bias level of n-wells (CSPA[1:0]) and the p-wells (CSPA[3:2]). CSPA[1:0]: 00 Increased back bias applied to the n-wells (appropriate for system voltage of 1.0 V or less). 01 Moderate back bias applied to the n-wells (appropriate for system voltage of 1.1 V or less) 10 Decreased back bias applied to the n-wells (appropriate for system voltage of 1.2 V or less). 11 Minimum back bias applied to the n-wells (appropriate for system voltage of 1.2 V or less). CSPA[3:2]: 00 Increased back bias applied to the p-wells (appropriate for system voltage of 1.0 V or less). 01 Moderate back bias applied to the p-wells.(appropriate for system voltage of 1.1 V or less). 10 Decreased back bias applied to the p-wells (appropriate for system voltage of 1.2 V or less). 11 Minimum back bias applied to the p-wells (appropriate for system voltage of 1.2 V or less).
8–7	Reserved
6 CPFA	ARM platform well bias frequency adjustment. 0 The free-running oscillator is in reduced mode.(frequency is approximately 0.75 times the full frequency (default) 1 The free-running oscillator is set to full mode.
5–0	Reserved

15.3.3.22 Power Management Control Register 2 (PMCR2)

Offset 0x0060 (PMCR2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ARM MEM DWN	VSTBY	OSC24M_DOWN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ARM MEMON CNT								ARM CLKON CNT				0	0	DVFS_REQ	DVFS_ACK
W																
Reset	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0

Figure 15-24. Power Management Control Register 2 (PMCR2)

Table 15-30. PCMR2 Field Descriptions

31–19	Reserved
18 ARM MEM DWN	This bit is used to control negation of arm_mem_on in doze and stop mode. 0 arm_mem_on and arm_mem_pwrdwn will be kept invalid in doze and stop mode. 1 arm_mem_on will be negated and arm_mem_pwrdwn will be asserted in doze and stop mode.
17 VSTBY	This bit controls whether or not the core voltage is lowered to 1.0 V in stop mode. 0 Core voltage is not lowered in stop mode. 1 Core voltage is lowered to 1.0 V in stop mode.
16 OSC24M_DOWN	This bit controls the OSC24M power supply during stop mode. 0 OSC24M will not be powered off in stop mode. 1 OSC24M will be powered off in stop mode.
15–8 ARM MEMON CNT	This field controls the delay before arm_mem_on is asserted. 0 1 OSC24M clock cycle delay 1 2 OSC24M clock cycles delay. ... 255 256 OSC24M clock cycles delay.
7–4 ARM MEMON CNT	This field controls the delay between arm_mem_on and arm_clock_on. 0 1 OSC24M clock cycle delay 1 2 OSC24M clock cycles delay. ... 15 256 OSC24M clock cycles delay.
3–2	Reserved

Table 15-30. PCMR2 Field Descriptions (continued)

1 DVFS_REQ	This bit controls the dvfs_req signal (frequency change request) to EMI M3IF. This bit is automatically cleared by the dvfs_ack signal. 0 no dvfs_req signal 1 dvfs_req signal asserted
0 DVFS_ACK	This read-only bit shows whether or not EMI has closed the access for SDRAM. 0 No ack received 1 Ack received

15.3.3.23 Miscellaneous Control Register (MCR)

Offset 0x0064 (MCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	USB XTAL MUX	CLKO EN	CLKO DIV						CLKO SEL				ESAI CLK MUX	SSI2 CLK MUX	SSI1 CLK MUX	USB CLK MUX
Reset	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PER CLK MUX															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-25. Miscellaneous Control Register (MCR)

Table 15-31. MCR Field Descriptions

31 USB XTAL MUX	This bit selects output of UXTAL1_CKIN. 0 External 24 MHz clock 1 24 MHz source is divided from the UPLL 240 MHz. (fixed divided by 10)
30 CLKO EN	This bit is used to enable/disable the CLKO debug function. When set to 1, the selected clock will be routed to CLKO pad. 0 Disable 1 Enable
29–24 CLKO DIV	These bits control the divider for CLKO, the source clock selected by CLKO SEL. See Figure 15-28. 000000 Divide by 1 000001 Divide by 2 … 111110 Divide by 63 111111 Divide by 64

Table 15-31. MCR Field Descriptions

23–20 CLKO SEL	These bits select which clock is to be reflected on the clock output CKO 000032 kHz clock. 0001 Input 133 MHz clock for PLL reference 0010 Ungated arm_clk. 0011 Ungated ahb_clk. 0100 Ungated ipg_clk. 0101 per_clk_0 clock source. 0110 usb_clk clock source. 0111 Gated arm_clk. 1000 Gated ahb_clk. 1001 Gated ipg_clk. 1010 per_clk_0. 1011 per_clk_2. 1100 per_clk_13. 1101 per_clk_14. 1110 usb_clk. 1111 Reserved
19 ESAI CLK MUX	This bit controls which divider will be selected for per_clk_2. 0 Normal divider, which is same as other per_clk except esai, ssi1, ssi2. 1 A dedicated divider which divides the 240 MHz UPLL clock output to 24.61 MHz.
18 SSI2 CLK MUX	This bit controls which divider will be selected for per_clk_14. 0 Normal divider, which is same as other per_clk except ESAI, SSI1, SSI2. 1 A dedicated divider which divides the 240 MHz UPLL clock output to 24.61 MHz.
17 SSI1 CLK MUX	This bit controls which divider will be selected for per_clk_13. 0 Normal divider, which is same as other per_clk except ESAI, SSI1, SSI2. 1 A dedicated divider which divides the 240 MHz UPLL clock output to 24.61 MHz.
16 USB CLK MUX	This bit determines the clock source for usb_clk. 0 UPLL output 1 HCLK
15–0 PER CLK MUX	These bits determine the clock source for each per_clk. That is, bit *n* determines the source for per_clk_*n*. 0 HCLK 1 UPLL output

15.3.3.24 Low Power Interrupt Mask Registers (LPIMR0)

Offset 0x0068 (LPIMR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPIM[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPIM[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-26. Low Power Interrupt Mask Register 0 (LPIMR0)

Table 15-32. LPIMR0 Field Descriptions

31–0 LPIM	Low power interrupt mask. Each bit enables/disables the corresponding interrupt to CCM in low-power mode. 0 Interrupt is enabled. 1 Interrupt is masked in low-power mode.

15.3.3.25 Low Power Interrupt Mask Registers (LPIMR1)

Offset 0x006C (LPIMR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPIM[63:48]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPIM[47:32]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-27. Low Power Interrupt Mask Register 1 (LPIMR1)

Table 15-33. LPIMR1 Field Descriptions

31–0 LPIM[63:32]	Low power interrupt mask. Each bit enables/disables the corresponding interrupt to CCM in low-power mode. 0 Interrupt is enabled. 1 Interrupt is masked in low-power mode.

The LPIMR0 and LPIMR1control 64 interrupt requests to wake-up CCM from low-power mode, and resume clocks. Table 15-34 lists the wake-up interrupt sources.

Table 15-34. Wake-up Interrupt Sources

IRQ	Interrupt Source	Interrupt Description
0	CSPI3	—
1	GPT4	—
2	OWIRE	—
3	I2C1	—
4	I2C2	—
5	UART4	—
6	RTIC	—
7	ESAI	—
8	ESDHC2	—
9	ESDHC1	—
10	I2C3	—
11	SSI2	—
12	SSI1	—
13	CSPI2	—
14	CSPI1	—
15	ATA	—
16	GPIO3	Combined Interrupts - 1 Bit Int Or Of 32
17	CSI	—
18	UART3	—
19	IIM	—
20	SIM1	—
21	SIM2	—
22	RNGB	—
23	GPIO4	Combined Interrupts - 1 Bit Int Or Of 32
24	KPP	Keypad Interrupt
25	Dry-Ice	Consolidated RTC Interrupt
26	PWM	—
27	EPIT2	—
28	EPIT1	—
29	GPT3	—
30	POWER FAIL	power fail interrupt from external PAD

Table 15-34. Wake-up Interrupt Sources (continued)

IRQ	Interrupt Source	Interrupt Description
31	CCM	—
32	UART2	—
33	NANDFC	Consolidated Nand Flash Controller Interrupt
34	SDMA	“AND” of all 32 interrupts from all the channels
35	USB-HTG	—
36	PWM2	—
37	USB-OTG	—
38	SLCDC	—
39	LCDC	—
40	UART5	—
41	PWM3	—
42	PWM4	—
43	CAN1	—
44	CAN2	—
45	UART1	—
46	TSC	—
47	Reserved	—
48	ECT	—
49	SCC SCM	SCM interrupt
50	SCC SMN	SMN interrupt
51	GPIO2	Combined Interrupts - 1 Bit Int Or Of 32
52	GPIO1	Combined Interrupts - 1 Bit Int Or Of 32
53	GPT2	—
54	GPT1	—
55	WDOG	—
56	Dry-Ice	—
57	FEC	—
58	EXT_INT5	external interrupt for Pwr Management using GPIO-1[5]
59	EXT_INT4	external interrupt for Temp using GPIO-1[6]
60	EXT_INT3	external interrupt for Sensor using GPIO-1[0]

Table 15-34. Wake-up Interrupt Sources (continued)

IRQ	Interrupt Source	Interrupt Description
61	EXT_INT2	external interrupt for Sensor using GPIO-1[1]
62	EXT_INT1	external interrupt for Watch-dog using GPIO-1[2]
63	EXT_INT0	external interrupt for TV using GPIO-1[3]

15.4 Functional Description

15.4.1 Clock Control and Gating

Figure 15-28 shows the overall clock generation scheme.

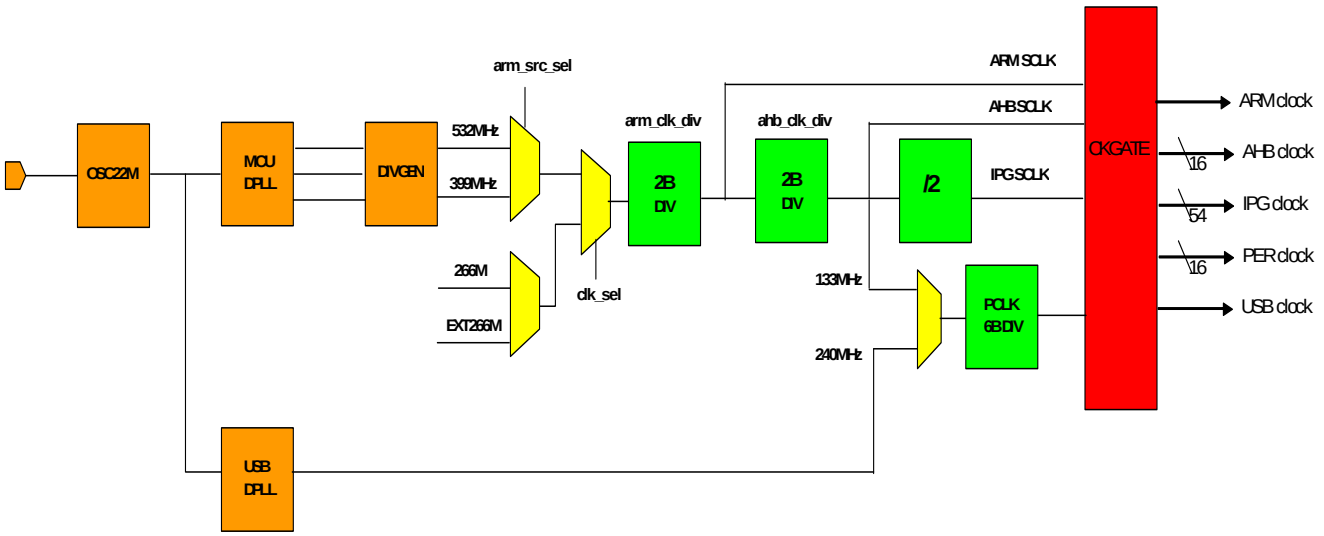


Figure 15-28. i.MX25 Clock Generation Scheme 1

[1] This figure does not show scan multiplexer and clock gating. For the scan multiplexer refer to Section 15.4.1.4, "Scan Mode Clock Multiplexers"

15.4.1.1 Clock Sources

There are two clock sources:

- 24 MHz external crystal
- 32 kHz clock input

15.4.1.1.1 24 MHz External Crystal

An external 24 MHz crystal is required for the functioning of the internal oscillator. The 24 MHz signal is used as a core PLL and peripheral PLL reference clock, and also serves as the reference frequency for USBPHY.

The internal 24 MHz oscillator can be bypassed through the CLKMOD pins, so that the external 24 MHz clk can enter the device directly.

Figure 15-28 describes these functions.

15.4.1.1.2 32 kHz Clock Input

The 32 kHz clock is input from the DryIce module. It is synched with the IPG clock, then sent to each module.

15.4.1.2 Core PLL Domain Clock Generation

There are two DPLLs in the system that generate all high-frequency clocks for each module.

The MPLL is configured by the MPCTL register. Figure 15-28 shows the clock generation scheme. The two ports generate edge-aligned clocks. The configuration of the clock switch is defined in the register map. The ARM and AHB clock frequency ranges are 133–532 MHz and 66.5–133MHz, respectively. Internal logic guarantees that the ARM/AHB clock switches concurrently at the ARM/AHB aligned rising edge.

15.4.1.3 USB PLL (UPLL) Domain Clock Generation

The UPLL is configured by the UPCTL register, and has the same design as the MPLL. Its dpgdck port generates a 240 MHz clock for USB/SSI/MSHC/UART/LCDC/CSI baud-rate clock generation. The baud-rate clocks of these modules need not be synched with the AHB/IPG clock, since most are used to generate data to communicate externally.

15.4.1.4 Scan Mode Clock Multiplexers

Scan mode clock multiplexers enable external sources of clocks (from I/O pins) to bypass the clock generation scheme. This is useful in scan mode and reduces the number of sources to different subdomains in transition scan mode. Figure 15-29 illustrates the scan mode clock multiplexer structure.

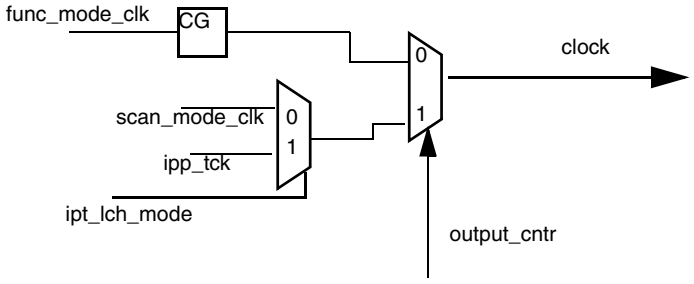


Figure 15-29. Scan Mode Clock Multiplexer Illustration

15.4.2 Reset Module

This section includes the following:

- A description of the reset module
- The reset negation sequence from POR
- A description of other reset events

15.4.2.1 Functional Description of the Reset Module

The reset module controls or distributes all of the system reset signals used by ti.MX25. Figure 15-30 gives a simplified block diagram of the reset module. The reset signals shown in Figure 15-30 are described in Table 15-35, and the timing parameters are described in Table 15-36.

Table 15-35. Reset Module Signals

Signal Name	Description
ipp_por_b	Resets OSC24M, which is input from external source
por_reset_b	Resets DPLLs, fusebox and IIM
emi_reset_b	Resets EMI module
scc_reset_b	Resets SCC and security modules
hreset_b	Resets ARM platform
per_reset	Resets peripheral modules

Table 15-36. Reset Module Timing Parameters

Timing Parameter	Description
t_0	Time for the power-up sequence of all the supplies. After t_0, OSC24M starts to work.
t_1	t_1 depends on boot mode: • If boot mode is PROD mode (0b01) then t_1 is 8 32-kHz cycles • If boot mode is not PROD mode, then t_1 is 256 32-kHz cycles After t_1, IIM, fusebox, and the DPLLs start to work
t_2	4 32-kHz cycles+ 1 HCLK cycle. After t_2, EMI starts to work.
t_3	• For NAND boot, t_3 is 32 32-kHz cycles + 1 HCLK cycle • Otherwise t_3 is 4 cycles 32-kHz + 1 HCLK cycle After t_3, the security modules and SCC start to work.
t_4	• 1 HCLK cycle for hreset_b • 1 ipg_clk cycle for per_reset_b After t_4, all other modules start to work
t_5	Out of reset

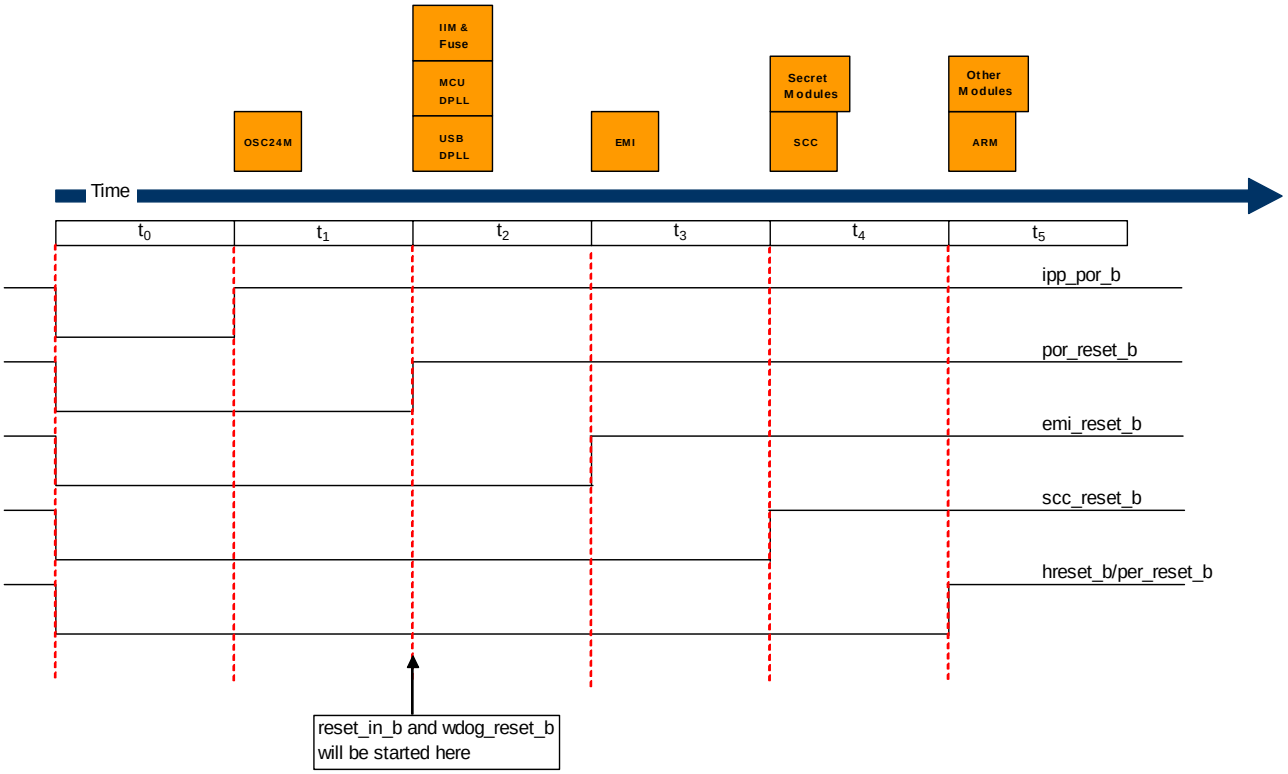


Figure 15-30. Reset Module Clock Diagram

15.4.2.2 Other Reset Events

Other reset events include the following:

- Watchdog resets
- reset_in_b source
- JTAG software reset

These resets are described in the following subsections.

15.4.2.2.1 Watchdog Resets

There are two different watchdog reset events: time out event or software reset. For more information about these events, refer to the watchdog specification.

A watchdog module reset, wdog_reset_b causes a reset of the chip, so that the CCM thereby generates a reset pulse. When the core comes out of reset it can check the “reset source” bits in the CCM module.

15.4.2.2.2 reset_in_b Source

reset_in_b can trigger the EMI, peripheral and core resets, but does not reset the PLL, fusebox, or CCM.

15.4.2.2.3 JTAG Software Reset

The device can be reset using software from a JTAG module, either by a signal or by a general-purpose bit.

15.4.3 Power Management

CCM supports several power management techniques that reduce active and static power consumption:

- **Clock gating** reduces the active power if the module is not active. CCM has low power mode power gating to gate the source clocks, and has register control the clock gating of each module.
- **Dynamic voltage frequency scaling** (DVFS) reduces active power consumption by scaling voltage and frequency accordingly to required MIPs.
- **Dynamic process temperature compensation** (DPTC) reduces active power consumption by adjusting supply voltage according to specific process cases, the manner in which the chip was fabricated, and the ambient temperature.
- **Active well bias** (AWB) reduces static power consumption by applying back bias on transistors. AWB can be applied on ARM9P. ARM9P is not functional when AWB is applied.

15.4.3.1 Power Domains

The i.MX25 is partitioned into several power domains, but all digital logics are included in one common power domain. Digital core logic can run at 133 MHz, 266 MHz, or 399 MHz while the voltage is above 1.1 V (worst case scenario).

15.4.3.2 Power Modes

The i.MX25 supports a versatile definition of power modes, including power and clock domains status and applied power techniques. The power modes described in the following sections were defined taking into account static and dynamic power consumption of blocks in different operational modes, the power consumption of clock sources, and the time required to exit low power modes.

15.4.3.2.1 Run Mode

This is the normal/functional operating mode of ARM9. The ARM clock frequency can vary between fmax (532 MHz) and fmin (133 MHz), and the voltage between Vmax (1.65 V) and Vmin (1.10 V).

15.4.3.2.2 Wait Mode

In wait mode, the processor clock is gated, but ARM9P MAX and all peripherals clocks are available. This mode is entered by the core executing a STANDBY FOR INTERRUPT command. The arm_clk_off output of ARM Platform is asserted and the clock to the processor arm_clk is gated by the CCM. The processor exits the wait mode and enters the run mode when a negation of arm_clk_off signal is detected. Clocks for specific modules can be gated off in wait mode by programming CGR registers.

Note some modules must trigger handshake processes in order to close the clock: this includes SDMA and EMI. These processes are handled automatically by the CCM and the respective modules.

The wait mode entry procedure is as follows:

1. The arm_clk_off signal is asserted from the ARM9P core by executing the instruction.
2. If the SDMA clock is configured to be gated in wait mode, then the CCM sends out the stby_req signal to the corresponding modules.
3. If the EMI module is configured to be gated in wait mode, then the CCM sends out the sd_lpmd to EMI.
4. Once all ack signals have been received from the modules listed above, then the 4-bit IPG clock counter is started.
5. If the FlexCAN module is configured to remain open in wait mode, then the CCM sends the doze_request signal to the FlexCAN module. Otherwise if the FlexCAN module is configured to be gated in wait mode, then the CCM sends the stop_request signal to the FlexCAN module.
6. After the ack from the FlexCAN module is received and the 4-bit counter counts 9 IPG cycles, then the lpm_sm signal is in WAIT_MODE.

Wait mode is exited by negating the arm_clk_off signal. The system then immediately returns to RUN mode.

15.4.3.2.3 Doze Mode

In doze mode the processor and MAX clocks are gated. The clock source is still available, and peripherals that do not require MAX and core functionality can be active.

Clocks for specific modules can be gated off in doze mode by programming the CGR register. Note some modules (such as SDMA, IPU, and EMI) must trigger a handshake process while trying to close the clock: this process is handled automatically by the modules and the CCM.

Doze mode is entered by programming the LP CTL field in the CCTL register to 0b10, then issuing a standby for interrupt command (which asserts the arm_clk_off signal). The doze mode entry procedure is as follows:

1. CCM sends out the stby_req to modules that are configured as clock-gated in doze mode.
2. CCM sends out the max_halt request.
3. Once stby ack from IPU/SDMA and max halt ack are received, then CCM sends out the emi_lpm_sd req if EMI is configured as clock gated in doze mode.
4. Once all ack signals (SDMA/MAX/EMI) are received, then if FlexCAN is configured to keep clock in doze mode the CCM will send out the can_doze request: otherwise if FlexCAN is configured to gate clock in doze mode, then CCM will send out the can_stop request.
5. Once the ack signal is received from the FlexCAN module and the 4-bit counter finishes counting nine IPG cycles, then lpm_sm changes into doze mode, and crm_int_holdoff is negated.
6. If the CPEN bit is set, then CCM asserts the mcu_wb_en, mcu_fbd_b, and mem_pwrdn signals to the ARM9P core and negates the mem_on signal to the ARM9P core.

The doze mode exit process proceeds as follows:

1. If the CPEN bit is not set to 1, the system returns immediately to RUN mode.

2. If the CPEN bit is set to 1, then the mcu_mem_pwrdn and mcu_wb_en signals are immediately cleared. After the wb_counter_finished signal is asserted, then mcu_mem_on is asserted, mcu_wbcp_fbd_b is negated, and the system returns to run mode.

15.4.3.2.4 Stop Mode

Stop mode stops the PLL and all clocks which are generated from PLL. If the VSTBY bit is set to 1, then CCM informs the external PMIC to lower the core supply to 1.0 V after all PLLs are closed. If the OSC24M_DOWN bit is set to 1, then the CCM powers down OSC24M to reduce the system power to minimum.

Stop mode is entered by setting the LP CTL field in the CCTL register to 0b11, then issuing a standby for interrupt command. The stop mode entry procedure is as follows:

1. SDMA, MAX, CAN, and EMI standby requests are sent out to those modules which are not clock gated before stop mode.
2. After all ack signals are received CCM waits eight IPG clk cycles, then requests CKIL to switch from IPG synched to not synched. sm_mode changes to stop, and the ARM clock is gated.
3. If the wb_en signal is asserted, then CCM asserts mcu_wb_en (emi_wb_en), mcu_fbd_b (emi_fbd_b), mem_pwrdn for ARM9P and negates mem_on for ARM9P. This is the same as doze mode.
4. If the PLL lock flag is negated, then crm_int_holdoff is negated to allow interrupt.
5. If the VSTBY bit is set to 1, then CCM sends out the vstby_pmic to PMIC through pad. This requests the external PMIC to lower down core logic to 1.0 V for state retention.
6. If the OSC24M_PWRDN bit is set to 1, then CCM sends out the osc24m_pwr signal to power down OSC24M (which also stops the internal 32 kHz clock, since it is divided from OSC24M). This special mode (“static mode”) can be considered as a submode of stop mode.

State retention mode is exited by any internal or external interrupt.

The stop mode exit procedure is as follows:

1. The arm_clk_off signal is negated, and the FSM changes to the exit process.
2. If the system in state retention mode, then vstby_pmic is negated to request the external PMIC to restore the voltage supply from 1.0 V to normal voltage.
3. If OSC24M is powered down, then the CCM negates the osc_pwr signal, causing the OSC24M to power up. After power-up the CCM must wait for the 32 kHz clock counter to finish to ensure that the OSC24M output is stable.
4. If system was configured in voltage reduce mode, the CCM waits for the 32 kHz clock counter to finish or ipp_pmic_rdy signal ack from PMIC.
5. After the OSC24M is stable and PMIC is ready, the MPLL starts to relock.
6. If the wb_en bit is set to 1, then mcu_mem_pwrdn and mcu_wb_en (emi_wb_en) are immediately negated and wb_en counter begins.
7. After the PLL is ready and the ARM clock begins, the chips returns to run mode.

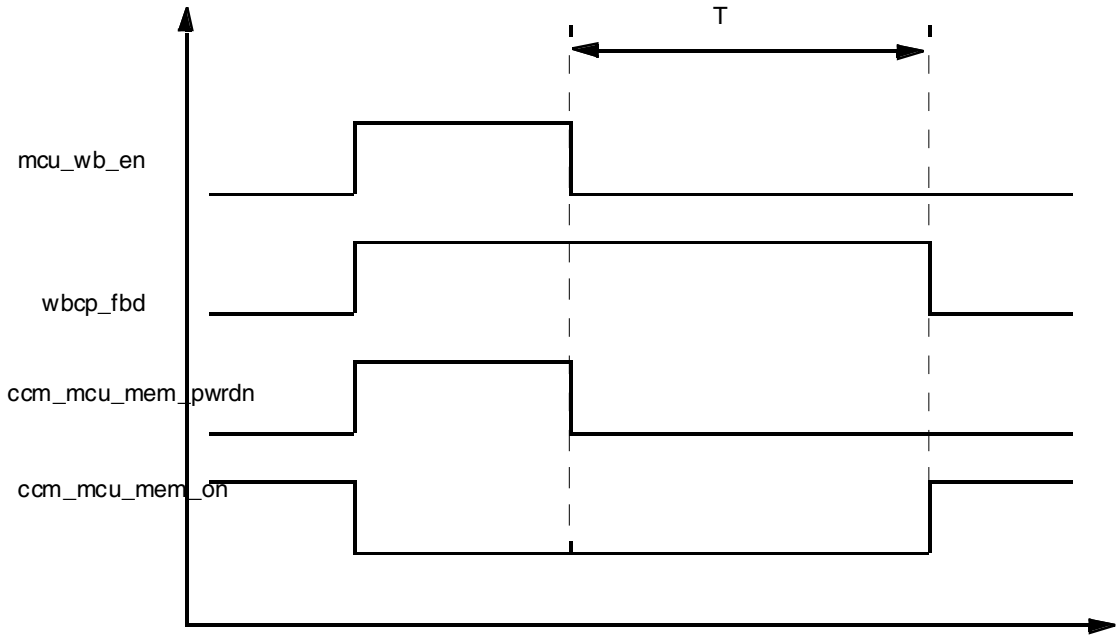


Figure 15-31. Well Bias Activation/Deactivation and ARM Core Power Gating for Kilobit Memories.

15.4.3.3 DVFS Support

The CCM allows simple software dynamic voltage frequency scaling. The frequency of the core clock domain and the voltage of the chip can be changed on the fly while all modules (including the core) continue their normal operation. The voltage of the chip can be changed by software through CSPI/I2C port between the i.MX25 and the external PMIC. The frequency of the core clock domain can be changed by configuring CCTL register.

The system bus frequency (AHB, IP) can also be changed according to core clock. The AHB frequency is configured by AHB_DIV, and the IP frequency is fixed at half of the AHB frequency.

The i.MX25 DVFS scheme requires additional software configuration. Figure 15-32 shows the DVFS software routine flow in i.MX25, after the triggering of a DVFS interrupt or DVFS DMA event.

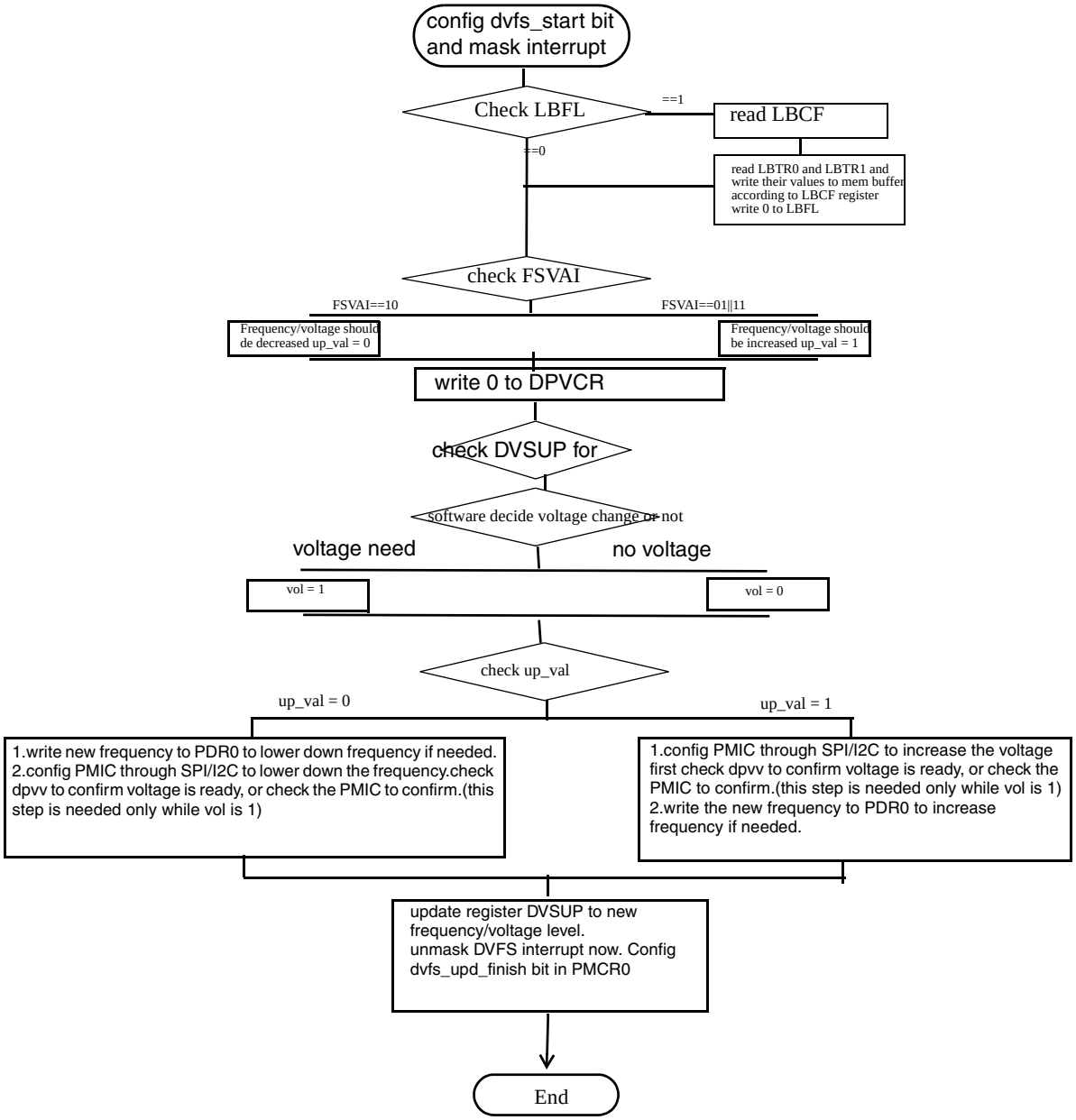


Figure 15-32. DVFS Frequency/Voltage Change Process

The DVFS load-tracking block allows hardware tracking on the core load and the generation of an interrupt when a frequency change is requested.

15.4.3.4 Dynamic Process and Temperature Compensation (DPTC) Support

The DPTC module is a power management module. The purpose of the DPTC module is to detect the minimum operation voltage for the IC, taking into account the worst-case values for processing activity and ambient temperature for a given frequency. It inputs predefined values for process speed performance measurement, and generates an interrupt if the value of the supply voltage must be updated.

15.4.3.4.1 Synchronization Between DVFS and DPTC

In order to avoid a situation of DPTC trying to update events during voltage update done by DVFS a synchronization scheme has been created.

The DPTC machine can work in case the DPTC is enabled and the voltage is valid (DPVV = 1) and there is no voltage change request (DPVCR = 1).

15.4.3.5 Well Bias Support

Well bias is implemented with an on-chip charge pump. When well bias is activated, the voltage of wells is pumped to optimal values to allow minimum leakage. Parameters of the charge pump can be controlled in software by setting configuration bits in the PMCR1 register. The device has both ARM9P and EMI well bias.

15.4.3.5.1 Well Bias Activation

EMI well bias is similar to ARM well bias, but it works in stop mode only.

In the ARM domain, well bias can be activated in doze or state retention modes by asserting the mcu_wb_en signal and mcu_wbcp_fbd_b. The ARM platform contains memory blocks with standard threshold voltage transistors, which have relatively high power leakage. Well bias eliminates this leakage by power gating, so that there is no need for data saving or restoration. The interface to memory is power gated by asserting the crm_mcu_mem_pwrdn signal and negating the crm_mcu_mem_on signal.

15.4.3.5.2 Well Bias Deactivation

When ARM negates the arm_clk_off signal, the CCM negates the mcu_wb_en signal. The clock is provided to ARM9P after the well bias counter reaches the threshold defined by the WBCN field, or after any other event that lasts longer than 10 ns. The ARM9P memories interface power supply is restored by negation of the crm_mcu_mem_pwrdn signal, together with negation of mcu_wb_en and assertion of the crm_mcu_mem_on and mcu_wbcp_fbd_b signals upon clock restoration.

EMI well bias also deactivates automatically.

15.4.3.6 State Retention Voltage Support

In STOP modes, the supply voltage of the i.MX25, can be reduced to a State Retention value, which allows a reduction of chip leakage current while embedded memory state is dormant. Cores and modules using embedded memory are not functional in this mode. Voltage will be reduced by PMIC after assertion of the lpm_sm_vstby_pmic signal, which in turn asserts the VSTBY I/O pin. Voltage will be restored to previous value upon negation of this signal. Valid voltage is indicated when the PMIC ready is asserted.

Chapter 16
ARM9 Platform Clock Control Module (CLKCTL)

16.1 Introduction

The CLKCTL module is used to enable or disable various peripherals on the platform. CLKCTL's registers interface with the AIPS module. The module also contains logic to determine when the ARM9's clock can be turned off. Figure 16-1 shows a general block diagram for the CLKCTL.

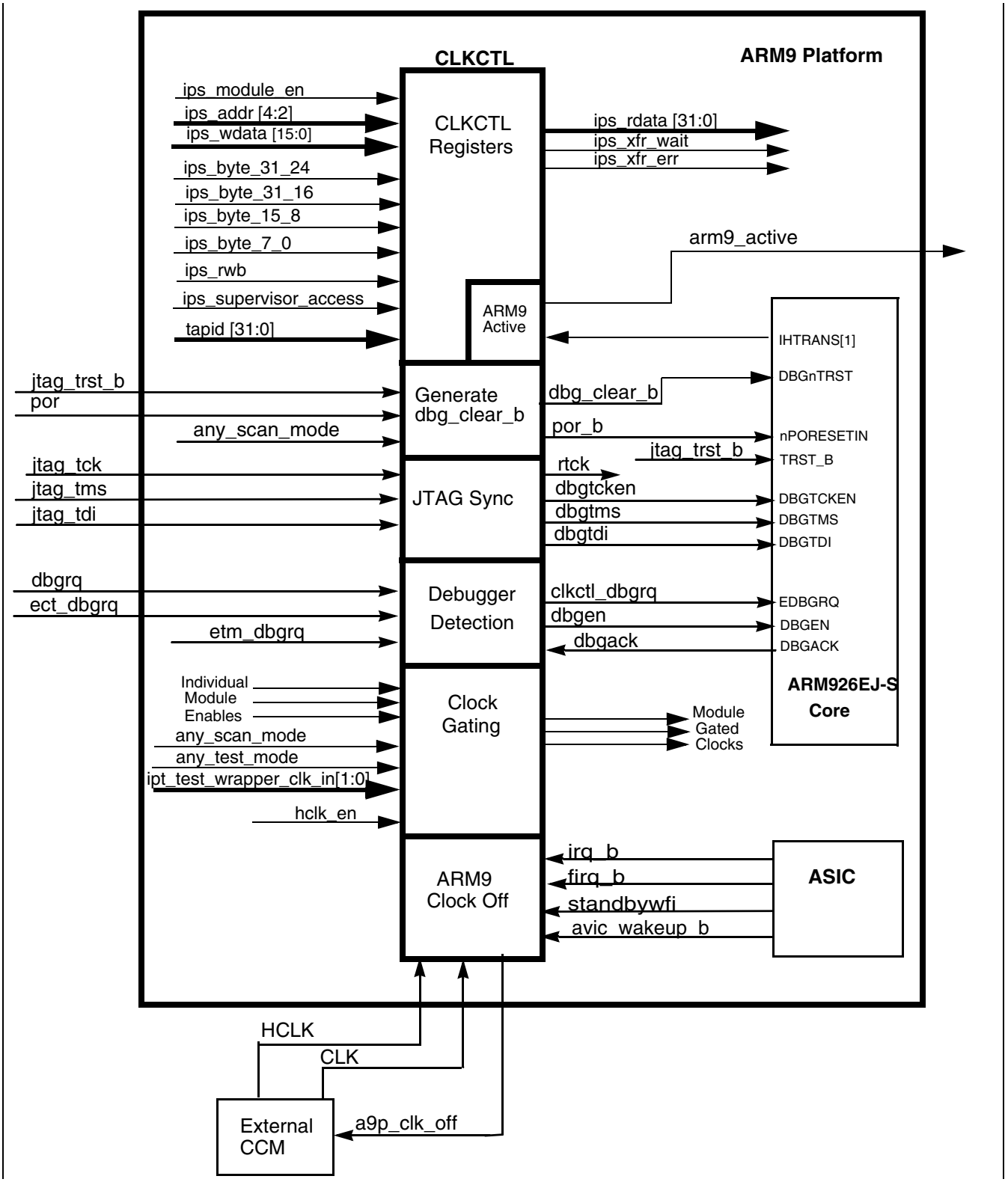


Figure 16-1. CLKCTL Block Diagram

16.2 Clock Gating

Table 16-1 shows the hardware clock gating control, based on the gated clocks' scan enable and module enable control signals.

Table 16-1. Hardware Clock Gating Control

Scan Enable	Module Enable	Gated Hclk
0	0	OFF
0	1	ON
1	0	ON
1	1	ON

Table 16-2 shows gated clocks and their corresponding control signals.

Table 16-2. Gated Clocks and Their Control Signals

Gated Clock	Module Enable	Scan Enable	Clock
etm_clk	dbgen	any_test_mode	clk_always
etb_clk	etb_clken	any_test_mode	clk_always
etb_hclk	etb_hclken	any_scan_mode	hclk
aape_hclk	aape_hclk_en	any_scan_mode	hclk
aape_reg_hclk	aape_module_en	any_scan_mode	hclk
aipsa_hclk	aipsa_hclk_en || aipsa_active	any_scan_mode	hclk
aipsb_hclk	aipsb_hclk_en || aipsb_active	any_scan_mode	hclk
asic_hclk	asic_hclk_en	any_scan_mode	hclk
asic_reg_hclk	asic_hclk_reg_en	any_scan_mode	hclk
max_reg_hclk	max_module_en	any_scan_mode	hclk
romc_hclk	romc_hclk_en	any_scan_mode	hclk
rompatch_hclk	rompatch_hclk_en	any_scan_mode	hclk
rompatch_reg_hclk	rompatch_hclk_reg_en	any_scan_mode	hclk
mbist_hclk	any_test_mode	any_test_mode	hclk
mbist_clk	any_test_mode	any_test_mode	clk
wrap_hclk	any_scan_mode	any_scan_mode	ipt_test_wrapper_clk_in[1]
wrap_clk	any_scan_mode	any_scan_mode	ipt_test_wrapper_clk_in[0]
tcu_hclk	any_scan_mode	any_scan_mode	hclk
tcu_clk	any_scan_mode	any_scan_mode	clk

In addition to the CLKCTL clock gating, each gated clock in Table 16-2 is controlled by a hardware enable. Each hardware enable signal may be ORed from several enable source signals (some from outside CLKCTL and some from within CLKCTL), and any one of the source signals can enable the gated clock.

16.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers. For the base address of a particular module instantiation, see the system memory map.

Table 16-3 is the module memory map.

Table 16-3. Module Memory Map

Base Address Offset (Register Abbreviation)	Register	Access[1]	Reset Value	Section/Page
0x0000 (GP_CTRL)	General-purpose control register	SR/SW	0x0000_0000	16.3.2.1/16-5
0x0004 (GP_SER)	Set register	SW	0x0000_0000	16.3.2.2/16-6
0x0008 (GP_CER)	Clear register	SW	0x0000_0000	16.3.2.3/16-7
0x0010 (TAPID)	Tap ID register	SR	0x0000_0000	16.3.2.4/16-7

[1] SR and SW denote supervisor read and supervisor write, respectively.

16.3.1 Register Summary

Table 16-4 is the register summary table.

Table 16-4. Module Register Summary

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (GP_CTRL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	GP_CTRL[11:0]											
	W																
0x0004 (GP_SER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W					write 1 to set adjacent bit in GP_CTRL											
0x0008 (GP_CER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W					write 1 to clear adjacent bit in GP_CTRL											

Table 16-4. Module Register Summary (continued)

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010 (TAPID)	R	Tapid[31:16]															
	W																
	R	Tapid[15:0]															
	W																

16.3.2 Register Descriptions

The CLKCTL registers are all 32-bit and can only be accessed in supervisor mode. Reads in user mode will return all zeros, and writes will cause an abort.

Only word-sized accesses are allowed. Half-word and byte accesses cause an abort.

16.3.2.1 General-Purpose Control Register (GP_CTRL)

Offset 0x0000 (GP_CTRL) Access: Supervisor read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	GP_CTRL [11]	0	0	0	GP_CTRL [7]	GP_CTRL [6]	0	GP_CTRL [4]	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-2. General Purpose Control Register

Table 16-5. General-Purpose Control Register Field Descriptions

Field	Description
31–12	Reserved.
11 GP_CTRL[11]	This register bit drives the signal which enables the slave port sharing widget (SPSW) in the MAX. 0 Slave port sharing disabled 1 Slave port sharing enabled
10–8	Reserved.
7 GP_CTRL[7]	When a9p_clk_off (ARM platform output) is asserted, the CCM starts the process to shut down the ARM platform clock. If the ARM platform is shut down, no debug activity is available—so by default a9p_clk_off is not allowed to be asserted if there is debug activity. Setting GP_CTL[7] to 1 overrides this restriction and allows assertion of a9p_clk_off when debug activity is present. 0 a9p_clk_off signal assertion is not allowed during debug activity (default). 1 a9p_clk_off signal assertion is allowed during debug activity.

Table 16-5. General-Purpose Control Register Field Descriptions (continued)

6 GP_CTRL[6]	Enables/disables clocks to the embedded trace buffer (ETB) 0 ETB clocks disabled 1 ETB clocks enabled
5	Reserved.
4 GP_CTRL[4]	Enables/disables the peripheral bus time-out monitors. 0 time-out monitors disabled 1 time-out monitors enabled
3–0	Reserved.

WARNING

When slave port sharing is enabled (GP_CNTRL[11]), running locked accesses to MAX Slave Port 1 and/or MAX Slave Port 0 can cause system deadlock. If you wish to run locked accesses to this memory region do not enable slave port sharing.

16.3.2.2 Set Control Register (GP_SER)

Offset 0x0004 (GP_SER) Access: Supervisor write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W					write 1 to set adjacent bit in GP_CTRL											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-3. Set Control Register

Table 16-6. Set Control Register Field Descriptions

Field	Description
31–12	Reserved.
11–0 GP_SER[11:0]	To set a bit in the GP_CTRL register, write 1 to the corresponding bit in this field.

16.3.2.3 Clear Control Register (GP_CER)

Offset 0x0008 (GP_CER) Access: Supervisor write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W					write 1 to clear corresponding bit in GP_CTRL											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-4. Clear Control Register

Table 16-7. Clear Control Register Field Descriptions

Field	Description
31–12	Reserved.
11–0 GP_CER[11:0]	To clear a bit in the GP_CTRL register, write 1 to the corresponding bit in this field.

16.3.2.4 Tap ID Register (TAPID)

Offset 0x0010 (TAPID) Access: Supervisor read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAPID[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TAPID[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 16-5. TAPID Register

Table 16-8. Tap ID Register Field Descriptions

Field	Description
31–0 TAPID	Tap ID. This is the JTAG ID register, described in the ARM9 platform overview chapter.

Chapter 17
CMOS Sensor Interface (CSI)

This chapter presents the CMOS Sensor Interface (CSI) architecture, operation principles, and programming model. The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only Bayer and statistics data, while smart sensors support CCIR656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

The capabilities of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.
- Support for CCIR656 video interface as well as traditional sensor interface.
- 8-bit data port for YCC, YUV, or RGB data input.
- 8-bit/10-bit/16-bit data port for Bayer data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit/pixel data format to 32-bit receive FIFO packing.
- 128 × 32 FIFO to store received image pixel data.
- Receive FIFO overrun protection mechanism.
- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support 2D DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support double buffering two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full, FIFO overrun, DMA transfer done, CCIR error and AHB bus response error.
- Configurable master clock frequency output to sensor.
- Statistic data generation for Auto Exposure (AE) and Auto White Balance (AWB) control of the camera (only for Bayer data and 8-bit/pixel format).

17.1 CSI Architecture

Figure 17-1 shows the block diagram of the CMOS Sensor Interface.

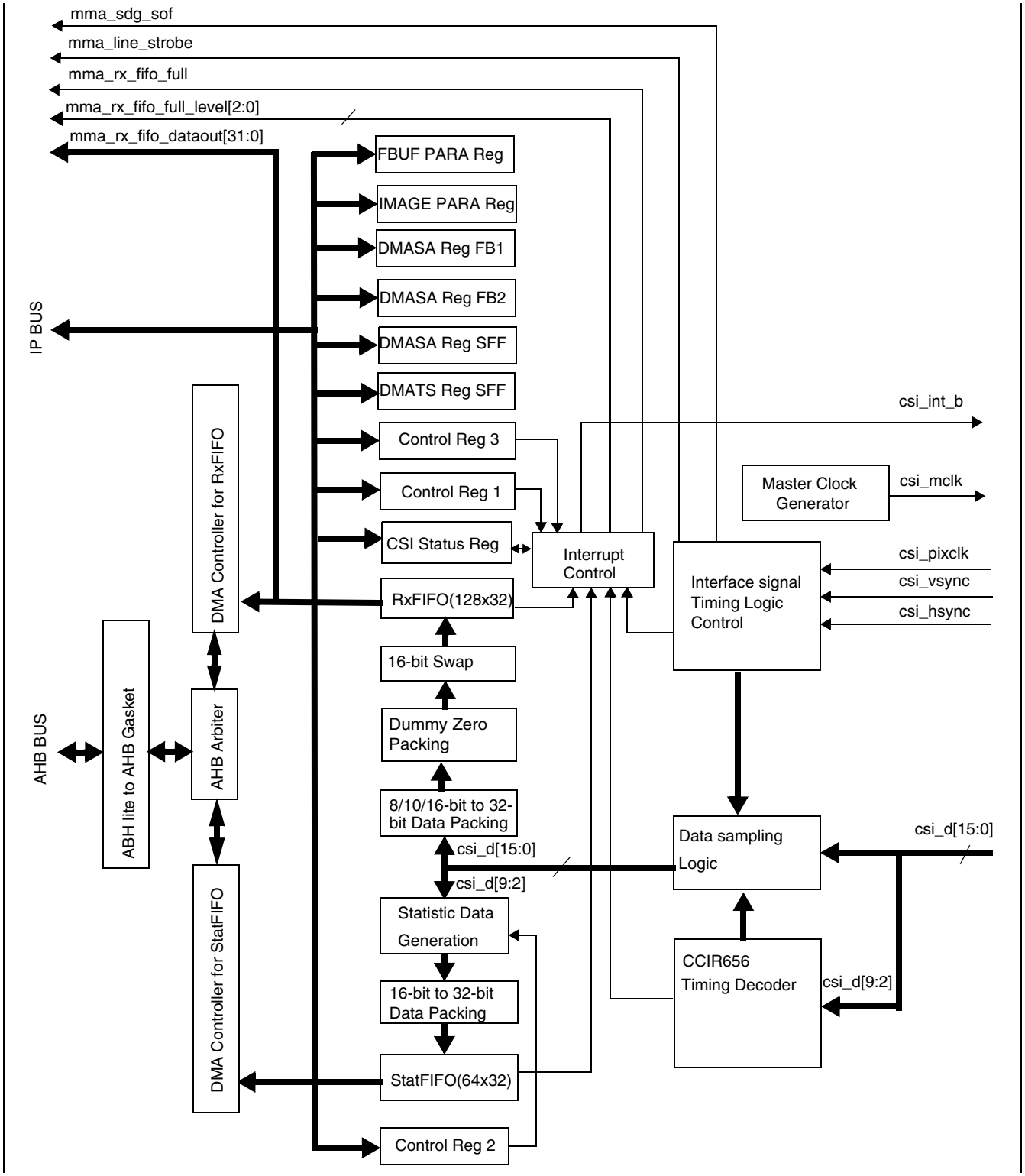


Figure 17-1. CSI Block Diagram

17.2 CSI Interface Signal Description

Table 17-1 provides a listing of the input and output signals between the CSI module and an external CMOS sensor.

Table 17-1. Signals Between CSI and Sensor

CSI Signals	Direction	Description
CSI_VSYNC	Input	Vertical Sync (Start Of Frame)
CSI_HSYNC	Input	Horizontal Sync (Blank Signal)
CSI_D[15:0]	Input	16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)
CSI_MCLK	Output	Sensor Master Clock
CSI_PIXCLK	Input	Pixel Clock

17.3 Principles of Operation

This section describes the modes of operation of the sensor interface.

The CSI is designed to support generic sensor interface timing as well as CCIR656 video interface timing. Traditional CMOS sensors typically use VSYNC, HSYNC, and PIXCLK signals to output Bayer or YUV data. Smart CMOS sensors, that come with on-chip imaging processing, usually support video mode transfer. They use an embedded timing codec to replace the SOF and BLANK signal. The timing codec is defined by the CCIR656 standard.

The CSI can support to connect with the sensor as below, one 8-bit sensor or one 10-bit sensor or one 16-bit sensor. To connect with one 8-bit sensor, the data interfaces should be connected to DATA[9:2]. To connect with one 10-bit sensor, the data interfaces should be connected to DATA[9:0]. To connect with one 16-bit sensor, the data interfaces should be connected to DATA[15:0].

17.3.1 Data Transfer With The Embedded DMA Controllers

The CSI has two embedded DMA controllers, one for the receive FIFO and the other for the statistic FIFO. The CSI supports 2D DMA transfer from the receive FIFO to the frame buffers in the external memory and linear DMA transfer from the statistic FIFO.

To transfer data from the RxFIFO to the external memory, the user needs to set the start address in the frame buffer registers for where the data is to be stored. Additionally the image size parameters need to be set in the CSI registers for the DMA controller to determine how much data needs to be transferred to trigger a DMA complete interrupt after each frame. Users can have two hardware controlled frame buffers in the external memory. Each one stores a single frame image coming from the sensor. Initially the embedded DMA controller writes the frame to buffer1 first and then the second frame to buffer2. These two frame buffers are written in a circular fashion. When an entire frame is transferred, that triggers a DMA complete interrupt and the start address pointer for the other buffer is loaded into the DMA engine for the next frame. The start addresses should be word aligned and should be set in the CSIDMASA-FB1 and CSIDMASA-FB2 registers. In the CSIFBUF_PARA register, user should set the stride of the frame buffer to show how many words to skip before starting to write the next row of the image. In the CSIIMAG_PARA register, user should set the width and height of the image coming from the sensor. This

parameter is essential for the DMA engine to determine how much data needs to be transferred for a complete frame. The RxFF_LEVEL and DMA_REQ_EN_RFF bits in the CSICR3 registers also need to be set before the data transfer starts, to enable when DMA transfers occur.

When the number of the words of data in the RxFIFO reaches the water-mark trigger level, a DMA request is sent from the RxFIFO to the embedded DMA controller and a single burst of data is read out from the RxFIFO and written through the AHB bus into the external frame buffers. No additional bursts of data are transferred until the RxFIFO reaches the water-mark trigger level again. The DMA burst type for the transfer can be INCR4, INCR8 and INCR16 by setting the DMA_BURST_TYPE_RFF bits in the CSICR2 register. After all data of a complete frame is transferred, the DMA_TSF_DONE_FB1 or DMA_TSF_DONE_FB2 bit is set in the CSISR register and the interrupt is triggered if the corresponding enable bit is set in the CSICR1 register.

The DMA_REFLASH_RFF bit in CSICR3 is used to activate or re-start the embedded DMA controller. Setting the reflash bit resets the DMA engine by reloading the start address of CSIDMASA_FB1 and reloading the image size parameters from the CSIIMAG_PARA register. If the image parameters from the sensor ever change these registers should be updated accordingly and the DMA_REFLASH_RFF bit should be set to re-initialize the DMA engine.

The RxFIFO has an overrun protection mechanism in case the RxFIFO is overrun during data transfer. Overrun occurs if the RxFIFO is full and more data needs to be received during the data transfer. If this occurs the RxFIFO is over-written continuously and all 128 words data in the RxFIFO before overrun occurred, are discarded and the corresponding 128 words in the external memory space for the frame buffer keep the previous values.

Similarly to transfer data from the statistic FIFO to the external memory, user need to set the start address of the external memory for where the transferred data is to be stored and the total transfer size. The start address and the transfer sizes are all word aligned and should be set in the CSIDMASA-STATFIFO and CSIDMATS-STATFIFO registers. The STATFF_LEVEL and DMA_REQ_EN_SFF bits in the CSICR3 registers also need to be set before the data transfer starts. When the number of words of data in the STATFIFO reaches the water-mark trigger level, a DMA request is sent from the STATFIFO to the embedded DMA controller and a single burst of data is read out from the STATFIFO and written through AHB bus into the external memory. The burst type for the transfer can be INCR4, INCR8 and INCR16 by setting the DMA_BURST_TYPE_SFF bits in the CSICR2 register. After all expected data (defined by the total transfer sizes) are transferred, the DMA_TSF_DONE_SFF bit is set in the CSISR register and an interrupt is triggered if the SFF_DMA_DONE_INTEN is enabled in the CSICR1 register. The DMA_REFLASH_SFF bit in CSICR3 can be used to activate or re-start the embedded DMA controller.

17.3.2 Gated Clock Mode

VSYNC, HSYNC, and PIXCLK signals are used in gated clock mode.

A frame starts with a rising edge on VSYNC, then HSYNC goes to HIGH and holds for the entire line. The Pixel clock is valid as long as HSYNC is HIGH. Data is latched at the rising edge of the valid pixel clocks. HSYNC goes to LOW at the end of line. Pixel clocks then become invalid and CSI stops receiving data from the stream. For the next line the HSYNC timing repeats. For the next frame the VSYNC timing repeats.

17.3.3 Non-Gated Clock Mode

In non-gated clock mode, only the VSYNC and PIXCLK signals are used; the HSYNC signal is ignored. Figure 17-2 is the timing diagram for non-gated clock mode.

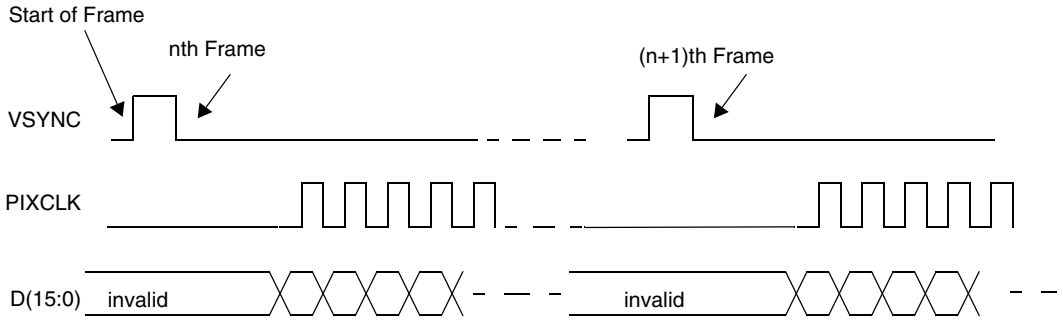


Figure 17-2. Non-Gated Clock Mode Timing Diagram

The overall timing of non-gated mode is the same as the gated-clock mode, except for the HSYNC signal. HSYNC signal is ignored by the CSI. All incoming pixel clocks are valid and cause data to be latched into RxFIFO. The PIXCLK signal is inactive (states low) until valid data is ready to be transmitted over the bus.

Figure 17-2 shows the timing using a typical sensor, other sensors may have slightly different timing from that shown. The CSI should be programed to support rising/falling-edge triggered VSYNC; active-high/low HSYNC; and rising/falling-edge triggered PIXCLK.

17.3.4 CCIR656 Interlace Mode

In CCIR656 mode, only the PIXCLK and DATA[9:2] signals are used. The start of frame and blank signals are replaced by a timing codec which is embedded in the data stream. Each active line starts with a SAV code and ends with a EAV code. In some cases, digital blanking is inserted in between EAV and SAV code. The CSI decodes and filters out the timing-coding from the data stream, thus recovering VSYNC and HSYNC signals for internal use, such as statistical block control. Data is forwarded to the data receive and packing block in a *sequential* manner without re-ordering—that is, field 1 followed by field 2. The fields must be re-ordered in software to get back the original image.

Change Of Field interrupt (COF) is triggered upon every field change. The interrupt service routine reads the status register to check for the current field.

According to the CCIR656 specification, the image must be in 625/50 PAL or 525/60 NTSC format. In addition, the image is interlaced into odd and even fields, with vertical and horizontal blank data being filled into certain lines. Data must be in YCC422 format, each pixel contains 2 bytes, either Y + Cr or Y + Cb. These requirements are set for TV systems. The CSI module supports PAL and NTSC format only. Figure 17-3 shows the frame structure in PAL system, showing vertical blanking and horizontal blanking.

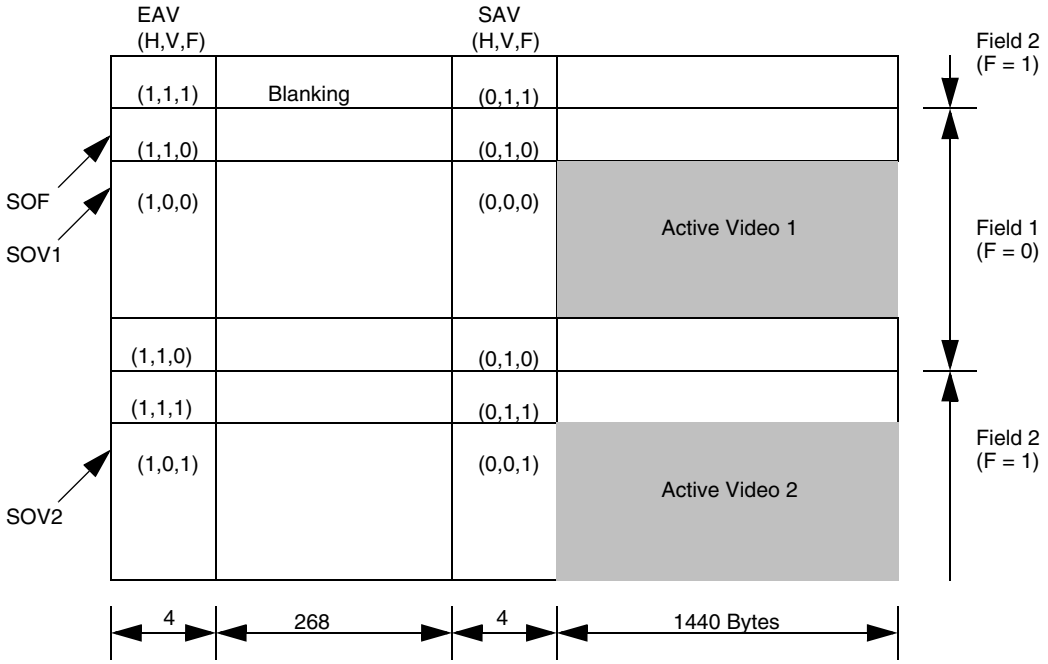


Figure 17-3. CCIR656 Interlace Mode (PAL)

Figure 17-4 shows the general timing for a single line, showing SAV and EAV.

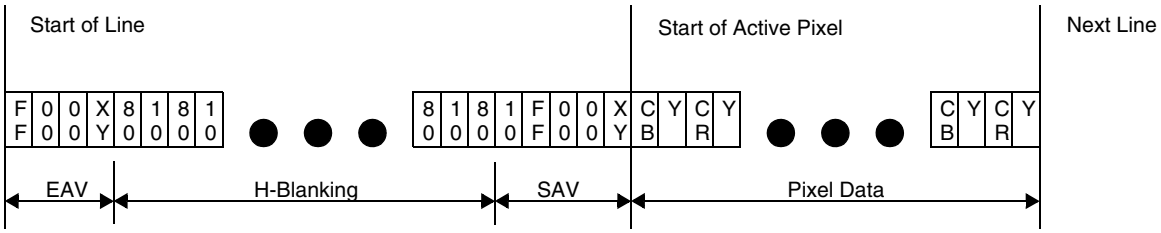


Figure 17-4. CCIR656 General Line Timing

The coding tables recommended by the CCIR656 specification are shown in Table 17-2, Table 17-3 and Table 17-4. It is used in the CCIR656 mode to decode the video stream. An interrupt is generated for SOF, which is decoded from the embedded timing codec.

Table 17-2. Coding for SAV and EAV

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P3
2	1	0	0	P2

Table 17-2. Coding for SAV and EAV (continued)

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
1	1	0	0	P1
0	1	0	0	P0

Table 17-3. Coding for Protection Bits

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Table 17-4. Representations by F-Bit

F-Bit	Representations
0	ODD FIELD (FIELD 1)
1	EVEN FIELD (FIELD 2)

17.3.5 CCIR656 Progressive Mode

For a CMOS camera system of VGA or CIF resolution, strict adherence to the interlace requirements stated in the CIR standard is not required. The image is considered to have only 1 active field which is scanned in a progressive manner. This active field is regarded as field 1 and the F-bit in the timing codec is ignored by the decoder. Most sensors support CCIR timing in this mode (progressive) by default.

Figure 17-5 shows the typical flow of progressive mode.

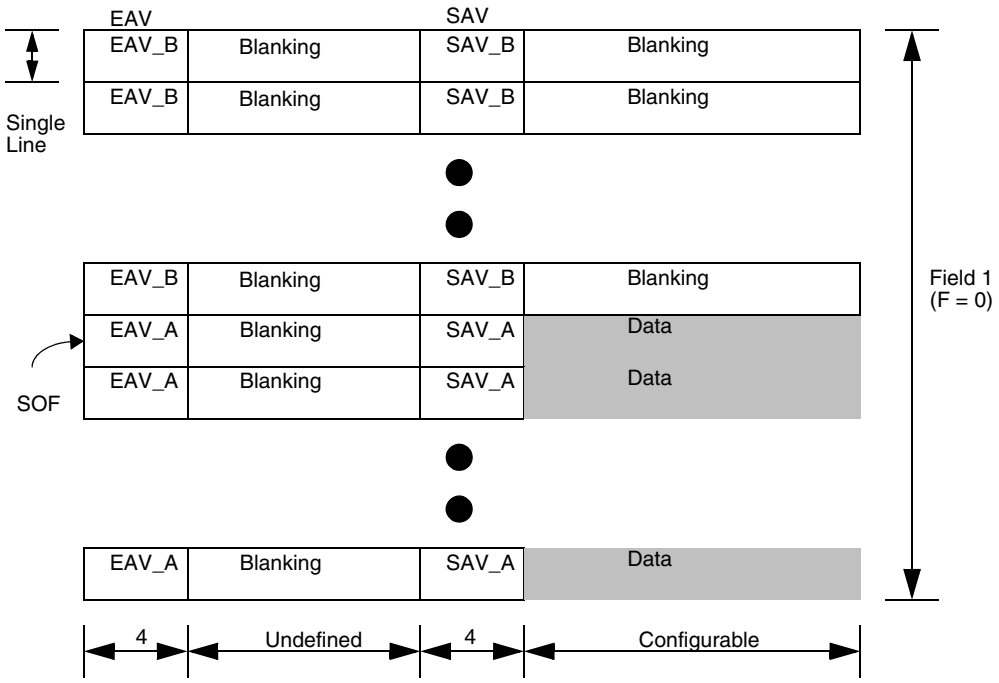


Figure 17-5. CCIR656 Progressive Mode (General Case)

An interrupt is generated for SOF but not for COF. In the general case, when SOF information is retrieved from the embedded coding, it is known as internal VSYNC mode. In other cases, when the VSYNC signal is provided by the sensor, it is known as external VSYNC mode. The CSI can be operated in internal or external VSYNC mode.

17.3.6 Error Correction for CCIR656 Coding

According to the algorithm for CCIR coding, protection bits in the SAV and EAV are encoded in the way that allows a 1-bit error to be corrected, or a 2-bit error to be detected by the decoder. This feature is supported by the CCIR decoder in CSI, for interlace mode only.

For the 1-bit error case, users can select the error to be corrected automatically, or simply shown as a status flag instead. For the 2-bit error case, because the decoder is unable to make a correction, the error would be shown as a status flag only.

An interrupt can be generated upon the detection of an error. This signal can be enabled or disabled without affecting the operation of the status bit.

17.4 Interrupt Generation

This section describes CSI events that generate interrupts.

17.4.1 Start Of Frame Interrupt (SOF_INT)

The source of an SOF interrupt is dependent on the mode of operation.

In traditional mode, VSYNC signal is taken from sensor and SOF_INT is generated at the rising or falling edge (programmable) of VSYNC.

In CCIR interlace mode, the SOF interrupt information is retrieved from the embedded coding and SOF_INT is generated.

In CCIR progressive mode, there are two sources of an SOF interrupt:

- In *internal* VSYNC mode, SOF is retrieved from the embedded coding.
- In *external* VSYNC mode, VSYNC is taken from the sensor and SOF is generated at the rising edge of VSYNC.

17.4.2 End Of Frame Interrupt (EOF_INT)

An EOF interrupt is generated when the frame ends and the complete frame data in RXFIFO is read.

The EOF event triggering works with the RX count register (CSIRXCNT). Software sets the RX count register to the frame size (in words). The CSI RX logic then counts the number of pixel data being received and compares it with the RX count. If the preset value is reached, then an EOF interrupt is generated. If an SOF event is detected before this happens, then the EOF interrupt is not generated.

Users can trigger an EOF interrupt anywhere while the current frame is being transferred by setting the RX count register to a value smaller than the actual frame size. However this would not be a true end of frame.

17.4.3 Change Of Field Interrupt (COF_INT)

The Change of Field interrupt is only valid in CCIR Interlace mode. The COF interrupt is generated when the field toggles, either from field 1 to field 2, or field 2 to field 1.

Software should first check on COF_INT bit in the CSI Status Register (CSISTAT), before checking that F1_INT or F2_INT is turned on.

In PAL systems, the field changes at the beginning of the frame and coincides with SOF. For the first field, a COF interrupt is not generated, only an SOF is generated. The COF interrupt is generated for the second field.

17.4.4 CCIR Error Interrupt (ECC_INT)

The CCIR Error Interrupt is only valid for CCIR Interlace mode. An ECC interrupt is generated when an error is found on the SAV or EAV codes in the incoming stream. When this happens, the ECC_INT status bit is set.

17.4.5 RxFIFO Full Interrupt (RxFF_INT)

A RxFIFO full interrupt is generated when the number of data in RXFIFO reaches the water mark defined by RxFF_LEVEL in CSICR3.

17.4.6 Statistic FIFO Full Interrupt (STATFF_INT)

A StatFIFO full interrupt is generated when the number of data in STATFIFO reaches the water mark defined by STATFF_LEVEL in CSICR3.

17.4.7 RxFIFO Overrun Interrupt (RFF_OR_INT)

A RxFIFO Overrun interrupt is generated when the RxFIFO has 128 words data and more data is being written in.

17.4.8 Statistic FIFO Overrun Interrupt (SFF_OR_INT)

A StatFIFO Overrun interrupt is generated when the STATFIFO has 64 words data and more data is being written in.

17.4.9 Frame Buffer1 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB1)

A DMA transfer done interrupt of frame buffer1 is generated when one frame of data are transferred from RxFIFO to the frame buffer1 in the external memory.

17.4.10 Frame Buffer2 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB2)

A DMA transfer done interrupt of frame buffer2 is generated when one frame of data are transferred from RxFIFO to the frame buffer2 in the external memory.

17.4.11 Statistic FIFO DMA Transfer Done Interrupt (DMA_TSF_DONE_SFF)

A StatFIFO DMA transfer done interrupt is generated when all the data are transferred from StatFIFO to the external memory. The transfer size is defined in the STATFIFO DMA transfer size register.

17.4.12 AHB Bus Response Error Interrupt (HRESP_ERR_INT)

An AHB Bus response error interrupt is generated when an hresponse error is detected on AHB bus.

17.5 Data Packing Style

Owing to different port sizes at different stages of the image capture path, the endianess of data is important. To enable flexible packing of image data, the CSI module provides data swapping through the PACK_DIR and the SWAP16_EN bits in CSI Control Register 1 (CSICR1) which enables data swapping before it is presented to the FIFOs. The CSI module accepts 8-bit, 10-bit or 16-bit data format from the sensor by configuring PIXEL_BIT bit in CSI Control Register 1 (CSICR1) and 16BIT_SENSOR bit in CSI Control Register3 (CSICR3).

For 10-bit per pixel data format, each pixel is expanded to 16 bits by adding 6 bits zero from MSB. Then data is packed from 8-bit or 16-bit to 32-bit according to the setting of PACK_DIR bit, and then put into the RX FIFO according to the setting of the SWAP16_EN bit.

CSI can support 16-bit data format. The input data is packed from 16-bit to 32-bit according to the setting of PACK_DIR bit, and then put into the RX FIFO according to the setting of the SWAP16_EN bit.

17.5.1 RX FIFO Path

Bayer data is a type of raw data from the image sensor. This byte-wide data must be converted to the RGB space or YUV space by software. The data path for Bayer data is from the CSI to memory. If the system is in little endian, then the PACK_DIR bit should be set to 0. For 8-bit/pixel data format from sensor, it is packed to 32-bit as P3.P2.P1.P0, where P0 is the pixel coming in time slot 0 (first data), while P3 is the pixel coming in time slot 3 (last data). When the data is addressed as bytes by software, P0 goes out first, and ends up with P3. For 10-bit/pixel data format from sensor, it is packed to 32-bit as 000000.P1.000000.P0, where P0 is the 10-bit data coming in time slot 0 (first pixel), while P1 is the 10-bit data coming in time slot 1 (second pixel). For 16-bit data format from sensor, it is packed to 32-bit as P1.P0, where P0 is the 16-bit data coming in time slot 0, while P1 is the 16-bit data coming in time slot 1.

17.5.1.1 RGB565 Data

RGB565 data is processed data from the image sensor, which can be put directly into the display buffer. The data is 16 bits wide. The data path is from CSI to memory, memory to LCDC. On the sensor side, data must be output as P0 first, followed by P1, and so on. Within each pixel, either MSB or LSB will come out first. This is controlled by the endian style of the sensor. Data is 16 bits wide with the MSB labeled RG, and the LSB labeled GB. So for P0, it is represented as RG0, GB0, and so on for P1.

CSI receives data in one of the following sequence:

- RG0, GB0, RG1, GB1, while RG0 comes out at time slot 0 (first data), and GB1 comes out at time slot 3 (last data), or
- GB0, RG0, GB1, RG1.

Using the first sequence as an example, and assuming the system is running in little endian the data is presented as:

- 8-bit data from sensor: RG0, GB0, RG1, GB1, …
- 32-bit data before CSI RX FIFO (PACK_DIR bit = 1): RG0GB0RG1GB1
- 32-bit data in CSI RX FIFO (SWAP16_EN bit enabled): RG1GB1RG0GB0
- 32-bit transfer to system memory: RG1GB1RG0GB0
- 16-bit read by LCDC: RG0GB0, RG1GB1

17.5.1.2 RGB888 Data

This is another kind of processed data from image sensor, which can be used for further image processing directly. Each of the data consist of 8-bit Red, 8-bit Green, and 8-bit Blue data. An example of a possible timing scheme is shown in Figure 17-6.

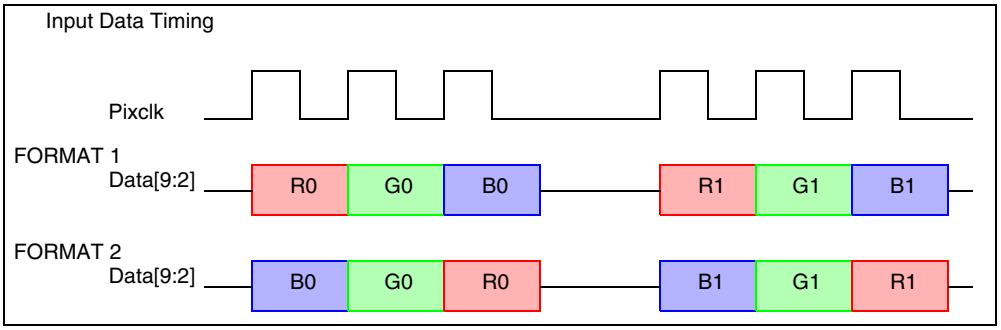


Figure 17-6. Sample Timing Diagram for RGB888 Data

To improve the data transfer, an optional dummy byte packing scheme is provided. For every group of 3 bytes data, a dummy zero is packed to form a 32-bit word as shown in Figure 17-7. The dummy zero is always packed at the LSB position.

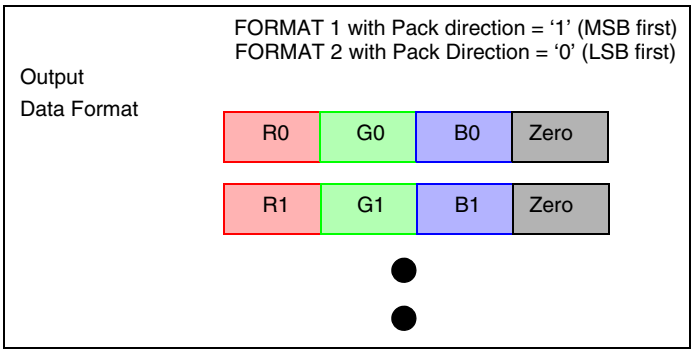


Figure 17-7. Optional Dummy Byte Packing Scheme

17.5.2 STAT FIFO Path

Statistics only works for Bayer data in 8-bit per pixel format. It generates 16-bit statistical output from the 8-bit Bayer input (DATA[9:2]). The outputs are Sum of Green (G), Sum of Red (R), Sum of Blue (B), and Auto Focus (F). Each output is 16-bits wide.

The settings of PACK_DIR and SWAP16_EN bits in the CSICR1 register have no effect on the input path. The PACK_DIR only controls how the 16-bit stat output is packed into the 32-bit STAT FIFO.

When the PACK_DIR bit = 1, the stat data is packed as:

First 32-bit: RG

Second 32-bit: BF

…

When the PACK_DIR bit = 0, the stat data is packed as:

First 32-bit GR
Second 32-bit: FB
…

17.6 Memory Map and Register Definition

All the 32-bit registers of the CSI module are summarized in Table 17-5. Table 17-7 summarizes the registers and offset addresses.

17.6.1 CSI Memory Map

Table 17-5 shows the CSI memory map.

Table 17-5. CSI Memory Map

Address	Use	Access	Reset Value	Section/Page
0xBASE_0000 (CSICR1)	CSI Control Register 1	R/W	0x4000_0800	17.6.3/17-17
0xBASE_0004 (CSICR2)	CSI Control Register 2	R/W	0x0000_0000	17.6.4/17-21
0xBASE_0008 (CSICR3)	CSI Control Register 3	R/W	0x0000_0000	17.6.5/17-22
0xBASE_000C (CSISTATFIFO)	CSI Statistic FIFO Register	R	0x0000_0000	17.6.6/17-24
0xBASE_0010 (CSIRFIFO)	CSI RX FIFO Register	R	0x0000_0000	17.6.7/17-25
0xBASE_0014 (CSIRXCNT)	CSI RX Count Register	R/W	0x0000_9600	17.6.8/17-25
0xBASE_0018 (CSISR)	CSI Status Register	R/W	0x0000_4000	17.6.9/17-26
0xBASE_0020 (CSIDMASA-STATFIFO)	CSI DMA Start Address Register - for STATFIFO	R/W	0x0000_0000	17.6.10/17-29
0xBASE_0024 (CSIDMATS-STATFIFO)	CSI DMA Transfer Size Register - for STATFIFO	R/W	0x0000_0000	17.6.11/17-30
0xBASE_0028 (CSIDMASA-FB1)	CSI DMA Start Address Register - for Frame Buffer1	R/W	0x0000_0000	17.6.12/17-30
0xBASE_002C (CSIDMASA-FB2)	CSI DMA Transfer Size Register - for Frame Buffer2	R/W	0x0000_0000	17.6.13/17-31
0xBASE_0030 (CSIFBUF_PARA)	CSI Frame Buffer Parameter Register	R/W	0x0000_0000	17.6.14/17-32
0xBASE_0034 (CSIIMAG_PARA)	CSI Image Parameter Register	R/W	0x0000_0000	17.6.15/17-32

17.6.2 Register Summary

Figure 17-8 shows the key to the register fields, and Table 17-6 shows the register figure conventions.

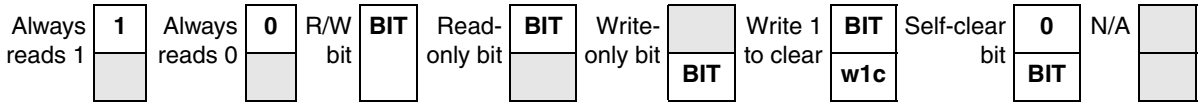


Figure 17-8. Key to Register Fields

Table 17-6. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 17-7 shows the CSI register summary.

Table 17-7. CSI Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_0000 (CSICR1)	R	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	CCIR_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	0	SFF_DMA_DONE_INTEN	STATFF_INTEN	FB2_DMA_DONE_INTEN	FB1_DMA_DONE_INTEN	RxFF_INTEN	SOF_POL	SOF_INTEN
	W																
	R	MCLKDIV				HSYNC_POL	CCIR_EN	MCLKEN	FCC	PACK_DIR	CLR_STATFIFO	CLR_RxFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	PIXEL_BIT
	W																

Table 17-7. CSI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_0004 (CSICR2)	R	DMA_BURST_TYPE_RFF		DMA_BURST_TYPE_SFF		0	DRM	AFS		SCE	0	0	BTS		LVRM		
	W																
	R	VSC								HSC							
	W																
0xBASE_0008 (CSICR3)	R	FRMCNT															
	W																
	R	0	0	0	DMA_REQ_EN_RFF	DMA_REQ_EN_SFF	STATFF_LEVEL			HRESP_ERR_EN	RxFF_LEVEL			16BIT_SENSOR	ZERO_PACK_EN	ECC_INT_EN	ECC_AUTO_EN
	W	FRMCNT_RST	DMA_REFLASH_RFF	DMA_REFLASH_SFF													
0xBASE_000C (CSISTATFIFO)	R	STAT															
	W																
	R	STAT															
	W																
0xBASE_0010 (CSIRFIFO)	R	IMAGE															
	W																
	R	IMAGE															
	W																
0xBASE_0014 (CSIRXCNT)	R	0	0	0	0	0	0	0	0	0	0	RXCNT					
	W																
	R	RXCNT															
	W																

Table 17-7. CSI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_0018 (CSISR)	R	0	0	0	0	0	0	SFF_OR_INT	RFF_OR_INT	0	DMA_TSF_DONE_SFF	STATFF_INT	DMA_TSF_DONE_FB2	DMA_TSF_DONE_FB1	RxFF_INT	EOF_INT	SOF_INT
	W							w1c	w1c		w1c		w1c	w1c		w1c	w1c
	R	F2_INT	F1_INT	COF_INT	0	0	0	0	0	HRESP_ERR_INT	0	0	0	0	0	ECC_INT	DRDY
	W			w1c						w1c						w1c	
0xBASE_0020 (CSIDMASA-STATFIFO)	R	DMA_START_ADDR_SFF[31:16]															
	W																
	R	DMA_START_ADDR_SFF[15:2]														0	0
	W																
0xBASE_0024 (CSIDMATS-STATFIFO)	R	DMA_TSF_SIZE_SFF[31:16]															
	W																
	R	DMA_TSF_SIZE_SFF[15:0]															
	W																
0xBASE_0028 (CSIDMASA-FB1)	R	DMA_START_ADDR_FB1[31:16]															
	W																
	R	DMA_START_ADDR_FB1[15:2]														0	0
	W																
0xBASE_002C (CSIDMASA-FB2)	R	DMA_START_ADDR_FB2[31:16]															
	W																
	R	DMA_START_ADDR_FB2[15:2]														0	0
	W																
0xBASE_0030 (CSIFBUF_PARA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	FBUF_STRIDE[15:0]															
	W																

Table 17-7. CSI Register Summary (continued)

Name		31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0xBASE_0034 (CSIIMAG_PARA)	R W	IMAGE_WIDTH[15:0]
	R W	IMAGE_HEIGHT[15:0]

17.6.3 CSI Control Register 1 (CSICR1)

This register, shown in Figure 17-9 and Table 17-8, controls the sensor interface timing and interrupt generation. The interrupt enable bits in this register control the interrupt signals and the status bits. That means status bits function only when the corresponding interrupt bits are enabled.

0xBASE_0000 (CSICR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	CCIR_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	0	SFF_DMA_DONE_INTEN	STATFF_INTEN	FB2_DMA_DONE_INTEN	FB1_DMA_DONE_INTEN	RxFF_INTEN	SOF_POL	SOF_INTEN
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MCLKDIV				HSYNC_POL	CCIR_EN	MCLKEN	FCC	PACK_DIR	CLR_STATFIFO	CLR_RxFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	PIXEL_BIT
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 17-9. CSPI Control Register 1 (CSICR1)

Table 17-8. CSI Control Register 1 Field Descriptions

Field	Description
31 SWAP16_EN	SWAP 16-Bit Enable. This bit enables the swapping of 16-bit data. Data is packed from 8-bit or 10-bit to 32-bit first (according to the setting of PACK_DIR and then swapped as 16-bit words before putting into the RX FIFO. The action of the bit only affects the RX FIFO and has no affect on the STAT FIFO. **Note:** Example of swapping enabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x 33441122 **Note:** Example of swapping disabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x11223344 0 Disable swapping 1 Enable swapping
30 EXT_VSYNC	External VSYNC Enable. This bit controls the operational VSYNC mode. **Note:** This only works when the CSI is in CCIR progressive mode. 0 Internal VSYNC mode 1 External VSYNC mode
29 EOF_INT_EN	End-of-Frame Interrupt Enable. This bit enables and disables the EOF interrupt. 0 EOF interrupt is disabled. 1 EOF interrupt is generated when RX count value is reached.
28 PrP_IF_EN	CSI—PrP Interface Enable. This bit controls the CSI to PrP bus. When enabled the RxFIFO is detached from the AHB bus and connected to PrP. All CPU reads or DMA accesses to the RxFIFO register are ignored. All CSI interrupts are also masked. 0 CSI to PrP bus is disabled 1 CSI to PrP bus is enabled
27 CCIR_MODE	CCIR Mode Select. This bit controls the CCIR mode of operation. This bit only works in CCIR interface mode. 0 Progressive mode is selected 1 Interlace mode is selected
26 COF_INT_EN	Change Of Image Field (COF) Interrupt Enable. This bit enables the COF interrupt. This bit works only in CCIR interlace mode which is when CCIR_EN = 1 and CCIR_MODE = 1. 0 COF interrupt is disabled 1 COF interrupt is enabled
25 SF_OR_INTEN	STAT FIFO Overrun Interrupt Enable. This bit enables the STATFIFO overrun interrupt. 0 STATFIFO overrun interrupt is disabled 1 STATFIFO overrun interrupt is enabled
24 RF_OR_INTEN	RxFIFO Overrun Interrupt Enable. This bit enables the RX FIFO overrun interrupt. 0 RxFIFO overrun interrupt is disabled 1 RxFIFO overrun interrupt is enabled
23	Reserved.
22 SFF_DMA_DONE_INTEN	STATFIFO DMA Transfer Done Interrupt Enable. This bit enables the interrupt of STATFIFO DMA transfer done. 0 STATFIFO DMA Transfer Done interrupt disable 1 STATFIFO DMA Transfer Done interrupt enable
21 STATFF_INTEN	STATFIFO Full Interrupt Enable. This bit enables the STAT FIFO interrupt. 0 STATFIFO full interrupt disable 1 STATFIFO full interrupt enable

Table 17-8. CSI Control Register 1 Field Descriptions (continued)

Field	Description
20 FB2_DMA_DONE_INTEN	Frame Buffer2 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer2 DMA transfer done. 0 Frame Buffer2 DMA Transfer Done interrupt disable 1 Frame Buffer2 DMA Transfer Done interrupt enable
19 FB1_DMA_DONE_INTEN	Frame Buffer1 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer1 DMA transfer done. 0 Frame Buffer1 DMA Transfer Done interrupt disable 1 Frame Buffer1 DMA Transfer Done interrupt enable
18 RXFF_INTEN	RxFIFO Full Interrupt Enable. This bit enables the RxFIFO full interrupt. 0 RxFIFO full interrupt disable 1 RxFIFO full interrupt enable
17 SOF_POL	SOF Interrupt Polarity. This bit controls the condition that generates an SOF interrupt. 0 SOF interrupt is generated on SOF falling edge 1 SOF interrupt is generated on SOF rising edge
16 SOF_INTEN	Start Of Frame (SOF) Interrupt Enable. This bit enables the SOF interrupt. 0 SOF interrupt disable 1 SOF interrupt enable
15–12 MCLKDIV	Sensor Master Clock (MCLK) Divider. This field contains the divisor MCLK. The MCLK is derived from the PERCLK. 0000 Divided by 1 0001 Divided by 2 0010 Divided by 4 0011 Divided by 6 0100 Divided by 8 0101 Divided by 10 0110 Divided by 12 0111 Divided by 14 1000 Divided by 16 1001 Divided by 18 1010 Divided by 20 1011 Divided by 22 1100 Divided by 24 1101 Divided by 26 1110 Divided by 28 1111 Divided by 30
11 HSYNC_POL	HSYNC Polarity Select. This bit controls the polarity of HSYNC. **Note:** This bit only works in gated-clock—that is, GCLK_MODE = 1 and CCIR_EN = 0. 0 HSYNC is active low 1 HSYNC is active high
10 CCIR_EN	CCIR656 Interface Enable. This bit selects the type of interface used. When the CCIR656 timing decoder is enabled, it replaces the function of timing interface logic. 0 Traditional interface is selected. Timing interface logic is used to latch data. 1 CCIR656 interface is selected.
9 MCLKEN	Sensor Master Clock (MCLK) Enable. This bit enables or disables the MCLK input to the sensor. 0 MCLK disable 1 MCLK enable

Table 17-8. CSI Control Register 1 Field Descriptions (continued)

Field	Description
8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected. **Note:** FCC should only be used when CSI DMA burst size and FIFO water-mark fill level match. If they do not match and FCC bit is set this may cause corrupted images.
7 PACK_DIR	Data Packing Direction. This bit Controls how 8-bit/10-bit image data is packed into 32-bit RX FIFO, and how 16-bit statistical data is packed into 32-bit STAT FIFO. 0 Pack from LSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x44332211 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xBBBBAAAA in STAT FIFO. 1 Pack from MSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x11223344 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xAAAABBBB in STAT FIFO.
6 CLR_STATFIFO	Asynchronous STATFIFO Clear. This bit clears the STATFIFO and Reset STAT block. **Note:** This bit works only in async FIFO clear mode—that is, FCC = 0. Otherwise this bit is ignored. Writing 1 will clear STATFIFO and reset STAT block immediately, STATFIFO and STAT block then wait and restart after the arrival of next SOF. The bit is restored to 0 automatically after finish. Normally reads 0.
5 CLR_RXFIFO	Asynchronous RXFIFO Clear. This bit clears the RXFIFO. This bit works only in async FIFO clear mode—that is, FCC = 0. Otherwise this bit is ignored. Writing 1 clears the RXFIFO immediately, RXFIFO restarts immediately after that. The bit is restore to 0 automatically after finish. Normally reads 0.
4 GCLK_MODE	Gated Clock Mode Enable. Controls if CSI is working in gated or non-gated mode. **Note:** This bit works only in traditional mode—that is, CCIR_EN = 0. Otherwise this bit is ignored. 0 Non-gated clock mode. All incoming pixel clocks are valid. HSYNC is ignored. 1 Gated clock mode. Pixel clock signal is valid only when HSYNC is active.
3 INV_DATA	Invert Data Input. This bit enables or disables internal inverters on the data lines. 0 CSI_D[7:0] data lines are directly applied to internal circuitry 1 CSI_D[7:0] data lines are inverted before applied to internal circuitry
2 INV_PCLK	Invert Pixel Clock Input. This bit determines if the Pixel Clock (CSI_PIXCLK) is inverted before it is applied to the CSI module. 0 CSI_PIXCLK is directly applied to internal circuitry 1 CSI_PIXCLK is inverted before applied to internal circuitry
1 REDGE	Valid Pixel Clock Edge Select. Selects which edge of the CSI_PIXCLK is used to latch the pixel data. 0 Pixel data is latched at the falling edge of CSI_PIXCLK 1 Pixel data is latched at the rising edge of CSI_PIXCLK
0 PIXEL_BIT	Pixel Bit. This bit indicates the bayer data width for each pixel. This bit should be configured before activating or re-starting the embedded DMA controller. 0 8-bit data for each pixel 1 10-bit data for each pixel

17.6.4 CSI Control Register 2 (CSICR2)

This register, shown in Figure 17-10 and Table 17-9, provides the statistic block with data about which live view resolution is being used, and the starting sensor pixel of the Bayer pattern. It also contains the horizontal and vertical count used to determine the number of pixels to skip between the 64 × 64 blocks of statistics when generating statistics on live view image that are greater than 512 × 384.

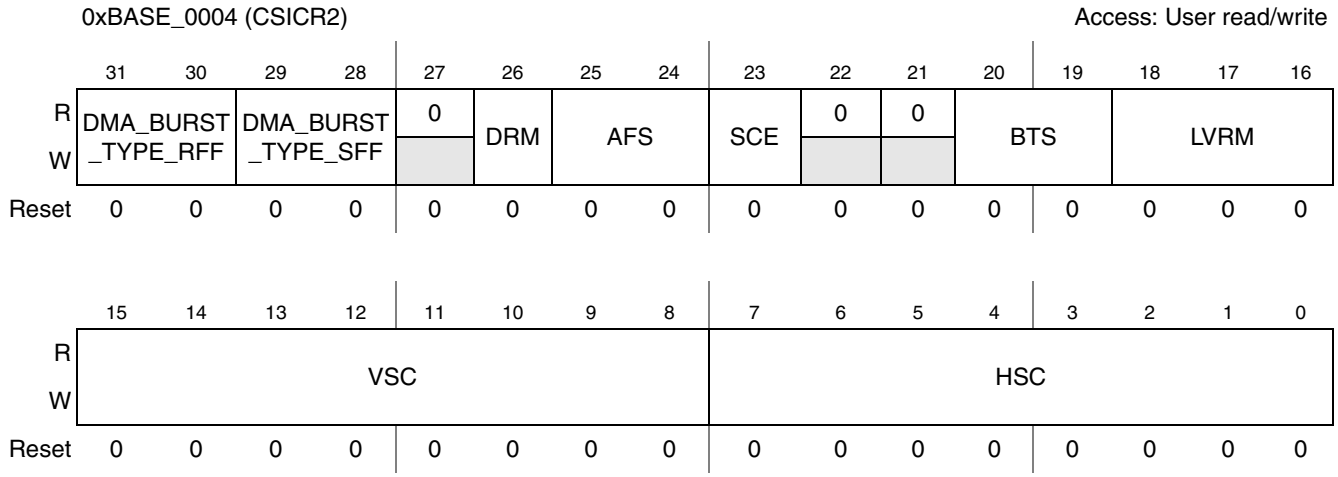


Figure 17-10. CSI Control Register 2 (CSICR2)

Table 17-9. CSI Control Register 2 Description

Field	Description
31–30 DMA_BURST_TYPE_RFF	Burst Type of DMA Transfer from RxFIFO. Selects the burst type of DMA transfer from RxFIFO. 00 INCR8 01 INCR4 10 INCR8 11 INCR16 **Note:** The optimal setting is INCR8 so that the CSI burst size matches the ESDRAMC burst size.
29–28 DMA_BURST_TYPE_SFF	Burst Type of DMA Transfer from STATFIFO. Selects the burst type of DMA transfer from STATFIFO. 00 INCR8 01 INCR4 10 INCR8 11 INCR16 **Note:** The optimal setting is INCR8 so that the CSI burst size matches the ESDRAMC burst size.
27	Reserved.
26 DRM	Double Resolution Mode. Controls size of statistics grid. 0 Stats grid of 8 × 6 1 Stats grid of 8 × 12
25–24 AFS	Auto Focus Spread. Selects which green pixels are used for auto-focus. 00 Abs Diff on consecutive green pixels 01 Abs Diff on every third green pixels 1x Abs Diff on every four green pixels
23 SCE	Skip Count Enable. Enables or disables the skip count feature. 0 Skip count disable 1 Skip count enable

Table 17-9. CSI Control Register 2 Description (continued)

Field	Description
22–21	Reserved.
20–19 BTS	Bayer Tile Start. Controls the Bayer pattern starting point. 00 GR 01 RG 10 BG 11 GB
18–16 LVRM	Live View Resolution Mode. Selects the grid size used for live view resolution. 0 512 × 384 1 448 × 336 2 384 × 288 3 384 × 256 4 320 × 240 5 288 × 216 6 400 × 300
15–8 VSC	Vertical Skip Count. Contains the number of rows to skip. SCE must be 1, otherwise VSC is ignored. 0–255 Number of rows to skip minus 1
7–0 HSC	Horizontal Skip Count. Contains the number of pixels to skip. SCE must be 1, otherwise HSC is ignored. 0–255 Number of pixels to skip minus 1

17.6.5 CSI Control Register 3 (CSICR3)

This read/write register, shown in Figure 17-11 and Table 17-10, acts as an extension of the functionality of the CSI Control register 1, adding control and features.

0xBASE_0008 (CSICR3) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FRMCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0													
W	FRMCNT_RST	DMA_REFLASH_RFF	DMA_REFLASH_SFF	DMA_REQ_EN_RFF	DMA_REQ_EN_SFF	STATFF_LEVEL			HRESP_ERR_EN	RxFF_LEVEL			16BIT_SENSOR	ZERO_PACK_EN	ECC_INT_EN	ECC_AUTO_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-11. CSI Control Register 3 (CSICR3)

Table 17-10. CSI Control Register 3 Field Descriptions

Field	Description
31–16 FRMCNT	Frame Counter. This is a 16-bit Frame Counter (Wrap around automatically after reaching the maximum)
15 FRMCNT_RST	Frame Count Reset. Resets the Frame Counter. 0 Do not reset 1 Reset frame counter immediately (Cleared automatically after reset is done)
14 DMA_REFLASH_RFF	Reflash DMA Controller for RxFIFO. This bit reflash the embedded DMA controller for RxFIFO. It should be reflashed before the embedded DMA controller starts to work. 0 No reflashing 1 Reflash the embedded DMA controller (Cleared automatically after reflashing is done)
13 DMA_REFLASH_SFF	Reflash DMA Controller for STATFIFO. This bit reflash the embedded DMA controller for STATFIFO. It should be reflashed before the embedded DMA controller starts to work. 0 No reflashing 1 Reflash the embedded DMA controller (Cleared automatically after reflashing is done)
12 DMA_REQ_EN_RFF	DMA Request Enable for RxFIFO. This bit enables the DMA request from RxFIFO to the embedded DMA controller. 0 Disable the DMA request 1 Enable the DMA request
11 DMA_REQ_EN_SFF	DMA Request Enable for STATFIFO. This bit enables the DMA request from STATFIFO to the embedded DMA controller. 0 Disable the DMA request 1 Enable the DMA request

Table 17-10. CSI Control Register 3 Field Descriptions

Field	Description
10-8 STATFF_LEVEL	STATFIFO Full Level. When the number of data in STATFIFO reach this level, STATFIFO full interrupt is generated, or STATFIFO DMA request is sent. 000 4 Words 001 8 Words 010 12 Words 011 16 Words 100 24 Words 101 32 Words 110 48 Words 111 not support
7 HRESP_ERR_EN	Hresponse Error Enable. This bit enables the hresponse error interrupt. 0 Disable hresponse error interrupt 1 Enable hresponse error interrupt
6-4 RxFF_LEVEL	RxFIFO Full Level. When the number of data in RxFIFO reach this level, a RxFIFO full interrupt is generated, or an RXFIFO DMA request is sent. 000 4 Words 001 8 Words 010 16 Words 011 24 Words 100 32 Words 101 48 Words 110 64 Words 111 96 Words
3 16BIT_SENSOR	16-bit Sensor Mode. This bit indicates one 16-bit sensor is connected to the 16-bit data ports. This bit should be configured before activating or re-starting the embedded DMA controller. 0 16-bit sensor is not connected. 1 16-bit sensor is connected.
2 ZERO_PACK_EN	Dummy Zero Packing Enable. This bit causes a dummy zero to be packed with every 3 incoming bytes, forming a 32-bit word. The dummy zero is always packed to the LSB position. This packing function is only available in 8-bit/pixel mode. 0 Zero packing disabled 1 Zero packing enabled
1 ECC_INT_EN	Error Detection Interrupt Enable. This bit enables and disables the error detection interrupt. This feature only works in CCIR interlace mode. 0 No interrupt is generated when error is detected. Only the status bit ECC_INT is set. 1 Interrupt is generated when error is detected.
0 ECC_AUTO_EN	Automatic Error Correction Enable. This bit enables and disables the automatic error correction. If an error occurs and error correction is disabled only the ECC_INT status bit is set. This feature only works in CCIR interlace mode. 0 Auto Error correction is disabled. 1 Auto Error correction is enabled.

17.6.6 CSI STATFIFO Register (CSISTATFIFO)

The StatFIFO, shown in Figure 17-12, is a read-only register containing statistic data from the sensor. Writing to this register has no effect.

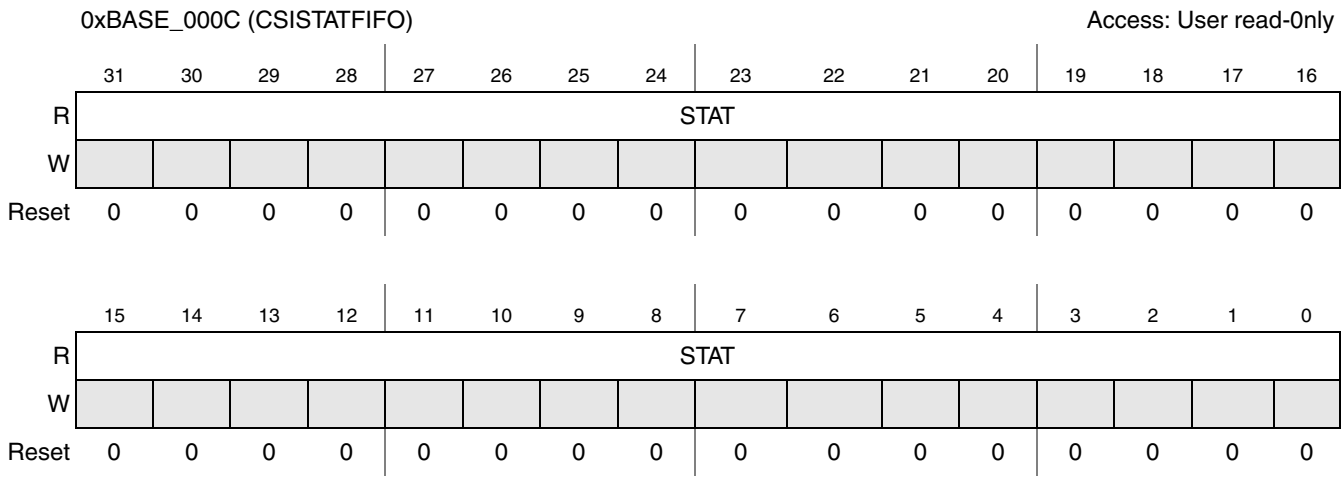


Figure 17-12. CSI STATFIFO Register (CSISTATFIFO)

17.6.7 CSI RxFIFO Register (CSIRFIFO)

This read-only register, shown in Figure 17-13, contains received image data. Writing to this register has no effect.

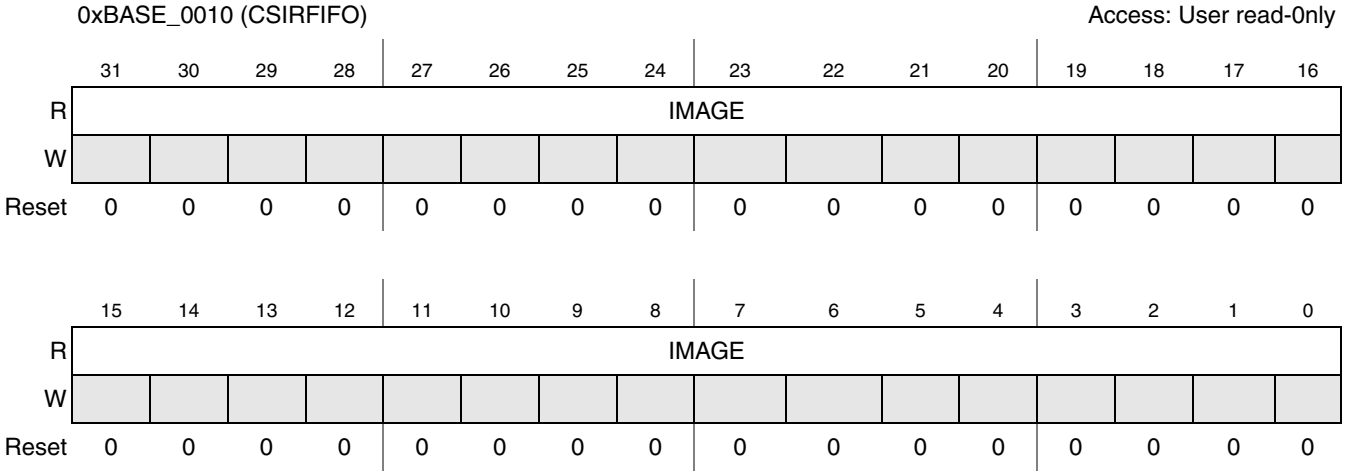


Figure 17-13. CSI RxFIFO Register (CSIRFIFO)

17.6.8 CSI RX Count Register (CSIRXCNT)

This register, shown in Figure 17-14 and Table 17-11, works for EOF interrupt generation. It should be set to the number of words to receive that would generate an EOF interrupt.

There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or the embedded DMA controller, the counter value is updated and compared with this register. If the values match, then an EOF interrupt is triggered. For example, one FIFO word is 4 bytes, so for a picture of 640 × 480 using 2 bytes/pixel, RXCNT should be set to 640 × 480 × 2/4.

0xBASE_0014 (CSIRXCNT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	RXCNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXCNT															
W																
Reset	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0

Figure 17-14. CSI RX Count Register (CSIRXCNT)

Table 17-11. CSI RX Count Register Field Descriptions

Field	Description
31–22	Reserved.
21–0 RXCNT	RxFIFO Count. This 22-bit counter for RXFIFO is updated each time the RXFIFO is read by CPU or DMA.This counter should be set to the expected number of words to receive that would generate an EOF interrupt.

17.6.9 CSI Status Register (CSISR)

This read/write register, shown in Figure 17-15 and Table 17-12, indicates sensor interface status, and which kind of interrupt is being generated. The corresponding interrupt bits must be set for the status bit to function. Status bits should function normally even if the corresponding interrupt enable bits are not enabled.

0xBASE_0018 (CSISR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	SFF_OR_INT	RFF_OR_INT	0	DMA_TSF_DONE_SFF	STATFF_INT	DMA_TSF_DONE_FB2	DMA_TSF_DONE_FB1	RxFF_INT	EOF_INT	SOF_INT
W							w1c	w1c		w1c		w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F2_INT	F1_INT	COF_INT	0	0	0	0	0	HRESP_ERR_INT	0	0	0	0	0	ECC_INT	DRDY
W			w1c						w1c						w1c	
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-15. CSI Status Register (CSISR)

Table 17-12. CSI Status Register Field Descriptions

Field	Description
31–26	Reserved.
25 SF_OR_INT	STATFIFO Overrun Interrupt Status. Indicates the overflow status of the STATFIFO register. 0 STATFIFO has not overflowed. 1 STATFIFO has overflowed. (Cleared by writing 1)
24 RF_OR_INT	RxFIFO Overrun Interrupt Status. Indicates the overflow status of the RxFIFO register. 0 RXFIFO has not overflowed. 1 RXFIFO has overflowed. (Cleared by writing 1)
23	Reserved.
22 DMA_TSF_DONE_SFF	DMA Transfer Done from StatFIFO. Indicates that the DMA transfer from StatFIFO is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writing 1 or reflashing the StatFIFO DMA controller in CSICR3. 0 DMA transfer is not completed. 1 DMA transfer is completed. (Cleared by writing 1)

Table 17-12. CSI Status Register Field Descriptions (continued)

Field	Description
21 STATFF_INT	STATFIFO Full Interrupt Status. Indicates the number of data in the STATFIFO reaches the trigger level. 0 STATFIFO is not full. 1 STATFIFO is full. (this bit is cleared automatically by reading the STATFIFO)
20 DMA_TSF_DONE_FB2	DMA Transfer Done in Frame Buffer2. Indicates that the DMA transfer from RxFIFO to Frame Buffer2 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writing 1 or reflashing the RxFIFO DMA controller in CSICR3. 0 DMA transfer is not completed. 1 DMA transfer is completed. (Cleared by writing 1)
19 DMA_TSF_DONE_FB1	DMA Transfer Done in Frame Buffer1. Indicates that the DMA transfer from RxFIFO to Frame Buffer1 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writing 1 or reflashing the RxFIFO DMA controller in CSICR3. 0 DMA transfer is not completed. 1 DMA transfer is completed. (Cleared by writing 1)
18 RxFF_INT	RXFIFO Full Interrupt Status. Indicates the number of data in the RxFIFO reaches the trigger level. 0 RxFIFO is not full. 1 RxFIFO is full. (this bit is cleared automatically by reading the RxFIFO)
17 EOF_INT	End of Frame (EOF) Interrupt Status. Indicates when EOF is detected. 0 EOF is not detected. 1 EOF is detected. (Cleared by writing 1)
16 SOF_INT	Start of Frame Interrupt Status. Indicates when SOF is detected. 0 SOF is not detected. 1 SOF is detected. (Cleared by writing 1)
15 F2_INT	CCIR Field 2 Interrupt Status. Indicates the presence of field 2 of video in CCIR mode. **Note:** Only works in CCIR Interlace mode. 0 Field 2 of video is not detected 1 Field 2 of video is about to start (Cleared automatically when current field does not match)
14 F1_INT	CCIR Field 1 Interrupt Status. Indicates the presence of field 1 of video in CCIR mode. **Note:** Only works in CCIR Interlace mode. 0 Field 1 of video is not detected. 1 Field 1 of video is about to start. (Cleared automatically when current field does not match)
13 COF_INT	Change Of Field Interrupt Status. Indicates that a change of the video field has been detected. Only works in CCIR Interlace mode. Software should read this bit first and then dispatch the new field from F1_INT and F2_INT. 0 Video field has no change. 1 Change of video field is detected. (Cleared by writing 1)
12–8	Reserved.

Table 17-12. CSI Status Register Field Descriptions (continued)

Field	Description
7 HRESP_ERR_INT	Hresponse Error Interrupt Status. Indicates that a hresponse error has been detected. 0 No hresponse error. 1 Hresponse error is detected. (Cleared by writing 1)
6– 2	Reserved.
1 ECC_INT	CCIR Error Interrupt. This bit indicates an error has occurred. This only works in CCIR Interlace mode. 0 No error detected 1 Error is detected in CCIR coding (Cleared by writing 1)
0 DRDY	RXFIFO Data Ready. Indicates the presence of data that is ready for transfer in the RxFIFO. 0 No data (word) is ready 1 At least 1 data (word) is ready in RXFIFO. (Cleared automatically by reading FIFO)

17.6.10 CSI STATFIFO DMA Start Address Register (CSIDMASA-STATFIFO)

This register, shown in Figure 17-16 and Table 17-13, provides the start address for the embedded DMA controller of STATFIFO. The embedded DMA controller will read data from STATFIFO and write it to the external memory from the start address. This register should be configured before activating or re-starting the embedded DMA controller.

0xBASE_0020 (CSIDMASA-STATFIFO) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_SFF[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMA_START_ADDR_SFF[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-16. CSI STATFIFO DMA Start Address Register (CSIDMASA-STATFIFO)

Table 17-13. CSI STATFIFO DMA Start Address Register Field Descriptions

Field	Description
31–2 DMA_START_ADDR_SFF	DMA Start Address for STATFIFO. Indicates the start address to write data. The embedded DMA controller will read data from STATFIFO and write it from this address through AHB bus. The address should be word aligned.
1-0	Reserved.

17.6.11 CSI STATFIFO DMA Transfer Size Register (CSIDMATS-STATFIFO)

This register, shown in Figure 17-17 and Table 17-14, provides the total transfer size for the embedded DMA controller of STATFIFO. This register should be configured before activating or re-starting the embedded DMA controller.

0xBASE_0024 (CSIDMATS-STATFIFO) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_TSF_SIZE_SFF[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_TSF_SIZE_SFF[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-17. CSI STATFIFO DMA Transfer Size Register (CSIDMATS-STATFIFO)

Table 17-14. CSI STATFIFO DMA Transfer Size Register Field Descriptions

Field	Description
31–0 DMA_TSF_SIZE_SFF	DMA Transfer Size for STATFIFO. Indicates how many words to be transferred by the embedded DMA controller. The size should be word aligned.

17.6.12 CSI Frame Buffer1 DMA Start Address Register (CSIDMASA-FB1)

This register, shown in Figure 17-18 and Table 17-15, provides the start address in the frame buffer1 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer1 from the start address. This register should be configured before activating or re-starting the embedded DMA controller.

0xBASE_0028 (CSIDMASA-FB1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA_START_ADDR_FB1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMA_START_ADDR_FB1[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-18. CSI Frame Buffer1 DMA Start Address Register (CSIDMASA-FB1)

Table 17-15. CSI Frame Buffer1 DMA Start Address Register Field Descriptions

Field	Description
31–2 DMA_START_ADDR_FB1	DMA Start Address in Frame Buffer1. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be word aligned.
1-0	Reserved.

17.6.13 CSI Frame Buffer2 DMA Start Address Register (CSIDMASA-FB2)

This register, shown in Figure 17-19 and Table 17-16, provides the start address in the frame buffer2 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer2 from the start address. This register should be configured before activating or re-starting the embedded DMA controller.

0xBASE_002C (CSIDMASA-FB2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA_START_ADDR_FB2[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMA_START_ADDR_FB2[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-19. CSI Frame Buffer2 DMA Start Address Register (CSIDMASA-FB2)

Table 17-16. CSI Frame Buffer2 DMA Start Address Register Field Descriptions

Field	Description
31–2 DMA_START_ADDR_FB2	DMA Start Address in Frame Buffer2. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. The address should be word aligned.
1-0	Reserved.

17.6.14 CSI Frame Buffer Parameter Register (CSIFBUF_PARA)

This register, shown in Figure 17-20 and Table 17-17, provides the stride of the frame buffer to show how many words to skip before starting to write the next row of the image. The width of the frame buffer minus the width of the image is the stride. This register should be configured before activating or re-starting the embedded DMA controller.

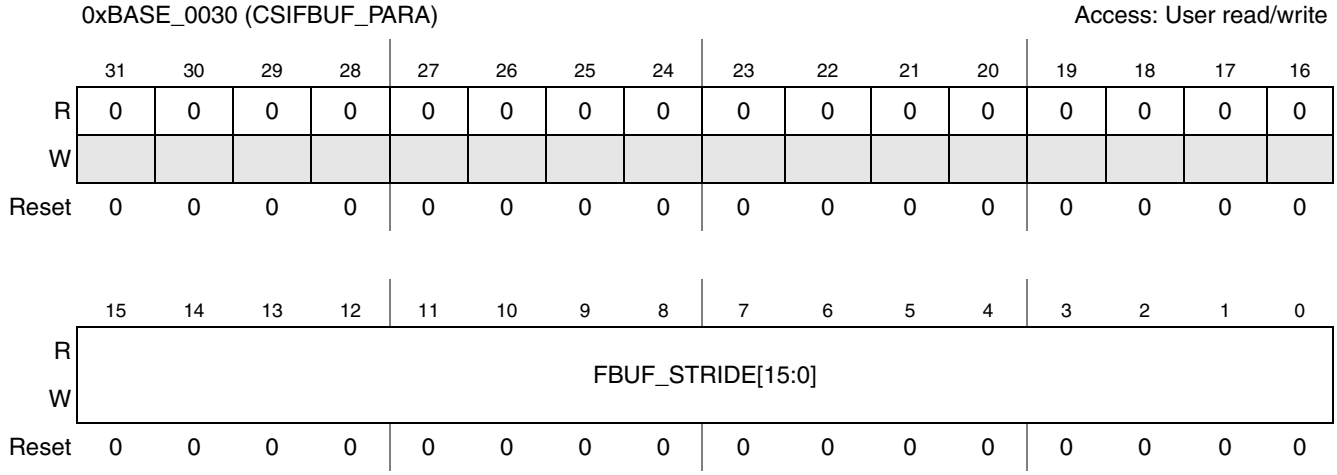


Figure 17-20. CSI Frame Buffer Parameter Register (CSIFBUF_PARA)

Table 17-17. CSI Frame Buffer Parameter Register Field Descriptions

Field	Description
31-16	Reserved.
15–0 FBUF_STRIDE	Frame Buffer Parameter. Indicates the stride of the frame buffer. The width of the frame buffer (in word) minus the width of the image (in word) is the stride. The stride should be word aligned. The embedded DMA controller will skip the stride before starting to write the next row of the image.

17.6.15 CSI Image Parameter Register (CSIIMAG_PARA)

This register, shown in Figure 17-21 and Table 17-18, provides the width and the height of the image from the sensor. The width and height should be aligned in pixel. The width of the image multiplied by the height is the total pixel size that will be transferred in a frame by the embedded DMA controller. This register should be configured before activating or re-starting the embedded DMA controller.

0xBASE_0034 (CSIIMAG_PARA) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMAGE_WIDTH[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMAGE_HEIGHT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 17-21. CSI Image Parameter Register (CSIIMAG_PARA)

Table 17-18. CSI Image Parameter Register Field Descriptions

Field	Description
31–16 IMAGE_WIDTH	Image Width. Indicates how many pixels in a line of the image from the sensor. If the input data from the sensor is 8-bit/pixel format, the IMAGE_WIDTH should be a multiple of 4 pixels. If the input data from the sensor is 10-bit/pixel or 16-bit/pixel format, the IMAGE_WIDTH should be a multiple of 2 pixels.
15–0 IMAGE_HEIGHT	Image Height. Indicates how many pixels in a column of the image from the sensor.

Chapter 18
Configurable Serial Peripheral Interface (CSPI)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

18.1 Overview

The Configurable Serial Peripheral Interface (CSPI) module is a full-duplex, synchronous, four-wire serial communication module. The CSPI module contains an 8 × 32 receive buffer (RXFIFO) and an 8 × 32 transmit buffer (TXFIFO). With data FIFOs, the CSPI module allows rapid data communication with fewer software interrupts. Figure 18-1 shows a block diagram of the CSPI.

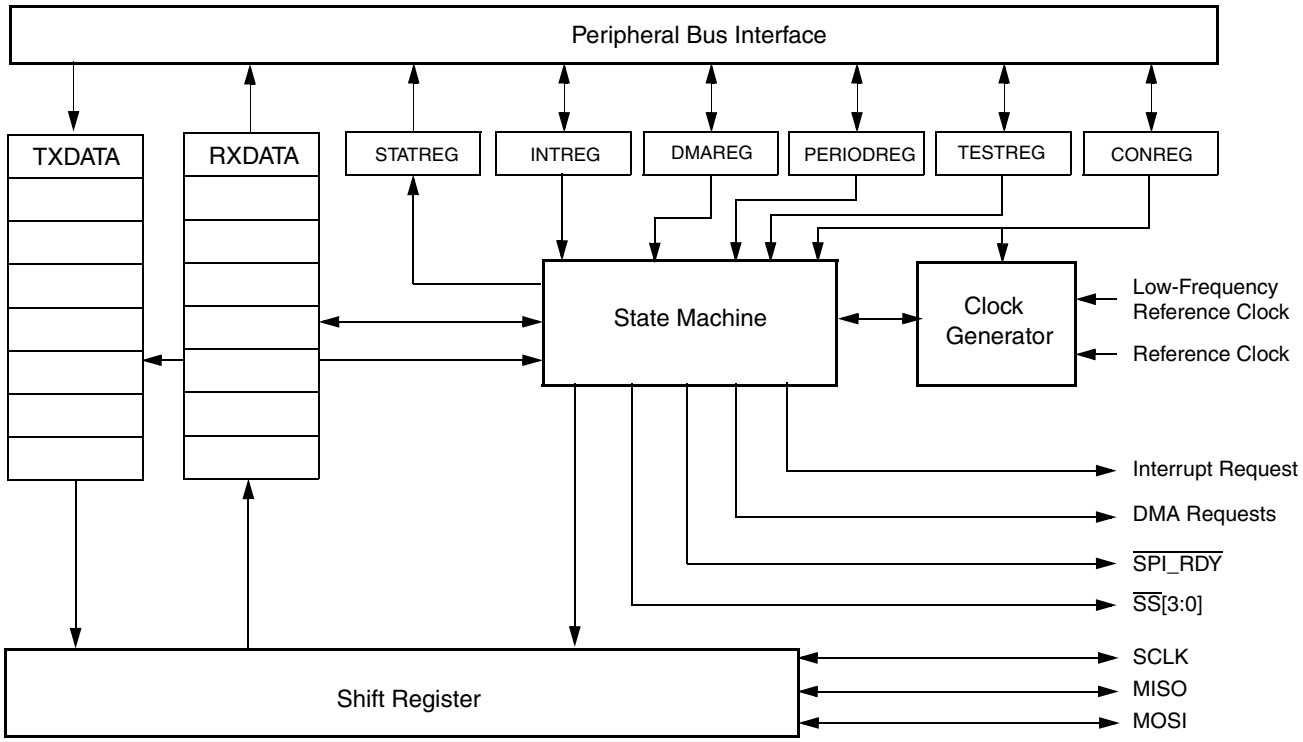


Figure 18-1. CSPI Block Diagram

18.1.1 Features

Key features of the CSPI module include:

- Full-duplex synchronous serial interface

- Master/Slave configurable
- Four Chip Select (SS) signals to support multiple peripherals
- Transfer continuation function allows unlimited length data transfers
- 32-bit wide by 8-entry FIFO for both transmit and receive data
- Polarity and phase of the Chip Select (SS) and SPI Clock (SCLK) are configurable
- Direct Memory Access (DMA) support
- Max operation frequency up to one-quarter of the reference clock frequency.

18.1.2 Modes and Operations

The CSPI supports the modes described in the indicated sections:

- Section 18.4.1, “Operating Modes”:
  - Section 18.4.1.1, “Master Mode”
  - Section 18.4.1.2, “Slave Mode”
- Section 18.4.2, “Low Power Modes”

As described in Section 18.4.3, “Operations,” the CSPI supports the operations described in the indicated sections:

- Section 18.4.3.1, “Typical Master Mode”:
  - Section 18.4.3.1.1, “Master Mode with SPI_RDY”
  - Section 18.4.3.1.2, “Master Mode with Wait States”
  - Section 18.4.3.1.3, “Master Mode with SSCTL Control”
  - Section 18.4.3.1.4, “Master Mode with Phase Control”
- Section 18.4.3.2, “Typical Slave Mode”

18.2 External Signals

Conventions: Table 18-1 lists conventions for representing signals.

Table 18-1. Module Signal Conventions

Category	Convention	Example(s)
Off-chip signal	Uppercase (all capital letters)	TXD
Internal signal[1]	Lowercase italics	*core_int*
Active low signal	_B (_b) suffix or overbar	RESET_EN_B or $\overline{\text{RESET\_EN}}$
Range of bussed or commonly named signals	Beginning and end points of the range are: • Separated by a colon. • Surrounded by square brackets.	ADDR[31:0] CSE_B[7:0] or $\overline{\text{CSE}}$[7:0]
Individual signal in a range of bussed or commonly named signals	Individual number in the range appears without a colon or square brackets	ADDR31 CSE0_B or $\overline{\text{CSE0}}$

[1] Internal signals are for reference only in descriptions of internal module or SoC functionality.

Table 18-2 describes all CSPI signals that connect off-chip.

Table 18-2. Off-Chip Module Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/ Down[1]
$\overline{SS}$[3:0]	I/O	Chip selects	1	—
SCLK	I/O	SPI clock	0	Active
MISO	I/O	Master data in; slave data out	0	Passive
MOSI	I/O	Master data out; slave data in	0	—
$\overline{SPI\_RDY}$	I	Master data out; slave data in	0	Active

1 The reset state values and pull-up/down requirements provided in this table are from the module-level perspective. To understand how the module is integrated at the SoC level, the system software developer must see discussions of the module in the appropriate SoC-level chapter(s). For example, a module signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

Figure 18-2 shows the CSPI module in master mode connected to four external devices in a one-way communication link.

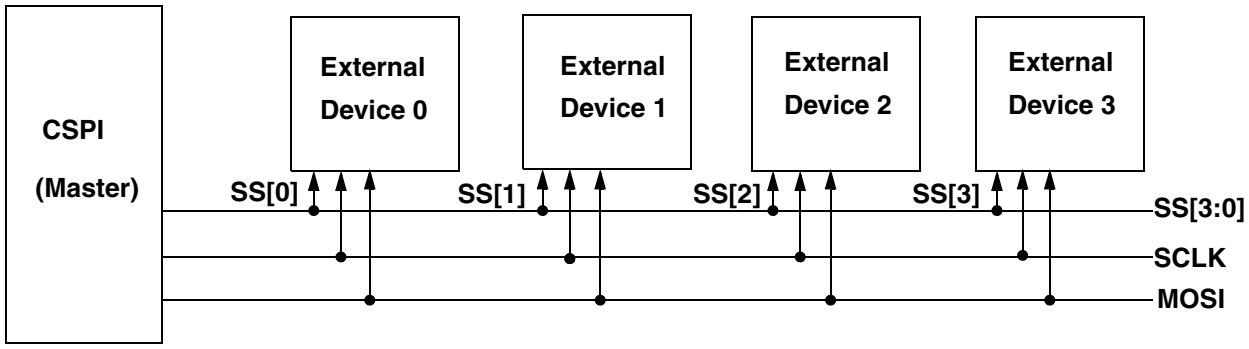


Figure 18-2. Example Connection Diagram

18.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers. For the base address of a particular module instantiation, see the system memory map.

18.3.1 Memory Map

Table 18-3 is the module memory map.

Table 18-3. Module Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
General Registers				
0x0000 (RXDATA)	Receive Data Register (RXDATA)	R	0x0000_0000	18.3.3.1/18-6

Table 18-3. Module Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0004 (TXDATA)	Transmit Data Register (TXDATA)	W	0x0000_0000	18.3.3.2/18-7
0x0008 (CONREG)	Control Register (CONREG)	R/W	0x0000_0000	18.3.3.3/18-7
0x000C (INTREG)	Interrupt Control Register (INTREG)	R/W	0x0000_0000	18.3.3.4/18-10
0x0010 (DMAREG)	DMA Control Register (DMAREG)	R/W	0x0000_0000	18.3.3.5/18-11
0x0014 (STATREG)	Status Register (STATREG)	R/W	0x0000_0003	18.3.3.6/18-12
0x0018 (PERIODREG)	Sample Period Control Register (PERIODREG)	R/W	0x0000_0000	18.3.3.7/18-13
0x001C (TESTREG)	Test Control Register (TESTREG)	R/W	0x0000_0000	18.3.3.8/18-14

18.3.2 Register Summary

Table 18-4 is the register summary table.

Table 18-4. Module Register Summary

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (RXDATA)	R	RXDATA[31:16]															
	W																
	R	RXDATA[15:0]															
	W																
0x0004 (TXDATA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TXDATA[31:16]															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TXDATA[15:0]															
0x0008 (CONREG)	R	BURST LENGTH												0	DATA RATE		
	W																
	R	0	0	CHIP SELECT		0	0	DRCTL		SSP OL	SSC TL	PHA	POL	SMC	XCH	MO DE	EN
	W																
0x000C (INTREG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	TCE N	ROE N	RFE N	RHE N	RRE N	TFE N	THE N	TEE N
	W																

Table 18-4. Module Register Summary (continued)

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010 (DMAREG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	RF DEN	RH DEN	0	0	TH DEN	TE DEN
	W																
0x0014 (STATREG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	TC	RO	RF	RH	RR	TF	TH	TE
	W									w1c							
0x0018 (PERIODREG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CSRC	SAMPLE PERIOD[14:0]														
	W																
0x001C (TESTREG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SWAP	LBC	0	0	—				RXCNT				TXCNT			
	W																

18.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 18-3 and Table 18-5 explain conventions used in register diagrams and tables.

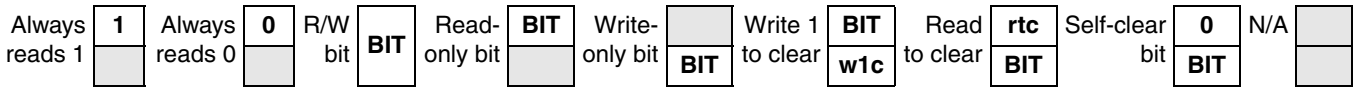


Figure 18-3. Register Field Conventions

Table 18-5. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.

Table 18-5. General Register Conventions (continued)

Convention	Description
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

18.3.3.1 Receive Data Register (RXDATA)

The Receive Data register (RXDATA) is a read-only register that forms the top word of the 8 × 32 receive FIFO. This register holds the data received from an external SPI device during a data transaction. Only word-sized read operations are allowed. Figure 18-4 shows the register. Table 18-6 describes the register fields.

Offset 0x0000 (RXDATA) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXDATA[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDATA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-4. RXDATA Register Diagram

Table 18-6. RXDATA Register Field Description

Field	Description
31–0 RXDATA	Receive Data. This register holds the top word of the receive data FIFO. The FIFO is advanced for each read of this register. The data read is undefined when the Receive Data Ready (RR) bit in the Interrupt Control/Status register is cleared. Zeros are read when CSPI is disabled.

18.3.3.2 Transmit Data Register (TXDATA)

The Transmit Data (TXDATA) register is a write-only data register that forms the bottom word of the 8 × 32 TXFIFO. The TXFIFO can be written to as long as it is not full, even when the SPI Exchange bit (XCH) in CONREG is set. This allows software to write to the TXFIFO during a SPI data exchange process. Writes to this register are ignored when the CSPI module is disabled (CONREG[EN] bit is cleared). Figure 18-5 shows the register. Table 18-7 describes the register fields.

Offset 0x0004 (TXDATA) Access: User write-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TXDATA[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TXDATA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-5. TXDATA Register Diagram

Table 18-7. TXDATA Register Field Description

Field	Description
31–0 TXDATA	Transmit Data. This register holds the top word of data loaded into the FIFO. Data written to this register must be a word operation. The number of bits actually transmitted is determined by the BURST_LENGTH field of the corresponding SPI Control register. If this field contains more bits than the number specified by BURST_LENGTH, the extra bits are ignored. For example, to transfer 10 bits of data, a 32-bit word must be written to this register. Bits 9-0 are shifted out and bits 31-10 are ignored. When the CSPI module is operating in Slave mode, zeros are shifted out when the FIFO is empty. Zeros are read when CSPI is disabled.

18.3.3.3 Control Register (CONREG)

The Control Register (CONREG) allows software to enable the CSPI module, configure its operating modes, specify the divider value, phase, and polarity of the clock, configure the Chip Select (SS) and SPI_RDY control signal, and define the transfer length.

Figure 18-6 shows the register. Table 18-8 describes the register fields.

Offset 0x0008 (CONREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BURST LENGTH												0	DATA RATE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	CHIP SELECT		0	0	DRCTL		SSPOL	SSCTL	PHA	POL	SMC	XCH	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-6. CONREG Register Diagram

Table 18-8. CONREG Register Field Description

Field	Description
31–20 BURST LENGTH	Burst Length. This field defines the length of a SPI burst to be transferred. The Chip Select (SS) will remain asserted until all bits in a SPI burst are shifted out. A maximum of 2^12 bits can be transferred in a single SPI burst. In master mode, it controls the number of bits per SPI burst. Since the shift register always loads 32-bit data from transmit FIFO, only the n least-significant (n = BURST LENGTH + 1) will be shifted out. The remaining bits will be ignored. In slave mode, only when SSCTL is cleared, this field will take effect in the transfer. Number of Valid Bits in a SPI burst. 0x000 A SPI burst contains the 1 LSB in a word. 0x001 A SPI burst contains the 2 LSB in a word. 0x002 A SPI burst contains the 3 LSB in a word. ..... 0x01F A SPI burst contains all 32 bits in a word. 0x020 A SPI burst contains the 1 LSB in first word and all 32 bits in second word. 0x021 A SPI burst contains the 2 LSB in first word and all 32 bits in second word. ..... 0xFFE A SPI burst contains the 31 LSB in first word and 2^7 -1 words. 0xFFF A SPI burst contains 2^7 words.
19	Reserved
18–16 DATA RATE	SPI Data Rate Control. This field selects the baud rate of the SCLK based on a division of the reference clock. These bits allow the CSPI to synchronize with different external SPI devices. The max frequency is one quarter of the reference clock. The divide ratio is determined according to the following table using the equation: $2^{(n+2)}$. SPI Data Rate Control (Master Mode only) 000 Divide by 4. 001 Divide by 8. 010 Divide by 16. 011 Divide by 32. 100 Divide by 64. 101 Divide by 128. 110 Divide by 256. 111 Divide by 512.
15 –14	Reserved

Table 18-8. CONREG Register Field Description (continued)

Field	Description
13 –12 CHIP SELECT	CHIP SELECT. Select one of four external SPI Master/Slave Devices. In master mode, these two bits select the external slave devices by asserting the Chip Select (SSn) outputs. Only the selected Chip Select (SSn) signal can be active at a given time; the remaining three signals will be negated. Chip Select 00 Chip Select 0 (SS0) will be asserted. 01 Chip Select 1 (SS1) will be asserted. 10 Chip Select 2 (SS2) will be asserted. 11 Chip Select 3 (SS3) will be asserted.
11–10	Reserved
9–8 DRCTL	SPI Data Ready Control. This field selects the utilization of the $\overline{\text{SPI\_RDY}}$ signal in master mode. CSPI checks this field before it starts an SPI burst. 00 The $\overline{\text{SPI\_RDY}}$ signal is a don't care. 01 Burst will be triggered by the falling edge of the SPI_RDY signal (edge-triggered). 10 Burst will be triggered by a low level of the SPI_RDY signal (level-triggered). 11 Reserved.
7 SSPOL	SPI SS Polarity Select. In both Master and Slave modes, this bit selects the polarity of the Chip Select (SS) signal. 0 Active low. 1 Active high.
6 SSCTL	SPI SS Wave Form Select. In master mode, this bit controls the output wave form of the Chip Select (SS) signal when SMC is cleared. The SSCTL bit will be ignored if the SMC (Start Mode Control) bit is set. 0 Only one SPI burst will be transmitted. 1 Negate Chip Select (SS) signal between SPI bursts. Multiple SPI bursts will be transmitted. The SPI transfer will automatically stop when the TXFIFO is empty. In slave mode, this bit controls when the SPI burst is completed. 0 A SPI burst is completed when the number of bits received in the shift register is equal to BURST LENGTH + 1. Only n least-significant bits (n = BURST LENGTH[4:0] + 1) of the first received word are valid. All bits subsequent to the first received word in RXFIFO are valid. 1 A SPI burst is completed by the Chip Select (SS) signal edges. (SSPOL = 0: rising edge; SSPOL = 1: falling edge) The RXFIFO is advanced whenever a Chip Select (SS) signal edge is detected or the shift register contains 32-bits of valid data.
5 PHA	SPI Clock/Data Phase Control. This bit controls the clock/data phase relationship. See Figure 18-19 for more information. 0 Phase 0 operation. 1 Phase 1 operation.
4 POL	SPI Clock Polarity Control. This bit controls the polarity of the SCLK signal. See Figure 18-19 for more information. 0 Active high polarity (0 = Idle). 1 Active low polarity (1 = Idle).
3 SMC	Start Mode Control. This bit applies only when the module is configured in Master mode (MODE = 1). It controls how the CSPI starts a SPI burst, either through the SPI exchange bit, or immediately when the TXFIFO is written to. 0 SPI Exchange Bit (XCH) controls when a SPI burst can start. Setting the XCH bit will start a SPI burst or multiple bursts. This is controlled by the SPI SS Wave Form Select (SSCTL) bit. Refer to XCH and SSCTL bit descriptions. 1 Immediately starts a SPI burst when data is written in TXFIFO.

Table 18-8. CONREG Register Field Description (continued)

Field	Description
2 XCH	SPI Exchange Bit. This bit applies only when the module is configured in Master mode (MODE = 1). If the Start Mode Control (SMC) bit is cleared, writing a 1 to this bit starts one SPI burst or multiple SPI bursts according to the SPI SS Wave Form Select (SSCTL bit). The XCH bit remains set while either the data exchange is in progress, or when the CSPI is waiting for an active input if SPIRDY is enabled through DRCTL. This bit is cleared automatically when all data in the TXFIFO and the shift register has been shifted out. 0 Idle. 1 Initiates exchange (write) or busy (read).
1 MODE	SPI Function Mode Select. This bit selects the operating mode of the CSPI. 0 Slave mode. 1 Master mode.
0 EN	SPI Module Enable Control. This bit enables the CSPI module. This bit must be set before writing to other registers or initiating an exchange. Writing zero to this bit disables the module and resets the internal logic with the exception of the CONREG. The module's internal clocks are gated off whenever the module is disabled. 0 Disable the module. 1 Enable the module.

18.3.3.4 Interrupt Control Register (INTREG)

The Interrupt Control Register (INTREG) enables the generation of interrupts to the host processor. If the CSPI is disabled, this register reads zero. Figure 18-7 shows the register. Table 18-9 describes the register fields.

Offset 0x000C (INTREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	TCEN	ROEN	RFEN	RHEN	RREN	TFEN	THEN	TEEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-7. INTREG Register Diagram

Table 18-9. INTREG Register Field Description

Field	Description
31–8	Reserved
7 TCEN	Transfer Completed Interrupt enable. This bit enables the Transfer Completed Interrupt. 0 Disable 1 Enable
6 ROEN	RXFIFO Overflow Interrupt enable. This bit enables the RXFIFO Overflow Interrupt. 0 Disable 1 Enable

Table 18-9. INTREG Register Field Description (continued)

Field	Description
5 RFEN	RXFIFO Full Interrupt enable. This bit enables the RXFIFO Full Interrupt. 0 Disable 1 Enable
4 RHEN	RXFIFO Half Full Interrupt enable. This bit enables the RXFIFO Half Full Interrupt. 0 Disable 1 Enable
3 RREN	RXFIFO Ready Interrupt enable. This bit enables the RXFIFO Ready Interrupt. 0 Disable 1 Enable
2 TFEN	TXFIFO Full Interrupt enable. This bit enables the TXFIFO Full Interrupt. 0 Disable 1 Enable
1 THEN	TXFIFO Half Empty Interrupt enable. This bit enables the TXFIFO Half Empty Interrupt. 0 Disable 1 Enable
0 TEEN	TXFIFO Empty Interrupt enable. This bit enables the TXFIFO Empty Interrupt. 0 Disable 1 Enable

18.3.3.5 DMA Control Register (DMAREG)

The Direct Memory Access Control Register (DMAREG) provides software a way to use an on-chip DMA controller for CSPI data. Internal DMA request signals enable direct data transfers between the CSPI FIFOs and system memory. The CSPI sends out DMA requests when the appropriate FIFO conditions are matched.

If the CSPI is disabled, this register is read as 0. Figure 18-8 shows the register. Table 18-10 describes the register fields.

Offset 0x0010 (DMAREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	RF DEN	RH DEN	0	0	TH DEN	TE DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-8. DMAREG Register Diagram

Table 18-10. DMAREG Register Field Description

Field	Description
31–6	Reserved
5 RFDEN	RXFIFO Full DMA Request Enable. This bit enables/disables the RXFIFO Full DMA Request. 0 Disable 1 Enable
4 RHDEN	RXFIFO Half Full DMA Request Enable. This bit enables/disables the RXFIFO Half Full DMA Request. 0 Disable 1 Enable
3–2	Reserved
1 THDEN	TXFIFO Half Empty DMA Request Enable. This bit enables/disables the TXFIFO Half Empty DMA Request. 0 Disable 1 Enable
0 TEDEN	TXFIFO Empty DMA Request Enable. This bit enables/disables the TXFIFO Empty DMA Request. 0 Disable 1 Enable

18.3.3.6 Status Register (STATREG)

The CSPI Status Register (STATREG) reflects the status of the CSPI module's operating condition. If the CSPI is disabled, this register reads 0x0000_0003. Figure 18-9 shows the register. Table 18-11 describes the register fields.

Offset 0x0014 (STATREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	TC	RO	RF	RH	RR	TF	TH	TE
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 18-9. STATREG Register Diagram

Table 18-11. STATREG Register Field Description

Field	Description
31–8	Reserved
7 TC	Transfer Completed Status bit. Writing 1 to this bit clears it. 0 Transfer in progress. 1 Transfer completed.

Table 18-11. STATREG Register Field Description (continued)

Field	Description
6 RO	RXFIFO Overflow. When set, this bit indicates that RXFIFO has overflowed. 0 RXFIFO has no overflow. 1 RXFIFO has overflowed.
5 RF	RXFIFO Full. This bit is set when the RXFIFO is full. 0 Not Full. 1 Full.
4 RH	RXFIFO Half Full. This bit is set when the RXFIFO reaches half full. 0 Less than 4 words are stored in RXFIFO. 1 Four or more words are available in RXFIFO.
3 RR	RXFIFO Ready. This bit is set when one or more words are stored in the RXFIFO. 0 No valid data in RXFIFO. 1 More than 1 word in RXFIFO.
2 TF	TXFIFO Full. This bit is set when if the TXFIFO is full. 0 TXFIFO is not Full. 1 TXFIFO is Full.
1 TH	TXFIFO Half empty. This bit is set when the TXFIFO reaches half empty. 0 TXFIFO holds more than 4 words. 1 TXFIFO holds 4 or fewer words.
0 TE	TXFIFO Empty. This bit is set if the TXFIFO is empty. 0 TXFIFO contains one or more words. 1 TXFIFO is empty.

18.3.3.7 Sample Period Control Register (PERIODREG)

The Sample Period Control Register (PERIODREG) provides software a way to insert delays (wait states) between consecutive SPI transfers. Control bits in this register select the clock source for the sample period counter and the delay count indicating the number of wait states to be inserted between data transfers.

The delay counts apply only when the module is configured in Master mode (CONREG[MODE] = 1).

Figure 18-10 shows the register. Table 18-12 describes the register fields.

Offset 0x0018 (PERIODREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CSRC	SAMPLE PERIOD														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-10. PERIODREG Register Diagram

Table 18-12. PERIODREG Register Field Description

Field	Description
31–16	Reserved
15 CSRC	Clock Source Control. This bit selects the clock source for the sample period counter. 0 SPI Clock (SCLK) 1 Low-Frequency Reference Clock (32.768 KHz)
14–0 SAMPLE PERIOD	Sample Period Control. These bits control the number of wait states to be inserted in data transfers. During the idle clocks, the state of the SS output will operate according to the SSCTL control field in the CONREG register. 0x0000 0 wait states inserted 0x0001 1 wait state inserted .... 0x7FFE 32766 wait states inserted 0x7FFF 32767 wait states inserted

18.3.3.8 Test Control Register (TESTREG)

The Test Control Register (TESTREG) provides software a mechanism to internally connect the receive and transmit devices of the CSPI module, swap the byte order for receive data, and monitor the contents of the receive and transmit FIFO.

Figure 18-11 shows the register. Table 18-13 describes the register fields.

Offset 0x001C (TESTREG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWAP	LBC	0	0	—				RXCNT				TXCNT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 18-11. TESTREG Register Diagram

Table 18-13. TESTREG Register Field Description

Field	Description
31–16	Reserved
15 SWAP	Data Swap. This bit is used to swap data as it is read from the RXFIFO. When this bit is set, data read from RXFIFO is swapped. RXDATA[31:0] is swapped as follows: {RXDATA[7:0], RXDATA[15:8], RXDATA[23:16], RXDATA[31:24]} 0 Data read from RXFIFO is unchanged. 1 Data read from RXFIFO is swapped.

Table 18-13. TESTREG Register Field Description (continued)

Field	Description
14 LBC	Loop Back Control. This bit is used in Master mode only. When this bit is set, the CSPI module connects the transmitter and receiver sections internally, and the data shifted out from the most-significant bit of the shift register is looped back into the least-significant bit of the Shift register. In this way, a self-test of the complete transmit/receive path can be made. The output pins are not affected, and the input pins are ignored. 0 Not connected. 1 Transmitter and receiver sections internally connected for Loopback.
13–12	Reserved
11–8	Reserved
7–4 RXCNT	RXFIFO Counter. These bits indicate the number of words in RXFIFO. RXFIFO Counter 0000 0 word in RXFIFO 0001 1 word in RXFIFO .... 0111 7 words in RXFIFO 1000 8 words in RXFIFO
3–0 TXCNT	TXFIFO Counter. These bits indicate the number of words in TXFIFO. TXFIFO Counter 0000 0 word in TXFIFO 0001 1 word in TXFIFO .... 0111 7 words in TXFIFO 1000 8 words in TXFIFO

18.4 Functional Description

This section provides a complete functional description of the CSPI. Figure 18-12 shows the relationship of SCLK and data lines while CSPI has been configured with different POL and PHA settings.

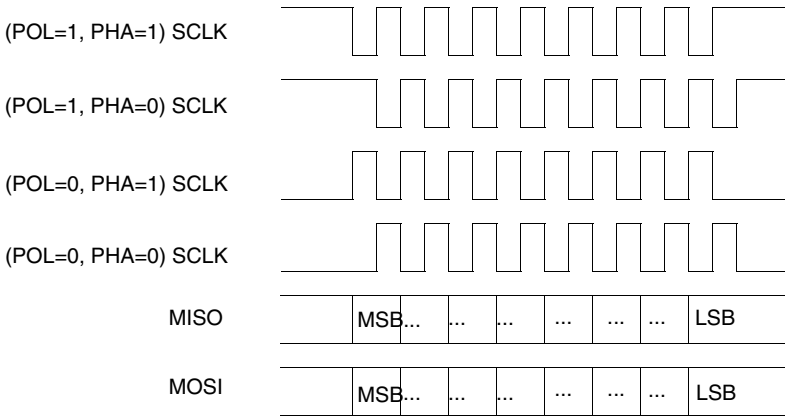


Figure 18-12. CSPI SCLK, MISO, and MOSI Relationship

18.4.1 Operating Modes

CSPI has two operating modes, master mode and slave mode. This section describes all functional operation modes of the CSPI.

18.4.1.1 Master Mode

When the CSPI module is configured as a master, it uses a serial link to transfer data between the CSPI and an external slave device. One of the Chip Select (SS) signals and the clock signal (SCLK) are used to transfer data between two devices. If the external device is a transmit-only device, the CSPI master's output port can be ignored and used for other purposes. In order to use the internal TXFIFO and RXFIFO, two auxiliary output signals, Chip Select (SS) and SPI_RDY, are used for data transfer rate control. Software can also configure the sample period control register to a fixed data transfer rate.

18.4.1.2 Slave Mode

When the CSPI module is configured as a slave, software can configure the CSPI Control register to match the external SPI master's timing. In this configuration, Chip Select ($\overline{SS}$) becomes an input signal, and is used to control data transfers through the Shift register, as well as to load/store the data FIFO.

18.4.2 Low Power Modes

The CSPI module does not operate under low power mode. It holds its operation when its clock is gated off in master mode. In slave mode, the CSPI module does not respond when its clock is gated off.

18.4.3 Operations

This section describes the CSPI's operations.

18.4.3.1 Typical Master Mode

The CSPI master uses the Chip Select (SS) signal to enable an external SPI device, and uses the SCLK signal to transfer data in and out of the Shift register. The SPI_RDY enables fast data communication with fewer software interrupts. By programming the PERIODREG register accordingly, the CSPI can be used for a fixed data transfer rate.

When the CSPI module is in Master mode the SS, SCLK, and MOSI are output signals, and the MISO signal is an input. Figure 18-13 shows a typical SPI burst.

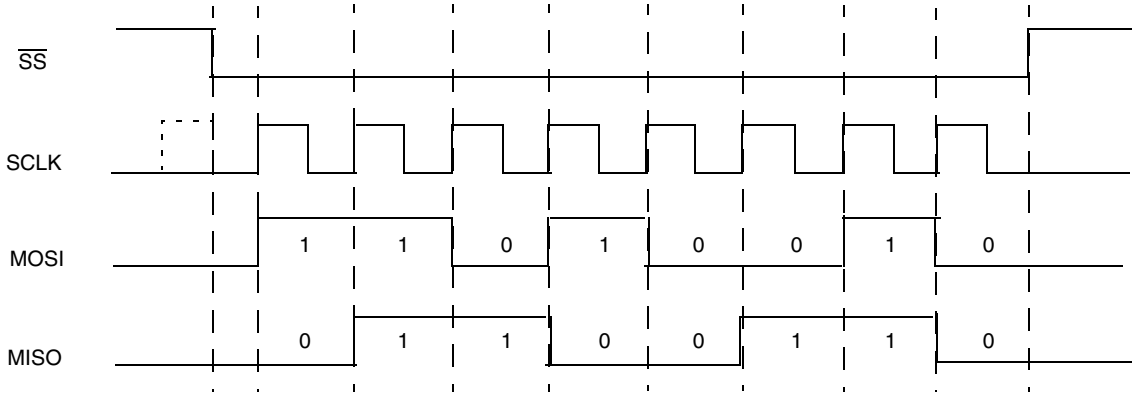


Figure 18-13. Typical SPI Burst (8-bit Transfer)

In Figure 18-13, the Chip Select (SS) signal enables the selected external SPI device, and the SCLK synchronizes the data transfer. The MOSI and MISO signals change on rising edge of SCLK and the MISO signal is latched on the falling edge of the SCLK. Figure 18-13 shows a data of 0xD2 is shifted out, and a data of 0x66 is shifted in.

18.4.3.1.1 Master Mode with SPI_RDY

By default, the CSPI does not use the SPI_RDY signal in master mode (MODE =1). A SPI burst begins when the following events happen:

- The CSPI is enabled, TXFIFO has data in it, and CONREG[XCH] bit or the CONREG[SMC] bit is set.
- When the SPI Data Ready Control (CONREG[DRCTL]) bits contains either 01 or 10, the SPI_RDY signal controls when a SPI burst starts.

A SPI burst is defined as a bus transaction that starts when the slave select is asserted and ends when the slave select is negated. The Chip Select (SS) signal will remain asserted until all the bits in a SPI burst are shifted out.

If CONREG[DRCTL] is set to 01, the SPI burst can be triggered only if a falling edge of the SPI_RDY signal has been detected. Figure 18-14 shows the relationship between a SPI burst and the falling edge of SPI_RDY signal.

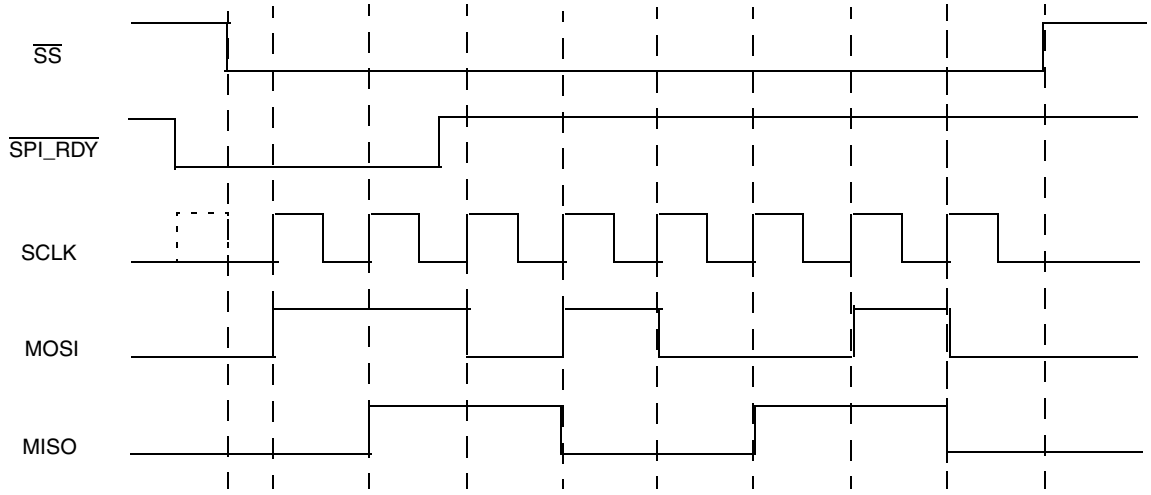


Figure 18-14. Relationship Between a SPI Burst and SPI_RDY: Falling-Edge Triggered

A SPI burst does not start until the falling edge of the SPI_RDY signal is detected. The next SPI burst starts when the next SPI_RDY falling edge is detected, after the last burst has finished.

If SPI Data Ready Control (CONREG[DRCTL]) is set to 10, the SPI burst can be triggered only if the SPI_RDY signal is low. Figure 18-15 shows the relationship between a SPI burst and the SPI_RDY signal. The SPI burst does not begin until the SPI_RDY signal goes low. The CSPI will keep transmitting SPI burst if the SPI_RDY signal remains low.

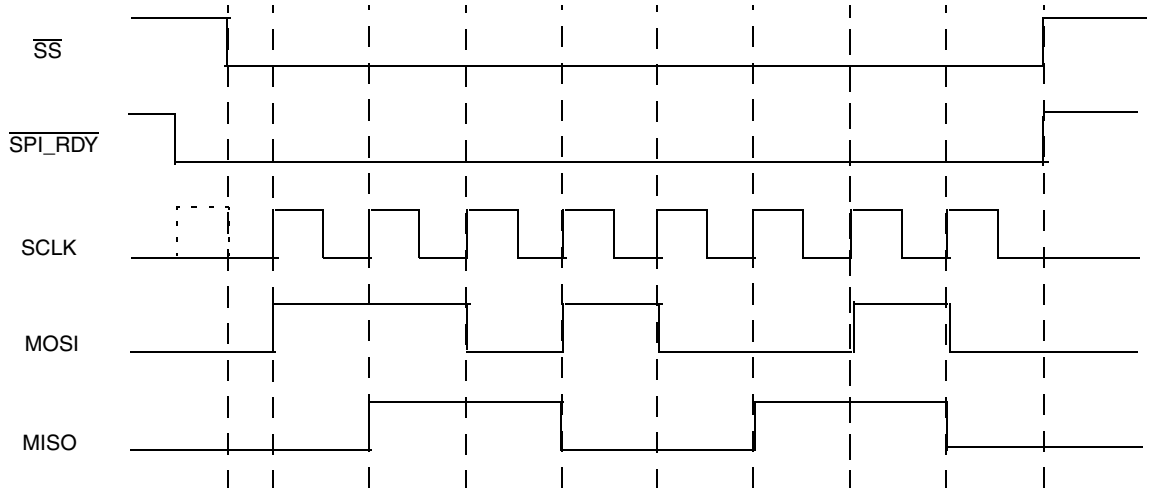


Figure 18-15. Relationship Between a SPI Burst and SPI_RDY: Low-Level Triggered

18.4.3.1.2 Master Mode with Wait States

Wait states can be inserted between SPI bursts. This provides a way for software to slow down the SPI burst to meet the timing requirements of a slower SPI device. Figure 18-16 shows wait states inserted between SPI bursts.

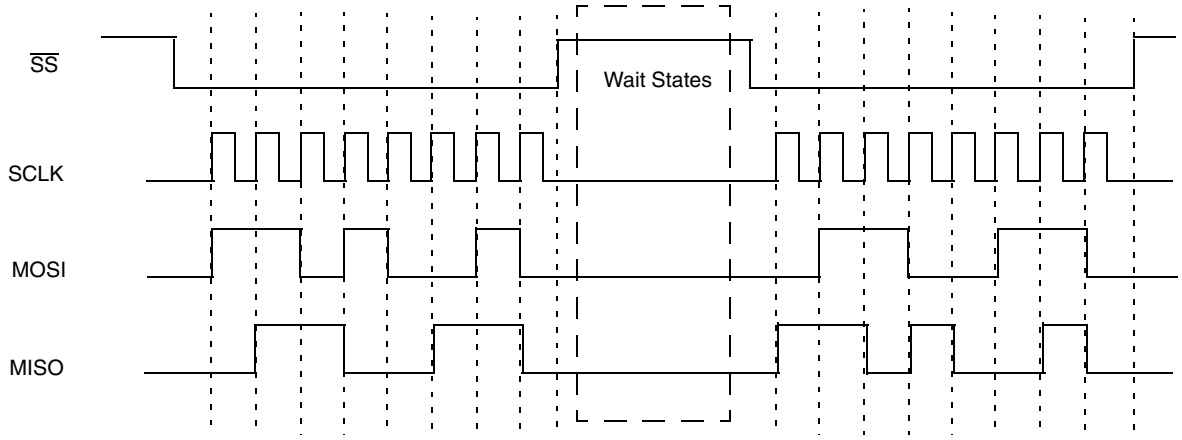


Figure 18-16. SPI Bursts with Wait States

In this case, the number of wait states is controlled by PERIODREG[SAMPLE PERIOD] and the wait states' clock source is selected by PERIODREG[CSRC].

18.4.3.1.3 Master Mode with SSCTL Control

The SPI SS Control (SSCTL) bit controls whether the current operation is single burst or multiple bursts. When the SPI SS Wave Form Select (SSCTL) bit is set, the current operation is multiple bursts transfer. When the SPI SS Wave Form Select (SSCTL) bit is cleared, the current operation is single burst transfer. A SPI burst can contains multiple words as defined in the BURST LENGTH field of the CONREG register.

Figure 18-17 shows one SPI burst while SSCTL is clear.

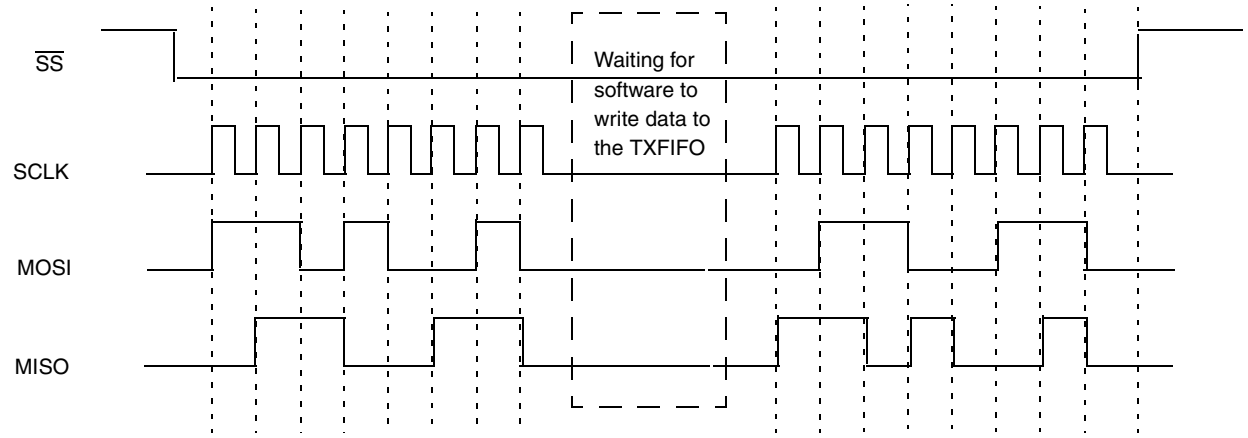


Figure 18-17. SPI Burst While SSCTL is Clear

In Figure 18-17, two 8-bit bursts in the TXFIFO have been combined and transmitted in one SPI burst. The maximum length of a single SPI burst is defined in the BURST LENGTH field of the CONREG control register. (Figure 18-17 corresponds to a BURST LENGTH of 8.) This provides a way for transferring a longer SPI burst by writing data into TXFIFO while the CSPI is transmitting.

Figure 18-18 shows two SPI bursts are transmitted while SSCTL is set.

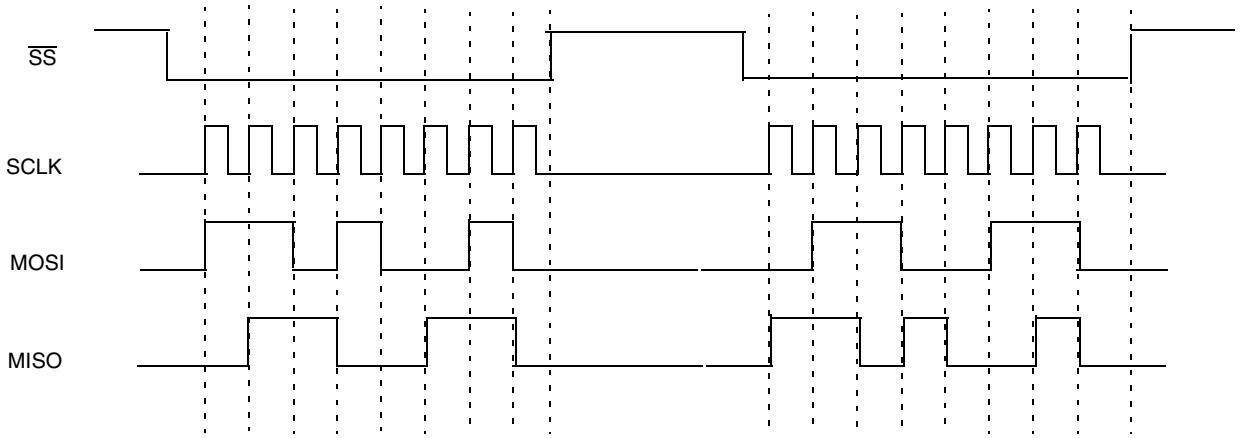


Figure 18-18. SPI Bursts While SSCTL is Set

In Figure 18-18, two FIFO entries are transmitted, one entry with each SPI burst. The CSPI will continue to transmit SPI bursts until the TXFIFO is empty. When wait states can be inserted between SPI bursts, the SS will negate between SPI bursts until the wait states finish.

18.4.3.1.4 Master Mode with Phase Control

The Phase Control (CONREG[PHA]) bit controls how the transmit data shifts out and the receive data shifts in.

When the Phase control (CONREG[PHA]) bit is set, the transmit data will shift out on the rising edge of SCLK, and the receive data is latched on the falling edge of SCLK. The most-significant bit is output on the first rising SCLK edge.

When CONREG[PHA] is cleared, the transmit data is shifted out on the falling edge of SCLK and the receive data is latched on the rising edge of SCLK. The MSB is output when the host processor loads the transmitted data.

Inverting the SCLK polarity does not impact the edge-triggered operations because they are internal to the serial peripheral interface master. Figure 18-19 shows a SPI burst using different POL and PHA configurations.

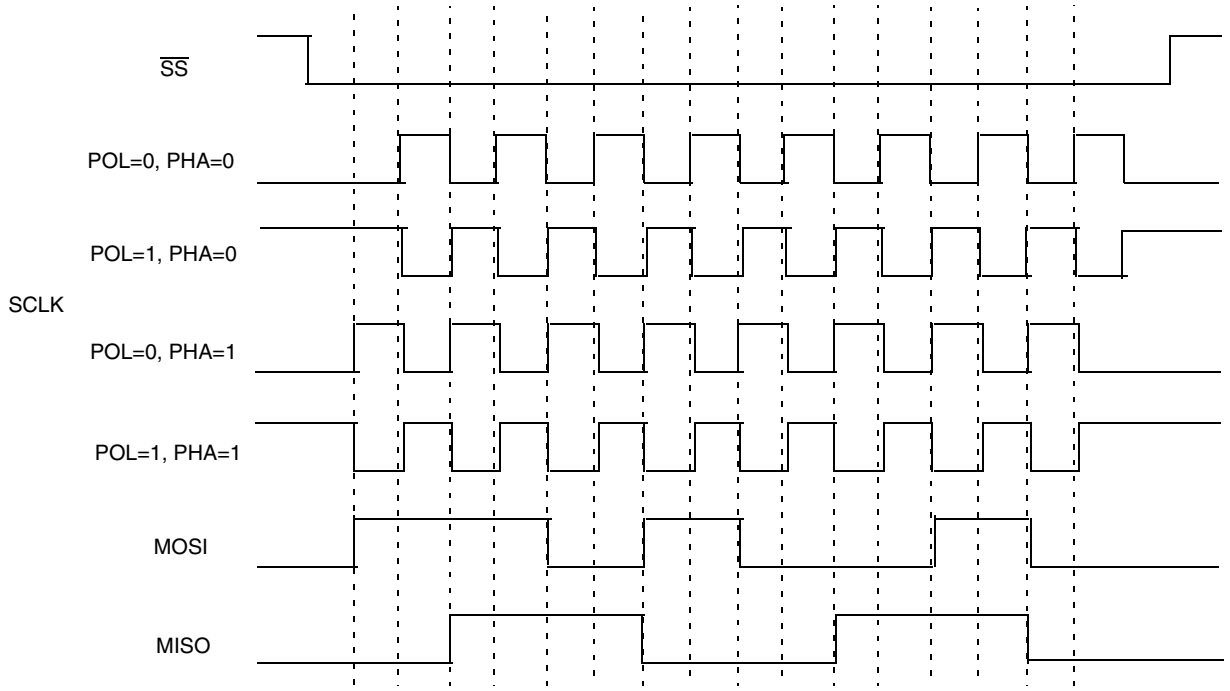


Figure 18-19. SPI Burst with Different POL and PHA Configurations

18.4.3.2 Typical Slave Mode

When the CSPI module is configured as a slave (Mode = 0), software can configure the CSPI Control register to match the external SPI master's timing. In this configuration, SS becomes an input signal, and is used to latch data in and out of the internal data Shift registers, as well as to advance the data FIFO.

The SS, SCLK, and MOSI are inputs and MISO is output. Most of the timing diagrams are similar to the diagrams shown previously for the SPI in Master mode (Mode = 1), because the inputs come from a SPI master device.

However, the timing is different when SS is used to advance the data FIFO. When the SSCTL is set while the CSPI is configured in Slave mode, the data FIFO will advance on the rising edge of the SS signal. When the polarity is reversed (SSPOL = 1), the data FIFO will advance on the falling edge of the SS signal.

Figure 18-20 shows a SPI burst in which the data FIFO is advanced by the rising edge of the SS signal.

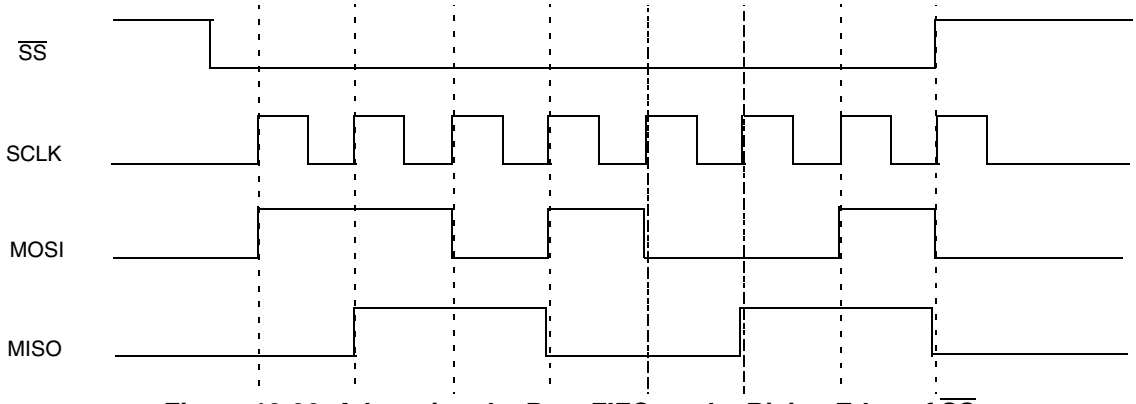


Figure 18-20. Advancing the Data FIFO on the Rising Edge of SS

In the above case, only the most significant 7 bits are loaded to the RXFIFO.

18.4.4 Clocks

This section describes clocks and special clocking requirements of the module.

CSPI has the following clock inputs:

- Reference Clock is the reference clock.
- Low-Frequency Reference Clock is a 32KHz input clock optionally used for counting wait states.

18.4.5 Reset

Whenever a device reset occurs, a reset is performed on the CSPI module, resetting all registers to their default values.

Software can reset the module using the CONREG[EN] bit; see Section 18.3.3.3, "Control Register (CONREG)."

18.4.6 Interrupts

Interrupt control provides a way to manage the CSPI FIFOs:

- For transmitting data, software can enable the *TXFIFO empty*, *TXFIFO half*, and *TXFIFO full* interrupts to maintain the TXFIFO using an interrupt service routine.
- For receiving data, software can enable the *RXFIFO ready*, *RXFIFO half*, and *RXFIFO full* interrupts to retrieve data from the RXFIFO using an interrupt service routine.

Other interrupt sources can be used to control or debug the SPI bursts:

- The *transfer-completed* interrupt means that there is no data left in the TXFIFO and that the data in the Shift register has been shifted out.
- The *RXFIFO overflow* interrupt means that the RXFIFO received more than 8 words and will not accept any other words.

Figure 18-21 shows a program sequence of SPI bursts using interrupt control.

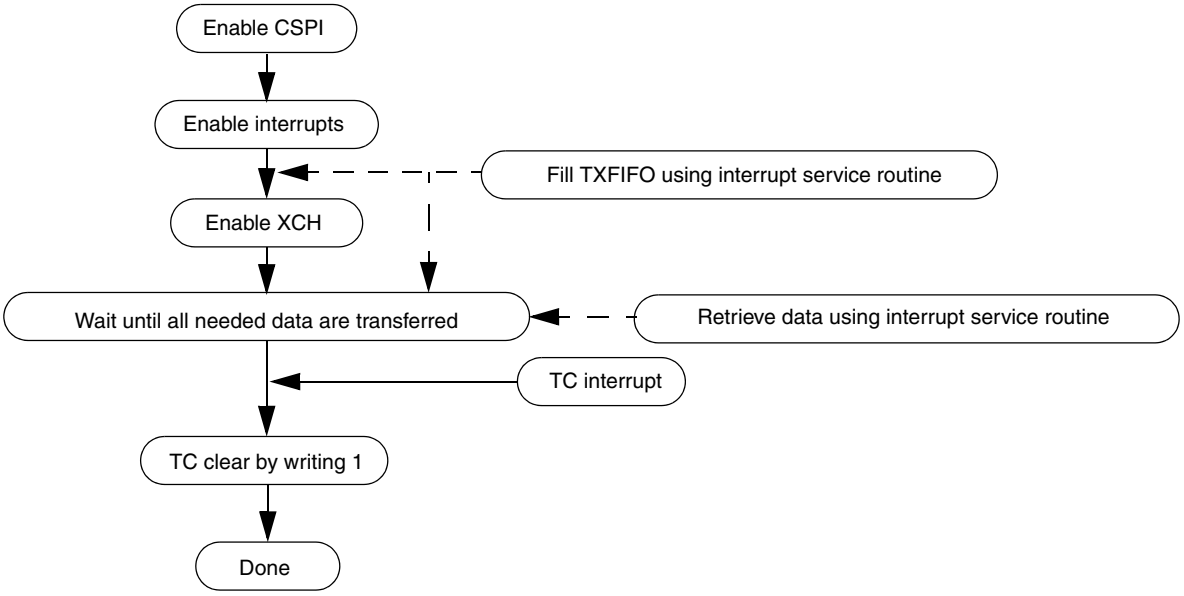


Figure 18-21. Program Sequence of SPI Burst Using Interrupt Control

18.4.7 DMA

DMA control provides another method to utilize the FIFOs in the CSPI module. By using DMA request and acknowledge signals, larger amounts of data can be transferred, and will reduce interrupts and host processor loading. When the appropriate conditions are matched, the module will send out a DMA request, and the DMA can deal with the following conditions:

- TXFIFO empty
- TXFIFO half
- RXFIFO half
- RXFIFO full

Figure 18-22 shows a program sequence of SPI bursts using DMA control.

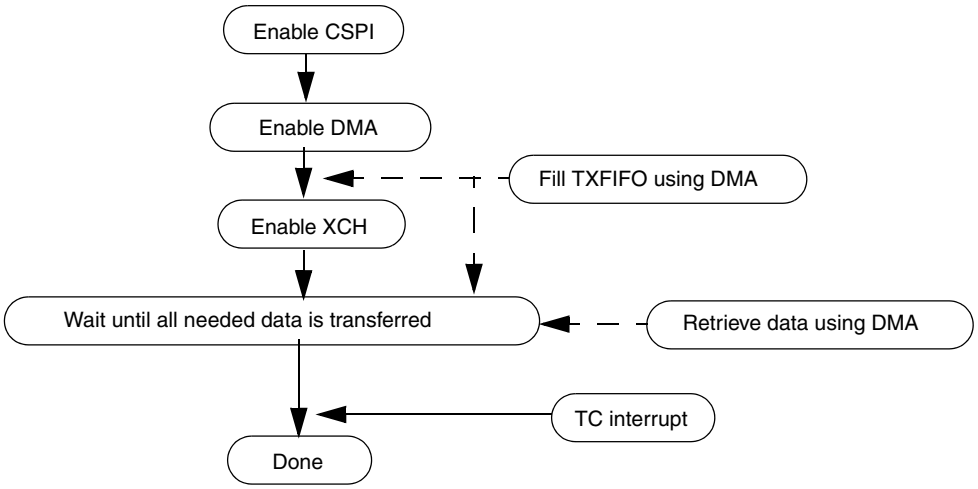


Figure 18-22. Program Sequence of SPI Burst Using DMA

18.4.8 Byte Order

Software can swap bytes for receive data using the TESTREG[SWAP] bit; see Section 18.3.3.8, "Test Control Register (TESTREG)."

18.5 Initialization

This section provides initialization information for CSPI.

To initialize the module:

1. Clear the EN bit in CONREG to reset the module.
2. Enable the clocks for CSPI.
3. Set the EN bit in CONREG to put CSPI out of reset.
4. Configure corresponding IOMUX for CSPI external signals.
5. Configure registers of CSPI properly according to the specifications of the external SPI device.

18.6 Applications

Figure 18-23 shows two flowcharts for the master and slave mode of operations supported by the CSPI module.

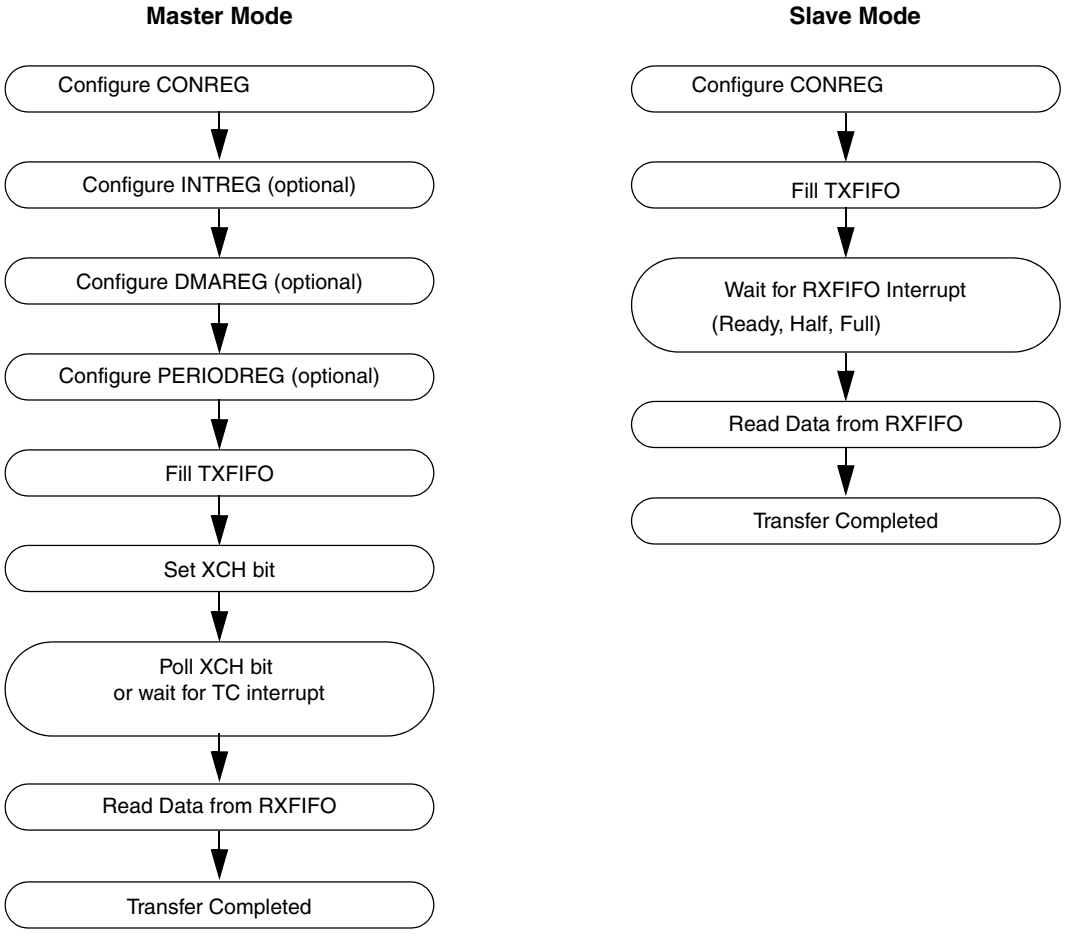


Figure 18-23. Flowchart of the CSPI Operation

Example 18-1 shows example code of CSPI operation using ARM instructions.

Example 18-1. CSPI Operation using ARM Instructions

```
LDR R0, =CSPI_BASE_ADDRESS          ; Load CSPI Base Address to R0
LDR R1, =0x01F00003                 ; Master Mode, 32-bit transaction
STR R1, [R0, #0x08]
LDR R1, =0x00000011                 ; Enable RXFIFO half and TXFIFO empty
STR R1, [R0, #0x0C]                 ; interrupt (Alternatively with DMA Mode)
LDR R1, =0x00000011                 ; Enable RXFIFO half and TXFIFO empty
STR R1, [R0, #0x10]                 ; DMA (Alternatively with interrupt)
LDR R5, =0x05                       ; R5 as number of words to be transferred.
```

```
        LDR R1, =0x11111111             ; R1 as increment to generate the data.
        LDR R2, =0x12345678             ; R2 load the data to be transferred.
Loop_00
        STR R2, [R2,#0x04]              ; Store data into TXFIFO.
        ADD R2, R2, R1                  ; Generating next data to be transferred.
        SUB R5, R5, #1                  ; Decrease the R5.
        CMP R5, #0x00                   ; Check R5 if it is zero.
        BNE Loop_00                     ; Loop until R5 is zero.
        LDR R1, =0x01F00007             ; set XCH bit to start transaction.
        STR R1, [R0, #0x08]
Loop_01
        LDR R1, [R0, #0x08]             ; check XCH bit if it is cleared.
        LDR R2, =0x00000004
        AND R1, R2, R1
        CMP R1, #0x00
        BEQ PASS_00                     ; if XCH bit is cleared then finish.
        B Loop_01                       ; if it isn't cleared then continue loop
        LDR R1, [R0, #0x00]             ; Read data from RXFIFO.
```

Chapter 19
Embedded Cross Trigger (ECT)

19.1 Introduction

Embedded cross trigger (ECT) is a new IP for real-time debug purpose. It is a programmable matrix allowing several sub-systems to interact with each other. ECT receives signals required for debugging purpose (from cores, peripherals, busses, external inputs, etc.) and propagates them (propagation programmed through software) to the different debug resources available within the SoC. Those debug resources are: module with time stamping capability, module with profiling capability, real-time tracer, debug interrupts, muxing at SoC level.

ECT allows sharing of debug resources between Cores and IPs from different domains.

ECT is based on a ARM. IP, delivered in the ETK11 kit. Note that information given into this document are in part extracted from the *Embedded Cross Trigger Technical Reference Manual* [1] provided by ARM

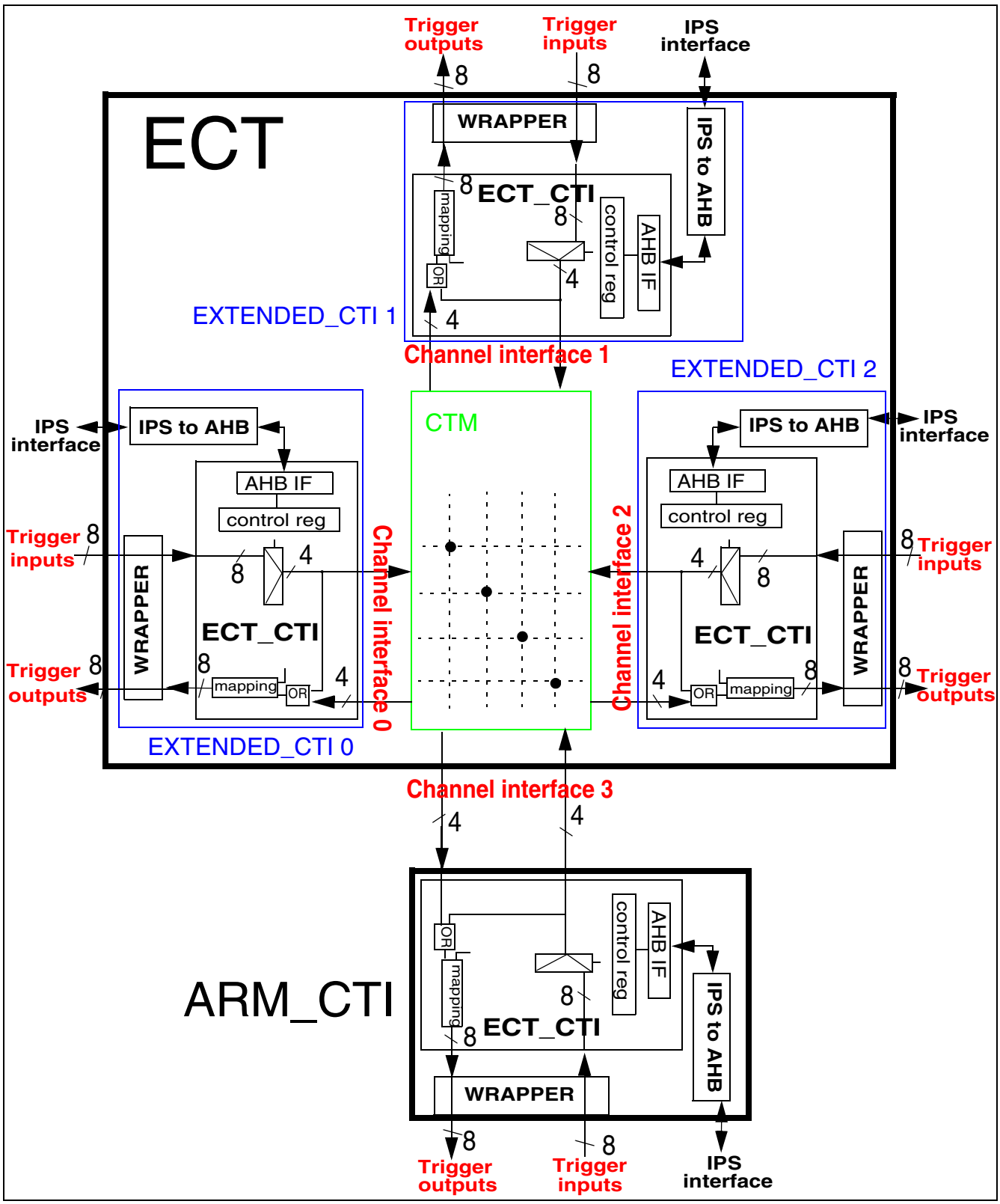


Figure 19-1. ECT Block Diagram

19.2 Overview

ECT is a key module in term of multi-cores and multi-IPs debug strategy:

- multi-Cores: ECT can propagate an event from one core to another one
- multi-IPs: all the IPs inside one domain can share the same debug resources (no need to duplicate counters, real-time tracers, etc.)

Initially, ECT is a complete module delivered by ARM Ltd. and is made of one Cross Trigger Matrix (CTM) and 2 Cross Trigger Interfaces (CTIs). But, as more trigger inputs/outputs were required for SoC debugging, 4 cross trigger interfaces were needed.

Thus, in order to support all projects with a common ECT architecture and as ARM11 platform already includes a CTI whereas ARM9 platform does not, ECT will have **3 Extended Cross Trigger Interfaces** (EXTENDED_CTI = extension of the CTI delivered by ARM Ltd) and **1 Cross Trigger Matrix** (CTM), as follows:

- For projects using an ARM9 platform, a fourth CTI will be delivered as a stand-alone block and this block will be plugged between ECT and ARM9 (reason why, in figure 1.1, ARM_CTI appears outside the ECT),
- For projects using an ARM11 platform, ECT will be directly plugged to the Cross Trigger Interface that resides in ARM11 platform.

Thus, ECT is made of 3 Extended Cross Trigger Interfaces (EXTENDED_CTI) and 1 Cross Trigger Matrix (CTM), a follows:

- EXTENDED_CTI includes the following:
  - A WRAPPER used to synchronize/hold the trigger events coming from asynchronous clock domains and to reformat the signals if necessary (pulse to level, inversion, etc.)
  - ECT_CTI (IP delivered by ARM Ltd.). This block combines and maps the trigger requests and broadcasts them to the CTM as channel events. The mapping of the 8 trigger inputs onto the 4 channel inputs and the mapping of the 4 channel outputs onto the 8 trigger outputs are programmed through memory-mapped registers.
  - an IPS to AHB bridge to interface with the AHB interface of ECT_CTI
- CTM includes:
  - ECT_CTM (IP delivered by ARM Ltd.). This block is a matrix providing 4 channel interfaces (4 channel inputs and 4 channel outputs per interface).
  - A wrapper to handle the CTM clock gating and additional observability functions.

19.3 Features

ECT includes the following features:

- 3 trigger interfaces with 8 trigger inputs/8 trigger outputs each
- 1 channel interface with 4 channel inputs/4 channel outputs
- hand-shaking and synchronization interfaces that can be bypassed for lower latency in case of synchronous sub-systems

- trigger signals reformatting
- application trigger: a trigger can be applied onto a channel by software
- memory-mapped register control
- secured register access
- test registers for internal test and SoC integration test

19.4 Modes of Operation

ECT has the following modes of operation:

- **Disabled mode:** all the mapping trigger inputs to channel inputs and trigger outputs to channel outputs is disabled.
- **Enabled mode:** the ECT receives trigger events and propagates those events to the selected trigger outputs (propagation programmed through some memory-mapped registers).

Because the ECT is accessed as a memory-mapped device and as it can be used to generate intrusive debug events, ECT should only be used during product development and its use in a production system should be prevented.

The following mechanisms are implemented to protect the ECT from unwanted accesses:

- **Debug enable:** two enable signals must be set to enable the ECT. The first one is software programmable through CTICONTROL register and the second one is an hardware input (ect_dbg_en).
- **Access:** a suitable key (32-bit word) has to be presented to the CTILOCK register to allow write access to the other EXTENDED_CTI registers.
- **Privileged access:** the CTIPROTECTION register restricts access to the configuration registers so that only privileged (supervisor mode) code can access them. In protection mode, user access is effectively masked, so any read from the ECT returns an inactive state.

19.5 Signals Description

19.6 Overview

ECT is a matrix that can receive signals (trigger inputs) from any point in SoC: cores, peripherals, busses, external inputs, etc. In the same way, ECT can propagate signals (trigger outputs) anywhere: essentially to debug resources but it could go to an external output through some I/O muxing. Indeed, one can assign any ECT trigger input/output to any signals.

The list of SoC signals connected to ECT trigger inputs/outputs should be specified in each project debug specification.

Table 19-1. ECT Signals

Signal Name	Signal Type	Signal Description
EXTENDED_CTI_0		
SYSTEM SIGNALS		
ipg_clk_0	Input	Global functional clock
ipg_clk_s_0	Input	IP bus clock
cti_trig_in_clk_0[7:0]	Input	Trigger input clocks
cti_trig_out_clk_0[7:0]	Input	Trigger output clocks
ipg_hard_async_reset_b_0	Input	IP bus registers reset
cti_trig_in_0[7:0]	Input	Trigger inputs
cti_trig_out_0[7:0]	Output	Trigger outputs
CONTROL SIGNALS		
cti_bus_if_sync_byp_0	Input	Bypass synchronization mechanism between the bus interface and the CTI control registers
trig_in_if_sync_byp_0[7:0]	Input	Bypass trigger input interface synchronization (wrapper <-> CTI)
trig_out_if_sync_byp_0[7:0]	Input	Bypass trigger output interface synchronization (wrapper <-> CTI)
cti_chan_out_hs_byp_0[3:0]	Input	Disable CTI channel output holder (CTI-> CTM)
cti_trig_out_hs_byp_0[7:0]	Input	Disable CTI trigger output holder (CTI -> wrapper)
inv_trig_in_0[7:0]	Input	Enable trigger input inversion
samp_trig_in_0[7:0]	Input	Enable trigger input sampling
trig_in_pulse_sel_0[7:0]	Input	Enable trigger input reshaping into a pulse
wrp_trig_in_hs_byp_0[7:0]	Input	Disable trigger input holder (wrapper -> CTI)
inv_trig_out_0[7:0]	Input	Enable trigger output inversion
samp_trig_out_0[7:0]	Input	Enable trigger output sampling
trig_out_pulse_sel_0[7:0]	Input	Enable trigger output reshaping into a pulse
wrp_trig_out_ack_byp_0[7:0]	Input	Disable trigger output acknowledgement (wrapper -> CTI)
IP BUS LINE SIGNALS		
ips_addr_0[11:2]	Input	IP bus address
ips_wdata_0[31:0]	Input	IP bus write data
ips_byte_31_24_0	Input	IP bus byte enable
ips_byte_23_16_0	Input	IP bus byte enable
ips_byte_15_8_0	Input	IP bus byte enable
ips_byte_7_0_0	Input	IP bus byte enable

Table 19-1. ECT Signals (continued)

Signal Name	Signal Type	Signal Description
ips_rwb_0	Input	IP bus read/write control
ips_module_en_0	Input	IP bus device select
ips_supervisor_access_0	Input	IP bus supervisor access
ips_rdata_0[31:0]	Output	IP bus read data
ips_xfr_wait_0	Output	IP bus transfer wait signal
ips_xfr_error_0	Output	IP bus transfer error signal
EXTENDED_CTI_1		
SYSTEM SIGNALS		
ipg_clk_1	Input	Global functional clock
ipg_clk_s_1	Input	IP bus clock
cti_trig_in_clk_1[7:0]	Input	Trigger input clocks
cti_trig_out_clk_1[7:0]	Input	Trigger output clocks
ipg_hard_async_reset_b_1	Input	IP bus registers reset
cti_trig_in_1[7:0]	Input	Trigger inputs
cti_trig_out_1[7:0]	Output	Trigger outputs
CONTROL SIGNALS		
cti_bus_if_sync_byp_1	Input	Bypass synchronization mechanism between the bus interface and the CTI control registers
trig_in_if_sync_byp_1[7:0]	Input	Bypass trigger input interface synchronization (wrapper <-> CTI)
trig_out_if_sync_byp_1[7:0]	Input	Bypass trigger output interface synchronization (wrapper <-> CTI)
cti_chan_out_hs_byp_1[3:0]	Input	Disable CTI channel output holder (CTI-> CTM)
cti_trig_out_hs_byp_1[7:0]	Input	Disable CTI trigger output holder (CTI -> wrapper)
inv_trig_in_1[7:0]	Input	Enable trigger input inversion
samp_trig_in_1[7:0]	Input	Enable trigger input sampling
trig_in_pulse_sel_1[7:0]	Input	Enable trigger input reshaping into a pulse
wrp_trig_in_hs_byp_1[7:0]	Input	Disable trigger input holder (wrapper -> CTI)
inv_trig_out_1[7:0]	Input	Enable trigger output inversion
samp_trig_out_1[7:0]	Input	Enable trigger output sampling
trig_out_pulse_sel_1[7:0]	Input	Enable trigger output reshaping into a pulse
wrp_trig_out_ack_byp_1[7:0]	Input	Disable trigger output acknowledgement (wrapper -> CTI)
IP BUS LINE SIGNALS		
ips_addr_1[11:2]	Input	IP bus address

Table 19-1. ECT Signals (continued)

Signal Name	Signal Type	Signal Description
ips_wdata_1[31:0]	Input	IP bus write data
ips_byte_31_24_1	Input	IP bus byte enable
ips_byte_23_16_1	Input	IP bus byte enable
ips_byte_15_8_1	Input	IP bus byte enable
ips_byte_7_0_1	Input	IP bus byte enable
ips_rwb_1	Input	IP bus read/$\overline{\text{write}}$ control
ips_module_en_1	Input	IP bus device select
ips_supervisor_access_1	Input	IP bus supervisor access
ips_rdata_1[31:0]	Output	IP bus read data
ips_xfr_wait_1	Output	IP bus transfer wait signal
ips_xfr_error_1	Output	IP bus transfer error signal
EXTENDED_CTI_2		
SYSTEM SIGNALS		
ipg_clk_2	Input	Global functional clock
ipg_clk_s_2	Input	IP bus clock
cti_trig_in_clk_2[7:0]	Input	Trigger input clocks
cti_trig_out_clk_2[7:0]	Input	Trigger output clocks
ipg_hard_async_reset_b_2	Input	IP bus registers reset
cti_trig_in_2[7:0]	Input	Trigger inputs
cti_trig_out_2[7:0]	Output	Trigger outputs
CONTROL SIGNALS		
cti_bus_if_sync_byp_2	Input	Bypass synchronization mechanism between the bus interface and the CTI control registers
trig_in_if_sync_byp_2[7:0]	Input	Bypass trigger input interface synchronization (wrapper <-> CTI)
trig_out_if_sync_byp_2[7:0]	Input	Bypass trigger output interface synchronization (wrapper <-> CTI)
cti_chan_out_hs_byp_2[3:0]	Input	Disable CTI channel output holder (CTI-> CTM)
cti_trig_out_hs_byp_2[7:0]	Input	Disable CTI trigger output holder (CTI -> wrapper)
inv_trig_in_2[7:0]	Input	Enable trigger input inversion
samp_trig_in_2[7:0]	Input	Enable trigger input sampling
trig_in_pulse_sel_2[7:0]	Input	Enable trigger input reshaping into a pulse
wrp_trig_in_hs_byp_2[7:0]	Input	Disable trigger input holder (wrapper -> CTI)
inv_trig_out_2[7:0]	Input	Enable trigger output inversion

Table 19-1. ECT Signals (continued)

Signal Name	Signal Type	Signal Description
samp_trig_out_2[7:0]	Input	Enable trigger output sampling
trig_out_pulse_sel_2[7:0]	Input	Enable trigger output reshaping into a pulse
wrp_trig_out_ack_byp_2[7:0]	Input	Disable trigger output acknowledgement (wrapper -> CTI)
IP BUS LINE SIGNALS		
ips_addr_2[11:2]	Input	IP bus address
ips_wdata_2[31:0]	Input	IP bus write data
ips_byte_31_24_2	Input	IP bus byte enable
ips_byte_23_16_2	Input	IP bus byte enable
ips_byte_15_8_2	Input	IP bus byte enable
ips_byte_7_0_2	Input	IP bus byte enable
ips_rwb_2	Input	IP bus read/$\overline{\text{write}}$ control
ips_module_en_2	Input	IP bus device select
ips_supervisor_access_2	Input	IP bus supervisor access
ips_rdata_2[31:0]	Output	IP bus read data
ips_xfr_wait_2	Output	IP bus transfer wait signal
ips_xfr_error_2	Output	IP bus transfer error signal
CTM SIGNALS		
SYSTEM SIGNALS		
ctm_clk	Input	System clock
ctm_chan_in_3[3:0]	input	Channel input port from CTI 3 (ARM_CTI)
ctm_chan_out_ack_3[3:0]	Input	Channel output acknowledge from CTI 3 (ARM_CTI)
ctm_chan_out_3[3:0]	Output	Channel output port to CTI 3 (ARM_CTI)
ctm_chan_in_ack_3[3:0]	Output	Channel input acknowledge to CTI 3 (ARM_CTI)
ctm_lines[3:0]	Output	System lines
CONTROL SIGNALS		
ctm_chan_out_hs_byp_0[3:0]	Input	Disable CTM channel output holder for port 0 (CTM->CTI0)
ctm_chan_out_hs_byp_1[3:0]	Input	Disable CTM channel output holder for port 1 (CTM->CTI1)
ctm_chan_out_hs_byp_2[3:0]	Input	Disable CTM channel output holder for port 2 (CTM->CTI2)
ctm_chan_out_hs_byp_3[3:0]	Input	Disable CTM channel output holder for port 3 (CTM->ARM_CTI)
ci_sync_byp_0	Input	Bypass channel interface synchronization for port 0 (CTI0 <-> CTM)
ci_sync_byp_1	Input	Bypass channel interface synchronization for port 1 (CTI1 <-> CTM)

Table 19-1. ECT Signals (continued)

Signal Name	Signal Type	Signal Description
ci_sync_byp_2	Input	Bypass channel interface synchronization for port 2 (CTI2 <-> CTM)
ci_sync_byp_3	Input	Bypass channel interface synchronization for port 3 (ARM_CTI <-> CTM)
GLOBAL SIGNALS		
ect_reset_b	Input	Global reset
ect_dbg_en	Input	Debug enable
ect_clk_en	Output	Functional clock enable

19.7 Detailed Signal Descriptions

19.7.1 EXTENDED_CTI_X signals (X = 0,1,2)

19.7.1.1 ipg_clk_X (input)

ipg_clk_X (input) is used to clock all EXTENDED_CTI_X registers except wrapper and bus interface, which excludes IPS to AHB module, AHB interface block and wrapper registers.

19.7.1.2 ipg_clk_s_X (input)

ipg_clk_s_X (input) is used to clock all bus interface registers, which includes IPS to AHB module and AHB interface block.

19.7.1.3 cti_trig_in_clk_X [7:0] (input)

cti_trig_in_clk_X [7:0] (input) is the clock of each trigger input, and is used to clock wrapper registers related to each trigger input.

19.7.1.4 cti_trig_out_clk_X [7:0] (output)

cti_trig_out_clk_X [7:0] (output) is the clock of each trigger output, and is used to clock wrapper registers related to each trigger output.

19.7.1.5 ipg_hard_async_reset_b_X (input)

ipg_hard_async_reset_b_X (input) is the asynchronous reset, active low, and is used to reset all bus interface registers, which includes IPS to AHB module and AHB interface block.

19.7.1.6 cti_trig_in_X [7:0] (input)

cti_trig_in_X [7:0] (input) triggers inputs from the SoC.

19.7.1.7 cti_trig_out_X [7:0] (output)

cti_trig_out_X [7:0] (output) triggers outputs to the SoC.

19.7.1.8 cti_bus_if_sync_byp_X (input)

cti_bus_if_sync_byp_X (input) is the bypass synchronization between the bus interface (IPS to AHB + AHB interface) and ECT_CTI control registers. Tie high when both sides of the interface are synchronous to bypass the synchronization circuitry.

19.7.1.9 trig_in_if_sync_byp_X [7:0] (input)

Tie trig_in_if_sync_byp_X [7:0] (input) high when ipg_clk_X and trigger input clocks are synchronous to bypass the synchronization circuitry.

19.7.1.10 trig_out_if_sync_byp_X [7:0] (input)

Tie trig_out_if_sync_byp_X [7:0] (input) high when ipg_clk_X and trigger output clocks are synchronous to bypass the synchronization circuitry.

19.7.1.11 cti_chan_out_hs_byp_X [3:0] (input)

Tie cti_chan_out_hs_byp_X [3:0] (input) high to disable the holder circuitry on ECT_CTI channel output going to CTM.

19.7.1.12 cti_trig_out_hs_byp_X [7:0] (input)

Tie cti_trig_out_hs_byp_X [7:0] (input) high to disable the holder circuitry on ECT_CTI trigger output going to wrapper. When high, both hardware acknowledgement from wrapper and software acknowledgement from CTIIINTACK register are disabled.

19.7.1.13 inv_trig_in_X [7:0] (input)

Tie inv_trig_in_X [7:0] (input) high to invert ECT trigger input.

19.7.1.14 samp_trig_in_X [7:0] (input)

Tie samp_trig_in_X [7:0] (input) high to sample ECT trigger input with its corresponding clock (cti_trig_in_clk_X [7:0]).

19.7.1.15 trig_in_pulse_sel_X [7:0] (input)

Tie trig_in_pulse_sel_X [7:0] (input) high to convert ECT trigger input into a pulse.

19.7.1.16 wrp_trig_in_hs_byp_X [7:0] (input)

Tie wrp_trig_in_hs_byp_X [7:0] (input) high to disable the holder circuitry on the trigger input going out from wrapper to ECT_CTI

19.7.1.17 inv_trig_out_X [7:0] (input)

Tie inv_trig_out_X [7:0] (input) high to invert ECT trigger output.

19.7.1.18 samp_trig_out_X [7:0] (input)

Tie samp_trig_out_X [7:0] (input) high to sample ECT trigger output with its corresponding clock (cti_trig_out_clk_X [7:0]).

19.7.1.19 trig_out_pulse_sel_X [7:0] (input)

Tie trig_out_pulse_sel_X [7:0] (input) high to convert ECT trigger output into a pulse.

19.7.1.20 wrp_trig_out_ack_byp_X [7:0] (input)

Tie wrp_trig_out_ack_byp_X [7:0] (input) high to disable the hardware trigger output acknowledgement going out from wrapper to ECT_CTI.

19.7.2 CTM signals (X= 0,1,2 / Y = 0,1,2,3)

19.7.2.1 ctm_clk (input)

ctm_clk (input) is the CTM clock, and is used to clock all CTM registers.

19.7.2.2 ctm_chan_in_3 [3:0] (input)

ctm_chan_in_3 [3:0] (input) is the channel input port from CTI 3 (ARM_CTI).

19.7.2.3 ctm_chan_out_ack_3 [3:0] (input)

ctm_chan_out_ack_3 [3:0] (input) is the channel output acknowledge from CTI 3 (ARM_CTI).

19.7.2.4 ctm_chan_out_3 [3:0] (output)

ctm_chan_out_3 [3:0] (output) is the channel output port to CTI 3 (ARM_CTI).

19.7.2.5 ctm_chan_in_ack_3 [3:0] (output)

ctm_chan_in_ack_3 [3:0] (output) is the channel input acknowledge to CTI 3 (ARM_CTI).

19.7.2.6 ctm_lines [3:0] (output)

Each bit reflects corresponding channel activity out of CTM. ctm_lines [3:0] (output) is routed to IOMUX for debug purpose.

19.7.2.7 ctm_chan_out_hs_byp_Y [3:0] (input)

Tie ctm_chan_out_hs_byp_Y [3:0] (input) high to disable the holder circuitry on CTM channel output going to EXTENDED_CTI_X/ ARM_CTI.

19.7.2.8 ci_sync_byp_Y (input)

ci_sync_byp_Y (input) is the bypass synchronization on the channel interface CTM <-> EXTENDED_CTI_X / ARM_CTI. Tie high when both sides of the channel interface are synchronous to bypass the synchronization circuitry.

19.7.3 Global signals (X = 0,1,2)

19.7.3.1 ect_reset_b (input)

ect_reset_b (input) is the asynchronous reset, active low. Used to reset all CTM registers, all EXTENDED_CTI_X registers except bus interface, which excludes IPS to AHB module & AHB interface block.

19.7.3.2 ect_dbg_en (input)

ect_dbg_en (input) is the debug enable signal. When 0, the cross trigger interface is disabled, no trigger can be generated or received and clocks should be disabled to avoid any power consumption (see ect_clk_en signal). When 0, all trigger outputs and CTM system lines are gated off.

19.7.3.3 ect_clk_en (output)

ect_clk_en (output) is the clock enable signal. This signal is a copy of ect_dbg_en. It should go to the clock reset module and should be used to enable or not EXTENDED_CTI_X clocks (ipg_clk_X), trigger input clocks (cti_trig_in_clk_X [7:0]), trigger output clocks (cti_trig_out_clk_X [7:0]) and CTM clock (ctm_clk).

Note that EXTENDED_CTI IP bus clocks (ipg_clk_s_X) should not be gated with this signal.

19.8 Memory Map/Register Definition

Only EXTENDED_CTI includes memory mapped registers; CTM cannot be configured by software.

The following subsections describe memory mapped registers for only one Extended_CTI. **All Extended_CTI have the same memory mapped register structure.**

Because Extended_CTIs stand on different IP busses or have different base addresses on the same bus, all register addresses are referenced with a $BASE<N><N> variable.

The following applies to all registers:

- Reserved or unused bits of registers must be written to 0, an ignore on read
- All registers bits are reset to 0 unless otherwise stated in the text
- only word accesses are allowed; an IP xfr error is generated if any other size is attempted.

All registers can be accessed even if EXTENDED_CTI is software disabled (GLBEN is low). However, if ect_dbg_en is low and ect_clk_en is used for clock gating, ECT clocks will be off. In this case, neither read nor write access is allowed, and any attempt will generate an ips_xfr_error.

19.9 Register Summary

KEY:

Always Reads One	Always Reads Zero	Read/ Write Bit	Read-Only Bit	Write-Only Bit	Write 1 to Clear	Self-Clear Bit	N/A
1	0	bit	bit	bit	bit w1c	0 bit	

Table 19-2. CTI Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTICONTROL ($BASE<N> + 0x000)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GLBEN
	W																
CTISTATUS ($BASE<N> + 0x004)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DGBEN	LOCKED
	W																
CTILOCK ($BASE<N> + 0x008)	R																
	W	LOCKKEY[31:16]															
	R																
	W	LOCKKEY[15:0]															
CTIPROTECTION ($BASE<N> +0x00C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PROT
	W																
CTIINTACK ($BASE<N> + 0x010)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0								
	W									INTACK[7:0]							
CTIAPPSET ($BASE<N> +0x014)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	APPSET[3:0]			
	W																
CTIAPPCLEAR ($BASE<N> +0x018)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 19-2. CTI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R	0	0	0	0	0	0	0	0	0	0	0	0				
	W													APPCLEAR[3:0]			
CTIAPPPULSE ($BASE<N> + 0x01C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0				
	W													APPPULSE[3:0]			
CTIINENn[1] ($BASE<N> + 0x020 + n*4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	TRIGINEN[3:0]			
	W																
CTIOUTENn[2] ($BASE<N> + 0x0A0 + n*4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	TRIGOUTEN[3:0]			
	W																
CTITRIGINSTA-TUS ($BASE<N> +0x130)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	TRIGINSTATUS[7:0]							
	W																
CTITRIGOUT-STATUS ($BASE<N> + 0x134)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	TRIGOUTSTATUS[7:0]							
	W																
CTICHINSTATUS ($BASE<N> + 0x138)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CHINSTATUS[3:0]			
	W																
CTICHOUTSTA-TUS ($BASE<N> + 0x13C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CHOUTSTATUS[3:0]			
	W																
CTIITCR ($BASE<N> +0x200)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEST_MODE[1:0]	
	W																

Table 19-2. CTI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTIITIP0 ($BASE<N> + 0x204)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ECTTRIGIN[7:0]							
	W																
CTIITIP1 ($BASE<N> + 0x208)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CTICHIN[3:0]			
	W																
CTIITIP2 ($BASE<N> + 0x20C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ECTTRIGOUTACK[7:0]							
	W																
CTIITIP3 ($BASE<N> + 0x210)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	ECTCHOUTACK[3:0]			
	W																
CTITIOP0 ($BASE<N> + 0x214)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ECTTRIGOUT[7:0]							
	W																
CTITIOP1 ($BASE<N> + 0x218)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CTICHOUT[3:0]			
	W																
CTITIOP2 ($BASE<N> + 0x21C)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ECTTRIGINACK[7:0]							
	W																
CTITIOP3 ($BASE<N> + 0x220)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CTICHINACK[3:0]			
	W																
CTIPERIPHID0 ($BASE<N> + 0xFE0)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 19-2. CTI Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R	0	0	0	0	0	0	0	0	Partnumber0[7:0]							
	W																
CTIPERIPHID1 ($BASE<N> + 0xFE4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Designer0[3:0]				Partnumber1[3:0]			
	W																
CTIPERIPHID2 ($BASE<N> + 0xFE8)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Revision[3:0]				Designer1[3:0]			
	W																
CTIPERIPHID3 ($BASE<N> + 0xFEC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	Configuration[7:0]							
	W																
CTIPCELLID0 ($BASE<N> + 0xFF0)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CTIPCELLID0[7:0]							
	W																
CTIPCELLID1 ($BASE<N> + 0xFF4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CTIPCELLID1[7:0]							
	W																
CTIPCELLID2 ($BASE<N> + 0xFF8)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CTIPCELLID2[7:0]							
	W																
CTIPCELLID3 ($BASE<N> + 0xFFC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CTIPCELLID3[7:0]							
	W																

[1] n = 0-7
[2] n = 0-7

19.10 Register Descriptions

Table 19-3. Register Terms

Term	Description
Grey bit	Unimplemented bit; always reads as zero; writing has no effect.
Access	
S	Supervisor mode only
-	Supervisor or user mode
Type	
r	Read only. Writing to this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change a bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than reset.
w1c	A status bit that can be read and cleared by writing a logic 1.
slfclr	Self-clearing bit. Writing a 1 has some effect on module, but it always reads as a 0.
Reset	
0	Resets to a logic 0.
1	Resets to a logic 1.
u	Unaffected by reset.
?	Reset state is unknown.

19.10.1 CTICONTROL Register

The CTICONTROL register, shown in Figure 19-2, enables the CTI.

Address $BASE<N> + 0x000 Access: Read/Write
Wait State: >3

	31																												1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GLBEN
W																														
Reset	All zeros																													

Figure 19-2. CTI Control Register (CTICONTROL)

Table 19-4 describes the CTICONTROL register.

Table 19-4. CTICONTROL Register Field Descriptions

Bits	Description
31–1	Reserved
0 GLBEN	Enables or Disables the ECT 0 Enabled 1 Disabled (When disabled, all cross triggering mapping logic functionality is disabled for this processor)

19.10.2 CTISTATUS Register

The CTISTATUS register, shown in Figure 19-3, provides the locked and enable status of the CTI.

Address $BASE<N> + 0x004 Access: Read only
Wait State: >3

	31																											2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DGBEN	LOCKED
W																														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	1

Figure 19-3. CTI Status Register (CTISTATUS)

Table 19-5 describes the CTISTATUS register.

Table 19-5. CTISTATUS Register Field Descriptions

Field	Description
31–2	Reserved
1 DGBEN	Shows the status of TI enable—reset values depends on ECTDBGEN port 0 Interfaces are enabled 1 Interfaces are disabled (even if CTI is disabled, register may be read)
0 LOCKED	Enables or Disables the ECT 0 Access to CTI is not locked 1 Access to CTI is locked

19.10.3 CTILOCK Register

The CTILOCK register, shown in Figure 19-4, enables or disables all other register write access by providing a LOCKED mode for the AHB interface. When the AHB interface is locked, write accesses to the CTI registers except CTILOCK are ignored.

To allow write access, a LOCKKEY value, 0x0ACCE550, is written into CTILOCK. All other values lock write access.

Address $BASE<N> + 0x008 Access: Write only
Wait State: >3

	31																													0
R																														
W	LOCKKEY																													
Reset	All zeros																													

Figure 19-4. CTI Lock Register (CTILOCK)

Table 19-6 describes the CTILOCK register.

Table 19-6. CTILOCK Register Field Descriptions

Bits	Name	Description
31–0	LOCKKEY	Lock key 0x0ACCE550—Write Access to all other register is allowed Else—Access locked

19.10.4 CTIPROTECTION Register

The CTIPROTECTION register, shown in Figure 19-5, enables or disables protected register access.

When enabled, only privileged mode (supervisor) accesses (read and write) can access CTI registers: that is when HPROT[1] is set high for the current transfer. If an access is made in user mode, all registers return 0.

When disabled, both user and privileged mode can access CTI registers, except for CTIPROTECTION register that can only be accessed in privileged mode.

Note that after reset, both user and supervisor access modes are allowed (except for CTIPROTECTION register that can only be accessed in privileged mode).

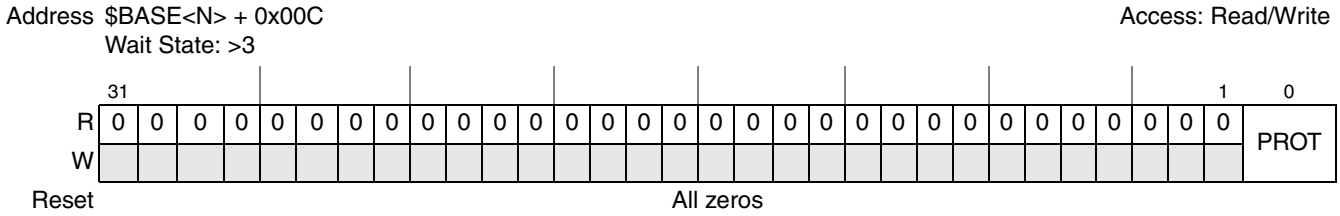


Figure 19-5. CTI Protection Register (CTIPROTECTION)

Table 19-7 describes the CTILOCK register.

Table 19-7. CTIPROTECTION Register Field Descriptions

Field	Description
31–1	Reserved
0 PROT	Enables protected mode 0 Protection mode disabled (both supervisor and user access can read/write CTI registers) 1 Protection mode enabled (only supervisor access can read/write CTI registers)

19.10.5 CTIINTACK Register

Any bit written as 1 causes the CTITRIGOUT output to be acknowledged.

This feature should be used when trigger output is used as an interrupt. In this case, the trigger output stay active until it is software acknowledged.

If trigger generator is still active, software acknowledge is kept active until trigger is off.

Figure 19-6 shows the CTIINTACK register.

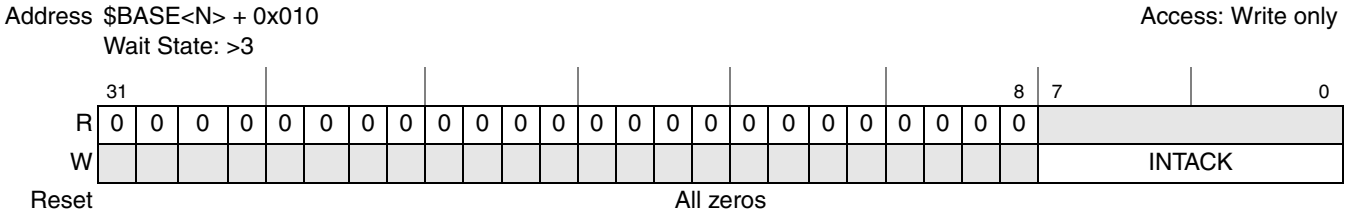


Figure 19-6. CTI Interrupt Acknowledge Register (CTIINTACK)

Table 19-8 describes the CTIPROTECTION register.

Table 19-8. CTIPROTECTION Register Field Descriptions

Field	Description
31–8	Reserved
7–0 INTACK	Interrupt acknowledge One bit of the register for each CTITRIGOUT 0 Nothing happens 1 CTITRIGOUT is acknowledged

19.10.6 CTIAPPSET Register

A write to the CTIAPPSET register, shown in Figure 19-7, causes a channel event to be raised, corresponding to the bit written to.

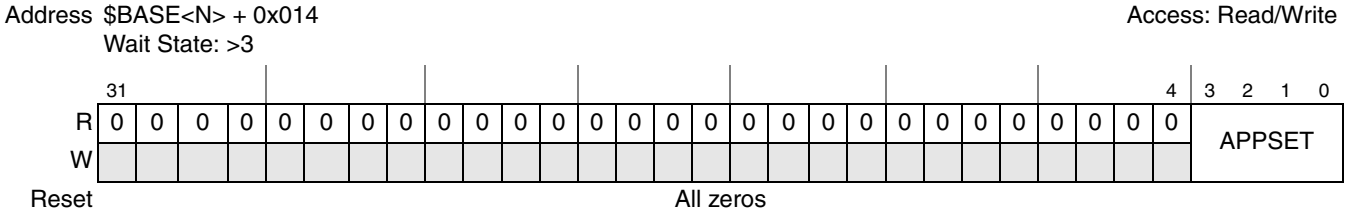


Figure 19-7. CTI Application Trigger Set Register (CTIAPPSET)

Table 19-9 describes the CTIAPPSET register.

Table 19-9. CTIAPPSET Register Field Descriptions

Field	Description
31–4	Reserved
3–0 APPSET	Setting a bit high generates a channel event for the selected channel Read: 0 Application trigger inactive 1 Application trigger active Write: 0 No effect 1 Generates channel event

19.10.7 CTIAPPCLEAR Register

A write to the CTIAPPCLEAR register, shown in Figure 19-8, causes a channel event to be cleared, corresponding to the bit written to.

Address $BASE<N> + 0x018
Wait State: >3

Access: Write only

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																											APPCLEAR
Reset	All zeros																										

Figure 19-8. CTI Application Trigger Clear Register (CTIAPPCLEAR)

Table 19-10 describes the CTIAPPCLEAR register.

Table 19-10. CTIAPPCLEAR Register Field Descriptions

Field	Description
31–4	Reserved
3–0 APPCLEAR	Clears corresponding bit in the APPSET register 0 No effect 1 Application trigger disabled in the APPSET register

19.10.8 CTIAPPPULSE Register

A write to the CTIAPPPULSE register, shown in Figure 19-9, causes a channel event pulse (one clock cycle length) to be generated, corresponding to the bit written to.

Address $BASE<N> + 0x01C
Wait State: >3

Access: Write only

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																											APPPULSE
Reset	All zeros																										

Figure 19-9. CTI Application Pulse Register (CTIAPPPULSE)

Table 19-11 describes the CTIAPPPULSE register.

Table 19-11. CTIAPPPULSE Register Field Descriptions

Field	Description
31–4	Reserved
3–0 APPPULSE	Setting a bit high generates a channel event pulse 0 No effect 1 Channel event pulse, 1 clk cycle length

19.10.9 CTIINEN0–7 Register

The CTIINEN0–7 registers, shown in Figure 19-10, enable the signalling of an event on (a) CTM channel(s) when the core issues a trigger (ECTTRIGIN) to the CTI.

There is a register for each of the eight ECTTRIGIN inputs. Within each register, there is one bit for each of the four channels implemented. These registers do not affect application trigger operations (APPSET register).

Address $BASE<N> + 0x020+ *n**4[1] Access: Read/Write
Wait State: >3

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRIGINEN	
W																											
Reset	All zeros																										

Figure 19-10. CTI Trigger to Channel Enable Register 0–7 (CTIINEN0–7)

[1] *n* = 0–7

Table 19-12 describes the CTIINEN0–7 registers.

Table 19-12. CTIINEN0–7 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 TRIGINEN	Enable a cross trigger event to the corresponding channel when a ECTTRIGIN is activated 0 Disable 1 Enable

Example 19-1.

When TRIGINEN[2] within CTIINEN5 register is set to 1, ECTTRIGIN[5] is mapped onto channel 2.

19.10.10 CTIOUTEN0–7 Register

The CTIOUTEN0–7 registers, shown in Figure 19-11, define which channel(s) can generate a ECTTRIGOUT output.

There is a register for each of the eight ECTTRIGOUT outputs. Within each register, there is one bit for each of the four channels implemented. These registers do not affect application trigger operations (APPSET register).

Address $BASE<N> + 0x0A0 + n*4[1] Access: Read/Write
Wait State: >3

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRIGOUTEN	
W																											
Reset	All zeros																										

Figure 19-11. CTI Channel to Trigger Register 0–7 (CTIOUTEN0–7)

[1] *n* = 0–7

Table 19-13 describes the CTIOUTEN0–7 registers.

Table 19-13. CTIOUTEN0–7 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 TRIGOUTEN	Enable a cross trigger event to the corresponding channel when a ECTTRIGOUT is activated 0 Disable 1 Enable

Example 19-2.

When TRIGINOUT[2] within CTIOUTEN5 register is set to 1, CTICHIN[2] is mapped onto ECTTRIGOUT[5].

19.10.11 CTITRIGINSTATUS Register

The CTITRIGINSTATUS register, shown in Figure 19-12, provides the status of the ECTTRIGIN inputs.

Address $BASE<N> + 0x130 Access: Read only
Wait State: >3

	31																								8	7	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRIGINSTATUS	
W																											
Reset	All zeros																										

Figure 19-12. CTI Trigger In Status Register (CTITRIGINSTATUS)

Table 19-14 describes the CTITRIGINSTATUS register.

Table 19-14. CTITRIGINSTATUS Register Field Descriptions

Field	Description
31–8	Reserved
7–0 TRIGINSTATUS	0 ECTTRIGIN is inactive 1 ECTTRIGIN is active

19.10.12 CTITRIGOUTSTATUS Register

The CTITRIGOUTSTATUS register, shown in Figure 19-13, provides the status of the ECTTRIGOUT outputs.

Address $BASE<N> + 0x134
Wait State: >3
Access: Read only

	31																				8	7		0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRIGOUTSTATUS
W																									
Reset	All zeros																								

Figure 19-13. CTI Trigger Out Status Register (CTITRIGOUTSTATUS)

Table 19-15 describes the CTITRIGOUTSTATUS register.

Table 19-15. CTITRIGOUTSTATUS Register Field Descriptions

Field	Description
31–8	Reserved
7–0 TRIGOUTSTATUS	0 ECTTRIGOUT is inactive 1 ECTTRIGOUT is active

19.10.13 CTICHINSTATUS Register

The CTICHINSTATUS register, shown in Figure 19-14, provides the status of the CTICHIN inputs.

Address $BASE<N> + 0x138
Wait State: >3
Access: Read only

	31																							4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTICHINSTATUS
W																										
Reset	All zeros																									

Figure 19-14. CTI Channel In Status Register (CTICHINSTATUS)

Table 19-16 describes the CTICHINSTATUS register.

Table 19-16. CTICHINSTATUS Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTICHINSTATUS	0 CTICHIN is inactive 1 CTICHIN is active

19.10.14 CTICHOUTSTATUS Register

The CTICHOUTSTATUS register, shown in Figure 19-15, provides the status of the CTICHOUT outputs.

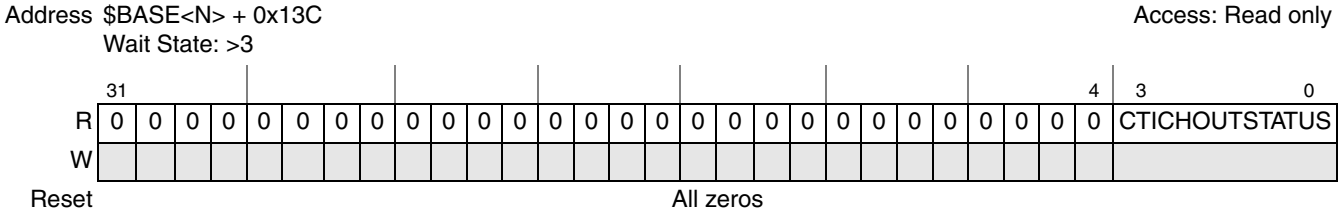


Figure 19-15. CTI Channel Out Status Register (CTICHOUTSTATUS)

Table 19-17 describes the CTICHOUTSTATUS register.

Table 19-17. CTICHOUTSTATUS Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTICHOUTSTATUS	0 CTICHOUT is inactive 1 CTICHOUT is active

19.10.15 CTITCR Register

The CTITCR register, shown in Figure 19-16, is a test controller register to be used only in test mode. This register controls the input and output test registers.

NOTE

This register is only used in test mode.

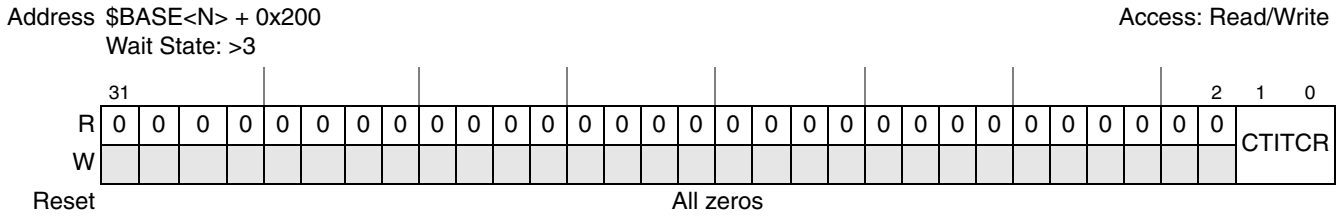


Figure 19-16. CTI Test Control Register (CTITCR)

Table 19-18 describes the CTITCR register.

Table 19-18. CTITCR Register Field Descriptions

Field	Description
31–2	Reserved

Table 19-18. CTITCR Register Field Descriptions (continued)

Field	Description
1–0 CTITCR	Test Control Register 00 Normal mode[1] 01 ITEN[2] = 1 10 soc_test[3] = 1 11 internal_test[4] = 1

Note:

1 Allows capture of all test registers (either input or output)

2 Forces test register values to be applied (both input and output)

3 Allows capture of all input test registers and forces output test register values to be applied on outputs for soc integration testing

4 Forces input test register values to be applied on inputs and allows capture of all output test registers for internal testing

19.10.16 CTIITIP0 Register

The CTIITIP0 register, shown in Figure 19-17, is used to control and read the values of the ECTTRIGIN inputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x204
Wait State: >3

Access: Read/Write

	31																8	7						0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTITIP0
W																									
Reset	All zeros																								

Figure 19-17. CTI Input Test Register 0 (CTIITIP0)

Table 19-19 describes the CTIITIP0 register.

Table 19-19. CTIITIP0 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTITIP0	Write: bypasses ECTTRIGIN input values with new value when ITEN or internal_test are high (cf. CTITCR register) Read: reads its own value when ITEN or internal_test are high, else reads ECTTRIGIN values

19.10.17 CTIITIP1 Register

The CTIITIP1 register, shown in Figure 19-18, is used to control and read the values of the CTICHIN inputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x208 Access: Read/Write
Wait State: >3

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIITIP1	
W																											
Reset	All zeros																										

Figure 19-18. CTI Input Test Register 1 (CTIITIP1)

Table 19-20 describes the CTIITIP1 register.

Table 19-20. CTIITIP1 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTIITIP1	Write: bypasses CTICHIN input values with new value when ITEN or internal_test are high (cf. CTITCR register) Read: reads its own value when ITEN or internal_test are high, else reads CTICHIN values

19.10.18 CTIITIP2 Register

The CTIITIP2 register, shown in Figure 19-19, is used to control and read the values of the ECTTRIGOUTACK inputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x20C Access: Read/Write
Wait State: >3

	31																								8	7	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIITIP2	
W																											
Reset	All zeros																										

Figure 19-19. CTI Input Test Register 2 (CTIITIP2)

Table 19-21 describes the CTIITIP2 register.

Table 19-21. CTIITIP2 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIITIP2	Write: bypasses ECTTRIGOUTACK input values with new value when ITEN or internal_test are high (cf. CTITCR register) Read: reads its own value when ITEN or internal_test are high, else reads ECTTRIGOUTACK values

19.10.19 CTIITIP3 Register

The CTIITIP3 register, shown in Figure 19-20, is used to control and read the values of the CTICHOUTACK inputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x210
Wait State: >3
Access: Read/Write

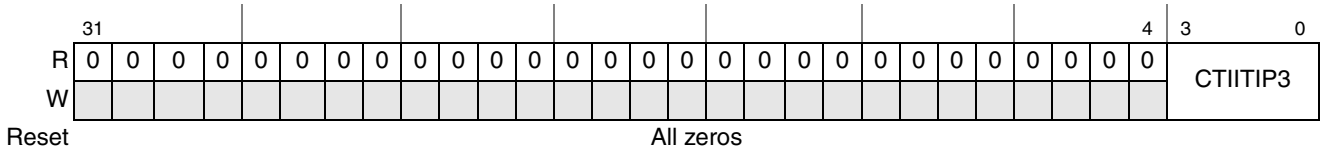


Figure 19-20. CTI Input Test Register 3 (CTIITIP3)

Table 19-22 describes the CTIITIP3 register.

Table 19-22. CTIITIP3 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTIITIP3	Write: bypasses CTICHOUTACK input values with new value when ITEN or internal_test are high (cf. CTITCR register) Read: reads its own value when ITEN or internal_test are high, else reads CTICHOUTACK values

19.10.20 CTIITOP0 Register

The CTIITOP0 register, shown in Figure 19-21, is used to control and read the values of the ECTTRIGOUT outputs

NOTE

This register can only be used in test mode.

Offset $BASE<N> + 0x214
Wait State: >3
Access: Read/Write

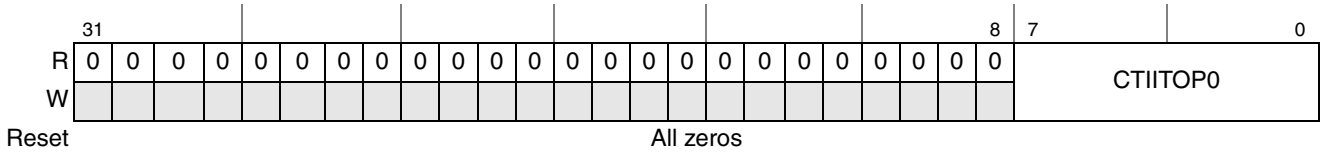


Figure 19-21. CTI Output Test Register 0 (CTIITOP0)

Table 19-23 describes the CTIITOP0 register.

Table 19-23. CTIITOP0 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIITOP0	Write: bypasses ECTTRIGOUT output values with new value when ITEN or soc_test are high (cf. CTITCR register) Read: reads its own value when ITEN or soc_test are high, else reads ECTTRIGOUT values

19.10.21 CTIITOP1 Register

The CTIITOP1 register, shown in Figure 19-22, is used to control and read the values of the CTICHOUT outputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x218
Wait State: >3
Access: Read/Write

	31																								4	3	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																												
Reset	All zeros																											

Figure 19-22. CTI Output Test Register 1 (CTIITOP1)

Table 19-24 describes the CTIITOP1 register.

Table 19-24. CTIITOP1 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTIITOP1	Write: bypasses CTICHOUT output values with new value when ITEN or soc_test are high (cf. CTITCR register) Read: reads its own value when ITEN or soc_test are high, else reads CTICHOUT values

19.10.22 CTIITOP2 Register

The CTIITOP2 register, shown in Figure 19-23, is used to control and read the values of the ECTTRIGINACK inputs

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x21C
Wait State: >3
Access: Read/Write

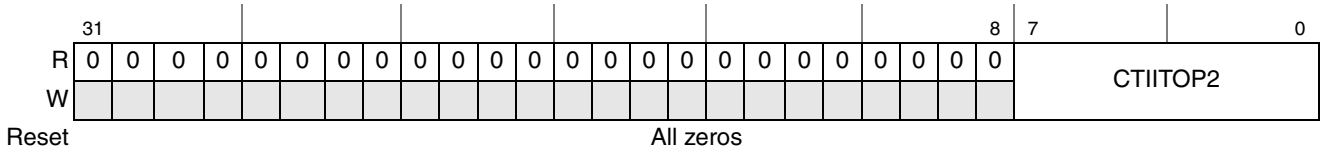


Figure 19-23. CTI Output Test Register 2 (CTIITOP2)

Table 19-25 describes the CTIITOP2 register.

Table 19-25. CTIITOP2 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIITOP2	Write: bypasses ECTTRIGINACK output values with new value when ITEN or soc_test are high (cf. CTITCR register Read: reads its own value when ITEN or soc_test are high, else reads ECTTRIGINACK values

19.10.23 CTIITOP3 Register

The CTIITOP3 register, shown in Figure 19-24, is used to control and read the values of the CTICHINACK inputs.

NOTE

This register can only be used in test mode.

Address $BASE<N> + 0x220
Wait State: >3
Access: Read/Write

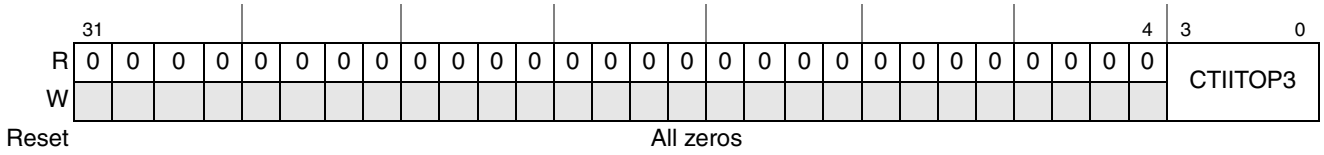


Figure 19-24. CTI Output Test Register 3 (CTIITOP3)

Table 19-26 describes the CTIITOP3 register.

Table 19-26. CTIITOP3 Register Field Descriptions

Field	Description
31–4	Reserved
3–0 CTIITOP3	Write: bypasses CTICHINACK output values with new value when ITEN or soc_test are high (cf. CTITCR register) Read: reads its own value when ITEN or soc_test are high, else reads CTICHINACK values

19.10.24 ARM Identification Registers

Extended_CTI have ARM dedicated registers for identification.

CTIPERIPHID (spread over CTIPERIPHID0–3 registers):

- Part Number [11:0]: identifies ARM peripheral. CTI is referenced as 0x900,
- Designer [7:0]: identifies designer of CTI. ARM Ltd. is referenced as 0x41,
- Revision Number [3:0],
- Configuration [7:0]: Configuration option for the design. Here it is 0x00.

CTIPCELLID (spread over CTIPCELLID0–3 registers):

- Standard Cross-Peripheral identification system, on 32 bits.

19.10.24.1 CTIPERIPHID0 Register

The CTIPERIPHID0 register, shown in Figure 19-25, is a read-only register that provide identifications code for the peripheral.

Offset $BASE<N> + 0xFE0 — Access: Read only
Wait State: >3

	31–8	7–0
R	0	Partnumber0
W		
Reset	All zeros	

Figure 19-25. CTI Peripheral Identification Register 0 (CTIPERIPHID0)

Table 19-27 describes the CTIPERIPHID0 register.

Table 19-27. CTIPERIPHID0 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 Partnumber0	First part of peripheral identification Read back as 0x00.

19.10.24.2 CTIPERIPHID1 Register

The CTIPERIPHID1 register, shown in Figure 19-26, is a read-only register that provides identifications code for the peripheral.

Offset $BASE<N> + 0xFE4 — Access: Read only
Wait State: >3

	31–8	7–4	3–0
R	0	Designer0	Partnumber0
W			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 1	0 0 0 1

Figure 19-26. CTI Peripheral Identification Register 1 (CTIPERIPHID1)

Table 19-28 describes the CTIPERIPHID1 register.

Table 19-28. CTIPERIPHID1 Register Field Descriptions

Field	Description
31–8	Reserved
7–4 Designer0	First part of designer identification read back as 0x1.
3–0 Partnumber1	Second part of peripheral identification Read back as 0x9.

19.10.24.3 CTIPERIPHID2 Register

The CTIPERIPHID2 register, shown in Figure 19-27, is a read-only register that provides identifications code for the peripheral.

Offset $BASE<N> + 0xFE8 — Access: Read only
Wait State: >3

	31																				8	7		4	3		0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Revision			Designer1		
W																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 19-27. CTI Peripheral Identification Register 2 (CTIPERIPHID2)

Table 19-29 describes the CTIPERIPHID2 register.

Table 19-29. CTIPERIPHID2 Register Field Descriptions

Field	Description
31–8	Reserved
7–4 Revision	Revision number read back as 0x0.
3–0 Designer1	Second part of designer identification Read back as 0x4.

19.10.24.4 CTIPERIPHID3 Register

The CTIPERIPHID3 register, shown in Figure 19-28, is a read-only register that provides identifications code for the peripheral.

Address $BASE<N> + 0xFEC — Access: Read only
Wait State: >3

	31																				8	7					0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Configuration	
W																											
Reset	All zeros																										

Figure 19-28. CTI Peripheral Identification Register 3 (CTIPERIPHID3)

Table 19-30 describes the CTIPERIPHID3 register.

Table 19-30. CTIPERIPHID3 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 Configuration	Indicates the number of interrupts supported read back as 0x00.

19.10.24.5 CTIPCELLID0 Register

The CTIPCELLID0 register, shown in Figure 19-29, is a read-only register that provides cross peripheral identification.

Address $BASE<N> + 0xFF0 Access: Read only
Wait State: >3

	31																				8	7							0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIPCELLID0			
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Figure 19-29. CTI Identification Register 0 (CTIPCELLID0)

Table 19-31 describes the CTIPCELLID0 register.

Table 19-31. CTIPCELLID0 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIPCELLID0	Identification Read back as 0x0D.

19.10.24.6 CTIPCELLID1 Register

The CTIPCELLID1 register, shown in Figure 19-30, is a read-only register that provides cross peripheral identification.

Address $BASE<N> + 0xFF4 Access: Read only
Wait State: >3

	31																				8	7							0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIPCELLID1			
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Figure 19-30. CTI Identification Register 1 (CTIPCELLID1)

Table 19-32 describes the CTIPCELLID1 register.

Table 19-32. CTIPCELLID1 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIPCELLID1	Identification Read back as 0xF0.

19.10.24.7 CTIPCELLID2 Register

The CTIPCELLID2 register, shown in Figure 19-31, is a read-only register that provides cross peripheral identification.

Address $BASE<N> + 0xFF8 Access: Read only
Wait State: >3

	31																				8	7						0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIPCELLID2		
W																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Figure 19-31. CTI Identification Register 2 (CTIPCELLID2)

Table 19-33 describes the CTIPCELLID2 register.

Table 19-33. CTIPCELLID2 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIPCELLID2	Identification Read back as 0x05.

19.10.24.8 CTIPCELLID3 Register

The CTIPCELLID3 register, shown in Figure 19-32, is a read-only register that provides cross peripheral identification.

Offset $BASE<N> + 0xFFC Access: Read only
Wait State: >3

	31																				8	7						0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CTIPCELLID3		
W																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Figure 19-32. CTI Identification Register 3 (CTIPCELLID3)

Table 19-34 describes the CTIPCELLID3 register.

Table 19-34. CTIPCELLID3 Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CTIPCELLID3	Identification Read back as 0xB1.

19.11 Functional Description

As explained in the overview, ECT is made of 3 EXTENDED_CTI and one CTM.

19.12 Extended Cross Trigger Interface (EXTENDED_CTI)

The EXTENDED_CTI is divided into the following sub-modules:

- a WRAPPER
- an ECT_CTI (IP delivered by ARM Ltd.)
- an IPS to AHB bridge to interface with ECT_CTI

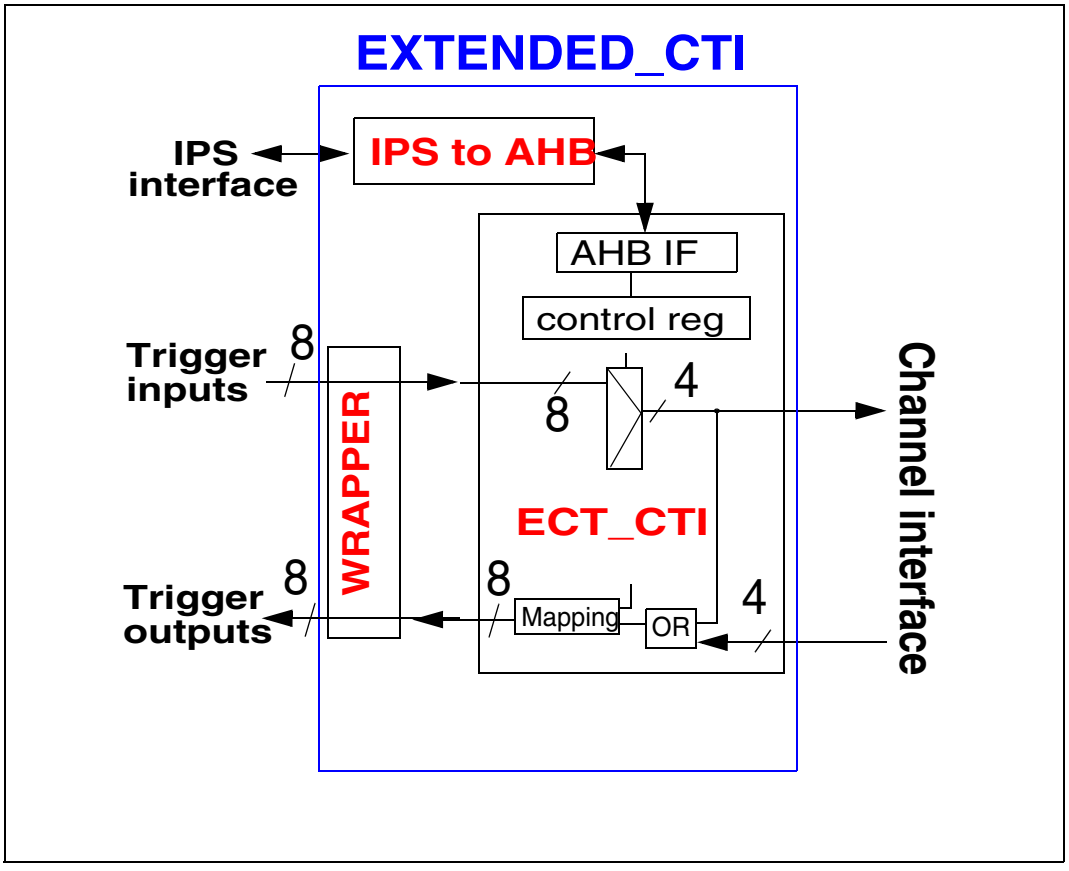


Figure 19-33. EXTENDED_CTI diagram

19.12.1 Wrapper

For a given EXTENDED_CTI, trigger inputs can emanate from asynchronous clock domains. Trigger inputs can also behave differently, some might be active high, others might be active low whereas ECT_CTI is only expecting events active high. For those reasons, trigger events have to pass through a wrapper to adapt their behavior.

Thus, the wrapper is used to do the following:

- Reshape signals: for example, a trigger transmitted as a level active high can be converted into a pulse or a trigger active low can be inverted.
- Sample trigger events: it can be useful to add a flip-flop on trigger inputs/outputs for timing purpose as wrapper might be far from trigger source/destination.
- Synchronize events between ECT_CTI clock and trigger clocks
- Hold values from a faster clock domain: for instance, a trigger input must stay active until an acknowledgement from the ECT_CTI is received

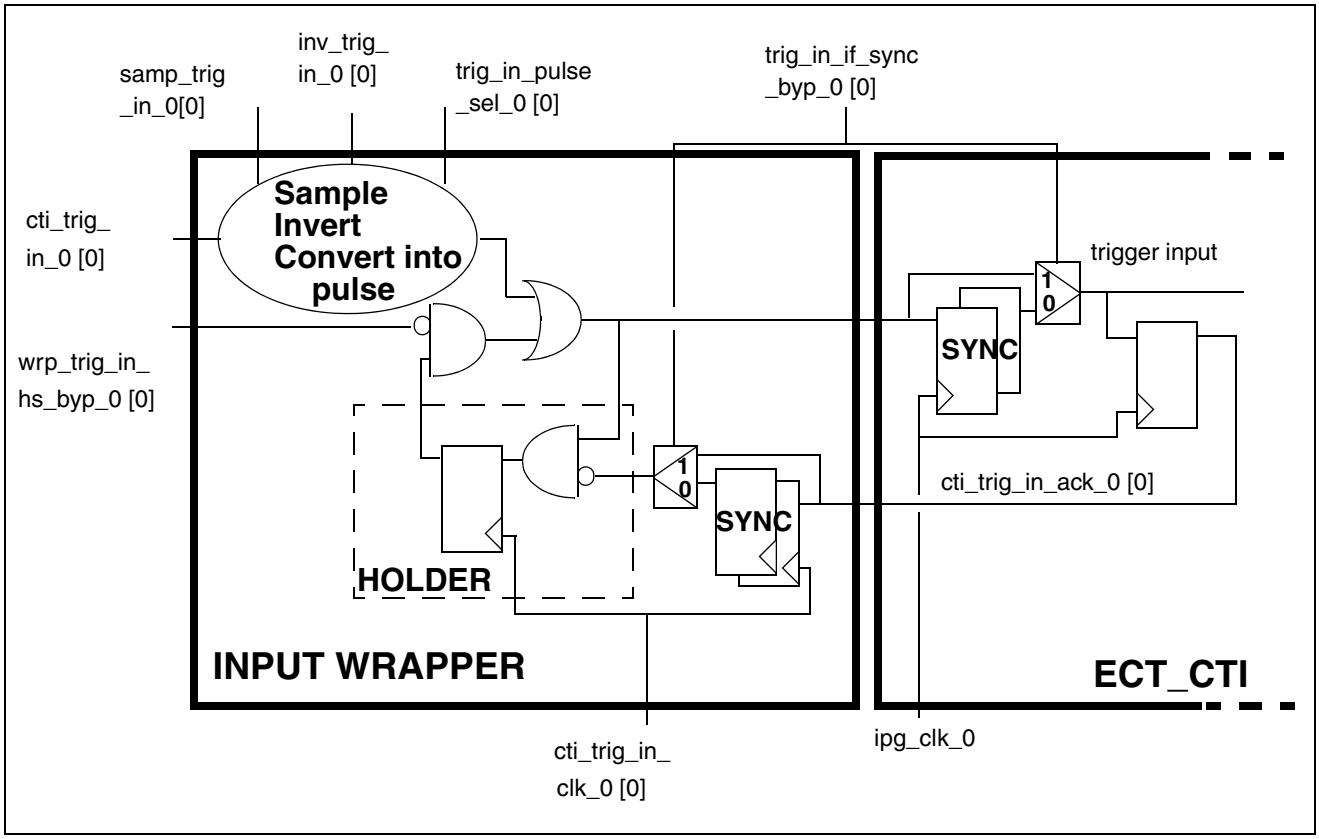


Figure 19-34. Input Wrapper and Hand-Shaking/Synchronization with ECT_CTI

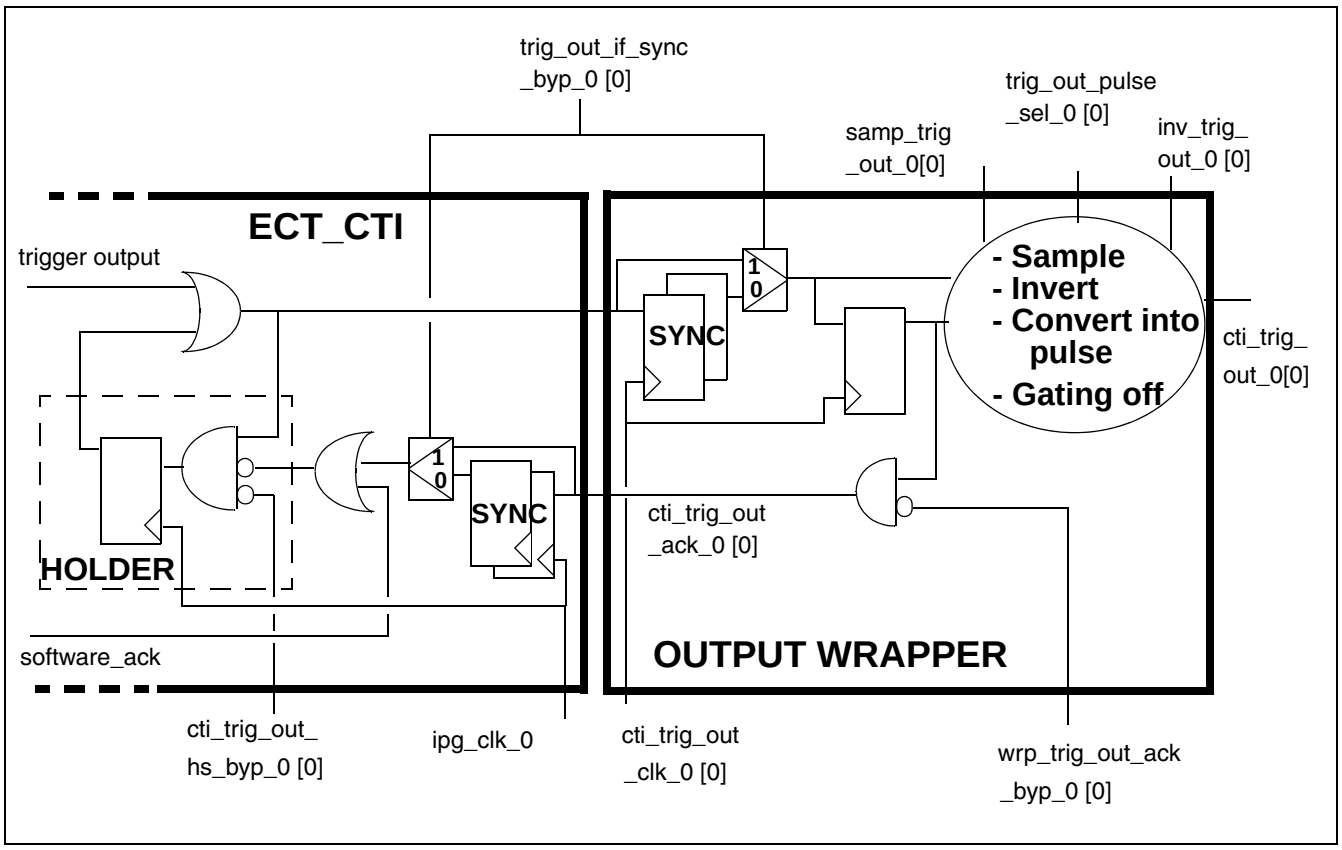


Figure 19-35. Output Wrapper and Hand-Shaking/Synchronization with ECT_CTI

All wrapper features such as sampling, inverter, reshaping, synchronization & holder can be activated or bypassed as wrapper control signals are brought to ECT top level.

The same mechanisms of hand-shaking/synchronization are used for CTM <-> EXTENDED_CTI/ARM CTI interfaces.

19.12.2 ECT_CTI

This block is the interface to CTM, it maps:

- trigger inputs onto channels
- channel onto trigger outputs.

ECT_CTI takes cross trigger event inputs from its associated subsystem, or from its own configuration registers, and generates outputs on the appropriate channels as defined by the configuration registers. Those ECT_CTI configuration registers are programmed by the sub-system connected to the ECT_CTI using the bus interface.

ECT_CTI also includes handshaking / synchronization mechanism on the 8 trigger inputs, 8 trigger outputs, 4 channel inputs and 4 channel outputs. All hand-shaking and synchronization mechanisms can be bypassed as the bypass control signals are brought to ECT top level.

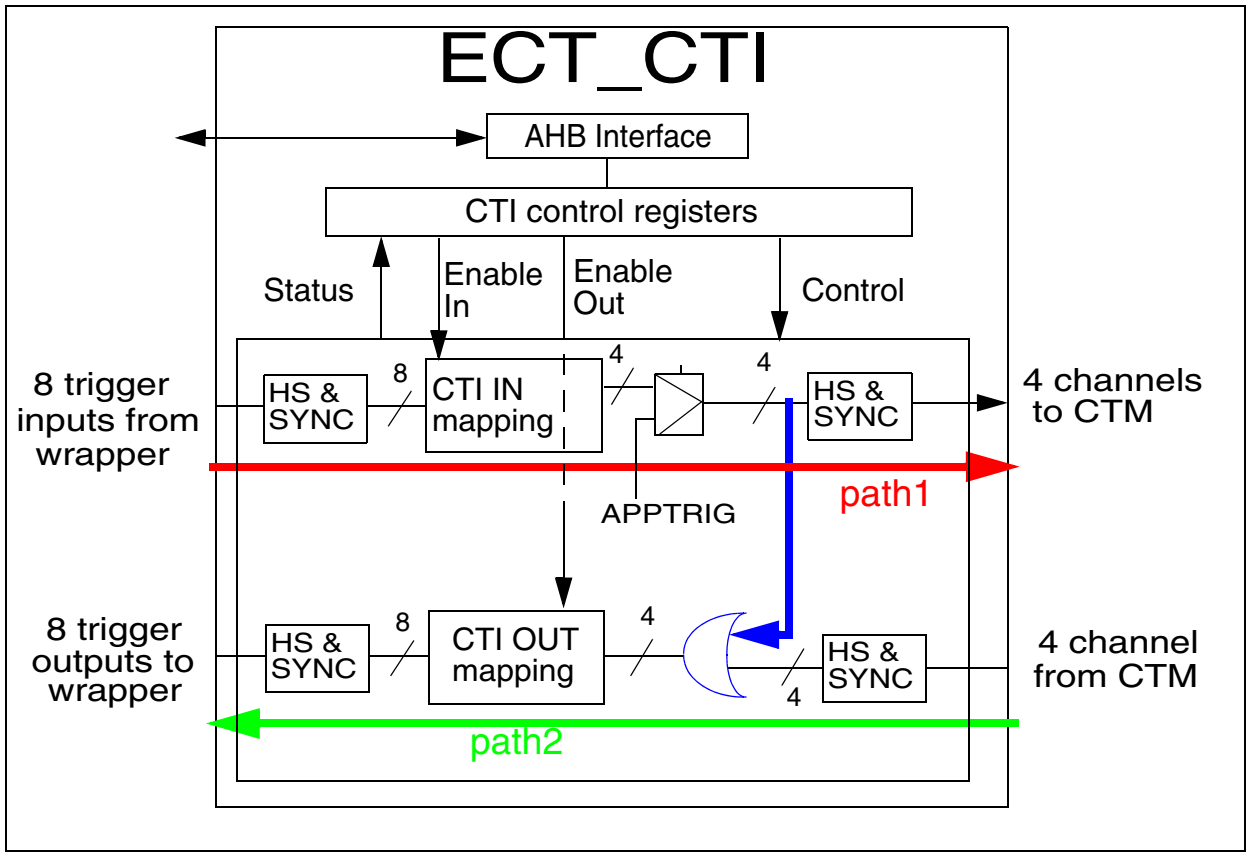


Figure 19-36. CTI Diagram

Note: The OR gate (in blue) is required because in the CTM, channel inputs are not routed back to their own channel outputs.

19.12.2.1 Mapping

On the wrapper side, ECT_CTI has 8 trigger inputs from the SoC and 8 trigger outputs to the SoC. On the CTM side, ECT_CTI has 4 channel inputs from the CTM and 4 channel outputs to the CTM.

Each trigger input can be connected to any of the 4 channels going to the CTM (path1). Note that several trigger inputs can be mapped to the same channel as actually, a channel input is an OR function of all enabled trigger inputs. The mapping triggers to channels is fully programmable using the memory-mapped registers CTIINEN (see Section 19.10.9, “CTIINEN0–7 Register”).

Each channel coming from the CTM can be mapped to any of the 8 trigger outputs of the ECT_CTI (path2). Note that several channels can be mapped to the same trigger output as a trigger output is an OR function of all the enabled channels. The mapping channels to triggers is fully programmable using the memory-mapped registers CTIOUTEN (see Section 19.10.10, “CTIOUTEN0–7 Register”).

19.12.2.2 Application Trigger

ECT_CTI enables to raise an event on one of the channels by writing a 1 to CTIAPPSET register (see Section 19.10.6, “CTIAPPSET Register”). To clear the application trigger, one has to assert to the

corresponding bit in CTIAPPCLEAR register (see Section 19.10.7, “CTIAPPCLEAR Register”). Each register has one bit corresponding to each of the 4 channels.

A pulse can also be generated by writing to APPPULSE register (see Section 19.10.8, “CTIAPPPULSE Register”).

19.12.3 IPS2AHB

This module is a bridge from the IP bus, where the EXTENDED_CTI stands, to the AHB (v2.0) interface provided by Arm Ltd.

Only 32-bits accesses are allowed otherwise an IP xfr error is generated.

Even if the bus clock is running, if ECT clocks are gated off (ect_clk_en = 0), neither read nor write access to memory mapped register are allowed and generates an ips_xfr_error if attempted.

19.13 Cross Trigger Matrix (CTM)

The Cross Trigger Matrix has 4 interfaces of 4 channel inputs and 4 channel outputs each.

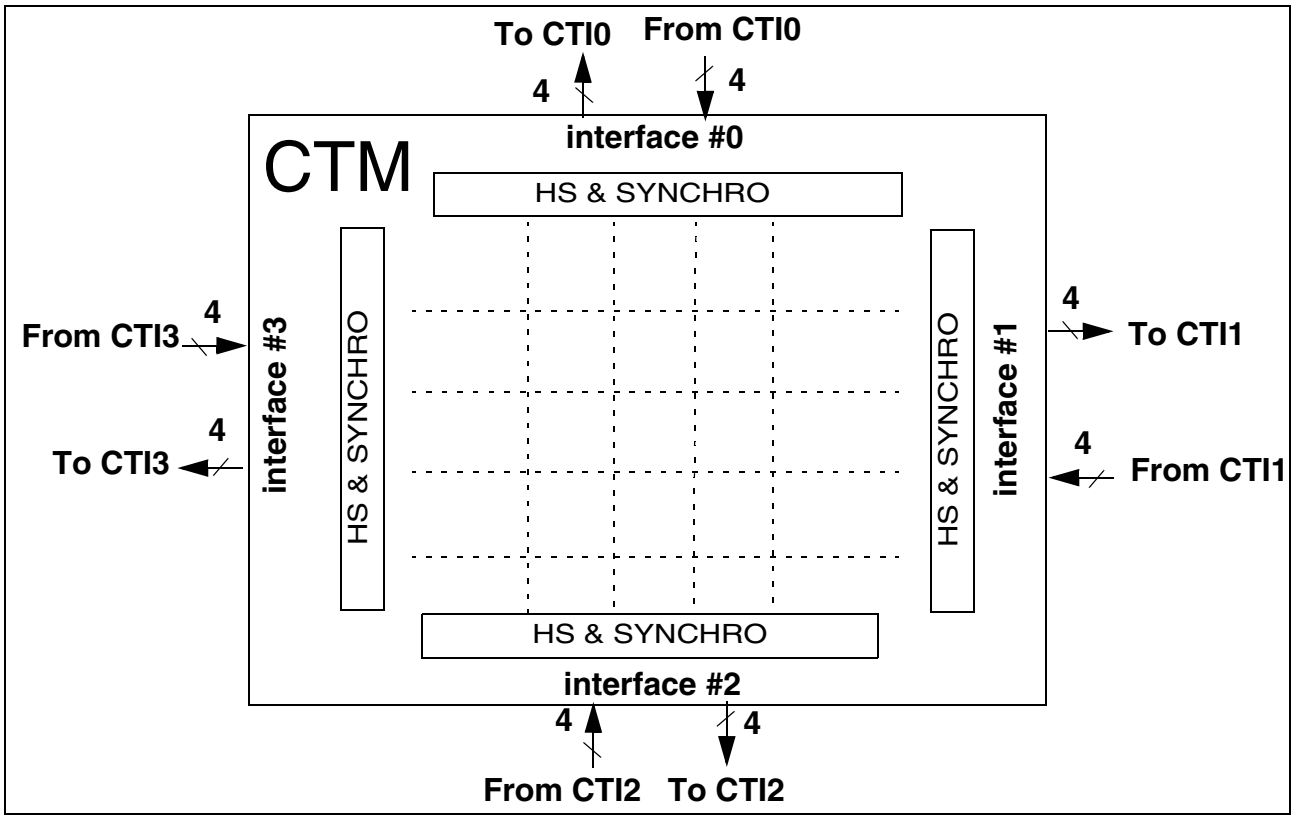


Figure 19-37. CTM Diagram

Each channel input of each interface is propagated to the other corresponding channel output of the 3 other interfaces so that when a channel input receives a signal, it is propagated to the channel output of the three

other ports. For example, the path in blue in the following figure shows the propagation of bit 0 of interface 2 (CHIN2[0]) to bit 0 of interfaces 0 (CHOUT0[0]), 1 (CHOUT1[0]), and 3 (CHOUT3[0]).

Moreover, each channel output of each interface is the logical OR of the 3 other channel inputs. For example, as shown in red in the following figure, CHIN0[0], CHIN1[0], and CHIN3[0] are ORed together to generate CHOUT2[0].

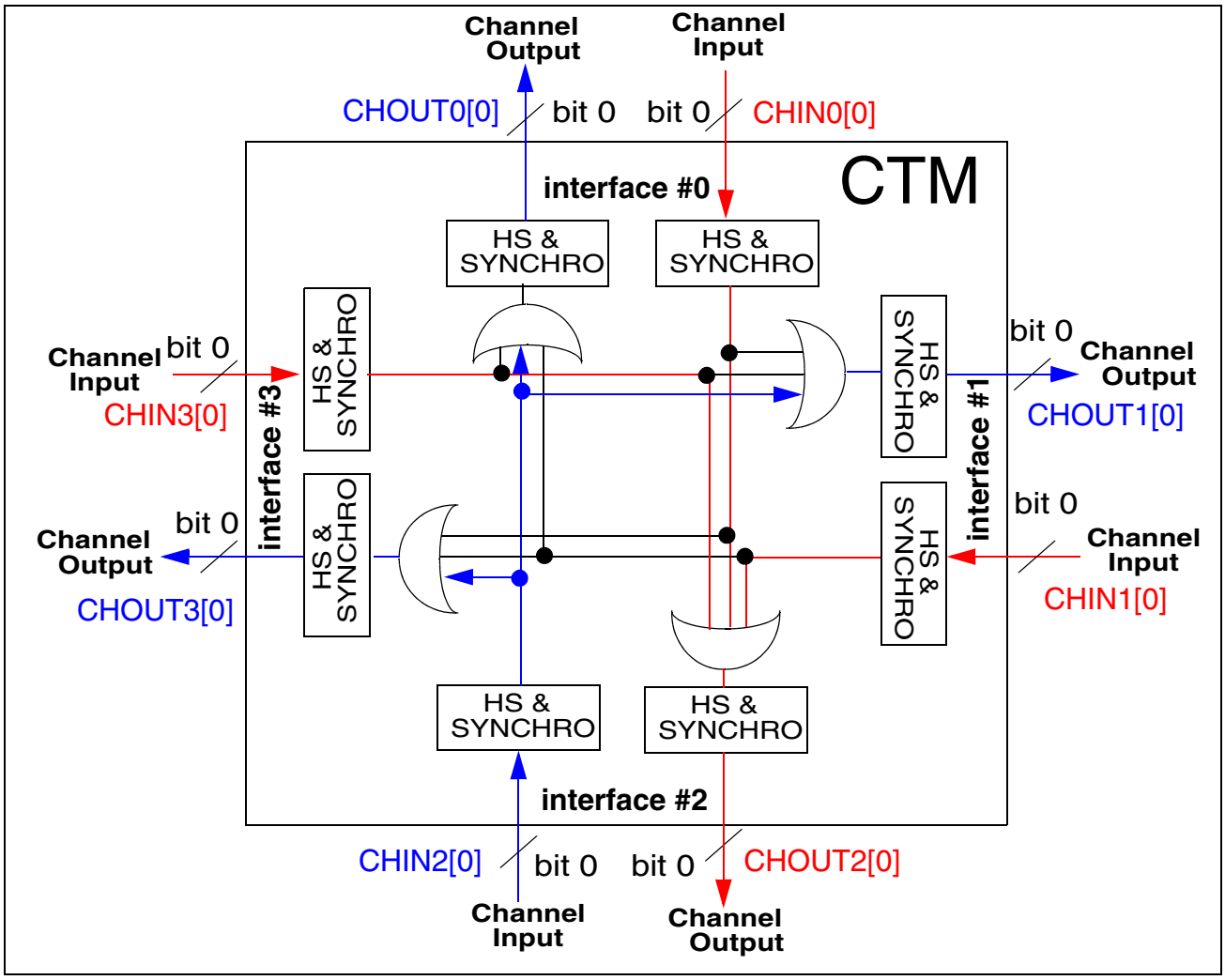


Figure 19-38. CTM Detailed Implementation for Channel 0

Each channel input is synchronized and acknowledged if required. Each channel output can be held until an acknowledgement is received.

All hand-shaking and synchronization mechanisms can be bypassed as the bypass control signals are brought to the ECT top level.

Note that in the CTM, channel inputs are not routed back to their own channel outputs, so this need to be performed in the ECT_CTI (OR gate in blue). Thus, a channel output is really a logical OR of the 4 corresponding channel inputs as shown below.

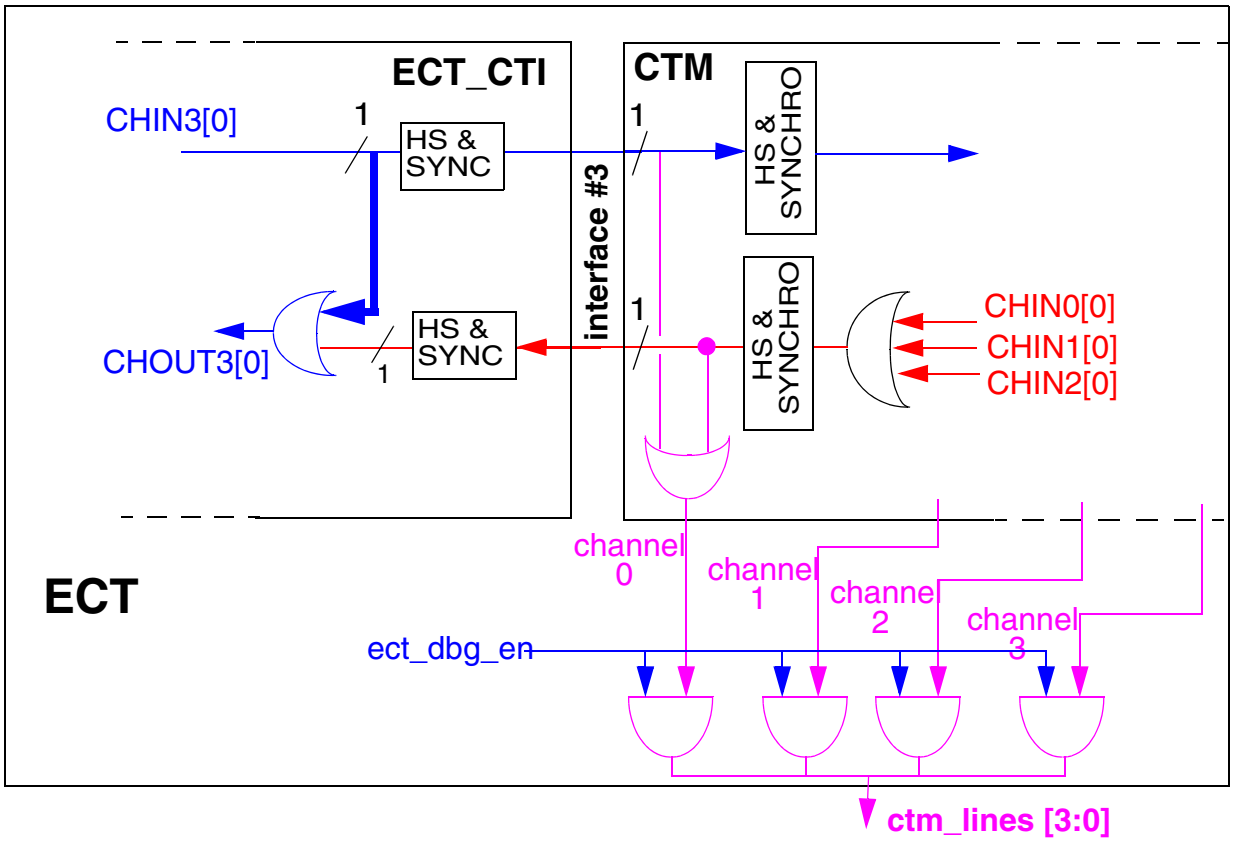


Figure 19-39. Channel Output and CTM Lines

Status of the 4 channels (0 to 3) can be observed through “ctm_lines[3:0]” that are brought to ECT top level.

19.14 Initialization/Application Information

19.15 Initialization

By default (reset), ECT is inactive. To enable cross-triggering functionality, the following steps are required:

1. **Unlock AHB interface**: to unlock AHB interface the correct 32-bit word (0x0ACCE550) has to be written in the CTILOCK register,
2. **Enable CTI logic**: To enable CTI, GLBEN bit in CTICONTROL register has to be set to 1.
3. **Enable mapping:** By default no mapping is enable. As a consequence, no signals can be propagated through CTI-CTM. To enable mapping following settings are needed:
   a) Trigger to channel mapping: Wanted mapping has to be specified for each CTI in CTIINEN registers,
   b) Channel to triggers mapping: Wanted mapping has to be specified for each CTI in CTIOUTEN registers.

4. **Access mode**: By default, both supervisor and user access modes are allowed for any read/write access of CTI registers. To only allow supervisor accesses, PROT bit in CTIPROTECTION register has to be set to 1(this operation need a supervisor write access).

NOTE

All those operations will be performed by software debug tools such as **RealView Debugger (RVD).**

19.16 Application information

If the hand-shaking is not bypassed, events close to one another (multi-shot) on the same trigger input are possibly merged into one event on an output trigger. Events arising from different interfaces and mapped to the same channel might also be merged due to the hand-shaking hardware.

Chapter 20
External Memory Interface (EMI)

The EMI provides the ability to connect the system to a wide variety of memory devices. This chapter contains technical information about the operation and configuration of the chip's EMI module, to allow the designer to quickly integrate external memory devices into new and existing designs. The chapter covers the following topics:

20.1 Overview

The EMI controls all IC external memory accesses (read/write/erase/program) from all the masters in the system to different external memories. All accesses are arbitrated by the multi-master memory interface (M3IF) submodule and controlled by the respective memory controller.

The EMI contains the following external memory controllers to support different types of memory devices:

- Enhanced SDRAM/LPDDR memory controller (ESDRAMC, also known as ESDRAMC/MDDRC, or ESDCTL/MDDRC). The ESDRAMC and ESDCTL mnemonics are equivalent; for historical reasons they are alternately used throughout the document.
- NAND Flash memory controller (NFC).
- Wireless external interface module (WEIM), which supports SRAM, PSRAM, and NOR Flash memory devices.

Figure 20-1 is a top-level diagram of the EMI that shows the functional organization of the block.

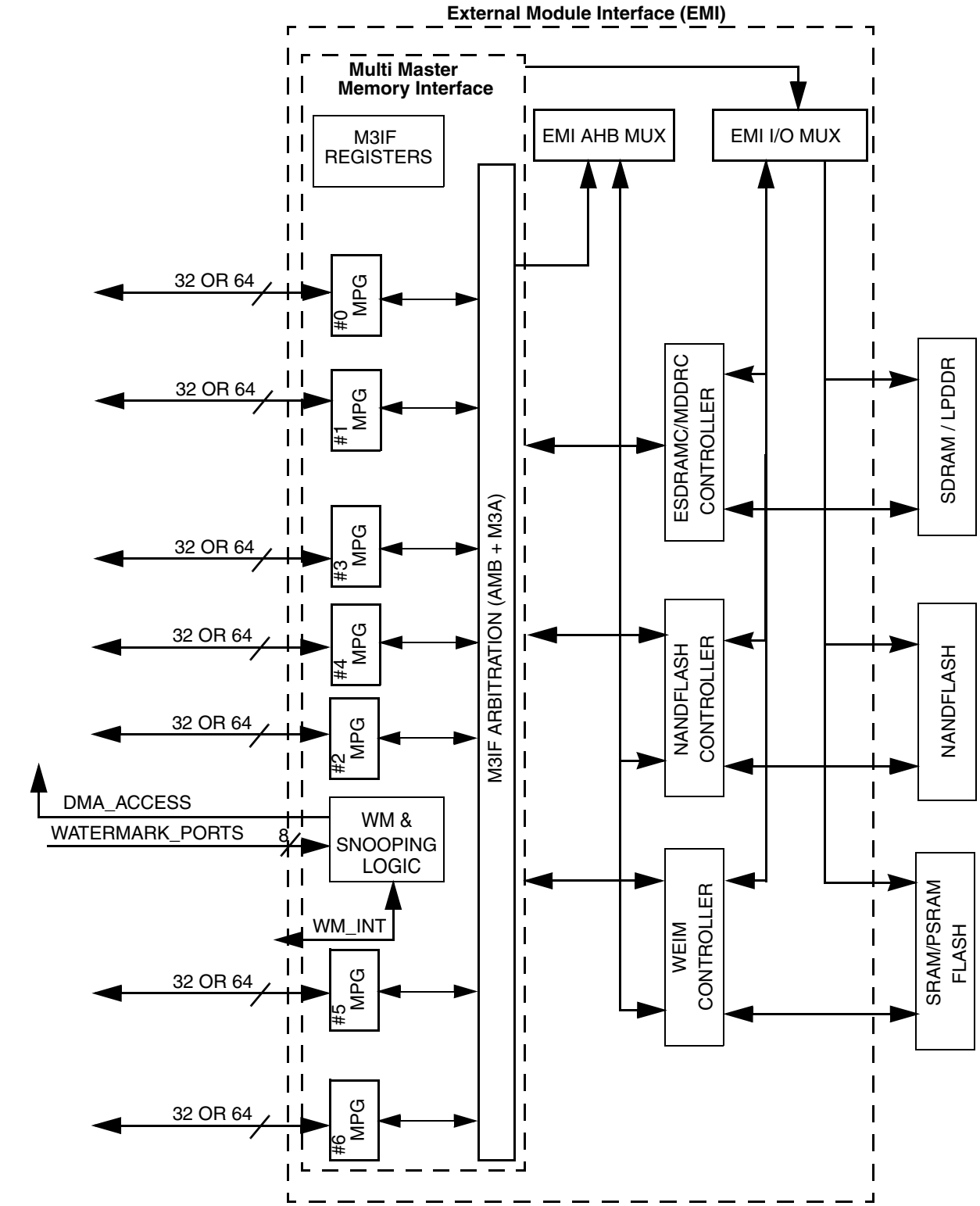


Figure 20-1. EMI System Block Diagram

20.1.1 Features

The EMI includes the following features:

- Multi master memory interface (M3IF)
  - Supports multiple requests from up to 8 masters through input ports interface.
  - Supports memory snooping: monitors a region of size 2 Kbytes–16 Mbytes in external memory for write accesses.
  - Supports memory watermark protection for up to 8 different chip selects for hardware-preselected masters.
- Enhanced SDRAM controller (ESDRAMC) / LPDDR controller (MDDRC)
  - Up to 2 chip selects (due to sharing of pins, 2 chip selects are supported only when the WEIM CS2 and CS3 are not is use).
  - Support x16 SDR SDRAM (up to 1-Gbit at133 MHz)
  - Support x16 LPDDR SDRAM (up to 1-Gbit at 266 MHz)
- NAND Flash controller (NFC)
  - Supports 8- and 16-bit NAND FLASH (up to 2 GB address space)
  - Internal 4.5 Kbyte RAM buffer.
- Wireless external interface memory controller (WEIM)
  - Up to five chip selects (due to sharing of pins, five chip selects are supported only when both ESDCTL/MDDRC and NAND FLASH chip selects are not in use).
  - Supports x16/x32 multiplexed/non-multiplexed mode for PSRAM and NOR Flash memory devices.
- Different memory controllers are accessible using AHB
- Pins are shared among memory controllers using the EMI AHB multiplexer and the EMI I/O multiplexer

20.2 EMI Input/Output Signals

Table 20-1 lists all of the EMI input and output signals. For the detailed description of each signal function, see the relevant module chapter in this document.

Table 20-1. EMI Signal Properties

Name	Port	Function	Reset State
AHB Interface Outputs			
M3IF_HREADY_M0	O	AHB access completion strobe to master #0	1
M3IF_HREADY_M1	O	AHB access completion strobe to master #1	1
M3IF_HREADY_M2	O	AHB access completion strobe to master #2	1
M3IF_HREADY_M3	O	AHB access completion strobe to master #3	1
M3IF_HREADY_M4	O	AHB access completion strobe to master #4	1

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HREADY_M5	O	AHB access completion strobe to master #5	1
M3IF_HREADY_M6	O	AHB access completion strobe to master #6	1
M3IF_HREADY_M7	O	AHB access completion strobe to master #7	1
M3IF_HRESP_M0	O	AHB error response to master #0	0
M3IF_HRESP_M1	O	AHB error response to master #1	0
M3IF_HRESP_M2	O	AHB error response to master #2	0
M3IF_HRESP_M3	O	AHB error response to master #3	0
M3IF_HRESP_M4	O	AHB error response to master #4	0
M3IF_HRESP_M5	O	AHB error response to master #5	0
M3IF_HRESP_M6	O	AHB error response to master #6	0
M3IF_HRESP_M7	O	AHB error response to master #7	0
M3IF_HRDATA_M0	O	AHB read data bus to master #0 (bus size is determined by system configuration)	0
M3IF_HRDATA_M1	O	AHB read data bus to master #1 (bus size is determined by system configuration)	0
M3IF_HRDATA_M2	O	AHB read data bus to master #2 (bus size is determined by system configuration)	0
M3IF_HRDATA_M3	O	AHB read data bus to master #3 (bus size is determined by system configuration)	0
M3IF_HRDATA_M4	O	AHB read data bus to master #4 (bus size is determined by system configuration)	0
M3IF_HRDATA_M5	O	AHB read data bus to master #5 (bus size is determined by system configuration)	0
M3IF_HRDATA_M6	O	AHB read data bus to master #6 (bus size is determined by system configuration)	0
M3IF_HRDATA_M7	O	AHB read data bus to master #7 (bus size is determined by system configuration)	0
M3IF and ESDRAMC/MDDRC Outputs			
IPP_DO_SDRC_SDCKE[1:0]	O	SDRAM/LPDDR clock enable	0
IPP_DO_EMI_DQM[3:0]	O	SDRAM data mask strobes. DQM0 corresponds to DQ0–DQ7, DQM1 corresponds to DQ8–DQ15, DQM2 corresponds to DQ16–DQ23 and DQM3 corresponds to DQ24–DQ31.	0
IPP_DO_DQS[3:0]	O	LPDDR data sample strobes for write accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0
IPP_OBE_DQS	O	DQS output enable strobe	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_DO_E M I _ADDR[25:0]	O	WEIM Address [25:0] multiplexed with SDR[13:0] (except for SDR[10])	0
IPP_DO_SDBA[1:0]	O	SDRAM/LPDDR bank address bits	0
IPP_DO_M3IF_MA10	O	SDRAM/LPDDR address bit A10	0
IPP_DO_M3IF_CAS_B	O	SDRAM/LPDDR CAS strobe	1
IPP_DO_M3IF_RAS_B	O	SDRAM/LPDDR RAS strobe	1
IPP_DO_SDRC	O	SDRAM/LPDDR WE strobe	1
M3IF_CHOOSEN_MASTER[2:0]	O	M3IF arbitration chosen master (for debug)	3
IPP_DO_SDRC_SDCLK	O	SDRAM/LPDDR clock (up to 133MHz)	0
LPACK	O	Low power mode acknowledge; toward CCM	1
SDRC_SF_WACK	O	Memory wake-up acknowledge indication to WDOG	0
NFC Ouputs			
IPI_INT_NFC_B	0	NFC interrupt (indicating an access completion)	1
IPP_NFC_ALE_OUT	O	NFC out NF_ALE	0
IPP_NFC_CE0_OUT	O	NFC out NF_CE0	0
IPP_NFC_CE1_OUT	O	NFC out NF_CE1	0
IPP_NFC_CE2_OUT	O	NFC out NF_CE2	0
IPP_NFC_CE3_OUT	O	NFC out NF_CE3	0
IPP_NFC_CLE_OUT	O	NFC out NF_CLE	0
IPP_NFC_RE_OUT	O	NFC out NF_RE	0
IPP_NFC_WE_OUT	O	NFC out NF_WE	0
IPP_NFC_WP_OUT	O	NFC out NF_WP	0
WEIM Outputs			
IPP_DO_WEIM_CS_B0	O	WEIM CS0 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_CS_B1	O	WEIM CS1 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_CS_B2_CSD0	O	WEIM CS2 or ESDRAMC/MDDRC CSD0 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_CS_B3_CSD1	O	WEIM CS2 or ESDRAMC/MDDRC CSD1 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_CS_B4	O	WEIM CS4 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_CS_B5	O	WEIM CS5 chip select toward I/O multiplexer/pins	1
IPP_DO_WEIM_BCLK	O	WEIM Burst Clock	0
IPP_DO_WEIM_LBA_B	O	WEIM Load Burst Address (LBA)	1

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPP_DO_WEIM_RW_B	O	WEIM read/write strobe	1
Global Ouputs			
M3IF_DMA_ACCESS	O	Snooping detection indication toward IPU module	0
IPP_OBE_DDR_EN	O	LPDDR active indication to ESDRAMC/MDDRC DATA pins	0
IPP_DO_EMI_ADDR[25:0]	O	EMI address out toward I/O multiplexer/pins	0
IPP_DO_NFC_WEIM_IO_DATA_OUT[15:0]	O	EMI WEIM/NFC data out toward I/O multiplexer/pins	0
IPP_DO_EMI_DATA[31:0]	O	EMI SDRAM/DDR data out toward I/O multiplexer/pins	0
IPP_OBE_EMI_DATA_DIR	O	EMI SDRAM/DDR data direction toward I/O multiplexer/pins	0
IPP_OBE_NFC_DIR_HIGH	O	EMI (NFC, WEIM) data direction toward I/O multiplexer/pins	0
IPP_OBE_NFC_DIR_LOW	O	EMI (NFC, WEIM) data direction toward I/O multiplexer/pins	0
IPP_DO_EMI_EB_B[1:0]	O	WEIM enable byte out toward I/O multiplexer/pins	0
IPP_DO_EMI_OE_B	O	EMI output enable toward I/O multiplexer/pins	0
IPP_OBE_IO_ADDR_DIR[1:0]	O	EMI output enable (dir) toward I/O ADDR/WEIM multiplexed DATA multiplexer/pins	0
AHB Interface Inputs			
M3IF_HADDR_M0[31:0]	I	AHB address bus from master #0	0
M3IF_HADDR_M1[31:0]	I	AHB address bus from master #1	0
M3IF_HADDR_M2[31:0]	I	AHB address bus from master #2	0
M3IF_HADDR_M3[31:0]	I	AHB address bus from master #3	0
M3IF_HADDR_M4[31:0]	I	AHB address bus from master #4	0
M3IF_HADDR_M5[31:0]	I	AHB address bus from master #5	0
M3IF_HADDR_M6[31:0]	I	AHB address bus from master #6	0
M3IF_HADDR_M7[31:0]	I	AHB address bus from master #7	0
M3IF_HWDATA_M0	I	AHB write data bus from master #0 (bus size is determined by system configuration). In case this master is a read only-master this signal is not routed as an EMI output.	0
M3IF_HWDATA_M1	I	AHB write data bus from master #1 (bus size is determined by system configuration)	0
M3IF_HWDATA_M2	I	AHB write data bus from master #2 (bus size is determined by system configuration)	0
M3IF_HWDATA_M3	I	AHB write data bus from master #3 (bus size is determined by system configuration)	0
M3IF_HWDATA_M4	I	AHB write data bus from master #4 (bus size is determined by system configuration)	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HWDATA_M5	I	AHB write data bus from master #5 (bus size is determined by system configuration)	0
M3IF_HWDATA_M6	I	AHB write data bus from master #6 (bus size is determined by system configuration)	0
M3IF_HWDATA_M7	I	AHB write data bus from master #7 (bus size is determined by system configuration)	0
M3IF_HBURST_M0[2:0]	I	AHB burst size bus from master #0	0
M3IF_HBURST_M1[2:0]	I	AHB burst size bus from master #1	0
M3IF_HBURST_M2[2:0]	I	AHB burst size bus from master #2	0
M3IF_HBURST_M3[2:0]	I	AHB burst size bus from master #3	0
M3IF_HBURST_M4[2:0]	I	AHB burst size bus from master #4	0
M3IF_HBURST_M5[2:0]	I	AHB burst size bus from master #5	0
M3IF_HBURST_M6[2:0]	I	AHB burst size bus from master #6	0
M3IF_HBURST_M7[2:0]	I	AHB burst size bus from master #7	0
M3IF_HSIZE_M0[1:0]	I	AHB data transfer width bus from master #0	0
M3IF_HSIZE_M1[1:0]	I	AHB data transfer width bus from master #1	0
M3IF_HSIZE_M2[1:0]	I	AHB data transfer width bus from master #2	0
M3IF_HSIZE_M3[1:0]	I	AHB data transfer width bus from master #3	0
M3IF_HSIZE_M4[1:0]	I	AHB data transfer width bus from master #4	0
M3IF_HSIZE_M5[1:0]	I	AHB data transfer width bus from master #5	0
M3IF_HSIZE_M6[1:0]	I	AHB data transfer width bus from master #6	0
M3IF_HSIZE_M7[1:0]	I	AHB data transfer width bus from master #7	0
M3IF_HBSTRB_M0	I	Byte lane (8) bus from master #0 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M1	I	Byte lane (8) bus from master #1 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M2	I	Byte lane (4) bus from master #2 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M3	I	Byte lane (4) bus from master #3 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M4	I	Byte lane (4) bus from master #4 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M5	I	Byte lane (4) bus from master #5 (bus size is determined by system configuration)	0
M3IF_HBSTRB_M6	I	Byte lane (4) bus from master #6 (bus size is determined by system configuration)	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HBSTRB_M7	I	Byte lane (4) bus from master #7 (bus size is determined by system configuration)	0
M3IF_HTRANS_M0[1:0]	I	AHB transfer state bus from master #0	0
M3IF_HTRANS_M1[1:0]	I	AHB transfer state bus from master #1	0
M3IF_HTRANS_M2[1:0]	I	AHB transfer state bus from master #2	0
M3IF_HTRANS_M3[1:0]	I	AHB transfer state bus from master #3	0
M3IF_HTRANS_M4[1:0]	I	AHB transfer state bus from master #4	0
M3IF_HTRANS_M5[1:0]	I	AHB transfer state bus from master #5	0
M3IF_HTRANS_M6[1:0]	I	AHB transfer state bus from master #6	0
M3IF_HTRANS_M7[1:0]	I	AHB transfer state bus from master #7	0
M3IF_HWRITE_M0	I	AHB read/write signal from master #0	0
M3IF_HWRITE_M1	I	AHB read/write signal from master #1	0
M3IF_HWRITE_M2	I	AHB read/write signal from master #2	0
M3IF_HWRITE_M3	I	AHB read/write signal from master #3	0
M3IF_HWRITE_M4	I	AHB read/write signal from master #4	0
M3IF_HWRITE_M5	I	AHB read/write signal from master #5	0
M3IF_HWRITE_M6	I	AHB read/write signal from master #6	0
M3IF_HWRITE_M7	I	AHB read/write signal from master #7	0
M3IF_HPROT_M0	I	AHB protection mode signal from master #0	0
M3IF_HPROT_M1	I	AHB protection mode signal from master #1	0
M3IF_HPROT_M2	I	AHB protection mode signal from master #2	0
M3IF_HPROT_M3	I	AHB protection mode signal from master #3	0
M3IF_HPROT_M4	I	AHB protection mode signal from master #4	0
M3IF_HPROT_M5	I	AHB protection mode signal from master #5	0
M3IF_HPROT_M6	I	AHB protection mode signal from master #6	0
M3IF_HPROT_M7	I	AHB protection mode signal from master #7	0
M3IF_HUNALIGN_M0	I	Unalign access signal from master #0	0
M3IF_HUNALIGN_M1	I	Unalign access signal from master #1	0
M3IF_HUNALIGN_M2	I	Unalign access signal from master #2	0
M3IF_HUNALIGN_M3	I	Unalign access signal from master #3	0
M3IF_HUNALIGN_M4	I	Unalign access signal from master #4	0
M3IF_HUNALIGN_M5	I	Unalign access signal from master #5	0
M3IF_HUNALIGN_M6	I	Unalign access signal from master #6	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_HUNALIGN_M7	I	Unalign access signal from master #7	0
M3IF and ESDRAMC/MDDRC Inputs			
M3IF_HCLK	I	M3IF AHB system clock up to 133MHz	0
HCLK32	I	32 KHz clock for ESDRAMC refresh counter	0
IPP_IND_SDRC_SDCLK_FB	I	SDRAM/LPDDR feedback clock (up to 133MHz)	0
LPMD	I	Low power mode indication signal, “0”=STOP, “1”=RUN.	1
IPP_IND_DQS[3:0]	I	LPDDR data sample strobes for read accesses. DQS0 corresponds to DQ0–DQ7, DQS1 corresponds to DQ8–DQ15, DQS2 corresponds to DQ16–DQ23 and DQS3 corresponds to DQ24–DQ31.	0
SDCTL_CSD0_SEL_B	I	SDRAM/LPDDR CSD0 select multiplexed with CS2 (configurable using the system control register, FMCR)	0
SDCTL_CSD1_SEL_B	I	SDRAM/LPDDR CSD1 select multiplexed with CS3 (configurable using the system control register, FMCR)	0
NFC Inputs			
NF16_BOOT_B	I	Boot mode source is 16 bit NAND Flash memory.	Application dependent
NF8_BOOT_B	I	Boot mode source is 8-bit NAND Flash memory.	Application dependent
NF_16BIT_SEL	I	16-bit NAND Flash memory is use indication.	0
NFC_HCLK	I	NFC AHB input clock	0
NFC_RD_OE	I	NFC read output enable controls the direction of data bus	0
IPP_IND_FLASH_CLK	I	NAND Flash side clock with period of 40 nS	0
IPP_IND_NFC_RB_IN	I	NFC in NF_RB	0
WEIM Inputs			
WEIM_BOOT_CFG[2:0]	I	WEIM boot mode select (from ccm) For detailed boot description see the WEIM specification	Application dependent
IPP_IND_WEIM_ECB_B	I	WEIM end current burst	1
WEIM_HCLK	I	WEIM AHB input clock	0
IPP_IND_WEIM_DTACK_B	I	External DTACK acknowledge	1
Global Inputs			
IPP_IND_RESETB	I	Reset signal	1
IPP_IND_NFC_READ_DATA_IN[15:0]	I	External memories (non SDRAM) read data in from I/O multiplexer/pins	0
IPP_IND_EMI_DATA_IN[31:0]	I	EMI SDRAM/DDR data in from I/O multiplexer/pins	0
IPP_IND_ADDR_IN[15:0]	I	EMI WEIM multiplexed data in from I/O multiplexer/pins.	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
M3IF_BIGEND_M0	I	Endian mode signal from master #0	Master dependent
M3IF_BIGEND_M1	I	Endian mode signal from master #1	Master dependent
M3IF_BIGEND_M2	I	Endian mode signal from master #2	Master dependent
M3IF_BIGEND_M3	I	Endian mode signal from master #3	Master dependent
M3IF_BIGEND_M4	I	Endian mode signal from master #4	Master dependent
M3IF_BIGEND_M5	I	Endian mode signal from master #5	Master dependent
M3IF_BIGEND_M6	I	Endian mode signal from master #6	Master dependent
M3IF_BIGEND_M7	I	Endian mode signal from master #7	Master dependent
WATERMARK_PORT[7:0]	I	Selects the watermark ports. If watermark is not enabled this port is not routed as EMI port.	0
IPI_INT_WATERMARK		Watermark interrupt. If watermark is not enabled this port is not routed as EMI port.	0
WARM_RESET	I	Warm reset indication. If ESDRAMC doesn't support warm_reset, than this port is not routed as EMI port.	0
M3IF_HMASTLOCK_M0	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M1	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M2	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M3	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M4	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M5	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M6	—	Master arbitration locking signal	0
M3IF_HMASTLOCK_M7	—	Master arbitration locking signal	0
DMA_REQ	—	NFC DMA request	0
HRESET_B	—	Reset signal	1
HRESET_NEG_B	—	Reset signal	1
IPP_DO_EMI_OE_B	—	WEIM Data pads direction	0
IPP_IND_WEIM_BCLK_FB	—	WEIM feedback clock	0
IPT_MODE	—	Scan signal	0
IPT_MODE_BURNIN	—	Scan signal	0

Table 20-1. EMI Signal Properties (continued)

Name	Port	Function	Reset State
IPT_RAM_SE	—	Scan signal	0
IPT_SE	—	Scan signal	0
IPT_SE_ASYNC	—	Scan signal	0
IPT_SE_GATEDCLK	—	Scan signal	0
JTA_OR_IOB_BIST_BITMAPPING	—	NFC JTAG signal	0
JTAGC_BIST_NAND_CLOCK_DR	—	NFC JTAG signal	0
JTAGC_BIST_NAND_TDI	—	NFC JTAG signal	0
NAND_JTAGC_BIST_TDO	—	NFC JTAG signal	0
NFC_FMS	—	NFC 512-byte / 2-Kbyte page size	
NF_4K	—	8-bit ECC signal	
NF_BOOT_WITH_RESET	—	Flash memory is MLC type	
EMI_SPARE_PORT_IN[9:0]	—	Spare inputs	0
EMI_SPARE_PORT_OUT[9:0]	—	spare outputs	0
IPT_SI[59:0]	—	Scan inputs	0
IPT_SO[59:0]	—	Scan outputs	0
DELAY_LINE_REF_CLK	—	Reference clock to the delay line	0
DVFS_REQ	—	DVFS request signal	0
DVFS_GRANT	—	DVFS acknowledge signal	0
VNW	—	WT_SHORT signal	0
VPW	—	WT_SHORT signal	0
WT_EN_DNW	—	WT_SHORT signal	0
WT_EN	—	WT_SHORT signal	0
NON_MOBILE_DDR_FUSE	—	DDR Type select	0

20.3 Memory Map/Register Definition

Table 20-2 shows the address ranges for the four different controllers' registers (M3IF plus three external memory controllers). Refer to the separate memory controller chapters for detailed descriptions of the memory controllers' registers.

Table 20-2. EMI Registers Address Ranges

Address Range	Use	Access
0xB800_3000 – 0xB800_3FFF	M3IF registers space (4 Kbytes)	Read/write
0xB800_1000 – 0xB800_1FFF	ESDRAMC/MDDRC registers space (4 Kbytes)	Read/write

Table 20-2. EMI Registers Address Ranges (continued)

0xB800_2000 – 0xB800_2FFF	WEIM registers space (4 Kbytes)	Read/write
0xBB00_1E00 – 0xBB00_1EFF	NFC registers space (4 Kbytes)	Read/write

Table 20-3 shows the memory map, including memory spaces allocated to the three different external controllers.

Table 20-3. EMI Memory Map

Address Range	Use	Access
ESDRAMC/MDDRC Memory Space		
0x8000_0000 – 0x8FFF_FFFF	CSD0 SDRAM/LPDDR memory region (256 Mbytes)	Read/write
0x9000_0000 – 0x9FFF_FFFF	CSD1 SDRAM/LPDDR memory region (256 Mbytes)	Read/write
WEIM Memory Space		
0xA000_0000 – 0xA7FF_FFFF	WEIM CS0 memory region[1] (128 Mbytes)	Read/write
0xA800_0000 – 0xAFFF_FFFF	WEIM CS1 memory region (128 Mbytes)	Read/write
0xB000_0000 – 0xB1FF_FFFF	WEIM CS2 memory region (32 Mbytes)7	Read/write
0xB200_0000 – 0xB3FF_FFFF	WEIM CS3 memory region (32 Mbytes)	Read/write
0xB400_0000 – 0xB5FF_FFFF	WEIM CS4 memory region (32 Mbytes)	Read/write
0xB600_0000 – 0xB7FF_FFFF	WEIM CS5 memory region (32 Mbytes)	Read/write
NFC Memory Space		
0xBB00_0000 – 0xBB00_11FF	NFC memory region[1] (4.5 Kbytes, NAND Flash)	Read/write

1. Can be used as a boot memory region.

20.4 Functional Description

This section provides a functional description of the multi-master memory interface (M3IF), the three external memory controllers, and the EMI's AHB and I/O multiplexers.

20.4.1 Multi-Master Memory Interface (M3IF)

When a master requests a memory access, the access is immediately taken by the M3IF if no other access is in progress. The M3IF forwards the access to the respective memory controller (slave). When the access execution is completed, HREADY is asserted and a new request can be processed.

The interface between M3IF and ESDRAMC is optimized to reduce access latency by generating multiple accesses through the dedicated ESDRAMC arbitration (MAB) module, which controls the access to/from the ESDRAMC.

For the other memory interfaces, the M3IF receives masters' requests through the master port gasket (MPG) interfaces, performs arbitration, and forwards each request to the appropriate memory controller.

20.4.2 NAND Flash Controller (NFC)

The NFC provides an interface between standard NAND Flash devices and the IC and hides the complexities of accessing a NAND Flash memory device. It provides a glueless interface to 8- or 16-bit SLC or MLC NAND Flash devices with different page size. Figure 20-2 is a simplified block diagram of the NFC.

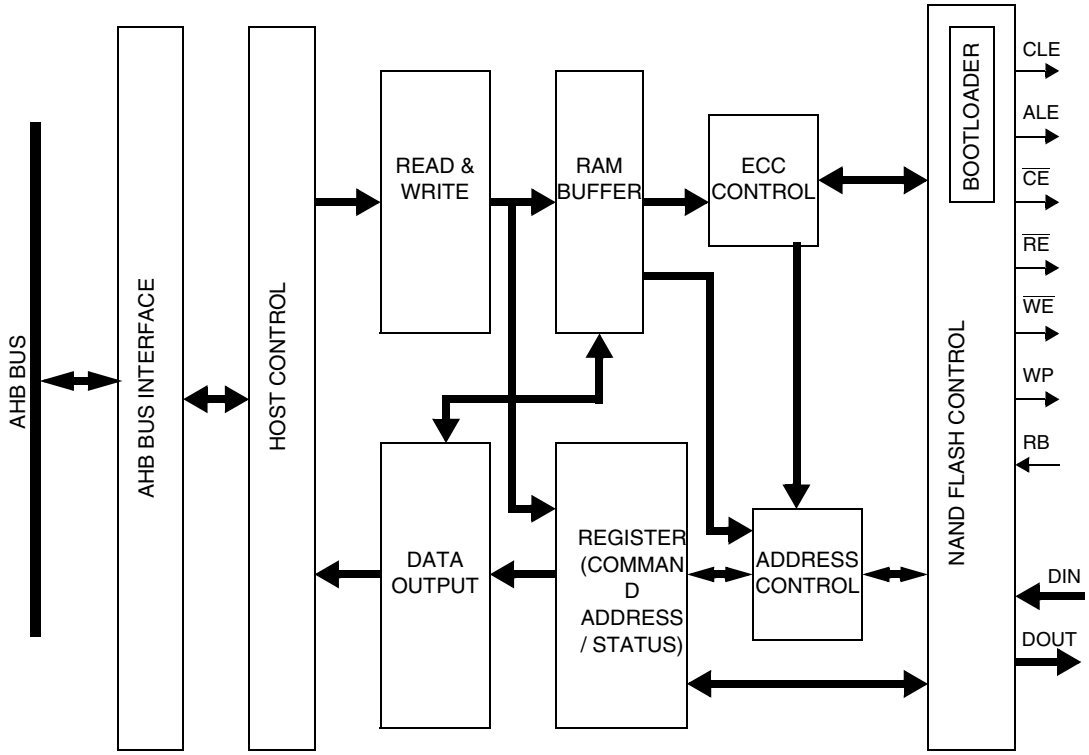


Figure 20-2. NAND Flash Controller Simplified Block Diagram

20.4.2.1 NFC Operation

Communication with a Flash memory device begins by the AHB host initiating a read from the NFC. This is accomplished by configuring the NFC and then waiting for an interrupt from the Flash memory device to be generated. When the NFC receives the interrupt, it inputs a page from the Flash memory device, and upon completion generates an interrupt to the AHB host.

When the AHB host receives the NFC interrupt, it reads the content from the internal RAM buffer of the NFC. To complete the operation the AHB host checks the status of the operation by reading the NFC status registers.

Data that is exchanged with the Flash memory device is temporarily maintained in the RAM buffer. This buffer is used as the boot RAM during a cold reset (if the IC is configured to boot from the NAND Flash device). After the boot load completes, the RAM is available as buffer RAM for normal Flash memory operations.

20.4.2.2 NFC Internal and External Communications

To ensure the greatest degree of flexibility, the NFC provides an internal interface to the AHB bus allowing 16-bit or 32-bit bus transfers, and a signal-selectable 8- or 16-bit interface to the external NAND Flash memory device.

All communication between the NFC and the ARM11 platform passes through the AHB host. The host configures and controls the NFC using the NFC registers.

Data integrity of the NAND Flash is maintained by the NFC. The NFC automatically generates the ECC for verification during a read or program operation.

20.4.2.3 NFC Sharing of I/O Pins

The NFC provides necessary logic to share I/O pins with other memory controllers. For example, when interfacing with a PSRAM, the 16 I/O signals of the NAND Flash controller share the same I/O pins with the data signals of the wireless external interface module (WEIM).

When a request to free the pins is asserted, the NFC state machine halts and the NAND Flash signals when it finishes the current transfer. The other memory controller is then able to gain control of the pins.

Since the NAND Flash memory accesses are typically long and relatively slow, priority is given to the other memory controller sharing the pins. The NFC waits until the other memory controller is finished with its operation and the pins are free before it continues its accesses.

20.4.3 Enhanced SDRAM Controller (ESDRAMC)

The ESDRAMC (equivalently denoted as ESDCTL) provides interface, configuration and control for many different types of synchronous SDRAM and low power mobile DDR (LPDDR) memories. Figure 20-3 shows the functional organization of the enhanced SDRAM controller.

The enhanced SDRAM controller consists of nine major blocks:

- SDRAM command state machine controller,
- Bank register (page and bank address comparators),
- Row/column address multiplexer,
- Configuration registers,
- Refresh request counter,
- Command sequencer,
- Size logic (splitting access),
- Data path (data aligner/multiplexer),
- LPDDR interface
- Power-down timer.

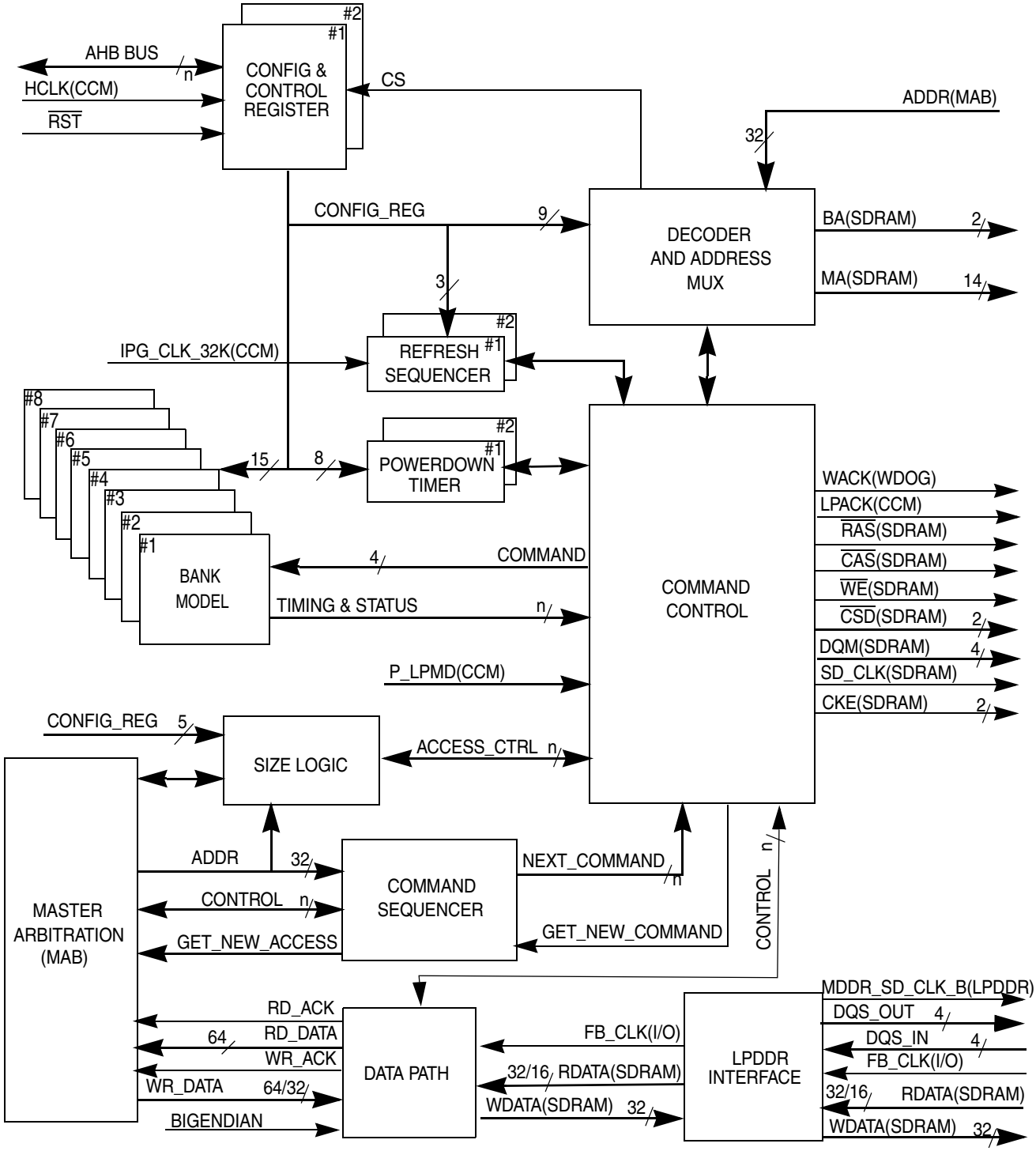


Figure 20-3. Enhanced SDR/LPDDR SDRAM Controller Block Diagram

20.4.4 EMI AHB Multiplexer

The EMI AHB multiplexer controls the traffic on the AHB bus (address and controls) between the memory controllers and the internal peripherals (using the M3IF). The EMI uses several muxes/glue logic to control the traffic on the AHB bus.

Only a few AHB signals/buses are routed through the EMI AHB multiplexer to the memory controllers. Most of the AHB buses (data, address and controls) are directly routed from the M3IF to the memory controllers.

20.4.4.1 Overview of EMI AHB Multiplexer Operation

Figure 20-4 illustrates the EMI AHB multiplexer block diagram. The interface is compatible with the ARM 11 AMBA-AHB lite standard (for instance, it does not support RETRY and SPLIT transfers). All AHB signals that are not shown in Figure 20-4 are directly routed between the M3IF and the relevant memory controllers. For the entire list of AHB signals, see Table 20-1 and to the relevant memory controller specification document.

The EMI AHB multiplexer generates the HSEL signals for all memory controllers, excluding the ESDRAMC (which is generated within the M3IF due to latency-hiding logic).

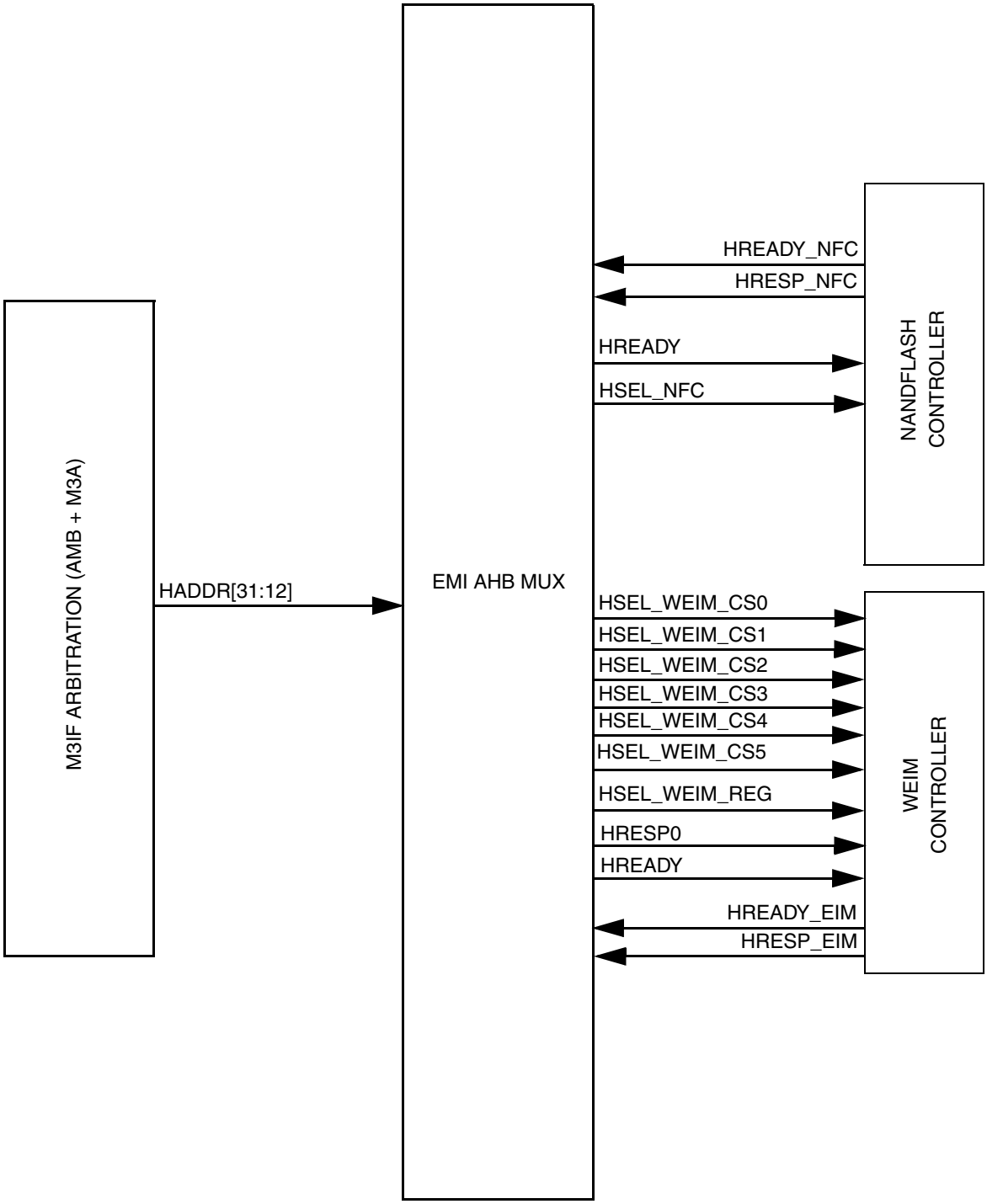


Figure 20-4. EMI AHB Multiplexer Interface Diagram

20.4.5 EMI I/O Multiplexer

The EMI I/O multiplexer controls the traffic (data, address and control signals) between the memory controllers and the external devices (using the IC IOMUX pins).

Figure 20-5 provides a top-level diagram of the EMI I/O multiplexer. Signals that share IC pins are routed through the EMI I/O multiplexer to the external devices. Signals that have dedicated pins (such as control signals) are not shown in Figure 20-5: these are directly routed from the memory controllers to the external devices. For a complete list of signals, see Table 20-1 and the relevant memory controller section.

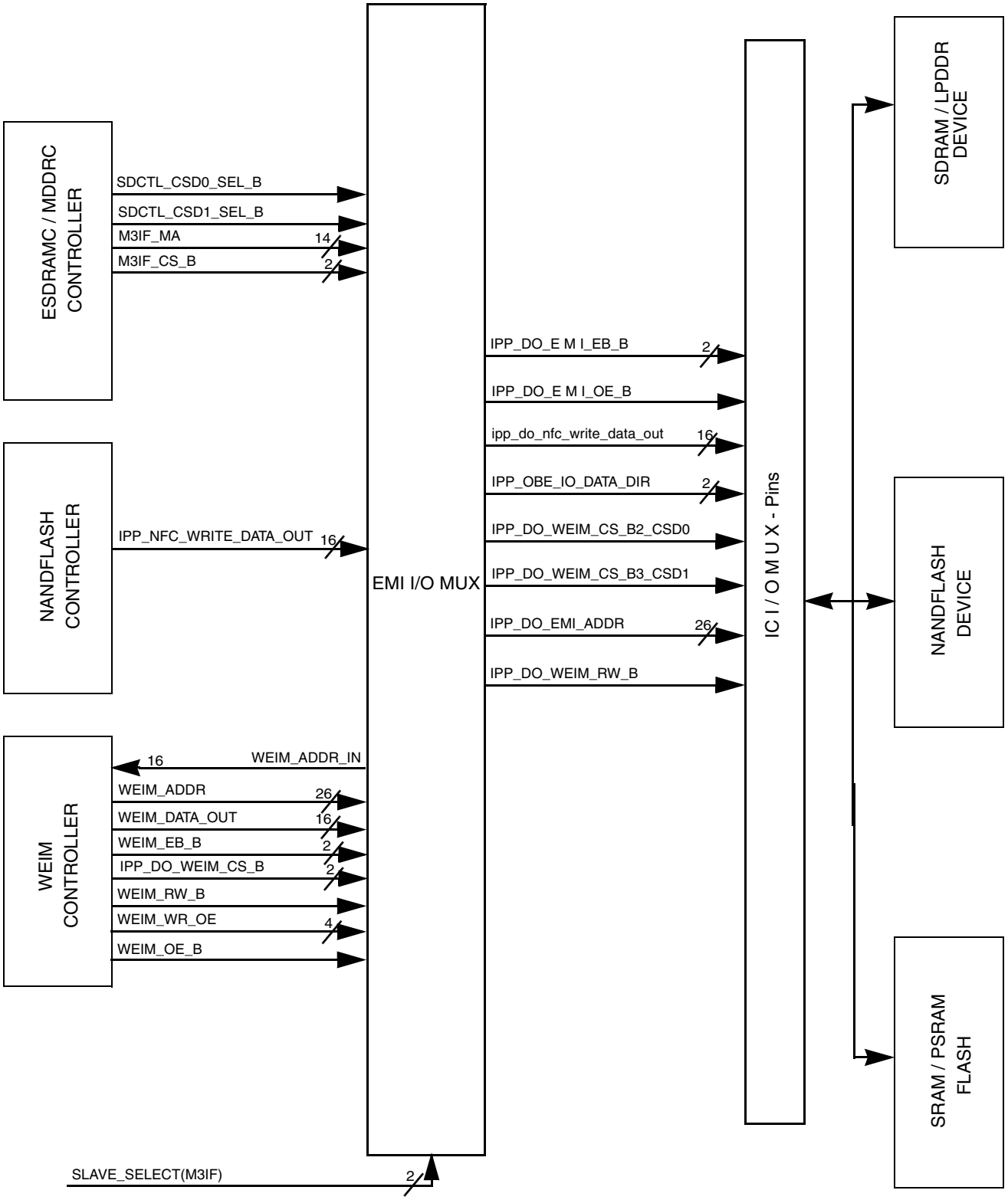


Figure 20-5. EMI I/O Multiplexer Interface Diagram

20.4.5.1 Overview of EMI I/O Multiplexer Operation

The active memory controller is determined by the SLAVE_SELECT[1:0] signal driven by the M3IF. Table 20-4 lists the memory controllers associated with different SLAVE_SELECT values.

Table 20-4. SLAVE_SELECT Memory Controller Selection

SLAVE_SELECT Value	Selected Memory Controller
00	ESDRAMC/MDDRC
01	WEIM
10	Reserved
11	NFC

Table 20-5 summarizes the EMI outputs to IC pins, including dedicated pins and pins which are shared between controllers.

Table 20-5. EMI Outputs to IC Pins

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
WEIM/ESDRAMC Address Multiplexing/WEIM Multiplexed Mode Data[15:0]					
MA[0]		IPP_DO_WEIM_ADDR_DATA_OUT[0]	—	IPP_DO_E M I _ADDR [0]	A0
—		IPP_IND_WEIM_ADDR_DATA_IN[0]		IPP_IND_ADDR_IN [0]	
MA[1]		IPP_DO_WEIM_ADDR_DATA_OUT[1]	—	IPP_DO_E M I _ADDR [1]	A1
—		IPP_IND_WEIM_ADDR_DATA_IN[1]		IPP_IND_ADDR_IN [1]	
MA[2]		IPP_DO_WEIM_ADDR_DATA_OUT[2]	—	IPP_DO_E M I _ADDR [2]	A2
—		IPP_IND_WEIM_ADDR_DATA_IN[2]		IPP_IND_ADDR_IN [2]	
MA[3]		IPP_DO_WEIM_ADDR_DATA_OUT[3]	—	IPP_DO_E M I _ADDR [3]	A3
—		IPP_IND_WEIM_ADDR_DATA_IN[3]		IPP_IND_ADDR_IN [3]	
MA[4]		IPP_DO_WEIM_ADDR_DATA_OUT[4]	—	IPP_DO_E M I _ADDR [4]	A4
—		IPP_IND_WEIM_ADDR_DATA_IN[4]		IPP_IND_ADDR_IN [4]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
MA[5]		IPP_DO_WEIM_ADDR_DATA_OUT[5]	—	IPP_DO_E M I _ADDR [5]	A5
—		IPP_IND_WEIM_ADDR_DATA_IN[5]		IPP_IND_ADDR_IN [5]	
MA[6]		IPP_DO_WEIM_ADDR_DATA_OUT[6]	—	IPP_DO_E M I _ADDR [6]	A6
—		IPP_IND_WEIM_ADDR_DATA_IN[6]		IPP_IND_ADDR_IN [6]	
MA[7]		IPP_DO_WEIM_ADDR_DATA_OUT[7]	—	IPP_DO_E M I _ADDR [7]	A7
—		IPP_IND_WEIM_ADDR_DATA_IN[7]		IPP_IND_ADDR_IN [7]	
MA[8]		IPP_DO_WEIM_ADDR_DATA_OUT[8]	—	IPP_DO_E M I _ADDR [8]	A8
—		IPP_IND_WEIM_ADDR_DATA_IN[8]		IPP_IND_ADDR_IN [8]	
MA[9]		IPP_DO_WEIM_ADDR_DATA_OUT[9]	—	IPP_DO_E M I _ADDR [9]	A9
—		IPP_IND_WEIM_ADDR_DATA_IN[9]		IPP_IND_ADDR_IN [9]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[10]	—	IPP_DO_E M I _ADDR [10]	A10
—		IPP_IND_WEIM_ADDR_DATA_IN[10]		IPP_IND_ADDR_IN [10]	
MA[11]		IPP_DO_WEIM_ADDR_DATA_OUT[11]	—	IPP_DO_E M I _ADDR [11]	A11
—		IPP_IND_WEIM_ADDR_DATA_IN[11]		IPP_IND_ADDR_IN [11]	
MA[12]		IPP_DO_WEIM_ADDR_DATA_OUT[12]	—	IPP_DO_E M I _ADDR [12]	A12
—		IPP_IND_WEIM_ADDR_DATA_IN[12]		IPP_IND_ADDR_IN [12]	
MA[13]		IPP_DO_WEIM_ADDR_DATA_OUT[13]	—	IPP_DO_E M I _ADDR [13]	A13
—		IPP_IND_WEIM_ADDR_DATA_IN[13]		IPP_IND_ADDR_IN [13]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[14]	—	IPP_DO_E M I _ADDR [14]	A14
		IPP_IND_WEIM_ADDR_DATA_IN[14]		IPP_IND_ADDR_IN [14]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
—		IPP_DO_WEIM_ADDR_DATA_OUT[15]	—	IPP_DO_E M I _ADDR [15]	A15
		IPP_IND_WEIM_ADDR_DATA_IN[15]		IPP_IND_ADDR_IN [15]	
—		IPP_DO_WEIM_ADDR_OUT[16]	—	IPP_DO_E M I _ADDR [16]	A16
—		IPP_DO_WEIM_ADDR[17]	—	IPP_DO_E M I _ADDR [17]	A17
—		IPP_DO_WEIM_ADDR[18]	—	IPP_DO_E M I _ADDR [18]	A18
—		IPP_DO_WEIM_ADDR[19]	—	IPP_DO_E M I _ADDR [19]	A19
—		IPP_DO_WEIM_ADDR[20]	—	IPP_DO_E M I _ADDR [20]	A20
—		IPP_DO_WEIM_ADDR[21]	—	IPP_DO_E M I _ADDR [21]	A21
—		IPP_DO_WEIM_ADDR[22]	—	IPP_DO_E M I _ADDR [22]	A22
—		IPP_DO_WEIM_ADDR[23]	—	IPP_DO_E M I _ADDR [23]	A23
—		IPP_DO_WEIM_ADDR[24]	—	IPP_DO_E M I _ADDR [24]	A24
—		IPP_DO_WEIM_ADDR[25]	—	IPP_DO_E M I _ADDR [25]	A25
Note: ESDRAMC address bit M3IF_MA[10] has a dedicated pin MA10 (required due to precharge all during auto refresh commands) ESDRAMC bank address bits have dedicated pins due to precharge bank during precharge timer timeout WEIM CRE signal is driven on A23 in multiplexed mode operation.					
M3IF_MA[10]		—	—	IPP_DO_M3IF _MA10	MA10
BA[0]		—	—	IPP_DO_SDBA[1:0]	SDBA0
BA[1]		—	—		SDBA1
Note: Since the SDBA pins are shared between the SDR/DDR SDRAM bank address CE‘, the ESDRAMC precharge timer cannot be used.					
SDRAM/LPDDR Dedicated Data Pins					
WR_DATA[0]		—	—	IPP_DO_EMII_DATA [0]	SD0
RD_DATA[0]				IPP_IND_EMII_DATA_IN [0]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
WR_DATA[1]		—	—	IPP_DO_EMII_DATA [1]	SD1
RD_DATA[1]				IPP_IND_EMII_DATA_IN [1]	
WR_DATA[2]		—	—	IPP_DO_EMII_DATA [2]	SD2
RD_DATA[2]				IPP_IND_EMII_DATA_IN [2]	
WR_DATA[3]		—	—	IPP_DO_EMII_DATA [3]	SD3
RD_DATA[3]				IPP_IND_EMII_DATA_IN [3]	
WR_DATA[4]		—	—	IPP_DO_EMII_DATA [4]	SD4
RD_DATA[4]				IPP_IND_EMII_DATA_IN [4]	
WR_DATA[5]		—	—	IPP_DO_EMII_DATA [5]	SD5
RD_DATA[5]				IPP_IND_EMII_DATA_IN [5]	
WR_DATA[6]		—	—	IPP_DO_EMII_DATA [6]	SD6
RD_DATA[6]				IPP_IND_EMII_DATA_IN [6]	
WR_DATA[7]		—	—	IPP_DO_EMII_DATA [7]	SD7
RD_DATA[7]				IPP_IND_EMII_DATA_IN [7]	
WR_DATA[8]		—	—	IPP_DO_EMII_DATA [8]	SD8
RD_DATA[8]				IPP_IND_EMII_DATA_IN [8]	
WR_DATA[9]		—	—	IPP_DO_EMII_DATA [9]	SD9
RD_DATA[9]				IPP_IND_EMII_DATA_IN [9]	
WR_DATA[10]		—	—	IPP_DO_EMII_DATA [10]	SD10
RD_DATA[10]				IPP_IND_EMII_DATA_IN [10]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
WR_DATA[11]		—	—	IPP_DO_EMII_DATA [11]	SD11
RD_DATA[11]				IPP_IND_EMII_DATA_IN [11]	
WR_DATA[12]		—	—	IPP_DO_EMII_DATA [12]	SD12
RD_DATA[12]				IPP_IND_EMII_DATA_IN [12]	
WR_DATA[13]		—	—	IPP_DO_EMII_DATA [13]	SD13
RD_DATA[13]				IPP_IND_EMII_DATA_IN [13]	
WR_DATA[14]		—	—	IPP_DO_EMII_DATA [14]	SD14
RD_DATA[14]				IPP_IND_EMII_DATA_IN [14]	
WR_DATA[15]		—	—	IPP_DO_EMII_DATA [15]	SD15
RD_DATA[15]				IPP_IND_EMII_DATA_IN [15]	
WEIM/NFC Data Multiplexing					
—		IPP_DO_WEIM_ADDR_DATA_OUT[16]	IPP_NFC_WRITE_DATA_OUT[0]	IPP_DO_NFC_WRITE_DATA_OUT [0]	D0
		IPP_IND_WEIM_ADDR_DATA_IN[16]	IPP_NFC_READ_DATA_IN[0]	IPP_IND_NFC_READ_ DATA_IN [0]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[17]	IPP_NFC_WRITE_DATA_OUT[1]	IPP_DO_NFC_WRITE_DATA_OUT [1]	D1
		IPP_IND_WEIM_ADDR_DATA_IN[17]	IPP_NFC_READ_DATA_IN[1]	IPP_IND_NFC_READ_ DATA_IN [1]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[18]	IPP_NFC_WRITE_DATA_OUT[2]	IPP_DO_NFC_WRITE_DATA_OUT [2]	D2
		IPP_IND_WEIM_ADDR_DATA_IN[18]	IPP_NFC_READ_DATA_IN[2]	IPP_IND_NFC_READ_ DATA_IN [2]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
—		IPP_DO_WEIM_ADDR_DATA_OUT[19]	IPP_NFC_WRITE_DATA_OUT[3]	IPP_DO_NFC_WRITE_DATA_OUT [3]	D3
		IPP_IND_WEIM_ADDR_DATA_IN[19]	IPP_NFC_READ_DATA_IN[3]	IPP_IND_NFC_READ_ DATA_IN [3]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[20]	IPP_NFC_WRITE_DATA_OUT[4]	IPP_DO_NFC_WRITE_DATA_OUT [4]	D4
		IPP_IND_WEIM_ADDR_DATA_IN[20]	IPP_NFC_READ_DATA_IN[4]	IPP_IND_NFC_READ_ DATA_IN [4]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[21]	IPP_NFC_WRITE_DATA_OUT[5]	IPP_DO_NFC_WRITE_DATA_OUT [5]	D5
		IPP_IND_WEIM_ADDR_DATA_IN[21	IPP_NFC_READ_DATA_IN[5]	IPP_IND_NFC_READ_ DATA_IN [5]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[22]	IPP_NFC_WRITE_DATA_OUT[6]	IPP_DO_NFC_WRITE_DATA_OUT [6]	D6
		IPP_IND_WEIM_ADDR_DATA_IN[22]	IPP_NFC_READ_DATA_IN[6]	IPP_IND_NFC_READ_ DATA_IN [6]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[23]	IPP_NFC_WRITE_DATA_OUT[7]	IPP_DO_NFC_WRITE_DATA_OUT [7]	D7
		IPP_IND_WEIM_ADDR_DATA_IN[23]	IPP_NFC_READ_DATA_IN[7]	IPP_IND_NFC_READ_ DATA_IN [7]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[24]	IPP_NFC_WRITE_DATA_OUT[8]	IPP_DO_NFC_WRITE_DATA_OUT [8]	D8
		IPP_IND_WEIM_ADDR_DATA_IN[24]	IPP_NFC_READ_DATA_IN[8]	IPP_IND_NFC_READ_ DATA_IN [8]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[25]	IPP_NFC_WRITE_DATA_OUT[9]	IPP_DO_NFC_WRITE_DATA_OUT [9]	D9
		IPP_IND_WEIM_ADDR_DATA_IN[25]	IPP_NFC_READ_DATA_IN[9]	IPP_IND_NFC_READ_ DATA_IN [9]	

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
—		IPP_DO_WEIM_ADDR_DATA_OUT[26]	IPP_NFC_WRITE_DATA_OUT[10]	IPP_DO_NFC_WRITE_DATA_OUT [10]	D10
		IPP_IND_WEIM_ADDR_DATA_IN[26]	IPP_NFC_READ_DATA_IN[10]	IPP_IND_NFC_READ_ DATA_IN [10]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[27]	IPP_NFC_WRITE_DATA_OUT[11]	IPP_DO_NFC_WRITE_DATA_OUT [11]	D11
		IPP_IND_WEIM_ADDR_DATA_IN[27]	IPP_NFC_READ_DATA_IN[11]	IPP_IND_NFC_READ_ DATA_IN [11]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[28]	IPP_NFC_WRITE_DATA_OUT[12]	IPP_DO_NFC_WRITE_DATA_OUT [12]	D12
		IPP_IND_WEIM_ADDR_DATA_IN[28]	IPP_NFC_READ_DATA_IN[12]	IPP_IND_NFC_READ_ DATA_IN [12]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[29]	IPP_NFC_WRITE_DATA_OUT[13]	IPP_DO_NFC_WRITE_DATA_OUT [13]	D13
		IPP_IND_WEIM_ADDR_DATA_IN[29]	IPP_NFC_READ_DATA_IN[13]	IPP_IND_NFC_READ_ DATA_IN [13]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[30]	IPP_NFC_WRITE_DATA_OUT[14]	IPP_DO_NFC_WRITE_DATA_OUT [14]	D14
		IPP_IND_WEIM_ADDR_DATA_IN[30]	IPP_NFC_READ_DATA_IN[14]	IPP_IND_NFC_READ_ DATA_IN [14]	
—		IPP_DO_WEIM_ADDR_DATA_OUT[31]	IPP_NFC_WRITE_DATA_OUT[15]	IPP_DO_NFC_WRITE_DATA_OUT [15]	D15
		IPP_IND_WEIM_ADDR_DATA_IN[31	IPP_NFC_READ_DATA_IN[15]	IPP_IND_NFC_READ_ DATA_IN [15]	
Mask (Byte Enable) Multiplexing					
—	—	IPP_DO_WEIM_EB_B[0]	—	IPP_DO_E M I _IO_EB_B[1:0]	EB0
—	—	IPP_DO_WEIM_EB_B[1]	—		EB1
SDRAM/LPDDR Mask (Byte Enable)					

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
DQM_X[0]		—	—	IPP_DO_EMI_DQM [3:0]	DQM0
DQM_X[1]		—	—		DQM1
Output Enable Multiplexing					
—	—	IPP_DO_WEIM_OE_B	—	IPP_DO_EMI_OE_B	OE
Chip Select Multiplexing					
—	—	IPP_DO_WEIM_CS_B[0]	—	IPP_DO_WEIM_CS_B0	CS0
—	—	IPP_DO_WEIM_CS_B[1]	—	IPP_DO_WEIM_CS_B1	CS1
CS_B[0]		IPP_DO_WEIM_CS_B[2]	—	IPP_DO_WEIM_CS_B2_CSD0	CS2
CS_B[1]		IPP_DO_WEIM_CS_B[3]	—	IPP_DO_WEIM_CS_B3_CSD1	CS3
—	—	IPP_DO_WEIM_CS_B[4]	—	IPP_DO_WEIM_CS_B4	CS4
—	—	IPP_DO_WEIM_CS_B[5]	—	IPP_DO_WEIM_CS_B5	CS5
The chip select selectors are the system control register bits SDCTL_CSD0_SEL and SDCTL_CSD1_SEL respectively. THE Default select for both chip selects is for the ESDRAMC/MDDRC.					
Write Enable Multiplexing					
—	—	IPP_DO_WEIM_RW_B	—	IPP_DO_WEIM_RW_B	RW
SDRAM/LPDDR Command Dedicated Pins					
RAS_B		—	—	IPP_DO_M3IF_RAS_B	RAS
CAS_B		—	—	IPP_DO_M3IF_CAS_B	CAS
WE_B		—	—	IPP_DO_SDRC_SDWE	SDWE
CKE[1]		—	—	IPP_DO_SDRC_SDCKE[1]	SDCKE[1]
CKE[0]		—	—	IPP_DO_SDRC_SDCKE[0]	SDCKE[0]
SDCLK_OUT		—	—	IPP_DO_SDRC_SDCLK	SDCLK

Table 20-5. EMI Outputs to IC Pins (continued)

Memory Controllers Outputs				EMI Output	IC Pin Name
SDRAMC	MDDRC	WEIM	NFC		
—	DQS_OUT [1]	—	—	IPP_DO_DQS[1]	DQS[1]
—	DQS_IN [1]	—	—	IPP_IND_DQS[1]	
—	DQS_OUT [0]	—	—	IPP_DO_DQS[0]	DQS[0]
—	DQS_IN [0]	—	—	IPP_IND_DQS[0]	
NFC Command Dedicated Pins					
—	—	—	IPP_NFC_WE_OUT	IPP_NFC_WE_OUT	NFWE_B
—	—	—	IPP_NFC_WP_OUT	IPP_NFC_WP_OUT	NFWP_B
—	—	—	IPP_NFC_RE_OUT	IPP_NFC_RE_OUT	NFRE_B
—	—	—	IPP_NFC_ALE_OUT	IPP_NFC_ALE_OUT	NFALE
—	—	—	IPP_NFC_CLE_OUT	IPP_NFC_CLE_OUT	NFCLE
—	—	—	IPP_NFC_CE0_OUT	IPP_NFC_CE0_OUT	NFCE0_B
—	—	—	IPP_NFC_CE1_OUT	IPP_NFC_CE1_OUT	NFCE1_B
—	—	—	IPP_NFC_CE2_OUT	IPP_NFC_CE2_OUT	NFCE2_B
—	—	—	IPP_NFC_CE3_OUT	IPP_NFC_CE3_OUT	NFCE3_B
—	—	—	IPP_NFC_RB_IN	IPP_IND_NFC_RB_IN	NFRB
WEIM Command Dedicated Pins					
—	—	IPP_DO_WEIM_LBA_B	—	IPP_DO_WEIM_LBA_B	LBA_B
—	—	IPP_DO_WEIM_BCLK	—	IPP_DO_WEIM_BCLK	BCLK
—	—	IPP_IND_WEIM_ECB_B	—	IPP_IND_WEIM_ECB_B	ECB

Chapter 21
Enhanced Periodic Interrupt Timer (EPIT)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

21.1 Overview

The enhanced periodic interrupt timer (EPIT) is a 32-bit set-and-forget timer that begins counting after the EPIT is enabled by software. It is capable of providing precise interrupts at regular intervals with minimal processor intervention. Figure 21-1 illustrates the block diagram of the EPIT.

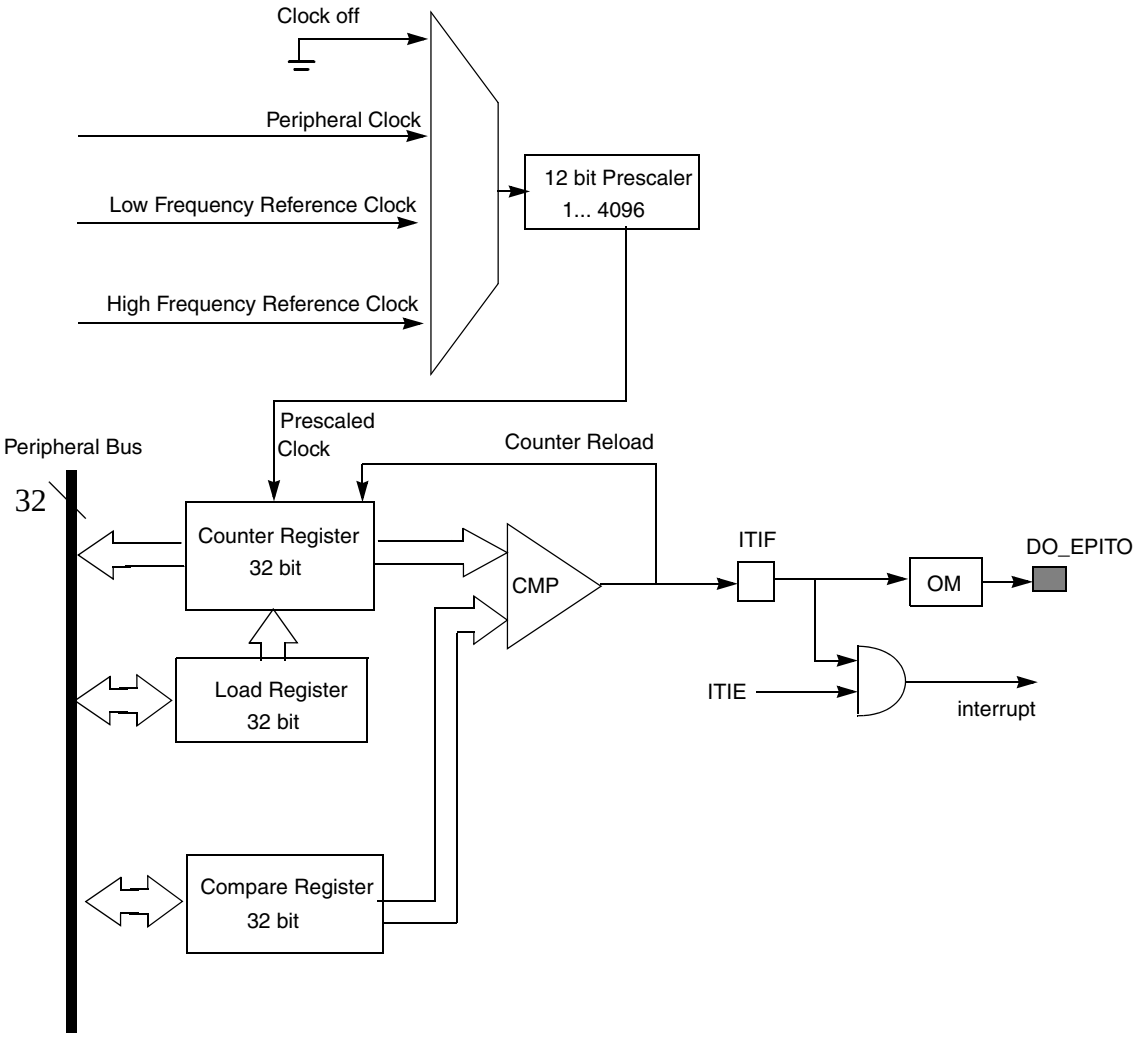


Figure 21-1. Enhanced Periodic Interrupt Timer Block Diagram

21.1.1 Features

Key features of the EPIT include:

- 32-bit down counter with clock source selection
- 12-bit prescaler for division of input clock frequency
- Counter value can be programmed on the fly
- Can be programmed to be active during low-power modes
- Interrupt generation when counter reaches the Compare value

21.1.2 Modes and Operations

The EPIT supports the modes described in the indicated sections:

- Section 21.4.1, “Operating Modes”

— Section 21.4.1.1, “Set-and-Forget Mode”
— Section 21.4.1.2, “Free-Running Mode”

The EPIT supports the operations as described in Section 21.4.2, “Operations.”

21.2 External Signals

Table 21-1 describes all EPIT signals that connect off-chip.

Table 21-1. Off-Chip Module Signals

Name	I/O	Description	Reset State	Pull Up
DO_EPITO	O	Output pin at chip boundary for indication of occurrence of output compare event through a specified transition.	0	—

21.3 Memory Map and Register Definitions

The EPIT module includes five user-accessible 32-bit registers. Table 21-2 summarizes these registers and their addresses.

Peripheral bus write access to the EPIT control register (EPITCR) and the EPIT load register (EPITLR) results in one cycle of wait state, while other valid peripheral bus accesses are with 0 wait state.

21.3.1 Memory Map

For the base address of a particular module instantiation, see the system memory map. Table 21-2 shows the memory map for the EPIT registers.

Table 21-2. EPIT Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0004 (EPITSR)	Control register	R/W	0x0000_0000	21.3.3.1/21-5
0x0004 (EPITSR)	Status register	R/W	0x0000_0000	21.3.3.2/21-7
0x0008 (EPITLR)	Load register	R/W	0xFFFF_FFFF	21.3.3.3/21-8
0x000C (EPITCMPR)	Compare register	R/W	0x0000_0000	21.3.3.4/21-9
0x0010 (EPITCNR)	Counter register	R	0xFFFF_FFFF	21.3.3.5/21-9

21.3.2 Register Summary

The EPIT registers are summarized in Table 21-3.

Table 21-3. EPIT Register Summary

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (EPITCR)	R	0	0	0	0	0	0	CLKSRC		OM		STOP EN	RES	WAIT EN	DBGE N	IOVW	SWR
	W																
	R	PRESCALER[15:4]												RLD	OCIE N	ENM OD	EN
	W																
0x0004 (EPITSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OCIF
	W																w1c
0x0008 (EPITLR)	R	LOAD[31:16]															
	W																
	R	LOAD[15:0]															
	W																
0x000C (EPITCMP R)	R	COMPARE[31:16]															
	W																
	R	COMPARE[15:0]															
	W																
0x0010 (EPITCNR)	R	COUNT[31:16]															
	W																
	R	COUNT[15:0]															
	W																

21.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers. Figure 21-2 and Table 21-4 explain conventions used in register diagrams and tables.

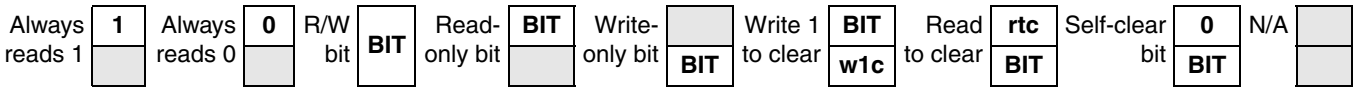


Figure 21-2. Register Field Conventions

Table 21-4. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.

Table 21-4. General Register Conventions (continued)

Convention	Description
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

21.3.3.1 EPIT Control Register

The EPIT control register (EPITCR) is used to configure the operating settings of the EPIT. It contains the clock division prescaler value and also the interrupt enable bit. Additionally it contains other control bit which are outlined below.

Peripheral Bus Write access to EPIT Control Register (EPITCR) results in one cycle of the wait state, while other valid peripheral bus accesses are with 0 wait state.

Figure 21-3 shows the register. Table 21-5 describes the register fields.

Offset 0x0000 (EPITCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	CLKSRC		OM		STOP EN	0	WAIT EN	DBG EN	IOVW	SWR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALAR[15:4]												RLD	OCIE N	ENM OD	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-3. EPIT Control Register

Table 21-5. EPIT Control Register Field Description

Field	Description
31–26	Reserved.
25–24 CLKSRC	Select clock source These bits determine which clock input is to be selected for running the counter. This field value should only be changed when the EPIT is disabled by clearing the EN bit in this register. For other programming requirements while changing clock source, refer to Section 21.5.1, “Change of Clock Source.” 00 Clock is off 01 Peripheral clock 10 High-frequency reference clock 11 Low-frequency reference clock
23–22 OM	EPIT output mode.This bit field determines the mode of EPIT output on the output pin. 00 EPIT output is disconnected from pad 01 Toggle output pin 10 Clear output pin 11 Set output pin
21 STOPEN	EPIT stop mode enable. This read/write control bit enables the operation of the EPIT during stop mode. This bit is reset by a hardware reset and unaffected by software reset. 0 EPIT is disabled in stop mode 1 EPIT is enabled in stop mode
20	Reserved.
19 WAITEN	This read/write control bit enables the operation of the EPIT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled in wait mode 1 EPIT is enabled in wait mode
18 DBGEN	This bit is used to keep the EPIT functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode.This bit is reset by hardware reset. A software reset does not affect this bit. 0 Inactive in debug mode 1 Active in debug mode
17 IOVW	EPIT counter overwrite enable. This bit controls the counter data when the modulus register is written. When this bit is set, all writes to the load register overwrites the counter contents and the counter starts subsequently counting down from the programmed value. 0 Write to load register does not result in counter value being overwritten. 1 Write to load register results in immediate overwriting of counter value.
16 SWR	Software reset. The EPIT module is reset when this bit is set to 1. It is a self clearing bit. This bit is set when the module is in reset state and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values, except for the EN, ENMOD, STOPEN, WAITEN and DBGEN bits in this control register 0 EPIT is out of reset 1 EPIT is undergoing reset
15–4 PRESCALAR	Counter clock prescaler value. This bit field determines the prescaler value by which the clock is divided before it goes to the counter 0x000 Divide by 1 0x001 Divide by 2 … 0xFFFF Divide by 4096

Table 21-5. EPIT Control Register Field Description (continued)

Field	Description
3 RLD	Counter reload control This bit is cleared by hardware reset. It decides the counter functionality, whether to run in free-running mode or set-and-forget mode. 0 When the counter reaches zero it rolls over to 0xFFFF_FFFF (free-running mode) 1 When the counter reaches zero it reloads from the modulus register (set-and-forget mode)
2 OCIEN	Output compare interrupt enable This bit enables the generation of interrupt on occurrence of compare event. 0 Compare interrupt disabled 1 Compare interrupt enabled
1 ENMOD	EPIT enable mode When EPIT is disabled (EN=0), then both main counter and prescaler counter freeze their count at current count values. ENMOD bit is a r/w bit that determines the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then main counter is loaded with the load value (If RLD=1)/ 0xFFFF_FFFF (If RLD=0) and prescaler counter is reset, when EPIT is enabled (EN=1). If ENMOD is programmed to 0 then both main counter and prescaler counter restart counting from their frozen values when EPIT is enabled (EN=1). If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT/DEBUG), then both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 Counter starts counting from the value it had when it was disabled. 1 Counter starts count from load value (RLD=1) or 0xFFFF_FFFF (If RLD=0)
0 EN	This bit enables the EPIT. EPIT counter and prescaler value when EPIT is enabled (EN =1), is dependent upon ENMOD and RLD bit as described for ENMOD bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 EPIT is disabled 1 EPIT is enabled

21.3.3.2 EPIT Status Register

The EPIT status register (EPITSR) has a single status bit for the output compare event. The bit is a write 1 to clear bit.

Figure 21-4 shows the register. Table 21-6 describes the register fields.

Offset 0x0004 (EPITSR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OCIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-4. EPIT Status Register

Table 21-6. EPIT Status Register Field Description

Field	Description
31–1	Reserved.
0 OCIF	Output compare interrupt flag. This bit is the interrupt flag that is set when the content of counter equals the content of the compare register (EPITCMPR). The bit is a write 1 to clear bit. 0 Compare event hasn't occurred 1 Compare event occurred

21.3.3.3 EPIT Load Register

The EPIT load register (EPITLR) contains the value that is to be loaded into the counter when EPIT counter reaches zero if the RLD bit in EPITCR is set. If the IOVW bit in the EPITCR is set then a write to this register overwrites the value of the EPIT counter register in addition to updating this registers value. This overwrite feature is active even if the RLD bit is not set.

Figure 21-5 shows the register. Table 21-7 describes the register fields.

Offset 0x0008 (EPITLR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LOAD[31:16]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOAD[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 21-5. EPIT Load Register

Table 21-7. EPIT Load Register Field Description

Field	Description
31–0 LOAD	Load value. Value that is loaded into the counter at the start of each count cycle.

21.3.3.4 EPIT Compare Register

The EPIT compare register (EPITCMPR) holds the value that determines when a compare event is generated.

Figure 21-6 shows the register. Table 21-8 describes the register fields.

Offset 0x000C (EPITCMPR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMPARE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 21-6. EPIT Compare Register

Table 21-8. EPIT Compare Register Field Description

Field	Description
31–0 COMPARE	Compare Value. When the counter value equals this bit field value a compare event is generated.

21.3.3.5 EPIT Counter Register

The EPIT counter register (EPITCNR) contains the current count value and can be read at any time without disturbing the counter. This is a read-only register and any attempt to write into it generates a transfer error. But if the IOVW bit in EPITCR is set, the value of this register can be overwritten with a write to EPITLR. This change is reflected when this register is subsequently read.

Figure 21-7 shows the register. Table 21-9 describes the register fields.

Offset 0x0010 (EPITCNR) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT[31:16]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 21-7. EPIT Counter Register

Table 21-9. EPIT Counter Register Field Description

Field	Description
31–0 COUNT	Counter value. This contains the current value of the counter.

21.4 Functional Description

This section provides a complete functional description of the module.

21.4.1 Operating Modes

This section describes all functional operation modes of the module. The EPIT can be programmed to function in set-and-forget or free-running modes.

21.4.1.1 Set-and-Forget Mode

This mode of operation is selected when the RLD bit in control register (EPITCR) is set to a value of one. The counter cannot be directly written from the module data bus. Instead, it gets its data from the load register (EPITLR). Whenever the counter reaches a count of zero, the value in EPITLR is loaded into the counter to be decremented toward zero. The counter can be directly initialized, without having to wait for the count to reach zero, when the EPITLR is written while the IOVW bit in EPITCR is set to 1.

21.4.1.2 Free-Running Mode

This mode of operation is selected when the RLD bit is cleared. In this mode, the counter rolls over from 0x0000_0000 to 0xFFFF_FFFF without reloading from the modulus register and continues to count down. In this mode when writing to EPITLR with the required initialization value after having set the IOVW bit, the counter can also be directly initialized.

21.4.2 Operations

The EPIT has a single 32-bit down counter which starts counting when the module is enabled by software. The start value of the counter is loaded from the EPIT load register which can be written to at any time by the processor. The value in the compare register determines the time of occurrence of the interrupt.

When EPIT is disabled (EN=0), then both main counter and prescaler counter freeze their count at current count values. ENMOD bit is a r/w bit which decides the counter value when the EPIT is enabled again by setting EN bit. If ENMOD bit is set, then main counter is loaded with the load value (If RLD=1)/ 0xFFFF_FFFF (If RLD=0) and prescaler counter is reset (0x000), when EPIT is enabled (EN=1). If ENMOD is programmed to 0 then both main counter and prescaler counter restart counting from their frozen values, when EPIT is enabled (EN=1). If EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), then both the main counter and the prescaler counter freeze at their current count values when EPIT enters low-power mode. When EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit.

A hardware reset resets all EPIT registers to their respective reset values. There is a software reset which has the same effect on all registers except for the EN, ENMOD, STOPEN and WAITEN bits in the control register. The state of these bits are not affected by software reset. A software reset can be asserted even when the EPIT is disabled.

21.4.3 Clocks

The clock that feeds the prescaler can be selected from among the following sources:

- High-frequency reference clock

  This clock is provided by the clock controller module (CCM). This clock remains on during low-power mode when the peripheral clock is turned off, allowing EPIT to use this clock in low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version.
- Low-frequency reference clock

  This 32 kHz reference clock is provided by the CCM. This clock remains on in low-power mode when the peripheral clock is turned off, so EPIT can use this clock during low-power mode. In normal mode, the CCM synchronizes this clock to ahb_clk; in low-power mode, CCM switches to an unsynchronized version. This clock is derived from the external 32kHz crystal.
- Peripheral clock

  This is the peripheral clock (PER Clock) which is provided (and optionally gated) by the CCM. This clock is typically used in normal operations. In low-power modes, if the EPIT is programmed to be disabled (using STOPEN or WAITEN), then the peripheral clock can be switched off.

The clock input source is determined by the CLKSRC field in the control register. The clock input to the prescaler can also be disabled by setting CLKSRC to 0b00. This field value should only be changed after first disabling the EPIT by clearing the EN bit in the EPITCR. For other programming requirements that apply while changing clock source, refer section Section 21.5.1, “Change of Clock Source.”

The PRESCALER field in the control register is used to select the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value between 1 and 4096. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency. Figure 21-8 shows the timing for a change in the prescaler value.

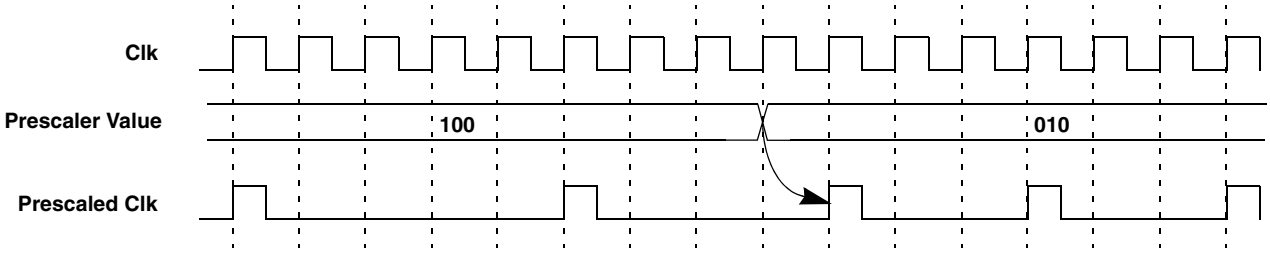


Figure 21-8. Prescaler Value Change Diagram

21.4.4 Compare Event

When the programmed value of EPITCMPR matches the value in EPITCNR a compare status flag is set, and an interrupt is generated if the OCIEN bit is set in the control register. The compare output pin is set, cleared, toggled, or not affected at all depending on the setting of the output mode (OM) bits in the control register. If an interrupt is required at rollover (when the counter value reaches 0x0000_0000 and the new value is loaded) then the compare register value should be set equal to the load register value in set-and-forget mode, or equal to 0xFFFF_FFFF in free-running mode.

Figure 21-9 shows the timing for a compare event and interrupt.

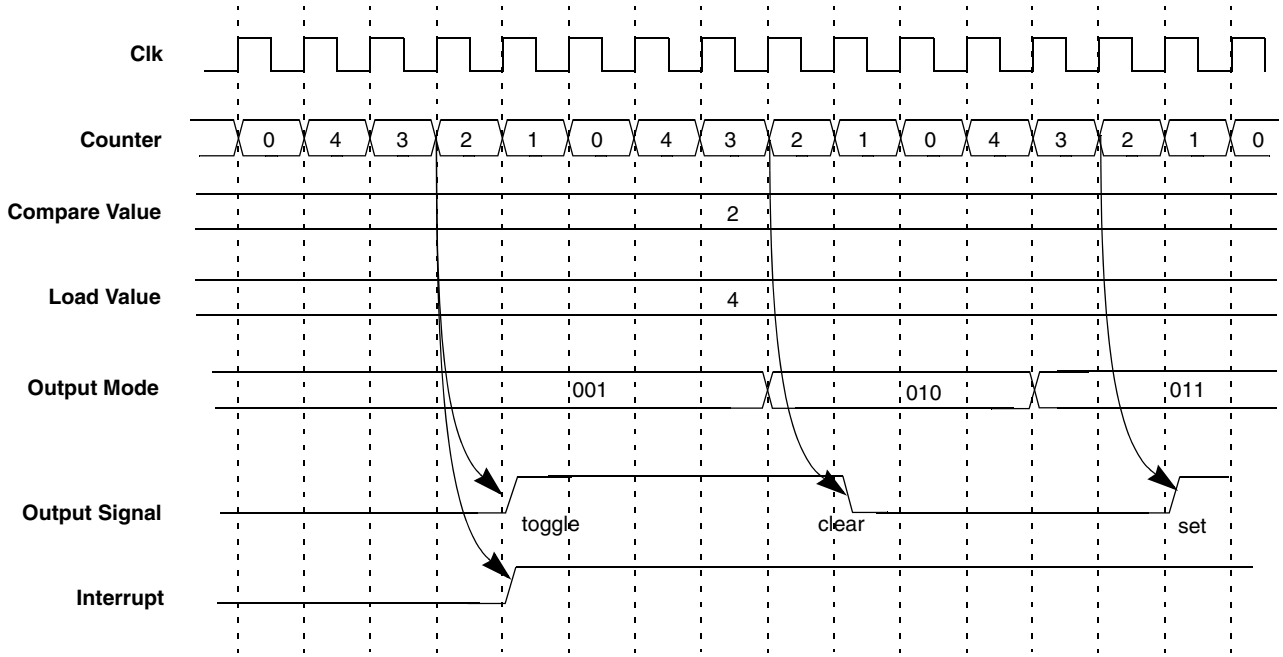


Figure 21-9. Compare Event and Interrupt Timing Diagram

21.4.4.1 Counter Value Overwrite

The EPIT counter value can be overwritten to acquire a desired value at any point of time. The procedure for this is to set the IOVW bit in the control register and then write the desired value into the load register. This results in the load register acquiring that value and also the counter being overwritten with it. If the EPIT is running the counter resumes counting from the overwritten value.

21.4.4.2 Low-Power Mode Behavior

The EPIT timer's behavior in low-power modes depends on which clock source is being used. If the selected clock source is available and the corresponding low-power enable bit is set, then the EPIT continues to function in the low-power mode. If the EPIT is programmed to be disabled in a low-power mode (STOP/WAIT), then main counter and the prescaler counter freeze at the current count values when the EPIT enters low-power mode. When the EPIT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit.

21.4.4.3 Debug Mode Behavior

In debug mode, the user has the option to run or halt the EPIT timers. If the DBGEN bit is reset in the EPIT control Register, the timer is halted. When debug mode is exited, the timer operation reverts to what it was prior to entering debug mode.

21.5 Initialization/Application Information

21.5.1 Change of Clock Source

The CLKSRC field in EPITCR determines the clock source. This field value should be changed only after disabling the EPIT (EN = 0). Below is the software sequence which must be followed while changing clock source.

1. Disable the EPIT by setting EN=0 in EPITCR.
2. Program OM=00 in the EPITCR.
3. Disable the EPIT interrupts.
4. Program CLKSRC to desired clock source in EPITCR.
5. Clear the EPIT status register (EPITSR) i.e. (w1c).
6. Enable the EPIT interrupts.
7. Set ENMOD= 1 in the EPITCR, to bring the EPIT Counter to defined state (EPITLR value or 0xFFFF_FFFF).
8. Enable EPIT (EN=1) in the EPITCR.

Chapter 22
Enhanced Serial Audio Interface (ESAI)

22.1 Introduction

The Enhanced Serial Audio Interface (ESAI) provides a full-duplex serial port for serial communication with a variety of serial devices, including industry-standard codecs, SPDIF transceivers, and other DSPs. The ESAI consists of independent transmitter and receiver sections, each section with its own clock generator.

The ESAI block diagram is shown in Figure 22-1. The ESAI is named synchronous because all serial transfers are synchronized to a clock. Additional synchronization signals are used to delineate the word frames. The normal mode of operation is used to transfer data at a periodic rate, one word per period. The network mode is similar in that it is also intended for periodic transfers; however, it supports up to 32 words (time slots) per period. This mode can be used to build time division multiplexed (TDM) networks. In contrast, the on-demand mode is intended for non-periodic transfers of data and to transfer data serially at high speed when the data becomes available.

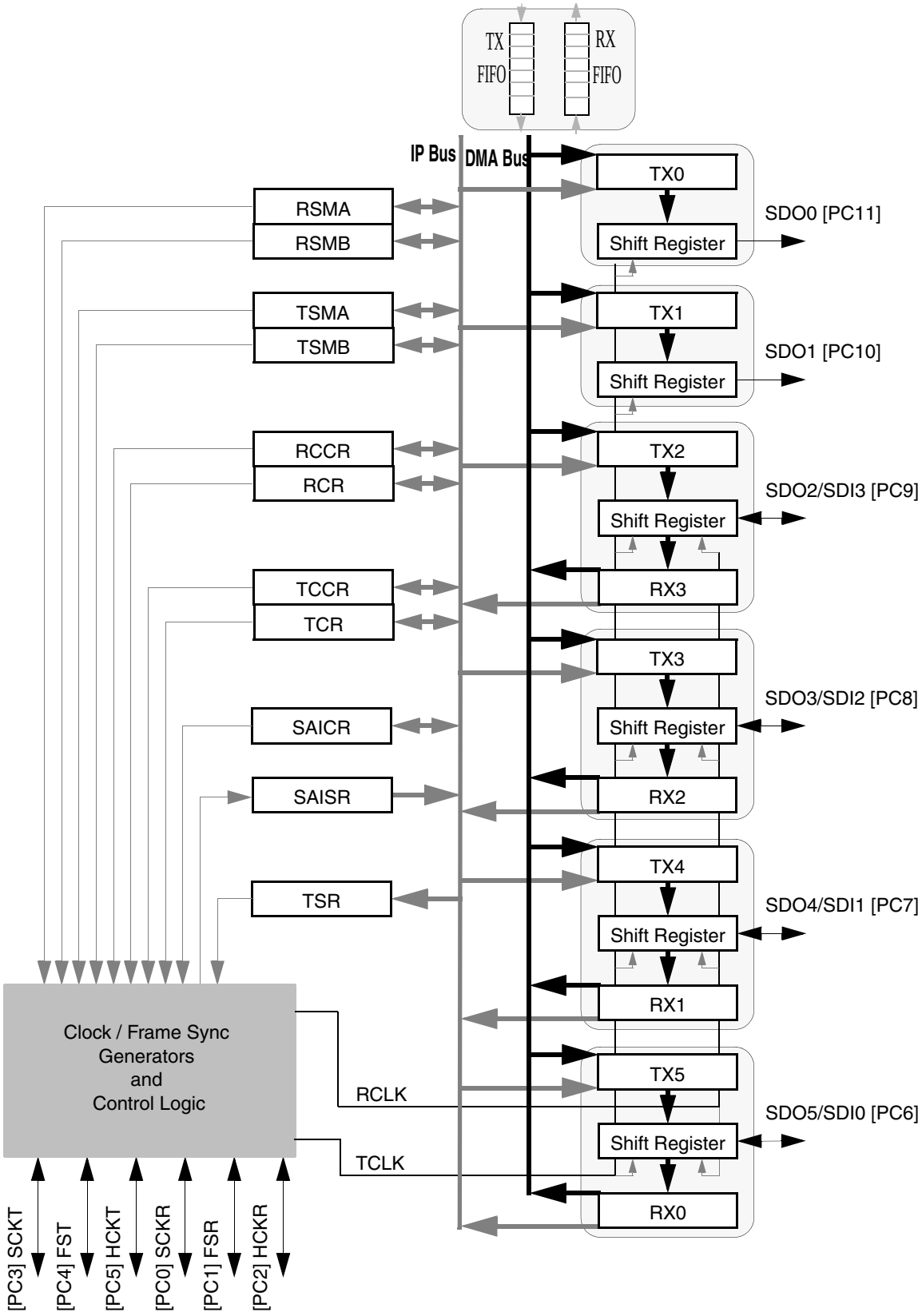


Figure 22-1. ESAI Block Diagram

22.1.1 Features

- Independent (asynchronous mode) or shared (synchronous mode) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Up to 6 transmitters and 4 receivers with SDO2/SDI3, SDO3/SDI2, SDO4/SDI1 and SDO5/SDI0 pins shared by transmitters 2 to 5 and receivers 0 to 3. SDO0 and SDO1 pins are used by transmitters 0 and 1 only.
- Programmable data interface modes supported are I2S, LSB aligned, MSB aligned
- Programmable word length (8, 12, 16, 20, or 24 bits)
- Flexible selection between system clock or external oscillator as input clock source, programmable internal clock divider and frame sync generation
- AC97 support
- Time Slot Mask Registers for reduced CPU overhead (both transmit and receive)
- 128-word Transmit FIFO shared by six transmitters
- 128-word Receive FIFO shared by four receivers

22.1.2 Modes of Operation

ESAI has three basic operating modes and many data/operation formats. ESAI operating mode are selected by the ESAI control registers (TCCR, TCR, RCCR, RCR, and SAICR). The main operating modes are described in the following paragraphs.

22.1.2.1 Normal/Network/On-Demand Mode Selection

Selecting between the normal mode and network mode is accomplished by clearing or setting the TMOD0-TMOD1 bits in the TCR register for the transmitter section, as well as in the RMOD0-RMOD1 bits in the RCR register for the receiver section.

For normal mode, the ESAI functions with one data word of I/O per frame (per enabled transmitter or receiver). The normal mode is typically used to transfer data to/from a single device.

For the network mode, 2 to 32 time slots per frame may be selected. During each frame, 0 to 32 data words of I/O may be received/transmitted. In either case, the transfers are periodic. The frame sync signal indicates the first time slot in the frame. Network mode is typically used in time division multiplexed (TDM) networks of codecs, DSPs with multiple words per frame, or multi-channel devices.

Selecting the network mode and setting the frame rate divider to zero (DC=00000) selects the on-demand mode. This special case does not generate a periodic frame sync. A frame sync pulse is generated only when data is available to transmit. The on-demand mode requires that the transmit frame sync be internal (output) and the receive frame sync be external (input). Therefore, for simplex operation, the synchronous mode could be used; however, for full-duplex operation, the asynchronous mode must be used. Data transmission that is data driven is enabled by writing data into each TX. Although the ESAI is double buffered, only one word can be written to each TX, even if the transmit shift register is empty. The receive and transmit interrupts function as usual using TDE and RDF; however, transmit underruns are impossible for on-demand transmission and are disabled.

22.1.2.2 Synchronous/Asynchronous Operating Modes

The transmit and receive sections of the ESAI may be synchronous or asynchronous, that is, the transmitter and receiver sections may use common clock and synchronization signals (synchronous operating mode), or they may have their own separate clock and sync signals (asynchronous operating mode). The SYN bit in the SAICR register selects synchronous or asynchronous operation. Since the ESAI is designed to operate either synchronously or asynchronously, separate receive and transmit interrupts are provided.

When SYN is cleared, the ESAI transmitter and receiver clocks and frame sync sources are independent. If SYN is set, the ESAI transmitter and receiver clocks and frame sync come from the transmitter section (either external or internal sources).

Data clock and frame sync signals can be generated internally by the ARM-core or may be obtained from external sources. If internally generated, the ESAI clock generator is used to derive high frequency clock, bit clock and frame sync signals from the ARM-core internal system clock.

22.1.2.3 Frame Sync Selection

The frame sync can be either a bit-long or word-long signal. The transmitter frame format is defined by the TFSL bit in the TCR register. The receiver frame format is defined by the RFSL bit in the RCR register.

1. In the word-long frame sync format, the frame sync signal is asserted during the entire word data transfer period. This frame sync length is compatible with codecs, SPI serial peripherals, serial A/D and D/A converters, shift registers and telecommunication PCM serial I/O.
2. In the bit-long frame sync format, the frame sync signal is asserted for one bit clock immediately before the data transfer period. This frame sync length is compatible with Intel and National components, codecs and telecommunication PCM serial I/O.

The relative timing of the word length frame sync as referred to the data word is specified by the TFSR bit in the TCR register for the transmitter section and by the RFSR bit in the RCR register for the receive section. The word length frame sync may be generated (or expected) with the first bit of the data word, or with the last bit of the previous word. TFSR and RFSR are ignored when a bit length frame sync is selected.

Polarity of the frame sync signal may be defined as positive (asserted high) or negative (asserted low). The TFSP bit in the TCCR register specifies the polarity of the frame sync for the transmitter section. The RFSP bit in the RCCR register specifies the polarity of the frame sync for the receiver section.

The ESAI receiver looks for a receive frame sync leading edge (trailing edge if RFSP is set) only when the previous frame is completed. If the frame sync goes high before the frame is completed (or before the last bit of the frame is received in the case of a bit frame sync or a word length frame sync with RFSR set), the current frame sync is not recognized, and the receiver is internally disabled until the next frame sync. Frames do not have to be adjacent, that is, a new frame sync does not have to immediately follow the previous frame. Gaps of arbitrary periods can occur between frames. Enabled transmitters are tri-stated during these gaps.

When operating in the synchronous mode (SYN=1), all clocks including the frame sync are generated by the transmitter section.

22.1.2.4 Shift Direction Selection

Some data formats, such as those used by codecs, specify MSB first while other data formats, such as the AES-EBU digital audio interface, specify LSB first. The MSB/LSB first selection is made by programming RSHFD bit in the RCR register for the receiver section and by programming the TSHFD bit in the TCR register for the transmitter section.

22.2 External Signal Description

Three to twelve pins are required for operation, depending on the operating mode selected and the number of transmitters and receivers enabled. The SDO0 and SDO1 pins are used by transmitters 0 and 1 only. The SDO2/SDI3, SDO3/SDI2, SDO4/SDI1 and SDO5/SDI0 pins are shared by transmitters 2 to 5 with receivers 0 to 3. The actual mode of operation is selected under software control. All transmitters operate fully synchronized under control of the same transmitter clock signals. All receivers operate fully synchronized under control of the same receiver clock signals.

22.2.1 Serial Transmit 0 Data Pin (SDO0)

SDO0 is used for transmitting data from the TX0 serial transmit shift register. SDO0 is an output when data is being transmitted from the TX0 shift register. In the on-demand mode with an internally generated bit clock, the SDO0 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO0 may be programmed as a disconnected pin (PC11) when the ESAI SDO0 function is not being used. (See Table 22-38.)

22.2.2 Serial Transmit 1 Data Pin (SDO1)

SDO1 is used for transmitting data from the TX1 serial transmit shift register. SDO1 is an output when data is being transmitted from the TX1 shift register. In the on-demand mode with an internally generated bit clock, the SDO1 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO1 can be programmed as a disconnected pin (PC10) when the ESAI SDO1 function is not being used. (See Table 22-38.)

22.2.3 Serial Transmit 2/Receive 3 Data Pin (SDO2/SDI3)

SDO2/SDI3 is used as the SDO2 for transmitting data from the TX2 serial transmit shift register when programmed as a transmitter pin, or as the SDI3 signal for receiving serial data to the RX3 serial receive shift register when programmed as a receiver pin. SDO2/SDI3 is an input when data is being received by the RX3 shift register. SDO2/SDI3 is an output when data is being transmitted from the TX2 shift register. In the on-demand mode with an internally generated bit clock, the SDO2/SDI3 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO2/SDI3 may be programmed as a disconnected pin (PC9) when the ESAI SDO2 and SDI3 functions are not being used. (See Table 22-38.)

22.2.4 Serial Transmit 3/Receive 2 Data Pin (SDO3/SDI2)

SDO3/SDI2 is used as the SDO3 signal for transmitting data from the TX3 serial transmit shift register when programmed as a transmitter pin, or as the SDI2 signal for receiving serial data to the RX2 serial receive shift register when programmed as a receiver pin. SDO3/SDI2 is an input when data is being received by the RX2 shift register. SDO3/SDI2 is an output when data is being transmitted from the TX3 shift register. In the on-demand mode with an internally generated bit clock, the SDO3/SDI2 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO3/SDI2 may be programmed as a disconnected pin (PC8) when the ESAI SDO3 and SDI2 functions are not being used. (See Table 22-38.)

22.2.5 Serial Transmit 4/Receive 1 Data Pin (SDO4/SDI1)

SDO4/SDI1 is used as the SDO4 signal for transmitting data from the TX4 serial transmit shift register when programmed as transmitter pin, or as the SDI1 signal for receiving serial data to the RX1 serial receive shift register when programmed as a receiver pin. SDO4/SDI1 is an input when data is being received by the RX1 shift register. SDO4/SDI1 is an output when data is being transmitted from the TX4 shift register. In the on-demand mode with an internally generated bit clock, the SDO4/SDI1 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO4/SDI1 may be programmed as a disconnected pin (PC7) when the ESAI SDO4 and SDI1 functions are not being used. (See Table 22-38.)

22.2.6 Serial Transmit 5/Receive 0 Data Pin (SDO5/SDI0)

SDO5/SDI0 is used as the SDO5 signal for transmitting data from the TX5 serial transmit shift register when programmed as transmitter pin, or as the SDI0 signal for receiving serial data to the RX0 serial shift register when programmed as a receiver pin. SDO5/SDI0 is an input when data is being received by the RX0 shift register. SDO5/SDI0 is an output when data is being transmitted from the TX5 shift register. In the on-demand mode with an internally generated bit clock, the SDO5/SDI0 pin becomes high impedance for a full clock period after the last data bit has been transmitted, assuming another data word does not follow immediately. If a data word follows immediately, there is no high-impedance interval.

SDO5/SDI0 may be programmed as a disconnected pin (PC6) when the ESAI SDO5 and SDI0 functions are not being used. (See Table 22-38.)

22.2.7 Receiver Serial Clock (SCKR)

SCKR is a bidirectional pin providing the receivers serial bit clock for the ESAI interface. The direction of this pin is determined by the RCKD bit in the RCCR register.The SCKR operates as a clock input or output used by all the enabled receivers in the asynchronous mode (SYN=0), or as serial flag 0 pin in the synchronous mode (SYN=1).

When this pin is configured as serial flag pin, its direction is determined by the RCKD bit in the RCCR register. When configured as the output flag OF0, this pin reflects the value of the OF0 bit in the SAICR register, and the data in the OF0 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When this pin is configured as the input flag IF0, the data value at the pin is stored in the IF0 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

SCKR may be programmed as a disconnected pin (PC0) when the ESAI SCKR function is not being used. (See Table 22-38.)

NOTE

Although the external ESAI serial clocks can be independent of and asynchronous to the internal 133 MHz ESAI system clock, the external ESAI serial clock frequency cannot exceed 133MHz/4 = 33.25 MHz and each external ESAI serial clock phase must exceed the minimum of 2 x 1/133MHz = 15.04ns.

For SCKR pin mode definitions, see Table 22-29.

Table 22-1 provides a list of asynchronous-mode receiver clock sources. For more information about EXTAL/ESAI clocking control bits (ERI,ERO), see Table 22-8.

Table 22-1. Receiver Clock Sources (Asynchronous Mode Only)

RHCKD	RFSD	RCKD	ERI	ERO	Receiver Bit Clock Source	OUTPUTS		
0	0	0	N/A	N/A	SCKR			
0	0	1	N/A	N/A	HCKR			SCKR
0	1	0	N/A	N/A	SCKR		FSR	
0	1	1	N/A	N/A	HCKR		FSR	SCKR
1	0	0	0	0	SCKR	HCKR		
1	0	0	0	1	SCKR	HCKR		
1	0	0	1	0	SCKR	HCKR		
1	0	0	1	1	SCKR	HCKR		
1	0	1	0	0	Fsys[1]	HCKR		SCKR
1	0	1	0	1	Fsys	HCKR		SCKR
1	0	1	1	0	EXTAL[2]	HCKR		SCKR

Table 22-1. Receiver Clock Sources (Asynchronous Mode Only) (Continued)

RHCKD	RFSD	RCKD	ERI	ERO	Receiver Bit Clock Source	OUTPUTS		
1	0	1	1	1	EXTAL	HCKR		SCKR
1	1	0	0	0	SCKR	HCKR	FSR	
1	1	0	0	1	SCKR	HCKR	FSR	
1	1	0	1	0	SCKR	HCKR	FSR	
1	1	0	1	1	SCKR	HCKR	FSR	
1	1	1	0	0	Fsys	HCKR	FSR	SCKR
1	1	1	0	1	Fsys	HCKR	FSR	SCKR
1	1	1	1	0	EXTAL	HCKR	FSR	SCKR
1	1	1	1	1	EXTAL	HCKR	FSR	SCKR

Note:
1. Fsys = 133MHz for i.MX35.
2. EXTAL is the on-chip clock sources other than ESAI system 133MHz clock. It is the 24.576MHz EXTAL_AUDIO clock in i.MX35 system.

22.2.8 Transmitter Serial Clock (SCKT)

SCKT is a bidirectional pin providing the transmitters serial bit clock for the ESAI interface. The direction of this pin is determined by the TCKD bit in the TCCR register. The SCKT is a clock input or output used by all the enabled transmitters in the asynchronous mode (SYN=0) or by all the enabled transmitters and receivers in the synchronous mode (SYN=1).

Table 22-2 provides a list of asynchronous-mode transmitter clock sources.

Table 22-2. Transmitter Clock Sources (Asynchronous Mode Only)

THCKD	TFSD	TCKD	ETI	ETO	Transmitter Bit Clock Source	OUTPUTS		
0	0	0	N/A	N/A	SCKT			
0	0	1	N/A	N/A	HCKT			SCKT
0	1	0	N/A	N/A	SCKT		FST	
0	1	1	N/A	N/A	HCKT		FST	SCKT
1	0	0	0	0	SCKT	HCKT		
1	0	0	0	1	SCKT	HCKT		
1	0	0	1	0	SCKT	HCKT		
1	0	0	1	1	SCKT	HCKT		
1	0	1	0	0	Fsys[1]	HCKT		SCKT

Table 22-2. Transmitter Clock Sources (Asynchronous Mode Only) (Continued)

THCKD	TFSD	TCKD	ETI	ETO	Transmitter Bit Clock Source	OUTPUTS		
1	0	1	0	1	Fsys	HCKT		SCKT
1	0	1	1	0	EXTAL[2]	HCKT		SCKT
1	0	1	1	1	EXTAL	HCKT		SCKT
1	1	0	0	0	SCKR	HCKT	FST	
1	1	0	0	1	SCKR	HCKT	FST	
1	1	0	1	0	SCKR	HCKT	FST	
1	1	0	1	1	SCKR	HCKT	FST	
1	1	1	0	0	Fsys	HCKT	FST	SCKT
1	1	1	0	1	Fsys	HCKT	FST	SCKT
1	1	1	1	0	EXTAL	HCKT	FST	SCKT
1	1	1	1	1	EXTAL	HCKT	FST	SCKT

Note:
1. Fsys = 133MHz for i.MX35.
2. EXTAL is the on-chip clock sources other than ESAI system 133MHz clock. It is the 24.576MHz EXTAL_AUDIO clock in i.MX35 system.

SCKT may be programmed as a disconnected pin (PC3) when the ESAI SCKT function is not being used. (See Table 22-38.)

For more information about EXTAL/ESAI clocking control bits (ETI, ETO), see Table 22-8.

NOTE

Although the external ESAI serial clock can be independent of and asynchronous to the internal 133 MHz ESAI system clock, the external ESAI serial clock frequency cannot exceed 133MHz/4 = 33.25MHz and each external ESAI serial clock phase must exceed the minimum of 2 x 1/133MHz = 15.04ns.

22.2.9 Frame Sync for Receiver (FSR)

FSR is a bidirectional pin providing the receivers frame sync signal for the ESAI interface. The direction of this pin is determined by the RFSD bit in RCR register. In the asynchronous mode (SYN=0), the FSR pin operates as the frame sync input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as either the serial flag 1 pin (TEBE=0), or as the transmitter external buffer enable control (TEBE=1, RFSD=1). For FSR pin mode definitions, see Table 22-30; for receiver clock signals, see Table 22-1.

When this pin is configured as serial flag pin, its direction is determined by the RFSD bit in the RCCR register. When configured as the output flag OF1, this pin reflects the value of the OF1 bit in the SAICR

register, and the data in the OF1 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When configured as the input flag IF1, the data value at the pin is stored in the IF1 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

FSR may be programmed as a disconnected pin (PC1) when the ESAI FSR function is not being used. (See Table 22-38.)

22.2.10 Frame Sync for Transmitter (FST)

FST is a bidirectional pin providing the frame sync for both the transmitters and receivers in the synchronous mode (SYN=1) and for the transmitters only in asynchronous mode (SYN=0) (see Table 22-2). The direction of this pin is determined by the TFSD bit in the TCR register. When configured as an output, this pin is the internally generated frame sync signal. When configured as an input, this pin receives an external frame sync signal for the transmitters (and the receivers in synchronous mode).

FST may be programmed as a disconnected pin (PC4) when the ESAI FST function is not being used. (See Table 22-38.)

22.2.11 High Frequency Clock for Transmitter (HCKT)

HCKT is a bidirectional pin providing the transmitters high frequency clock for the ESAI interface. The direction of this pin is determined by the THCKD bit in the TCCR register. In the asynchronous mode (SYN=0), the HCKT pin operates as the high frequency clock input or output used by all enabled transmitters. In the synchronous mode (SYN=1), it operates as the high frequency clock input or output used by all enabled transmitters and receivers. When programmed as input this pin is used as an alternative high frequency clock source to the ESAI transmitter rather than the ARM-core main clock. When programmed as output it can serve as a high frequency sample clock (to external DACs for example) or as an additional system clock (see Table 22-2).

HCKT may be programmed as a disconnected pin (PC5) when the ESAI HCKT function is not being used. (See Table 22-38.)

22.2.12 High Frequency Clock for Receiver (HCKR)

HCKR is a bidirectional pin providing the receivers high frequency clock for the ESAI interface. The direction of this pin is determined by the RHCKD bit in the RCCR register. In the asynchronous mode (SYN=0), the HCKR pin operates as the high frequency clock input or output used by all the enabled receivers. In the synchronous mode (SYN=1), it operates as the serial flag 2 pin. For HCKR pin mode definitions, see Table 22-31; for receiver clock signals, see Table 22-1.

When this pin is configured as serial flag pin, its direction is determined by the RHCKD bit in the RCCR register. When configured as the output flag OF2, this pin reflects the value of the OF2 bit in the SAICR register, and the data in the OF2 bit shows up at the pin synchronized to the frame sync being used by the transmitter and receiver sections. When configured as the input flag IF2, the data value at the pin is stored in the IF2 bit in the SAISR register, synchronized by the frame sync in normal mode or the slot in network mode.

HCKR may be programmed as a disconnected pin (PC2) when the ESAI HCKR function is not being used. (See Table 22-38.)

22.2.13 Serial I/O Flags

Three ESAI pins (FSR, SCKR and HCKR) are available as serial I/O flags when the ESAI is operating in the synchronous mode (SYN=1). Their operation is controlled by RCKD, RFSD, TEBE bits in the RCR, RCCR and SAICR registers.The output data bits (OF2, OF1 and OF0) and the input data bits (IF2, IF1 and IF0) are double buffered to/from the HCKR, FSR and SCKR pins. Double buffering the flags keeps them in sync with the TX and RX data lines.

Each flag can be separately programmed. Flag 0 (SCKR pin) direction is selected by RCKD, RCKD=1 for output and RCKD=0 for input. Flag 1 (FSR pin) is enabled when the pin is not configured as external transmitter buffer enable (TEBE=0) and its direction is selected by RFSD, RFSD=1 for output and RFSD=0 for input. Flag 2 (HCKR pin) direction is selected by RHCKD, RHCKD=1 for output and RHCKD=0 for input.

When programmed as input flags, the SCKR, FSR and HCKR logic values, respectively, are latched at the same time as the first bit of the receive data word is sampled. Because the input was latched, the signal on the input flag pin (SCKR, FSR or HCKR) can change without affecting the input flag until the first bit of the next receive data word. When the received data words are transferred to the receive data registers, the input flag latched values are then transferred to the IF0, IF1 and IF2 bits in the SAISR register, where they may be read by software.

When programmed as output flags, the SCKR, FSR and HCKR logic values are driven by the contents of the OF0, OF1 and OF2 bits in the SAICR register respectively, and they are driven when the transmit data registers are transferred to the transmit shift registers. The value on SCKR, FSR and HCKR is stable from the time the first bit of the transmit data word is transmitted until the first bit of the next transmit data word is transmitted. Software may change the OF0-OF2 values thus controlling the SCKR, FSR and HCKR pin values for each transmitted word. The normal sequence for setting output flags when transmitting data is as follows: wait for TDE (transmitter empty) to be set; first write the flags, and then write the transmit data to the transmit registers. OF0, OF1, and OF2 are double buffered so that the flag states appear on the pins when the transmit data is transferred to the transmit shift register, that is, the flags are synchronous with the data.

22.3 Memory Map and Register Definition

Section 22.3, “Memory Map and Register Definition” provides the detailed descriptions for all of the ESAI registers.

22.3.1 Memory Map

For the base address of a particular module instantiation, see the system memory map. Table 22-3 shows the ESAI memory map.

Table 22-3. ESAI Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x4000 (ETDR)	ESAI Transmit Data Register	W	0x0000_0000	22.3.3.1/22-18
0x4004 (ERDR)	ESAI Receive Data Register	R	0x0000_0000	22.3.3.2/22-19
0x4008 (ECR)	ESAI Control Register	R/W	0x0000_0000	22.3.3.3/22-20
0x400C (ESR)	ESAI Status Register	R	0x0000_0000	22.3.3.4/22-21
0x4010 (TFCR)	Transmit FIFO Configuration Register	R/W	0x0000_0000	22.3.3.5/22-22
0x4014 (TFSR)	Transmit FIFO Status Register	R	0x0000_0000	22.3.3.6/22-24
0x4018 (RFCR)	Receive FIFO Configuration Register	R/W	0x0000_0000	22.3.3.7/22-25
0x401C (RFSR)	Receive FIFO Status Register	R	0x0000_0000	22.3.3.8/22-26
0x4020–0x407C	Reserved	R	0x0000_0000	
0x4080–0x4094 (TX0 - TX5)	Transmit Data Register 0	W	0x0000_0000	22.3.3.9/22-27
0x4084 (TX1)	Transmit Data Register 1	W	0x0000_0000	22.3.3.9/22-27
0x4088 (TX2)	Transmit Data Register 2	W	0x0000_0000	22.3.3.9/22-27
0x408C (TX3)	Transmit Data Register 3	W	0x0000_0000	22.3.3.9/22-27
0x4090 (TX4)	Transmit Data Register 4	W	0x0000_0000	22.3.3.9/22-27
0x4094 (TX5)	Transmit Data Register 5	W	0x0000_0000	22.3.3.9/22-27
0x4098 (TSR)	Transmit Slot Register	W	0x0000_0000	22.3.3.11/22-31
0x409C	Reserved	R	0x0000_0000	
0x40A0–0x40AC (RX0 - RX3)	Receive Data Register 0	R	0x0000_0000	22.3.3.12/22-31
0x40A4 (RX1)	Receive Data Register 1	R	0x0000_0000	22.3.3.12/22-31
0x40A8 (RX2)	Receive Data Register 2	R	0x0000_0000	22.3.3.12/22-31
0x40AC (RX3)	Receive Data Register 3	R	0x0000_0000	22.3.3.12/22-31
0x40B0 to 0x40C8	Reserved	R	0x0000_0000	
0x40CC (SAISR)	Serial Audio Interface Status Register	R	0x0000_0000	22.3.3.14/22-32
0x40D0 (SAICR)	Serial Audio Interface Control Register	R/W	0x0000_0000	22.3.3.15/22-35
0x40D4 (TCR)	Transmit Control Register	R/W	0x0000_0000	22.3.3.16/22-37
0x40D8 (TCCR)	Transmit Clock Control Register	R/W	0x0000_0000	22.3.3.17/22-46
0x40DC (RCR)	Receive Control Register	R/W	0x0000_0000	22.3.3.18/22-49
0x40E0 (RCCR)	Receive Clock Control Register	R/W	0x0000_0000	22.3.3.19/22-53
0x40E4 (TSMA)	Transmit Slot Mask Register A	R/W	0x0000_FFFF	22.3.3.20/22-57
0x40E8 (TSMB)	Transmit Slot Mask Register B	R/W	0x0000_FFFF	22.3.3.21/22-57

Table 22-3. ESAI Memory Map (Continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x40EC (RSMA)	Receive Slot Mask Register A	R/W	0x0000_FFFF	22.3.3.22/22-58
0x40F0 (RSMB)	Receive Slot Mask Register B	R/W	0x0000_FFFF	22.3.3.23/22-59
0x40F8 (PRRC)	Port C Direction Register	R/W	0x0000_0000	22.3.3.25/22-61
0x40FC (PCRC)	Port C Control Register	R/W	0x0000_0000	22.3.3.26/22-61

22.3.2 Register Summary

Figure 22-2 shows the key to register tables and Table 22-4 shows the register figure conventions.

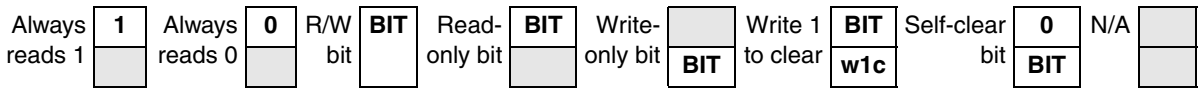


Figure 22-2. Key to Register Fields

Table 22-4. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 22-5 summarizes the ESAI registers.

Table 22-5. ESAI Register Summary

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4000 (ETDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	ETDR[31:16]															
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	ETDR[15:0]															
0x4004 (ERDR)	R	ERDR[31:16]															
	W																
	R	ERDR[15:0]															
	W																
0x4008 (ECR)	R	0	0	0	0	0	0	0	0	0	0	0	0	ETI	ETO	ERI	ERO
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ERST	ESAIEN
	W																
0x400C (ESR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	TINIT	RFF	TFE	TLS	TDE	TED	TD	RLS	RDE	RED	RD
	W																
0x4010 (TFCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	TIEN	TWA[2:0]		
	W																
	R	TFWM[7:0]								TE5	TE4	TE3	TE2	TE1	TE0	TFR	TFEN
	W																
0x4014 (TFSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	NTFO[2:0]			0	NTFI[2:0]			TFCNT[7:0]							
	W																
0x4018 (RFCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	REXT	RWA[2:0]		
	W																
	R	RFWM[7:0]								0	0	RE3	RE2	RE1	RE0	RFR	RFEN
	W																

Table 22-5. ESAI Register Summary (Continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x401C (RFSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	NRFI[1:0]		0	0	NRFO[1:0]		RFCNT[7:0]							
	W																
0x4080–0x4094 (TX0 - TX5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX0[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX0[15:0]															
0x4084 (TX1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX1[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX1[15:0]															
0x4088 (TX2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX2[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX2[15:0]															
0x408C (TX3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX3[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX3[15:0]															
0x4090 (TX4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX4[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX4[15:0]															
0x4094 (TX5)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TX5[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TX5[15:0]															

Table 22-5. ESAI Register Summary (Continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4098 (TSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									TSR[23:16]							
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	TSR[15:0]															
0x40A0–0x40AC (RX0 - RX3)	R	0	0	0	0	0	0	0	0	RX0[23:0]							
	W																
	R	RX0[15:0]															
	W																
0x40A4 (RX1)	R	0	0	0	0	0	0	0	0	RX1[23:0]							
	W																
	R	RX1[15:0]															
	W																
0x40A8 (RX2)	R	0	0	0	0	0	0	0	0	RX2[23:0]							
	W																
	R	RX2[15:0]															
	W																
0x40AC (RX3)	R	0	0	0	0	0	0	0	0	RX3[23:0]							
	W																
	R	RX3[15:0]															
	W																
0x40CC (SAISR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TODFE	TEDE
	W																
	R	TDE	TUE	TFS	0	0	RODF	REDF	RDF	ROE	RFS	0	0	0	IF2	IF1	IF0
	W																
0x40D0 (SAICR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	ALC	TEBE	SYN	0	0	0	OF2	OF1	OF0
	W																

Table 22-5. ESAI Register Summary (Continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40D4 (TCR)	R	0	0	0	0	0	0	0	0	TLIE	TIE	TDEIE	TEIE	TPR	0	PADC	TFSR
	W																
	R	TFSL	TSWS[4:0]					TMOD[1:0]		TWA	TSHFD	TE5	TE4	TE3	TE2	TE1	TE0
	W																
0x40D8 (TCCR)	R	0	0	0	0	0	0	0	0	THCKD	TFSD	TCKD	THCKP	TFSP	TCKP	TFP[3:2]	
	W																
	R	TFP[1:0]		TDC[4:0]					TPSR	TPM[7:0]							
	W																
0x40DC (RCR)	R	0	0	0	0	0	0	0	0	RLIE	RIE	RDEIE	REIE	RPR	0	0	RFSR
	W																
	R	RFSL	RSWS[4:0]					RMOD[1:0]		RWA	RSHFD	0	0	RE3	RE2	RE1	RE0
	W																
0x40E0 (RCCR)	R	0	0	0	0	0	0	0	0	RHCKD	RFSD	RCKD	RHCKP	RFSP	RCKP	RFP[3:2]	
	W																
	R	RFP[1:0]		RDC[4:0]					RPSR	RPM[7:0]							
	W																
0x40E4 (TSMA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TS[15:0]															
	W																
0x40E8 (TSMB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TS[31:16]															
	W																
0x40EC (RSMA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RS[15:0]															
	W																

Table 22-5. ESAI Register Summary (Continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40F0 (RSMB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	RS[31:16]															
	W																
0x40F8 (PRRC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PDC[11:0]											
	W																
0x40FC (PCRC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PC[11:0]											
	W																

22.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams in bit order.

22.3.3.1 ESAI Transmit Data Register (ETDR)

Offset 0x4000 (ETDR) Access: User write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	ETDR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	EDTR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-3. ESAI Transmit Data Register

Table 22-6. ESAI Transmit Data Register Field Descriptions

Field	Description
31–0 ETDR [31:0]	ESAI Transmit Data Register. Writing to this register stores the data written into the ESAI Transmit FIFO. Writing to this register when the Transmit FIFO is full causes the data written to be lost (the existing data within the FIFO is not overwritten). When multiple ESAI transmitters are enabled, the data for each transmitter must be interleaved from lowest transmitter to highest transmitter (for example, if transmitters 0, 2 and 3 are enabled then data must be written as follows: transmitter #0, transmitter #2, transmitter #3, transmitter #0, transmitter #2, transmitter #3, transmitter #0, …). Data within the ESAI Transmit FIFO is passed to the ESAI transmit shifter registers as defined by the Transmit Word Alignment configuration bits.

22.3.3.2 ESAI Receive Data Register (ERDR)

Offset 0x4004 (ERDR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERDR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERDR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-4. ESAI Receive Data Register

Table 22-7. ESAI Receive Data Register Field Descriptions

Field	Description
31–0 ERDR [31:0]	ESAI Receive Data Register. Reading this register returns the data within the ESAI Receive FIFO. Reading this register when the Receive FIFO is empty returns the last valid data word. When multiple ESAI receivers are enabled, the data for each receiver is interleaved from lowest receiver to highest receiver (for example, if receivers 0, 2 and 3 are enabled then data is returned as follows: receiver #0, receiver #2, receiver #3, receiver #0, receiver #2, receiver #3, receiver #0, etc). Data is passed from the ESAI receive shift registers to the ESAI Receive FIFO as defined by the Receiver Word Alignment configuration bits either zero or sign-extended based on the Receive Extension control bit.

22.3.3.3 ESAI Control Register (ECR)

Offset 0x4008 (ECR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	ETI	ETO	ERI	ERO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ERST	ESAI EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-5. ESAI Control Register

Table 22-8. ESAI Control Register Field Descriptions

Field	Description
31–20	Reserved
19 ETI	EXTAL Transmitter In. Mux EXTAL in place of the High Frequency Transmitter Clock input pin. HCKT can still be used to drive a divided down EXTAL or as GPIO. 0 HCKT pin has normal function. 1 EXTAL muxed into HCKT input.
18 ETO	EXTAL Transmitter Out. Drive the EXTAL input on the High Frequency Transmitter Clock pin. 0 HCKT pin has normal function. 1 EXTAL driven onto HCKT pin.
17 ERI	EXTAL Receiver In. Mux EXTAL in place of the High Frequency Receiver Clock input pin. HCKR can still be used to drive a divided down EXTAL or as GPIO. 0 HCKR pin has normal function. 1 EXTAL muxed into HCKR input.
16 ERO	EXTAL Receiver Out. Drive the EXTAL input on the High Frequency Receiver Clock pin. 0 HCKR pin has normal function. 1 EXTAL driven onto HCKR pin.
15–2	Reserved
1 ERST	ESAI Reset. Reset the ESAI core logic (including configuration registers) but not the ESAI FIFOs. 0 ESAI not reset. 1 ESAI reset.
0 ESAIEN	ESAI Enable. Enables/disables the ESAI logic clock. Enable the ESAI before reading or writing other ESAI registers. 0 ESAI disabled. 1 ESAI enabled.

22.3.3.4 ESAI Status Register (ESR)

Offset 0x400C (ESR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	TINIT	RFF	TFE	TLS	TDE	TED	TD	RLS	RDE	RED	RD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-6. ESAI Status Register

Table 22-9. ESAI Status Register Field Descriptions

Field	Description
31–11	Reserved
10 TINIT	Transmit Initialization. Indicates that the Transmit FIFO is writing the first word for each enabled transmitter into the Transmit Data Registers. This bit sets when the Transmit FIFO is enabled (provided Transmit Initialization is enabled) and clears after the Transmit Data Registers have been initialized. The Transmit Enable bits in the Transmit Control Register should not be set until this flag has cleared. 0 Transmitter has finished initializing the Transmit Data Registers (or Transmit FIFO is not enabled or Transmit Initialization is not enabled). 1 Transmitter has not finished initializing the Transmit Data Registers.
9 RFF	Receive FIFO Full. Indicates that the number of data words in the Receive FIFO has equaled or exceeded the Receive FIFO Watermark. This flag also drives the ESAI Receiver DMA request line. ESAI FIFO DMA requests see Section 22.4.3, “ESAI DMA Requests from the FIFOs”. 0 Number of words in Receive FIFO less than Receive FIFO watermark. 1 Number of words in Receive FIFO is equal to or greater than Receive FIFO watermark.
8 TFE	Transmit FIFO Empty. Indicates that the number of empty slots in the Transmit FIFO has met or exceeded the Transmit FIFO Watermark. This flag also drives the ESAI Transmitter DMA request line. ESAI FIFO DMA request see Section 22.4.3, “ESAI DMA Requests from the FIFOs”. 0 Number of empty slots in Transmit FIFO less than Transmit FIFO watermark. 1 Number of empty slots in Transmit FIFO is equal to or greater than Transmit FIFO watermark.
7 TLS	Transmit Last Slot. Reading this register when TLS is set will negate the Transmit Last Slot interrupt. 0 TLS is not the highest priority active interrupt. 1 TLS is the highest priority active interrupt.
6 TDE	Transmit Data Exception. 0 TDE is not the highest priority active interrupt. 1 TDE is the highest priority active interrupt.
5 TED	Transmit Even Data. 0 TED is not the highest priority active interrupt. 1 TED is the highest priority active interrupt.

Table 22-9. ESAI Status Register Field Descriptions (Continued)

Field	Description
4 TD	Transmit Data. 0 TD is not the highest priority active interrupt. 1 TD is the highest priority active interrupt.
3 RLS	Receive Last Slot. Reading this register when RLS is set will negate the Receive Last Slot interrupt. 0 RLS is not the highest priority active interrupt. 1 RLS is the highest priority active interrupt.
2 RDE	Receive Data Exception. 0 RDE is not the highest priority active interrupt. 1 RDE is the highest priority active interrupt.
1 RED	Receive Even Data. 0 RED is not the highest priority active interrupt. 1 RED is the highest priority active interrupt.
0 RD	Receive Data. 0 RD is not the highest priority active interrupt. 1 RD is the highest priority active interrupt.

22.3.3.5 Transmit FIFO Configuration Register (TFCR)

Offset 0x4010 (TFCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	TIEN	TWA[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFWM[7:0]								TE5	TE4	TE3	TE2	TE1	TE0	TFR	TFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-7. Transmit FIFO Configuration Register

Table 22-10. Transmit FIFO Configuration Register Field Descriptions

Field	Description
31–20	Reserved
19 TIEN	Transmitter Initialization Enable. Enables the initialization of the Transmit Data Registers when the Transmitter FIFO is enabled. 0 Transmit Data Registers are not initialized from the FIFO once the Transmit FIFO is enabled. Software must manually initialize the Transmit Data Registers separately. 1 Transmit Data Registers are initialized from the FIFO once the Transmit FIFO is enabled.

Table 22-10. Transmit FIFO Configuration Register Field Descriptions (Continued)

Field	Description
18–16 TWA [2:0]	Transmit Word Alignment. Configures the alignment of the data written into the ESAI Transmit Data Register and then passed to the relevant 24 bit Transmit shift register. 000 MSB of data is bit 31. Data bits 7–0 are ignored when passed to transmit shift register. 001 MSB of data is bit 27. Data bits 3–0 are ignored when passed to transmit shift register. 010 MSB of data is bit 23. 011 MSB of data is bit 19. Bottom 4 bits of transmit shift register are zeroed. 100 MSB of data is bit 15. Bottom 8 bits of transmit shift register are zeroed. 101 MSB of data is bit 11. Bottom 12 bits of transmit shift register are zeroed. 110 MSB of data is bit 7. Bottom 16 bits of transmit shift register are zeroed. 111 MSB of data is bit 3. Bottom 20 bits of transmit shift register are zeroed.
15–8 TFWM [7:0]	Transmit FIFO Watermark. These bits configure the threshold at which the Transmit FIFO Empty flag will set. The TFE is set when the number of empty slots in the Transmit FIFO equal or exceed the selected threshold.
7 TE5	Transmitter #5 FIFO Enable. This bit enables transmitter #5 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #5 is not using the Transmit FIFO. 1 Transmitter #5 is using the Transmit FIFO.
6 TE4	Transmitter #4 FIFO Enable. This bit enables transmitter #4 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #4 is not using the Transmit FIFO. 1 Transmitter #4 is using the Transmit FIFO.
5 TE3	Transmitter #3 FIFO Enable. This bit enables transmitter #3 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #3 is not using the Transmit FIFO. 1 Transmitter #3 is using the Transmit FIFO.
4 TE2	Transmitter #2 FIFO Enable. This bit enables transmitter #2 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #2 is not using the Transmit FIFO. 1 Transmitter #2 is using the Transmit FIFO.
3 TE1	Transmitter #1 FIFO Enable. This bit enables transmitter #1 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #1 is not using the Transmit FIFO. 1 Transmitter #1 is using the Transmit FIFO.
2 TE0	Transmitter #0 FIFO Enable. This bit enables transmitter #0 to use the Transmit FIFO. Do not change this bit when the Transmitter FIFO is enabled. 0 Transmitter #0 is not using the Transmit FIFO. 1 Transmitter #0 is using the Transmit FIFO.
1 TFR	Transmit FIFO Reset. This bit resets the Transmit FIFO pointers. 0 Transmit FIFO not reset. 1 Transmit FIFO reset.
0 TFE	Transmit FIFO Enable. This bit enables the use of the Transmit FIFO. 0 Transmit FIFO disabled. 1 Transmit FIFO enabled.

22.3.3.6 Transmit FIFO Status Register (TFSR)

Offset 0x4014 (TFSR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	NTFO[2:0]			0	NTFI[2:0]			TFCNT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-8. Transmit FIFO Status Register

Table 22-11. Transmit FIFO Status Register Field Descriptions

Field	Description
31–15	Reserved
14–12 NTFO [2:0]	Next Transmitter FIFO Out. Indicates which Transmit Data Register receives the top word of the Transmit FIFO. This will usually equal the lowest enabled transmitter, unless the transmit FIFO is empty. 000 Transmitter #0 receives next word from the Transmit FIFO. 001 Transmitter #1 receives next word from the Transmit FIFO. 010 Transmitter #2 receives next word from the Transmit FIFO. 011 Transmitter #3 receives next word from the Transmit FIFO. 100 Transmitter #4 receives next word from the Transmit FIFO. 101 Transmitter #5 receives next word from the Transmit FIFO. 110 Reserved. 111 Reserved.
11	Reserved
10–8 NTFI [2:0]	Next Transmitter FIFO In. Indicates which transmitter receives the next word written to the FIFO. 000 Transmitter #0 receives next word written to the Transmit FIFO. 001 Transmitter #1 receives next word written to the Transmit FIFO. 010 Transmitter #2 receives next word written to the Transmit FIFO. 011 Transmitter #3 receives next word written to the Transmit FIFO. 100 Transmitter #4 receives next word written to the Transmit FIFO. 101 Transmitter #5 receives next word written to the Transmit FIFO. 110 Reserved. 111 Reserved.
7–0 TFCNT [7:0]	Transmit FIFO Counter. These bits indicate the number of data words stored in the Transmit FIFO.

22.3.3.7 Receive FIFO Configuration Register (RFCR)

Offset 0x4018 (RFCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	REXT	RWA[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFWM[7:0]								0	0	RE3	RE2	RE1	RE0	RFR	RFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-9. Receive FIFO Configuration Register

Table 22-12. Receive FIFO Configuration Register Field Descriptions

Field	Description
31–20	Reserved
19 REXT	Receive Extension. Enables the receive data to be returned sign extended when the Receive Word Alignment is configured to return data where the MSB is not aligned with bit 31. 0 Receive data is zero extended. 1 Receive data is sign extended.
18–16 RWA [2:0]	Receive Word Alignment. Configures the alignment of the data passed from the relevant 24 bit Receive shift register and read out the ESAI Receive Data Register. 000 MSB of data is at bit 31. Data bits 7–0 are zeroed. 001 MSB of data is at bit 27. Data bits 3–0 are zeroed. 010 MSB of data is at bit 23. 011 MSB of data is at bit 19. Data bits 3–0 from receive shift register are ignored. 100 MSB of data is at bit 15. Data bits 7–0 from receive shift register are ignored. 101 MSB of data is at bit 11. Data bits 11–0 from receive shift register are ignored. 110 MSB of data is at bit 7. Data bits 15–0 from receive shift register are ignored. 111 MSB of data is at bit 3. Data bits 19–0 from receive shift register are ignored.
15–8 RFWM [7:0]	Receive FIFO Watermark. These bits configure the threshold at which the Receive FIFO Full flag will set. The RFF is set when the number of words in the Receive FIFO equal or exceed the selected threshold. It can be set to a non-zero value.
7–6	Reserved
5 RE3	Receiver #3 FIFO Enable. This bit enables receiver #3 to use the Receive FIFO. Do not change this bit when the Receiver FIFO is enabled. 0 Receiver #3 is not using the Receive FIFO. 1 Receiver #3 is using the Receive FIFO.
4 RE2	Receiver #2 FIFO Enable. This bit enables receiver #2 to use the Receive FIFO. Do not change this bit when the Receiver FIFO is enabled. 0 Receiver #2 is not using the Receive FIFO. 1 Receiver #2 is using the Receive FIFO.

Table 22-12. Receive FIFO Configuration Register Field Descriptions (Continued)

Field	Description
3 RE1	Receiver #1 FIFO Enable. This bit enables receiver #1 to use the Receive FIFO. Do not change this bit when the Receiver FIFO is enabled. 0 Receiver #1 is not using the Receive FIFO. 1 Receiver #1 is using the Receive FIFO.
2 RE0	Receiver #0 FIFO Enable. This bit enables receiver #0 to use the Receive FIFO. Do not change this bit when the Receiver FIFO is enabled. 0 Receiver #0 is not using the Receive FIFO. 1 Receiver #0 is using the Receive FIFO.
1 RFR	Receive FIFO Reset. This bit resets the Receive FIFO pointers. 0 Receive FIFO not reset. 1 Receive FIFO reset.
0 RFE	Receive FIFO Enable. This bit enables the use of the Receive FIFO. 0 Receive FIFO disabled. 1 Receive FIFO enabled.

22.3.3.8 Receive FIFO Status Register (RFSR)

Offset 0x401C (RFSR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	NRFI[1:0]		0	0	NRFO[1:0]		RFCNT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-10. Receive FIFO Status Register

Table 22-13. Receive FIFO Status Register Field Descriptions

Field	Description
31–14	Reserved
13–12 NRFI [1:0]	Next Receiver FIFO In. Indicates which Receiver Data Register the Receive FIFO will load next. This will usually equal the lowest enabled receiver, unless the receive FIFO is full. 00 Receiver #0 returns next word to the Receive FIFO. 01 Receiver #1 returns next word to the Receive FIFO. 10 Receiver #2 returns next word to the Receive FIFO. 11 Receiver #3 returns next word to the Receive FIFO.
11–10	Reserved

Table 22-13. Receive FIFO Status Register Field Descriptions (Continued)

Field	Description
9–8 NRFO [1:0]	Next Receiver FIFO Out. Indicates which receiver returns the top word of the Receive FIFO. 00 Receiver #0 returns next word from the Receive FIFO. 01 Receiver #1 returns next word from the Receive FIFO. 10 Receiver #2 returns next word from the Receive FIFO. 11 Receiver #3 returns next word from the Receive FIFO.
7–0 RFCNT [7:0]	Receive FIFO Counter. These bits indicate the number of data words stored in the Receive FIFO.

22.3.3.9 ESAI Transmit Data Registers (TX5, TX4, TX3, TX2,TX1,TX0)

TX5, TX4, TX3, TX2, TX1 and TX0 are 32-bit write-only registers. Data to be transmitted is written into these registers and is automatically transferred to the transmit shift registers (Figure 22-12 and Figure 22-13). The data written (8, 12, 16, 20, or 24 bits) should occupy the most significant portion of the TXx according to the ALC control bit setting. The unused bits (least significant portion and the 8 most significant bits when ALC=1) of the TXx are don't care bits. The Core is interrupted whenever the TXx becomes empty if the transmit data register empty interrupt has been enabled.

Offset 0x4080–0x4094 (TX0 - TX5) Access: User write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W									TXx[23:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TXx[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-11. ESAI Transmit Data Registers

Table 22-14. ESAI Transmit Data Registers

Field	Description
31-24	Reserved
23–0 TXx [23:0]	Stores the data to be transmitted and is automatically transferred to the transmit shift registers.

22.3.3.10 ESAI Transmit Shift Registers

The transmit shift registers contain the data being transmitted (Figure 22-12 and Figure 22-13). Data is shifted out to the serial transmit data pins by the selected (internal/external) bit clock when the associated

frame sync I/O is asserted. The number of bits shifted out before the shift registers are considered empty and may be written to again can be 8, 12, 16, 20, 24 or 32 bits (determined by the slot length control bits in the TCR register). Data is shifted out of these registers MSB first if TSHFD=0 and LSB first if TSHFD=1.

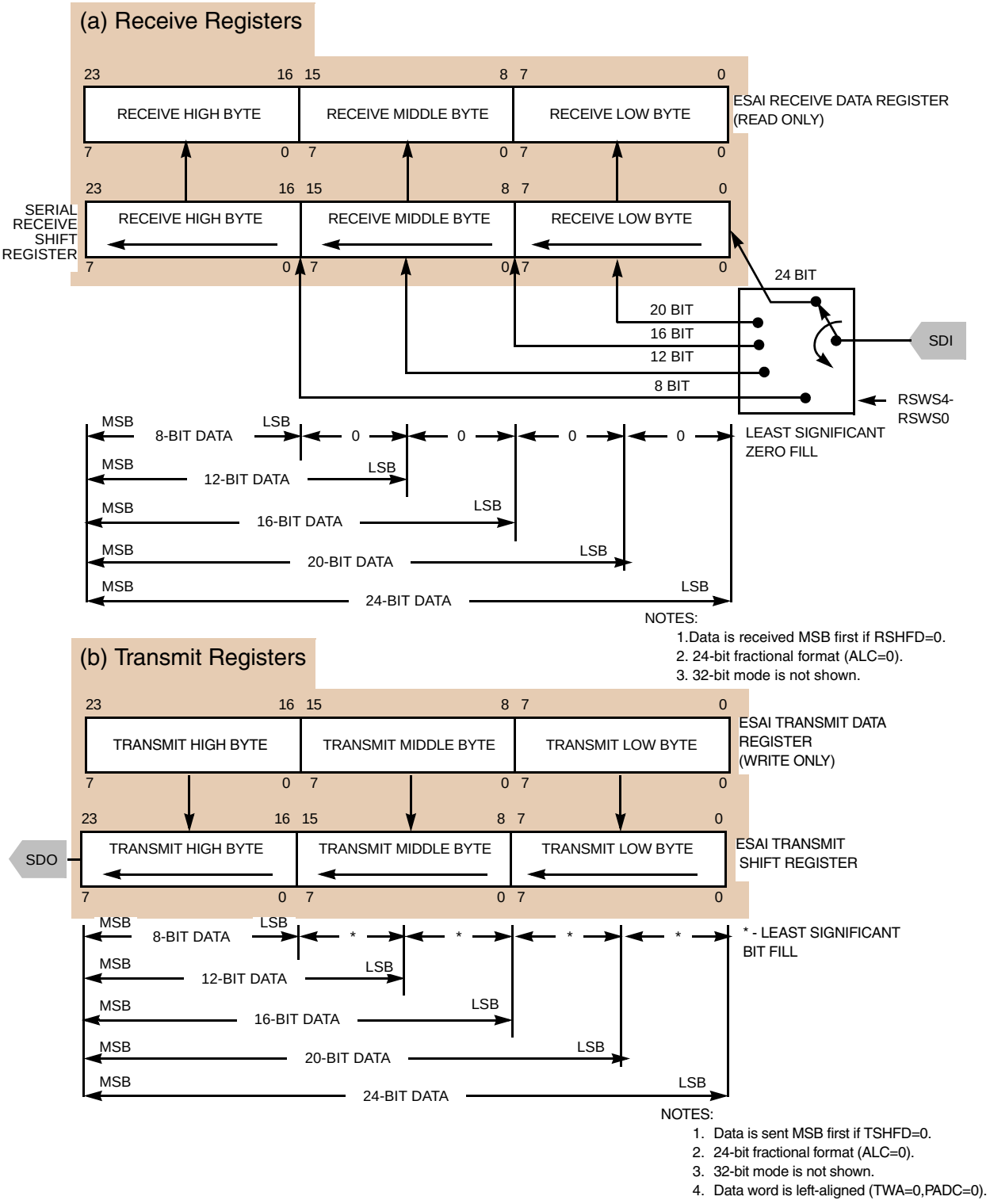


Figure 22-12. ESAI Data Path Programming Model ([R/T]SHFD=0)

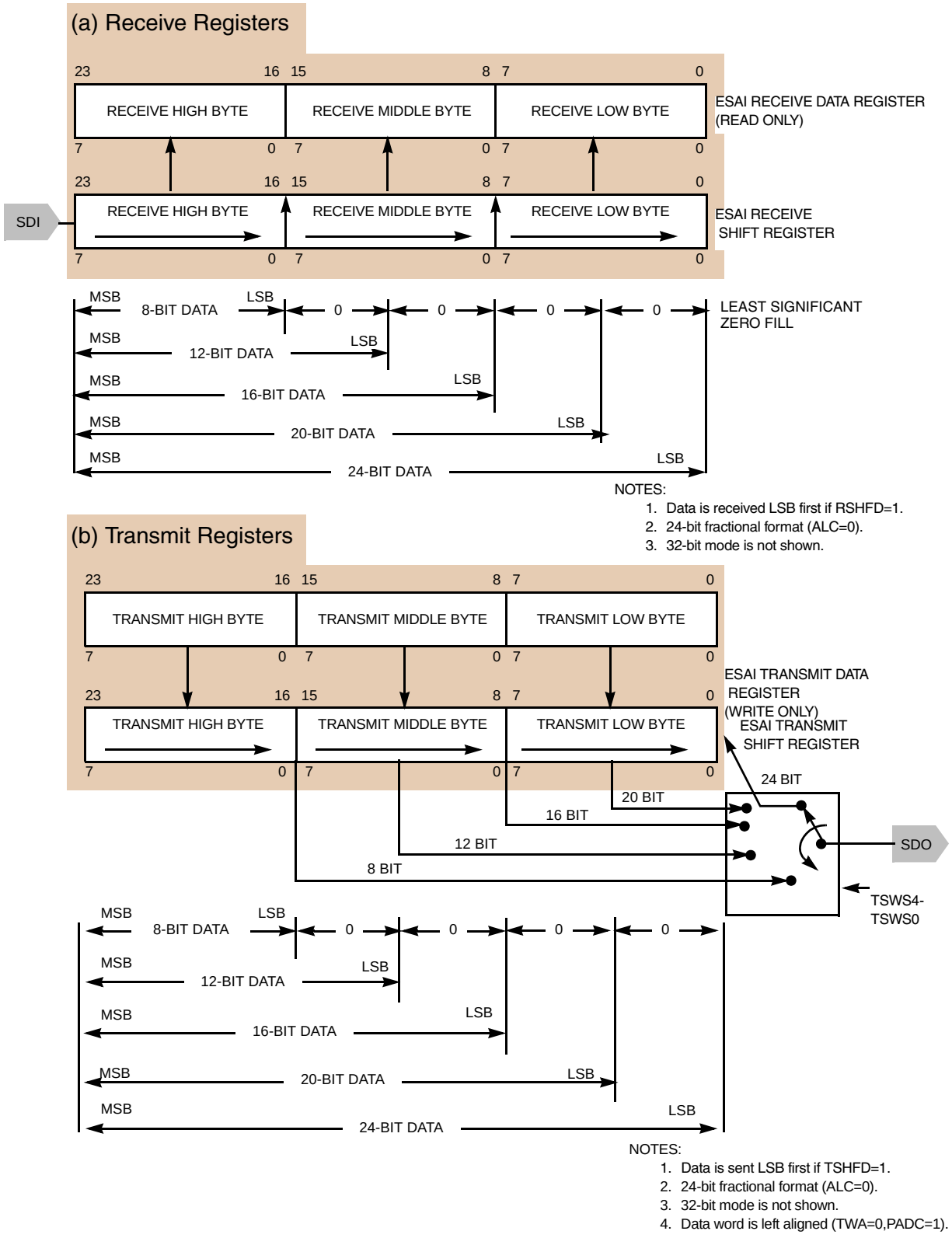


Figure 22-13. ESAI Data Path Programming Model ([R/T]SHFD=1)

22.3.3.11 ESAI Transmit Slot Register (TSR)

Offset 0x4098 (TSR) Access: User write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W									TSR[23:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	TSR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-14. ESAI Transmit Slot Register

Table 22-15. ESAI Transmit Slot Register

Field	Description
31-24	Reserved
23–0 TSR [23:0]	The write-only Transmit Slot Register (TSR) is effectively a null data register that is used when the data is not to be transmitted in the available transmit time slot. The transmit data pins of all the enabled transmitters are in the high-impedance state for the respective time slot where TSR has been written. The Transmitter External Buffer Enable pin (FSR pin when SYN=1, TEBE=1, RFSD=1) disables the external buffers during the slot when the TSR register has been written.

22.3.3.12 ESAI Receive Data Registers (RX3, RX2, RX1, RX0)

RX3, RX2, RX1, and RX0 are 32-bit read-only registers that accept data from the receive shift registers when they become full (Figure 22-12 and Figure 22-13). The data occupies the most significant portion of the receive data registers, according to the ALC control bit setting. The unused bits (least significant portion and 8 most significant bits when ALC=1) read as zeros. The Core is interrupted whenever RXx becomes full if the associated interrupt is enabled.

Offset 0x40A0–0x40AC (RX0 - RX3) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	RXx[23:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXx[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-15. ESAI Receive Data Registers

Table 22-16. ESAI Receive Data Register Field Descriptions

Field	Description
31–24	Reserved
23-0 RXx [23:0]	Accept data from the receive shift registers when they become full

22.3.3.13 ESAI Receive Shift Registers

The receive shift registers (Figure 22-12 and Figure 22-13) receive the incoming data from the serial receive data pins. Data is shifted in by the selected (internal/external) bit clock when the associated frame sync I/O is asserted. Data is assumed to be received MSB first if RSHFD=0 and LSB first if RSHFD=1. Data is transferred to the ESAI receive data registers after 8, 12, 16, 20, 24, or 32 serial clock cycles were counted, depending on the slot length control bits in the RCR register.

22.3.3.14 ESAI Status Register (SAISR)

The Status Register (SAISR) is a read-only status register used by the ARM-Core to read the status and serial input flags of the ESAI. The status bits are described in the following paragraphs.

Offset 0x40CC (SAISR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TODFE	TEDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TDE	TUE	TFS	0	0	RODF	REDF	RDF	ROE	RFS	0	0	0	IF2	IF1	IF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-16. ESAI Status Register

Table 22-17. ESAI Status Register Field Descriptions

Field	Description
31–18	Reserved.
17 TODFE	SAISR Transmit Odd-Data Register Empty. When set, TODFE indicates that the enabled transmitter data registers became empty at the beginning of an odd time slot. Odd time slots are all odd-numbered slots (1, 3, 5, and so on). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. This flag is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TODFE indicates that data should be written to all the TX registers of the enabled transmitters or to the transmit slot register (TSR). TODE is cleared when the Core writes to all the transmit data registers of the enabled transmitters, or when the Core writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TODFE is set. Hardware, software, ESAI individual reset clear TODFE.
16 TEDE	SAISR Transmit Even-Data Register Empty. When set, TEDE indicates that the enabled transmitter data registers became empty at the beginning of an even time slot. Even time slots are all even-numbered slots (0, 2, 4, 6, etc.). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. The zero time slot is considered even. This flag is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TEDE indicates that data should be written to all the TX registers of the enabled transmitters or to the transmit slot register (TSR). TEDE is cleared when the Core writes to all the transmit data registers of the enabled transmitters, or when the Core writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TEDE is set. Hardware, software, ESAI individual reset clear TEDE.
15 TDE	SAISR Transmit Data Register Empty. TDE is set when the contents of the transmit data register of all the enabled transmitters are transferred to the transmit shift registers; it is also set for a TSR disabled time slot period in network mode (as if data were being transmitted after the TSR was written). When set, TDE indicates that data should be written to all the TX registers of the enabled transmitters or to the transmit slot register (TSR). TDE is cleared when the Core writes to all the transmit data registers of the enabled transmitters, or when the Core writes to the TSR to disable transmission of the next time slot. If TIE is set, an ESAI transmit data interrupt request is issued when TDE is set. Hardware, software, ESAI individual reset clear TDE.

Table 22-17. ESAI Status Register Field Descriptions (Continued)

Field	Description
14 TUE	SAISR Transmit Underrun Error Flag. TUE is set when at least one of the enabled serial transmit shift registers is empty (no new data to be transmitted) and a transmit time slot occurs. When a transmit underrun error occurs, the previous data (which is still present in the TX registers that were not written) is retransmitted. If TEIE is set, an ESAI transmit data with exception (underrun error) interrupt request is issued when TUE is set. Hardware, software, ESAI individual reset clear TUE. TUE is also cleared by reading the SAISR with TUE set, followed by writing to all the enabled transmit data registers or to TSR.
13 TFS	SAISR Transmit Frame Sync Flag. When set, TFS indicates that a transmit frame sync occurred in the current time slot. TFS is set at the start of the first time slot in the frame and cleared during all other time slots. Data written to a transmit data register during the time slot when TFS is set is transmitted (in network mode), if the transmitter is enabled, during the second time slot in the frame. TFS is useful in network mode to identify the start of a frame. TFS is cleared by hardware, software, ESAI individual reset. TFS is valid only if at least one transmitter is enabled, that is, one or more of TE0, TE1, TE2, TE3, TE4 and TE5 are set. (In normal mode, TFS always reads as a one when transmitting data because there is only one time slot per frame – the “frame sync” time slot.)
12-11	Reserved.
10 RODF	SAISR Receive Odd-Data Register Full. When set, RODF indicates that the received data in the receive data registers of the enabled receivers have arrived during an odd time slot when operating in the network mode. Odd time slots are all odd-numbered slots (1, 3, 5, and so on). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. RODF is set when the contents of the receive shift registers are transferred to the receive data registers. RODF is cleared when the Core reads all the enabled receive data registers or cleared by hardware, software, ESAI individual resets.
9 REDF	SAISR Receive Even-Data Register Full. When set, REDF indicates that the received data in the receive data registers of the enabled receivers have arrived during an even time slot when operating in the network mode. Even time slots are all even-numbered slots (0, 2, 4, 6, and so on). Time slots are numbered from zero to N-1, where N is the number of time slots in the frame. The zero time slot is considered even. REDF is set when the contents of the receive shift registers are transferred to the receive data registers. REDF is cleared when the Core reads all the enabled receive data registers or cleared by hardware, software, ESAI individual resets. If REDIE is set, an ESAI receive even slot data interrupt request is issued when REDF is set.
8 RDF	SAISR Receive Data Register Full. RDF is set when the contents of the receive shift register of an enabled receiver is transferred to the respective receive data register. RDF is cleared when the Core reads the receive data register of all enabled receivers or cleared by hardware, software, ESAI individual reset. If RIE is set, an ESAI receive data interrupt request is issued when RDF is set.
7 ROE	SAISR Receive Overrun Error Flag. The ROE flag is set when the serial receive shift register of an enabled receiver is full and ready to transfer to its receiver data register (RXx) and the register is already full (RDF=1). If REIE is set, an ESAI receive data with exception (overrun error) interrupt request is issued when ROE is set. Hardware, software, ESAI individual reset clear ROE. ROE is also cleared by reading the SAISR with ROE set, followed by reading all the enabled receive data registers.
6 RFS	SAISR Receive Frame Sync Flag. When set, RFS indicates that a receive frame sync occurred during reception of the words in the receiver data registers. This indicates that the data words are from the first slot in the frame. When RFS is clear and a word is received, it indicates (only in the network mode) that the frame sync did not occur during reception of that word. RFS is cleared by hardware, software, ESAI individual reset. RFS is valid only if at least one of the receivers is enabled (REx=1). (In normal mode, RFS always reads as a one when reading data because there is only one time slot per frame – the “frame sync” time slot)
5-3	Reserved.

Table 22-17. ESAI Status Register Field Descriptions (Continued)

Field	Description
2 IF2	SAISR Serial Input Flag 2. The IF2 bit is enabled only when the HCKR pin is defined as ESAI in the Port Control Register, SYN=1 and RHCKD=0, indicating that HCKR is an input flag and the synchronous mode is selected. Data present on the HCKR pin is latched during reception of the first received data bit after frame sync is detected. The IF2 bit is updated with this data when the receive shift registers are transferred into the receiver data registers. IF2 reads as a zero when it is not enabled. Hardware, software, ESAI individual reset clear IF2.
1 IF1	SAISR Serial Inout Flag 1. The IF1 bit is enabled only when the FSR pin is defined as ESAI in the Port Control Register, SYN =1, RFSD=0 and TEBE=0, indicating that FSR is an input flag and the synchronous mode is selected. Data present on the FSR pin is latched during reception of the first received data bit after frame sync is detected. The IF1 bit is updated with this data when the receiver shift registers are transferred into the receiver data registers. IF1 reads as a zero when it is not enabled. Hardware, software, ESAI individual reset clear IF1.
0 IF0	SAISR Serial Input Flag 0. The IF0 bit is enabled only when the SCKR pin is defined as ESAI in the Port Control Register, SYN=1 and RCKD=0, indicating that SCKR is an input flag and the synchronous mode is selected. Data present on the SCKR pin is latched during reception of the first received data bit after frame sync is detected. The IF0 bit is updated with this data when the receiver shift registers are transferred into the receiver data registers. IF0 reads as a zero when it is not enabled. Hardware, software, ESAI individual reset clear IF0.

22.3.3.15 ESAI Common Control Register (SAICR)

The read/write Common Control Register (SAICR) contains control bits for functions that affect both the receive and transmit sections of the ESAI.

Offset 0x40D0 (SAICR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	ALC	TEBE	SYN	0	0	0	OF2	OF1	OF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-17. Receive Common Control Register

Table 22-18. Common Control Register Field Descriptions

Field	Description
31–9	Reserved
8 ALC	SAICR Alignment Control. The ESAI is designed for 24-bit fractional data, thus shorter data words are left aligned to the MSB (bit 23). Some applications use 16-bit fractional data. In those cases, shorter data words may be left aligned to bit 15. The Alignment Control (ALC) bit supports these applications. If ALC is set, transmitted and received words are left aligned to bit 15 in the transmit and receive shift registers. If ALC is cleared, transmitted and received word are left aligned to bit 23 in the transmit and receive shift registers. While ALC is set, 20-bit and 24-bit words may not be used, and word length control should specify 8-, 12-, or 16-bit words; otherwise, results are unpredictable.
7 TEBE	SAICR Transmit External Buffer Enable. The Transmitter External Buffer Enable (TEBE) bit controls the function of the FSR pin when in the synchronous mode. If the ESAI is configured for operation in the synchronous mode (SYN=1), and TEBE is set while FSR pin is configured as an output (RFSD=1), the FSR pin functions as the transmitter external buffer enable control to enable the use of an external buffers on the transmitter outputs. If TEBE is cleared, the FSR pin functions as the serial I/O flag 1. See Table 22-30 for a summary of the effects of TEBE on the FSR pin.
6 SYNC	SAICR Synchronous Mode Selection. The Synchronous Mode Selection (SYN) bit controls whether the receiver and transmitter sections of the ESAI operate synchronously or asynchronously with respect to each other (see Figure 22-18). When SYN is cleared, the asynchronous mode is chosen and independent clock and frame sync signals are used for the transmit and receive sections. When SYN is set, the synchronous mode is chosen and the transmit and receive sections use common clock and frame sync signals. When in the synchronous mode (SYN=1), the transmit and receive sections use the transmitter section clock generator as the source of the clock and frame sync for both sections. Also, the receiver clock pins SCKR, FSR and HCKR now operate as I/O flags. See Table 22-29, Table 22-30, and Table 22-31 for the effects of SYN on the receiver clock pins.
5-3	Reserved.
2 OF2	SAICR Serial Output Flag 2. The Serial Output Flag 2 (OF2) is a data bit used to hold data to be send to the OF2 pin. When the ESAI is in the synchronous clock mode (SYN=1), the HCKR pin is configured as the ESAI flag 2. If the receiver high frequency clock direction bit (RHCKD) is set, the HCKR pin is the output flag OF2, and data present in the OF2 bit is written to the OF2 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.
1 OF1	SAICR Serial Output Flag 1. The Serial Output Flag 1 (OF1) is a data bit used to hold data to be send to the OF1 pin. When the ESAI is in the synchronous clock mode (SYN=1), the FSR pin is configured as the ESAI flag 1. If the receiver frame sync direction bit (RFSD) is set and the TEBE bit is cleared, the FSR pin is the output flag OF1, and data present in the OF1 bit is written to the OF1 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.
0 OF0	SAICR Serial Output Flag 0. The Serial Output Flag 0 (OF0) is a data bit used to hold data to be send to the OF0 pin. When the ESAI is in the synchronous clock mode (SYN=1), the SCKR pin is configured as the ESAI flag 0. If the receiver serial clock direction bit (RCKD) is set, the SCKR pin is the output flag OF0, and data present in the OF0 bit is written to the OF0 pin at the beginning of the frame in normal mode or at the beginning of the next time slot in network mode.

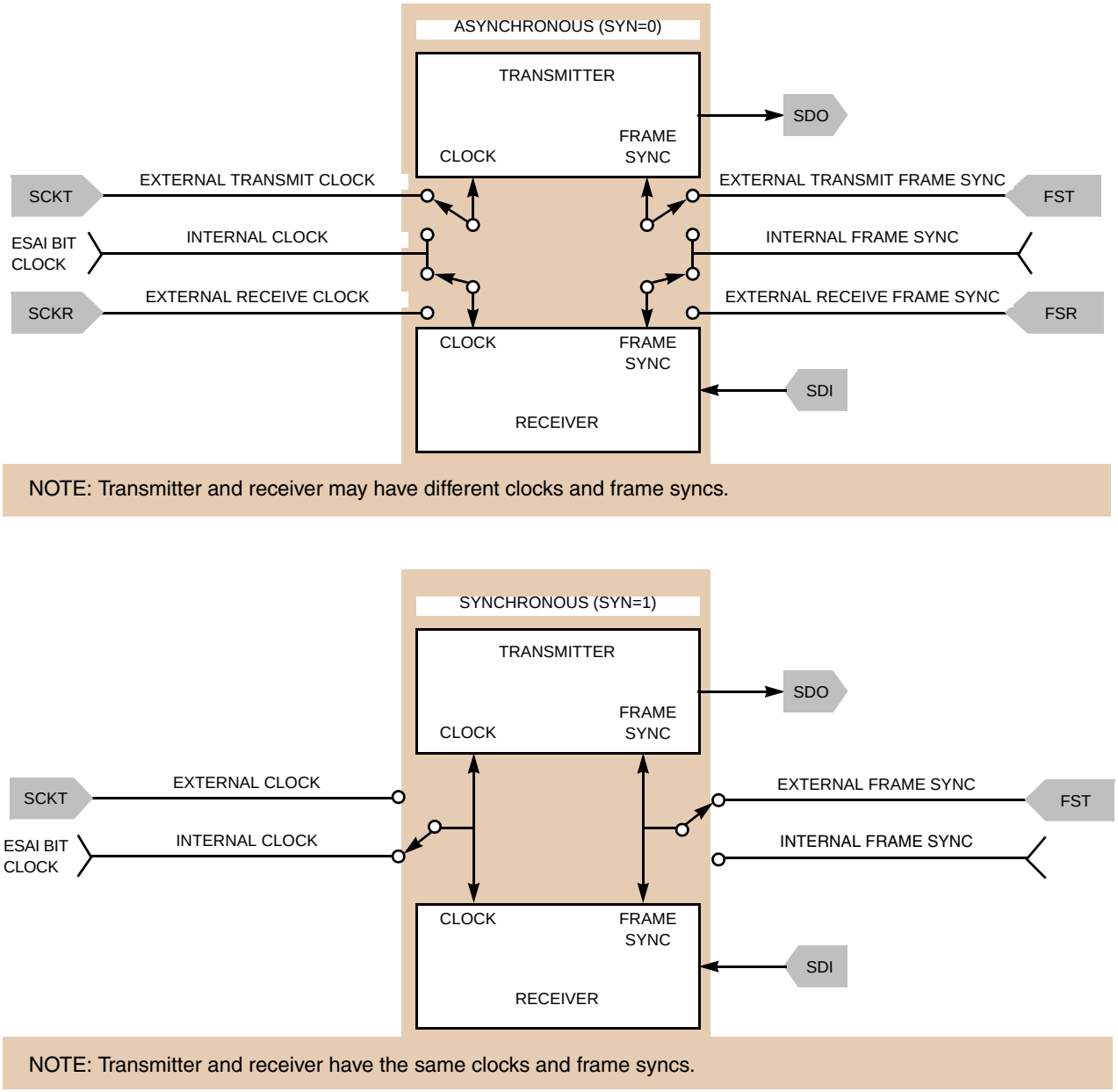


Figure 22-18. SAICR SYN Bit Operation

22.3.3.16 ESAI Transmit Control Register (TCR)

The read/write Transmit Control Register (TCR) controls the ESAI transmitter section. Interrupt enable bits for the transmitter section are provided in this control register. Operating modes are also selected in this register.

Offset 0x40D4 (TCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TLIE	TIE	TDEIE	TEIE	TPR	0	PADC	TFSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFSL	TSWS[4:0]					TMOD[1:0]		TWA	TSHFD	TE5	TE4	TE3	TE2	TE1	TE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-19. ESAI Transmit Control Register

Table 22-19. ESAI Transmit Control Register Field Descriptions

Field	Description
31-24	Reserved.
23 TLIE	TCR Transmit Last Slot Interrupt Enable. TLIE enables an interrupt at the beginning of last slot of a frame in network mode. When TLIE is set the Core is interrupted at the start of the last slot in a frame in network mode regardless of the transmit mask register setting. When TLIE is cleared the transmit last slot interrupt is disabled. TLIE is disabled when TDC[4:0]=$00000 (on-demand mode). The use of the transmit last slot interrupt is described in Section 22.4.2, “ESAI Interrupt Requests”.
22 TIE	TCR Transmit Interrupt Enable. The Core is interrupted when TIE and the TDE flag in the SAISR status register are set. When TIE is cleared, this interrupt is disabled. Writing data to all the data registers of the enabled transmitters or to TSR clears TDE, thus clearing the interrupt. Transmit interrupts with exception have higher priority than normal transmit data interrupts, therefore if exception occurs (TUE is set) and TEIE is set, the ESAI requests an ESAI transmit data with exception interrupt from the interrupt controller.
21 TEDIE	TCR Transmit Even Slot Data Interrupt Enable. The TEDIE control bit is used to enable the transmit even slot data interrupts. If TEDIE is set, the transmit even slot data interrupts are enabled. If TEDIE is cleared, the transmit even slot data interrupts are disabled. A transmit even slot data interrupt request is generated if TEDIE is set and the TEDE status flag in the SAISR status register is set. Even time slots are all even-numbered time slots (0, 2, 4, etc.) when operating in network mode. The zero time slot in the frame is marked by the frame sync signal and is considered to be even. Writing data to all the data registers of the enabled transmitters or to TSR clears the TEDE flag, thus servicing the interrupt. Transmit interrupts with exception have higher priority than transmit even slot data interrupts, therefore if exception occurs (TUE is set) and TEIE is set, the ESAI requests an ESAI transmit data with exception interrupt from the interrupt controller.
20 TEIE	TCR Transmit Exception Interrupt Enable. When TEIE is set, the Core is interrupted when both TDE and TUE in the SAISR status register are set. When TEIE is cleared, this interrupt is disabled. Reading the SAISR status register followed by writing to all the data registers of the enabled transmitters clears TUE, thus clearing the pending interrupt.

Table 22-19. ESAI Transmit Control Register Field Descriptions (Continued)

Field	Description
19 TPR	TCR Transmit Section Personal Reset. The TPR control bit is used to put the transmitter section of the ESAI in the personal reset state. The receiver section is not affected. When TPR is cleared, the transmitter section may operate normally. When TPR is set, the transmitter section enters the personal reset state immediately. When in the personal reset state, the status bits are reset to the same state as after hardware reset. The control bits are not affected by the personal reset state. The transmitter data pins are tri-stated while in the personal reset state; if a stable logic level is desired, the transmitter data pins should be defined as GPIO outputs, or external pull-up or pull-down resistors should be used. The transmitter clock outputs drive zeroes while in the personal reset state. Note that to leave the personal reset state by clearing TPR, the procedure described in Section 22.5.2, "ESAI Initialization Examples" should be followed.
18	Reserved.
17 PADC	TCR Transmit Zero Padding Control. When PADC is cleared, zero padding is disabled. When PADC is set, zero padding is enabled. PADC, in conjunction with the TWA control bit, determines the way that padding is done for operating modes where the word length is less than the slot length. See the TWA bit description in bit 7 for more details. Since the data word is shorter than the slot length, the data word is extended until achieving the slot length, according to the following rule: 1. If the data word is left-aligned (TWA=0), and zero padding is disabled (PADC=0), the last data bit is repeated after the data word has been transmitted. If zero padding is enabled (PADC=1), zeroes are transmitted after the data word has been transmitted. 2. If the data word is right-aligned (TWA=1), and zero padding is disabled (PADC=0), the first data bit is repeated before the transmission of the data word. If zero padding is enabled (PADC=1), zeroes are transmitted before the transmission of the data word.
16 TFSR	TCR Transmit Frame Sync Relative Timing. TFSR determines the relative timing of the transmit frame sync signal as referred to the serial data lines, for a word length frame sync only (TFSL=0). When TFSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When TFSR is set the word length frame sync starts one serial clock cycle earlier, that is, together with the last bit of the previous data word.
15 TFSL	TCR Transmit Frame Sync Length. The TFSL bit selects the length of frame sync to be generated or recognized. If TFSL is cleared, a word-length frame sync is selected. If TFSL is set, a 1-bit clock period frame sync is selected. See Figure 22-21 for examples of frame length selection.
14-10 TSWS [14:10]	TCR Tx Slot and Word Length Select (TSWS4-TSWS0). The TSWS4–TSWS0 bits are used to select the length of the slot and the length of the data words being transferred using the ESAI. The word length must be equal to or shorter than the slot length. The possible combinations are shown in Table 22-21. See also the ESAI data path programming model in Figure 22-12 and Figure 22-13.
9-8 TMOD [9:8]	TCR Transmit Network Mode Control (TMOD1-TMOD0). The TMOD1 and TMOD0 bits are used to define the network mode of ESAI transmitters, as shown in Table 22-20. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot, as shown in Figure 22-20. In network mode, it is possible to transfer a word for every time slot, as shown in Figure 22-20. For further details, see Section 22.1.2, "Modes of Operation." In order to comply with AC-97 specifications, TSWS4-TSWS0 should be set to 00011 (20-bit slot, 20-bit word length), TFSL and TFSR should be cleared, and TDC4-TDC0 should be set to $0C (13 words in frame). If TMOD[1:0]=$11 and the above recommendations are followed, the first slot and word will be 16 bits long, and the next 12 slots and words will be 20 bits long, as required by the AC97 protocol.

Table 22-19. ESAI Transmit Control Register Field Descriptions (Continued)

Field	Description
7 TWA	TCR Transmit Word Alignment Control. The Transmitter Word Alignment Control (TWA) bit defines the alignment of the data word in relation to the slot. This is relevant for the cases where the word length is shorter than the slot length. If TWA is cleared, the data word is left-aligned in the slot frame during transmission. If TWA is set, the data word is right-aligned in the slot frame during transmission. Since the data word is shorter than the slot length, the data word is extended until achieving the slot length, according to the following rule: 1. If the data word is left-aligned (TWA=0), and zero padding is disabled (PADC=0), the last data bit is repeated after the data word has been transmitted. If zero padding is enabled (PADC=1), zeroes are transmitted after the data word has been transmitted. 2. If the data word is right-aligned (TWA=1), and zero padding is disabled (PADC=0), the first data bit is repeated before the transmission of the data word. If zero padding is enabled (PADC=1), zeroes are transmitted before the transmission of the data word.
6 TSHFD	TCR Transmit Shift Direction. The TSHFD bit causes the transmit shift registers to shift data out MSB first when TSHFD equals zero or LSB first when TSHFD equals one (see Figure 22-12 and Figure 22-13).
5 TE5	TCR ESAI Transmit 5 Enable. TE5 enables the transfer of data from TX5 to the transmit shift register #5. When TE5 is set and a frame sync is detected, the transmit #5 portion of the ESAI is enabled for that frame. When TE5 is cleared, the transmitter #5 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX5 when TE5 is cleared but the data is not transferred to the transmit shift register #5. The SDO5/SDI0 pin is the data input pin for RX0 if TE5 is cleared and RE0 in the RCR register is set. If both RE0 and TE5 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE0 and TE5 should not be set at the same time. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE5 and setting it again disables the transmitter #5 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO5/SDI0 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE5 can be left enabled.
4 TE4	TCR ESAI Transmit 4 Enable. TE4 enables the transfer of data from TX4 to the transmit shift register #4. When TE4 is set and a frame sync is detected, the transmit #4 portion of the ESAI is enabled for that frame. When TE4 is cleared, the transmitter #4 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX4 when TE4 is cleared but the data is not transferred to the transmit shift register #4. The SDO4/SDI1 pin is the data input pin for RX1 if TE4 is cleared and RE1 in the RCR register is set. If both RE1 and TE4 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE1 and TE4 should not be set at the same time. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE4 and setting it again disables the transmitter #4 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO4/SDI1 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE4 can be left enabled.

Table 22-19. ESAI Transmit Control Register Field Descriptions (Continued)

Field	Description
3 TE3	TCR ESAI Transmit 3 Enable. TE3 enables the transfer of data from TX3 to the transmit shift register #3. When TE3 is set and a frame sync is detected, the transmit #3 portion of the ESAI is enabled for that frame. When TE3 is cleared, the transmitter #3 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX3 when TE3 is cleared but the data is not transferred to the transmit shift register #3. The SDO3/SDI2 pin is the data input pin for RX2 if TE3 is cleared and RE2 in the RCR register is set. If both RE2 and TE3 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE2 and TE3 should not be set at the same time. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE3 and setting it again disables the transmitter #3 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO3/SDI2 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE3 can be left enabled.
2 TE2	TCR ESAI Transmit 2 Enable. TE2 enables the transfer of data from TX2 to the transmit shift register #2. When TE2 is set and a frame sync is detected, the transmit #2 portion of the ESAI is enabled for that frame. When TE2 is cleared, the transmitter #2 is disabled after completing transmission of data currently in the ESAI transmit shift register. Data can be written to TX2 when TE2 is cleared but the data is not transferred to the transmit shift register #2. The SDO2/SDI3 pin is the data input pin for RX3 if TE2 is cleared and RE3 in the RCR register is set. If both RE3 and TE2 are cleared, the transmitter and receiver are disabled, and the pin is tri-stated. Both RE3 and TE2 should not be set at the same time. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE2 and setting it again disables the transmitter #2 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO2/SDI3 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE2 can be left enabled.
1 TE1	TCR ESAI Transmit 1 Enable. TE1 enables the transfer of data from TX1 to the transmit shift register #1. When TE1 is set and a frame sync is detected, the transmit #1 portion of the ESAI is enabled for that frame. When TE1 is cleared, the transmitter #1 is disabled after completing transmission of data currently in the ESAI transmit shift register. The SDO1 output is tri-stated, and any data present in TX1 is not transmitted, that is, data can be written to TX1 with TE1 cleared, but data is not transferred to the transmit shift register #1. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE1 and setting it again disables the transmitter #1 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO1 pin remains in the high-impedance state. The on-demand mode transmit enable sequence can be the same as the normal mode, or TE1 can be left enabled.
0 TE0	TCR ESAI Transmit 0 Enable. TE0 enables the transfer of data from TX0 to the transmit shift register #0. When TE0 is set and a frame sync is detected, the transmit #0 portion of the ESAI is enabled for that frame. When TE0 is cleared, the transmitter #0 is disabled after completing transmission of data currently in the ESAI transmit shift register. The SDO0 output is tri-stated, and any data present in TX0 is not transmitted, that is, data can be written to TX0 with TE0 cleared, but data is not transferred to the transmit shift register #0. The normal mode transmit enable sequence is to write data to one or more transmit data registers before setting TEx. The normal transmit disable sequence is to clear TEx, TIE and TEIE after TDE equals one. In the network mode, the operation of clearing TE0 and setting it again disables the transmitter #0 after completing transmission of the current data word until the beginning of the next frame. During that time period, the SDO0 pin remains in the high-impedance state.The on-demand mode transmit enable sequence can be the same as the normal mode, or TE0 can be left enabled.

Table 22-20. Transmit Network Mode Selection

TMOD1	TMOD0	TDC4–TDC0	Transmitter Network Mode
0	0	$0–$1F	Normal Mode
0	1	$0	On-Demand Mode
0	1	$1–$1F	Network Mode
1	0	X	Reserved
1	1	$0C	AC97

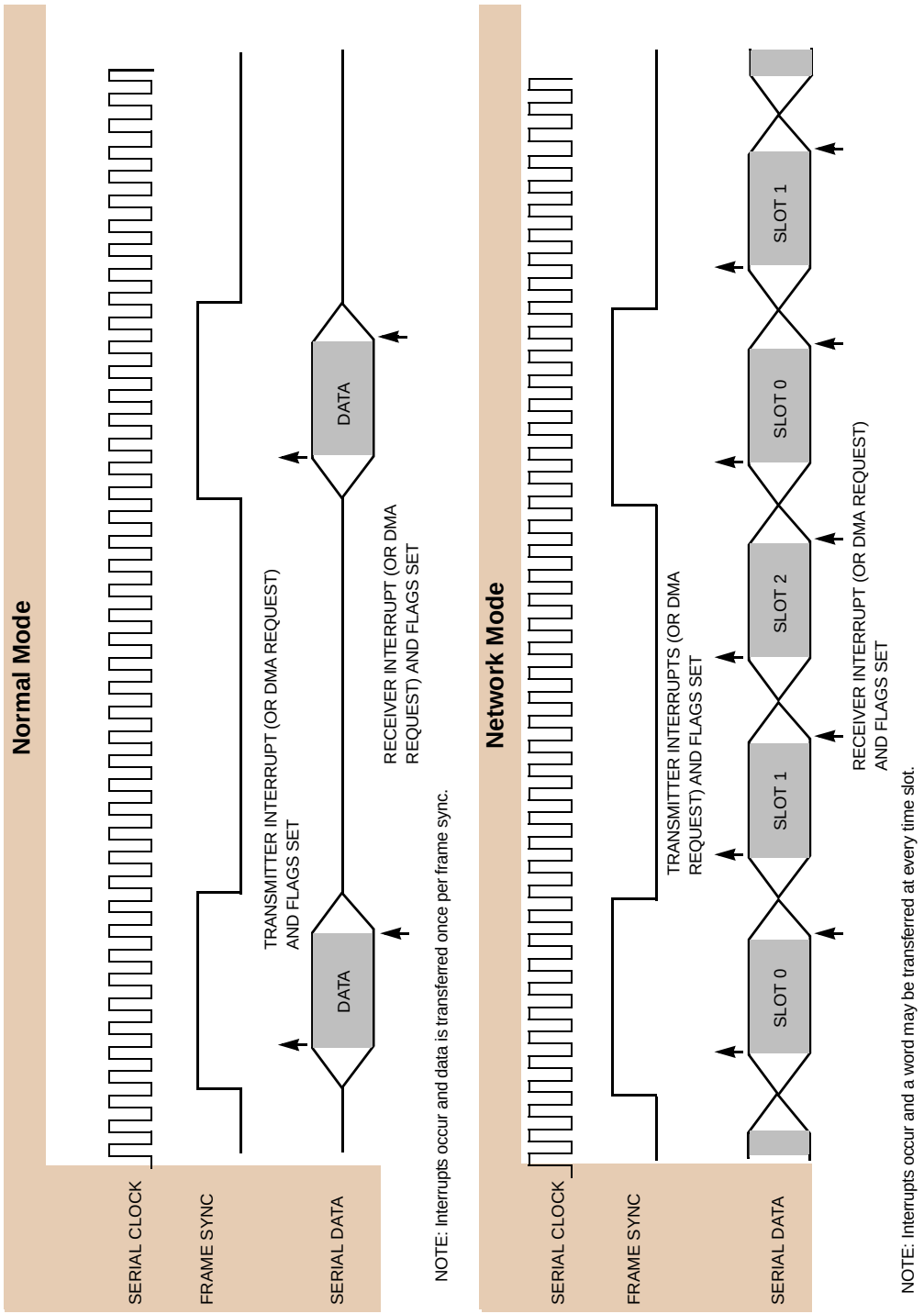


Figure 22-20. Normal and Network Operation

Table 22-21. ESAI Transmit Slot and Word Length Selection

TSWS4	TSWS3	TSWS2	TSWS1	TSWS0	SLOT LENGTH	WORD LENGTH
0	0	0	0	0	8	8
0	0	1	0	0	12	8
0	0	0	0	1		12
0	1	0	0	0	16	8
0	0	1	0	1		12
0	0	0	1	0		16
0	1	1	0	0	20	8
0	1	0	0	1		12
0	0	1	1	0		16
0	0	0	1	1		20
1	0	0	0	0	24	8
0	1	1	0	1		12
0	1	0	1	0		16
0	0	1	1	1		20
1	1	1	1	0		24
1	1	0	0	0	32	8
1	0	1	0	1		12
1	0	0	1	0		16
0	1	1	1	1		20
1	1	1	1	1		24
0	1	0	1	1	Reserved	
0	1	1	1	0		
1	0	0	0	1		
1	0	0	1	1		
1	0	1	0	0		
1	0	1	1	0		
1	0	1	1	1		
1	1	0	0	1		
1	1	0	1	0		
1	1	0	1	1		
1	1	1	0	0		
1	1	1	0	1		

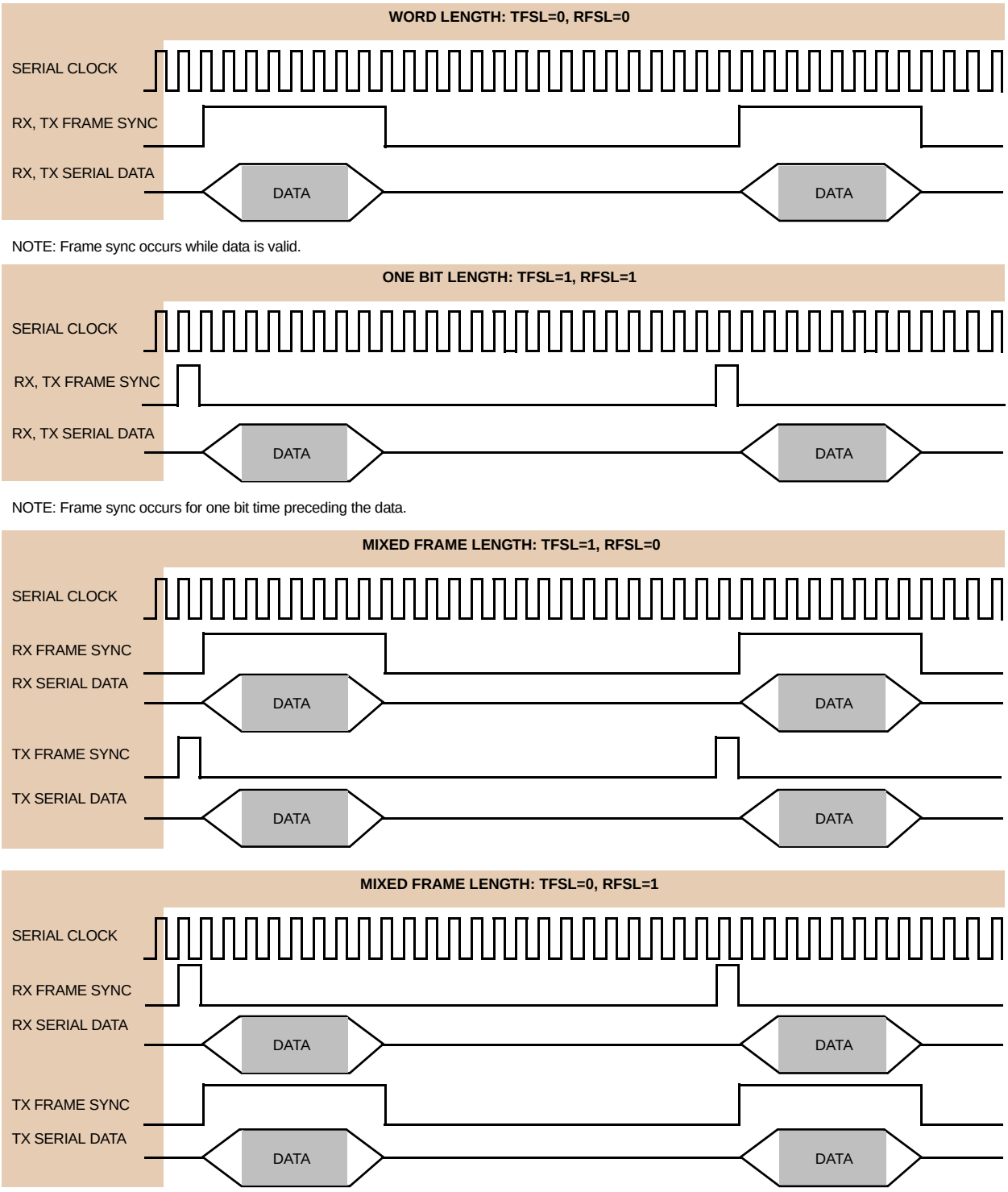


Figure 22-21. Frame Length Selection

22.3.3.17 ESAI Transmitter Clock Control Register (TCCR)

The read/write Transmitter Clock Control Register (TCCR) controls the ESAI transmitter clock generator bit and frame sync rates, the bit clock and high frequency clock sources and the directions of the HCKT, FST and SCKT signals. In the synchronous mode (SYN=1), the bit clock defined for the transmitter determines the receiver bit clock as well. TCCR also controls the number of words per frame for the serial data. Hardware and software reset clear all the bits of the TCCR register. Figure 22-22 shows the register. (Care should be taken in asynchronous mode whenever the frame sync clock (FSR, FST) is not sourced directly from its associated bit clock (SCKR,SCKT). Proper phase relationships must be maintained between these clocks in order to guarantee proper operation of the ESAI)

Offset 0x40D8 (TCCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	THCKD	TFSD	TCKD	THCKP	TFSP	TCKP	TFP[3:2]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TFP[1:0]		TDC[4:0]					TPSR	TPM[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

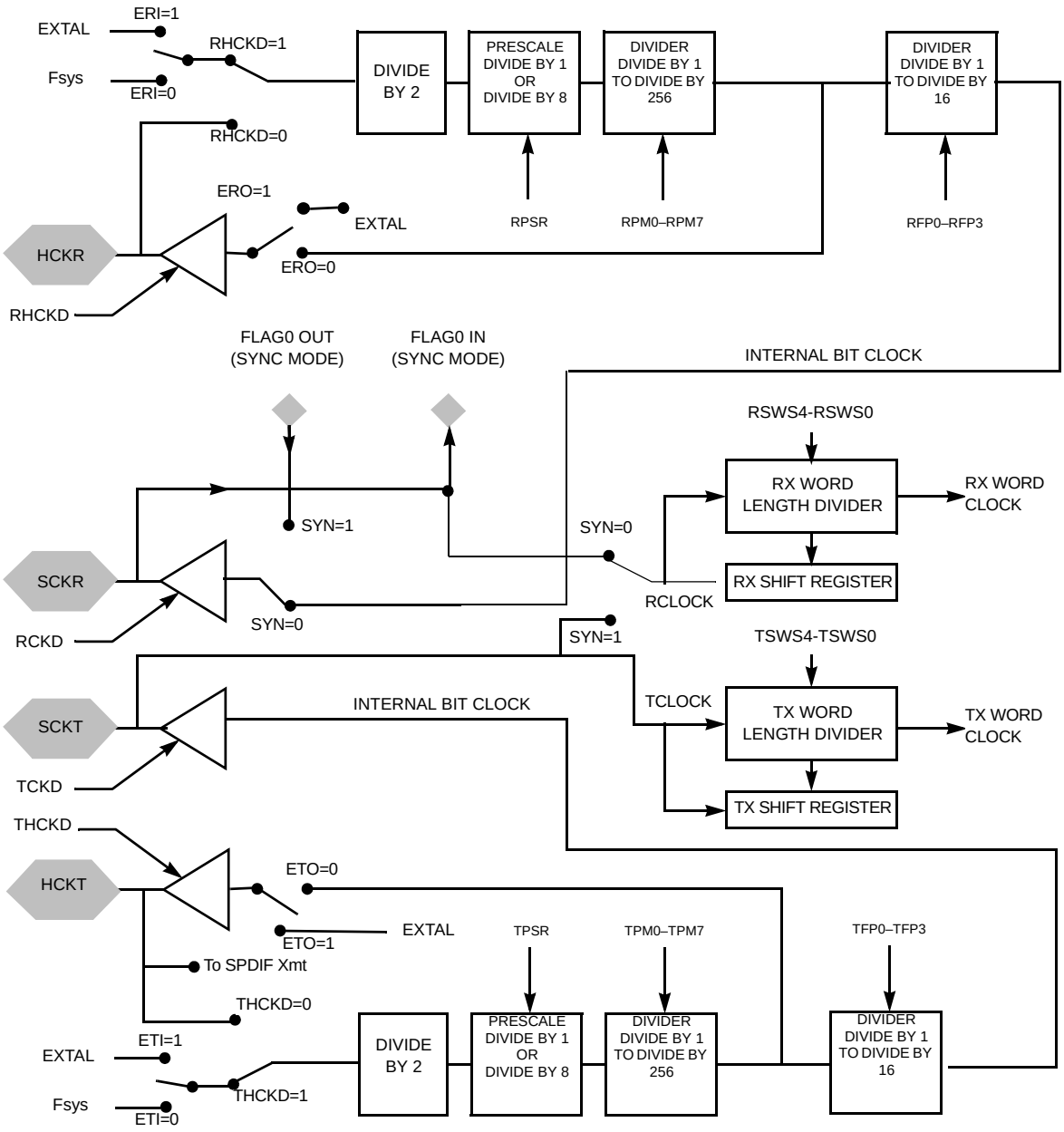
Figure 22-22. ESAI Transmitter Clock Control Register

Table 22-22. ESAI Transmitter Clock Control Register Field Descriptions

Field	Description
31-24	Reserved.
23 THCKD	TCCR Transmit High Frequency Clock Direction. THCKD controls the direction of the HCKT pin. When THCKD is cleared, HCKT is an input; when THCKD is set, HCKT is an output (see Table 22-2.).
22 TFSD	TCCR Transmit Frame Sync Signal Direction. TFSD controls the direction of the FST pin. When TFSD is cleared, FST is an input; when TFSD is set, FST is an output (see Table 22-2.).
21 TCKD	TCCR Transmit Clock Source Direction. The Transmitter Clock Source Direction (TCKD) bit selects the source of the clock signal used to clock the transmit shift registers in the asynchronous mode (SYN=0) and the transmit shift registers and the receive shift registers in the synchronous mode (SYN=1). When TCKD is set, the internal clock source becomes the bit clock for the transmit shift registers and word length divider and is the output on the SCKT pin. When TCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKT pin, and an external clock source may drive this pin (see Table 22-2.).
20 THCKP	TCCR Transmit High Frequency Clock Polarity. The Transmitter High Frequency Clock Polarity (THCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If THCKP is cleared the data and the frame sync are clocked out on the rising edge of the transmit high frequency bit clock and latched in on the falling edge of the transmit bit clock. If THCKP is set the falling edge of the transmit clock is used to clock the data out and frame sync and the rising edge of the transmit clock is used to latch the data and frame sync in.

Table 22-22. ESAI Transmitter Clock Control Register Field Descriptions (Continued)

Field	Description
19 TFSP	TCCR Transmit Frame Sync Polarity. The Transmitter Frame Sync Polarity (TFSP) bit determines the polarity of the transmit frame sync signal. When TFSP is cleared, the frame sync signal polarity is positive, that is, the frame start is indicated by a high level on the frame sync pin. When TFSP is set, the frame sync signal polarity is negative, that is, the frame start is indicated by a low level on the frame sync pin.
18 TCKP	TCCR Transmit Clock Polarity. The Transmitter Clock Polarity (TCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If TCKP is cleared the data and the frame sync are clocked out on the rising edge of the transmit bit clock and latched in on the falling edge of the transmit bit clock. If TCKP is set the falling edge of the transmit clock is used to clock the data out and frame sync and the rising edge of the transmit clock is used to latch the data and frame sync in.
17-14 TFP [3:0]	TCCR Tx High Frequency Clock Divider. The TFP3–TFP0 bits control the divide ratio of the transmitter high frequency clock to the transmitter serial bit clock when the source of the high frequency clock and the bit clock is the internal ARM-core clock. When the HCKT input is being driven from an external high frequency clock, the TFP3-TFP0 bits specify an additional division ratio in the clock divider chain. Table 22-23 shows the specification for the divide ratio. Figure 22-23 shows the ESAI high frequency clock generator functional diagram.
13-9 TDC [4:0]	TCCR Tx Frame Rate Divider Control. The TDC4–TDC0 bits control the divide ratio for the programmable frame rate dividers used to generate the transmitter frame clocks. In network mode, this ratio may be interpreted as the number of words per frame minus one. The divide ratio may range from 2 to 32 (TDC[4:0]=00001 to 11111) for network mode. A divide ratio of one (TDC[4:0]=00000) in network mode is a special case (on-demand mode). In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (TDC[4:0]=00000 to 11111) for normal mode. In normal mode, a divide ratio of 1 (TDC[4:0]=00000) provides continuous periodic data word transfers. A bit-length frame sync (TFSL=1) must be used in this case. The ESAI frame sync generator functional diagram is shown in Figure 22-24.
8 TPSR	TCCR Transmit Prescaler Range. The TPSR bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When TPSR is set, the fixed prescaler is bypassed. When TPSR is cleared, the fixed divide-by-eight prescaler is operational (see Figure 22-23). The maximum internally generated bit clock frequency is Fsys/4; the minimum internally generated bit clock frequency is Fsys/(2 x 8 x 256)=Fsys/4096. (Do not use the combination TPSR=1, TPM7–TPM0=$00, and TFP3–TFP0=$0 which causes synchronization problems when using the internal ARM-core clock as source (TCKD=1 or THCKD=1))
7-0 TPM [7:0]	TCCR Transmit Prescale Modulus Select. The TPM7–TPM0 bits specify the divide ratio of the prescale divider in the ESAI transmitter clock generator. A divide ratio from 1 to 256 (TPM[7:0]=$00 to $FF) may be selected. The bit clock output is available at the transmit serial bit clock (SCKT) pin. The bit clock output is also available internally for use as the bit clock to shift the transmit and receive shift registers. The ESAI transmit clock generator functional diagram is shown in Figure 22-23.



Notes:
1. ETI, ETO, ERI and ERO bit descriptions are covered in Table 22-8.
2. Fsys is the ESAI system 133 MHz clock.
3. EXTAL is the on-chip clock sources other than ESAI system 133MHz clock. In i.MX35 it is connected to the EXTAL_AUDIO clock.

Figure 22-23. ESAI Clock Generator Functional Block Diagram

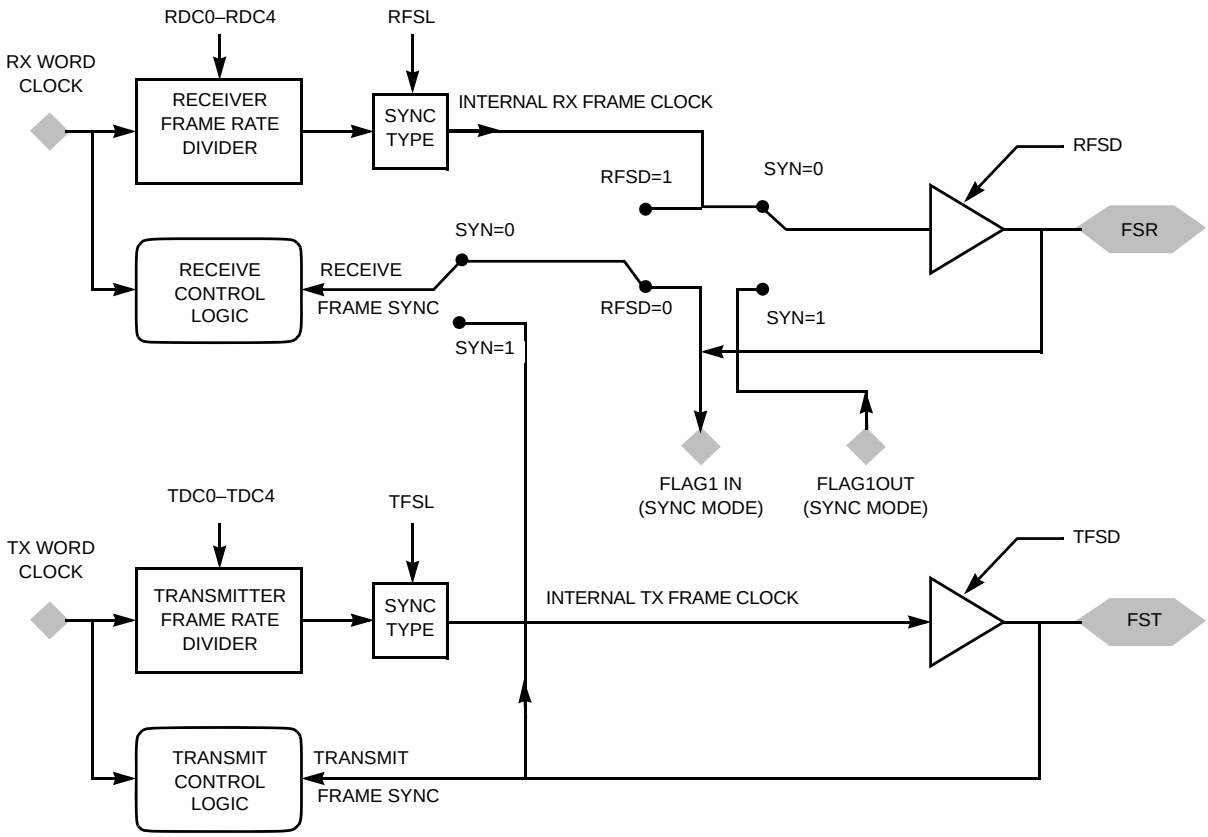


Figure 22-24. ESAI Frame Sync Generator Functional Block Diagram

Table 22-23. Transmitter High Frequency Clock Divider

TFP3-TFP0	Divide Ratio
$0	1
$1	2
$2	3
$3	4
...	...
$F	16

22.3.3.18 ESAI Receive Control Register (RCR)

The read/write Receive Control Register (RCR) controls the ESAI receiver section. Interrupt enable bits for the receivers are provided in this control register. The receivers are enabled in this register (0,1,2 or 3 receivers can be enabled) if the input data pin is not used by a transmitter. Operating modes are also selected in this register.

Offset 0x40DC (RCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	RLIE	RIE	REDIE	REIE	RPR	0	0	RFSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFSL	RSWS [4:0]					RMOD [1:0]		RWA[7:0]	RSHFD	0	0	RE3	RE2	RE1	RE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-25. ESAI Receive Control Register

Table 22-24. ESAI Receive Control Register Field Descriptions

Field	Description
31-24	Reserved.
23 RLIE	RCR Receive Last Slot Interrupt Enable. RLIE enables an interrupt after the last slot of a frame ended in network mode only. When RLIE is set the Core is interrupted after the last slot in a frame ended regardless of the receive mask register setting. When RLIE is cleared the receive last slot interrupt is disabled. Hardware and software reset clear RLIE. RLIE is disabled when RDC[4:0]=00000 (on-demand mode). The use of the receive last slot interrupt is described in Section 22.4.2, “ESAI Interrupt Requests”
22 RIE	RCR Receive Interrupt Enable. The Core is interrupted when RIE and the RDF flag in the SAISR status register are set. When RIE is cleared, this interrupt is disabled. Reading the receive data registers of the enabled receivers clears RDF, thus clearing the interrupt. Receive interrupts with exception have higher priority than normal receive data interrupts, therefore if exception occurs (ROE is set) and REIE is set, the ESAI requests an ESAI receive data with exception interrupt from the interrupt controller.
21 REDIE	RCR Receive Even Slot Data Interrupt Enable. The REDIE control bit is used to enable the receive even slot data interrupts. If REDIE is set, the receive even slot data interrupts are enabled. If REDIE is cleared, the receive even slot data interrupts are disabled. A receive even slot data interrupt request is generated if REDIE is set and the REDF status flag in the SAISR status register is set. Even time slots are all even-numbered time slots (0, 2, 4, etc.) when operating in network mode. The zero time slot is marked by the frame sync signal and is considered to be even. Reading all the data registers of the enabled receivers clears the REDF flag, thus servicing the interrupt. Receive interrupts with exception have higher priority than receive even slot data interrupts, therefore if exception occurs (ROE is set) and REIE is set, the ESAI requests an ESAI receive data with exception interrupt from the interrupt controller.
20 REIE	RCR Receive Exception Interrupt Enable. When REIE is set, the Core is interrupted when both RDF and ROE in the SAISR status register are set. When REIE is cleared, this interrupt is disabled. Reading the SAISR status register followed by reading the enabled receivers data registers clears ROE, thus clearing the pending interrupt.
19 RPR	RCR Receiver Section Personal Reset. The RPR control bit is used to put the receiver section of the ESAI in the personal reset state. The transmitter section is not affected. When RPR is cleared, the receiver section may operate normally. When RPR is set, the receiver section enters the personal reset state immediately. When in the personal reset state, the status bits are reset to the same state as after hardware reset.The control bits are not affected by the personal reset state.The receiver data pins are disconnected while in the personal reset state. Note that to leave the personal reset state by clearing RPR, the procedure described in Section 22.5.2, “ESAI Initialization Examples” should be followed.

Table 22-24. ESAI Receive Control Register Field Descriptions (Continued)

Field	Description
18-17	Reserved.
16 RFSR	RCR Receiver Frame Sync Relative Timing. RFSR determines the relative timing of the receive frame sync signal as referred to the serial data lines, for a word length frame sync only. When RFSR is cleared the word length frame sync occurs together with the first bit of the data word of the first slot. When RFSR is set the word length frame sync starts one serial clock cycle earlier, that is, together with the last bit of the previous data word.
15 RFSL	RCR Receiver Frame Sync Length. The RFSL bit selects the length of the receive frame sync to be generated or recognized. If RFSL is cleared, a word-length frame sync is selected. If RFSL is set, a 1-bit clock period frame sync is selected. See Figure 22-21 for examples of frame length selection.
14-10 RSWS [4:0]	RCR Receiver Slot and Word Select. The RSWS4–RSWS0 bits are used to select the length of the slot and the length of the data words being received using the ESAI. The word length must be equal to or shorter than the slot length. The possible combinations are shown in Table 22-26. See also the ESAI data path programming model in Figure 22-12 and Figure 22-13.
9-8 RMOD [1:0]	RCR Receiver Network Mode Control. The RMOD1 and RMOD0 bits are used to define the network mode of the ESAI receivers, as shown in Table 22-25. In the normal mode, the frame rate divider determines the word transfer rate – one word is transferred per frame sync during the frame sync time slot, as shown in Figure 22-20. In network mode, it is possible to transfer a word for every time slot, as shown in Figure 22-20. For more details, see Section 22.1.2, “Modes of Operation.” In order to comply with AC–97 specifications, RSWS4–RSWS0 should be set to 00011 (20-bit slot, 20-bit word); RFSL and RFSR should be cleared, and RDC4–RDC0 should be set to $0C (13 words in frame).
7 RWA	RCR Receiver Word Alignment Control. The Receiver Word Alignment Control (RWA) bit defines the alignment of the data word in relation to the slot. This is relevant for the cases where the word length is shorter than the slot length. If RWA is cleared, the data word is assumed to be left-aligned in the slot frame. If RWA is set, the data word is assumed to be right-aligned in the slot frame. If the data word is shorter than the slot length, the data bits which are not in the data word field are ignored. For data word lengths of less than 24 bits, the data word is right-extended with zeroes before being stored in the receive data registers.
6 RSHFD	RCR Receiver Shift Direction. The RSHFD bit causes the receiver shift registers to shift data in MSB first when RSHFD is cleared or LSB first when RSHFD is set (see Figure 22-12 and Figure 22-13).
5-4	Reserved.
3 RE3	RCR ESAI Receiver 3 Enable. When RE3 is set and TE2 is cleared, the ESAI receiver 3 is enabled and samples data at the SDO2/SDI3 pin. TX2 and RX3 should not be enabled at the same time (RE3=1 and TE2=1). When RE3 is cleared, receiver 3 is disabled by inhibiting data transfer into RX3. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX3 data register. If RE3 is set while some of the other receivers are already in operation, the first data word received in RX3 will be invalid and must be discarded.
2 RE2	RCR ESAI Receiver 2 Enable. When RE2 is set and TE3 is cleared, the ESAI receiver 2 is enabled and samples data at the SDO3/SDI2 pin. TX3 and RX2 should not be enabled at the same time (RE2=1 and TE3=1). When RE2 is cleared, receiver 2 is disabled by inhibiting data transfer into RX2. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX2 data register. If RE2 is set while some of the other receivers are already in operation, the first data word received in RX2 will be invalid and must be discarded.

Table 22-24. ESAI Receive Control Register Field Descriptions (Continued)

Field	Description
1 RE1	RCR ESAI Receiver 1 Enable. When RE1 is set and TE4 is cleared, the ESAI receiver 1 is enabled and samples data at the SDO4/SDI1 pin. TX4 and RX1 should not be enabled at the same time (RE1=1 and TE4=1). When RE1 is cleared, receiver 1 is disabled by inhibiting data transfer into RX1. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX1 data register. If RE1 is set while some of the other receivers are already in operation, the first data word received in RX1 will be invalid and must be discarded.
0 RE0	RCR ESAI Receiver 0 Enable. When RE0 is set and TE5 is cleared, the ESAI receiver 0 is enabled and samples data at the SDO5/SDI0 pin. TX5 and RX0 should not be enabled at the same time (RE0=1 and TE5=1). When RE0 is cleared, receiver 0 is disabled by inhibiting data transfer into RX0. If this bit is cleared while receiving a data word, the remainder of the word is shifted in and transferred to the RX0 data register. If RE0 is set while some of the other receivers are already in operation, the first data word received in RX0 will be invalid and must be discarded.

Table 22-25. ESAI Receive Network Mode Selection

RMOD1	RMOD0	RDC4–RDC0	Receiver Network Mode
0	0	$0–$1F	Normal Mode
0	1	$0	On-Demand Mode
0	1	$1–$1F	Network Mode
1	0	X	Reserved
1	1	$0C	AC97

Table 22-26. ESAI Receive Slot and Word Length Selection

RSWS4	RSWS3	RSWS2	RSWS1	RSWS0	SLOT LENGTH	WORD LENGTH
0	0	0	0	0	8	8
0	0	1	0	0	12	8
0	0	0	0	1		12
0	1	0	0	0	16	8
0	0	1	0	1		12
0	0	0	1	0		16
0	1	1	0	0	20	8
0	1	0	0	1		12
0	0	1	1	0		16
0	0	0	1	1		20

Table 22-26. ESAI Receive Slot and Word Length Selection (Continued)

RSWS4	RSWS3	RSWS2	RSWS1	RSWS0	SLOT LENGTH	WORD LENGTH
1	0	0	0	0	24	8
0	1	1	0	1		12
0	1	0	1	0		16
0	0	1	1	1		20
1	1	1	1	0		24
1	1	0	0	0	32	8
1	0	1	0	1		12
1	0	0	1	0		16
0	1	1	1	1		20
1	1	1	1	1		24
0	1	0	1	1	Reserved	
0	1	1	1	0		
1	0	0	0	1		
1	0	0	1	1		
1	0	1	0	0		
1	0	1	1	0		
1	0	1	1	1		
1	1	0	0	1		
1	1	0	1	0		
1	1	0	1	1		
1	1	1	0	0		
1	1	1	0	1		

22.3.3.19 ESAI Receiver Clock Control Register (RCCR)

The read/write Receiver Clock Control Register (RCCR) controls the ESAI receiver clock generator bit and frame sync rates, word length, and number of words per frame for the serial data. The RCCR control bits are described in the following paragraphs.

Offset 0x40E0 (RCCR) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	RHCKD	RFSD	RCKD	RHCKP	RFSP	RCKP	RFP[3:2]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RFP[1:0]		RDC[4:0]					RPSR	RPM[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-26. ESAI Receiver Clock Control Register

Table 22-27. ESAI Receiver Clock Control Register Field Descriptions

Field	Description
31-24	Reserved.
23 RHCKD	RCCR Receiver High Frequency Clock Direction. The Receiver High Frequency Clock Direction (RHCKD) bit selects the source of the receiver high frequency clock when in the asynchronous mode (SYN=0) and the IF2/OF2 flag direction in the synchronous mode (SYN=1). In the asynchronous mode, when RHCKD is set, the internal clock generator becomes the source of the receiver high frequency clock and is the output on the HCKR pin. In the asynchronous mode, when RHCKD is cleared, the receiver high frequency clock source is external; the internal clock generator is disconnected from the HCKR pin, and an external clock source may drive this pin. When RHCKD is cleared, HCKR is an input; when RHCKD is set, HCKR is an output. In the synchronous mode when RHCKD is set, the HCKR pin becomes the OF2 output flag. If RHCKD is cleared, the HCKR pin becomes the IF2 input flag. See Table 22-1 and Table 22-31.
22 RFSD	RCCR Receiver Frame Sync Signal Direction. The Receiver Frame Sync Signal Direction (RFSD) bit selects the source of the receiver frame sync signal when in the asynchronous mode (SYN=0) and the IF1/OF1/Transmitter Buffer Enable flag direction in the synchronous mode (SYN=1). In the asynchronous mode, when RFSD is set, the internal clock generator becomes the source of the receiver frame sync and is the output on the FSR pin. In the asynchronous mode, when RFSD is cleared, the receiver frame sync source is external; the internal clock generator is disconnected from the FSR pin, and an external clock source may drive this pin. In the synchronous mode when RFSD is set, the FSR pin becomes the OF1 output flag or the Transmitter Buffer Enable, according to the TEBE control bit. If RFSD is cleared, the FSR pin becomes the IF1 input flag. See Table 22-1 and Table 22-30.
21 RCKD	RCCR Receiver Clock Source Direction. The Receiver Clock Source Direction (RCKD) bit selects the source of the clock signal used to clock the receive shift register in the asynchronous mode (SYN=0) and the IF0/OF0 flag direction in the synchronous mode (SYN=1). In the asynchronous mode, when RCKD is set, the internal clock source becomes the bit clock for the receive shift registers and word length divider and is the output on the SCKR pin. In the asynchronous mode when RCKD is cleared, the clock source is external; the internal clock generator is disconnected from the SCKR pin, and an external clock source may drive this pin. In the synchronous mode when RCKD is set, the SCKR pin becomes the OF0 output flag. If RCKD is cleared, the SCKR pin becomes the IF0 input flag. See Table 22-1 and Table 22-29.

Table 22-27. ESAI Receiver Clock Control Register Field Descriptions (Continued)

Field	Description
20 RHCKP	RCCR Receiver High Frequency Clock Polarity. The Receiver High Frequency Clock Polarity (RHCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If RHCKP is cleared the data and the frame sync are clocked out on the rising edge of the receive high frequency bit clock and the frame sync is latched in on the falling edge of the receive bit clock. If RHCKP is set the falling edge of the receive clock is used to clock the data and frame sync out and the rising edge of the receive clock is used to latch the frame sync in.
19 RFSP	RCCR Receiver Frame Sync Polarity. The Receiver Frame Sync Polarity (RFSP) determines the polarity of the receive frame sync signal. When RFSP is cleared the frame sync signal polarity is positive, that is, the frame start is indicated by a high level on the frame sync pin. When RFSP is set the frame sync signal polarity is negative, that is, the frame start is indicated by a low level on the frame sync pin.
18 RCKP	The Receiver Clock Polarity (RCKP) bit controls on which bit clock edge data and frame sync are clocked out and latched in. If RCKP is cleared the data and the frame sync are clocked out on the rising edge of the receive bit clock and the frame sync is latched in on the falling edge of the receive bit clock. If RCKP is set the falling edge of the receive clock is used to clock the data and frame sync out and the rising edge of the receive clock is used to latch the frame sync in.
17-14 RFP 34:0]	RCCR Rx High Frequency Clock Divider. The RFP3–RFP0 bits control the divide ratio of the receiver high frequency clock to the receiver serial bit clock when the source of the receiver high frequency clock and the bit clock is the internal Core clock. When the HCKR input is being driven from an external high frequency clock, the RFP3–RFP0 bits specify an additional division ration in the clock divider chain. Table 22-28 provides the specification of the divide ratio. Figure 22-23 shows the ESAI high frequency generator functional diagram.
13-9 RDC [4:0]	RCCR Rx Frame Rate Divider Control. The RDC4–RDC0 bits control the divide ratio for the programmable frame rate dividers used to generate the receiver frame clocks. In network mode, this ratio may be interpreted as the number of words per frame minus one. The divide ratio may range from 2 to 32 (RDC[4:0]=00001 to 11111) for network mode. A divide ratio of one (RDC[4:0]=00000) in network mode is a special case (on-demand mode). In normal mode, this ratio determines the word transfer rate. The divide ratio may range from 1 to 32 (RDC[4:0]=00000 to 11111) for normal mode. In normal mode, a divide ratio of one (RDC[4:0]=00000) provides continuous periodic data word transfers. A bit-length frame sync (RFSL=1) must be used in this case. The ESAI frame sync generator functional diagram is shown in Figure 22-24.
8 RPSP	RCCR Receiver Prescaler Range. The RPSR controls a fixed divide-by-eight prescaler in series with the variable prescaler. This bit is used to extend the range of the prescaler for those cases where a slower bit clock is desired. When RPSR is set, the fixed prescaler is bypassed. When RPSR is cleared, the fixed divide-by-eight prescaler is operational (see Figure 22-23). The maximum internally generated bit clock frequency is Fsys/4, the minimum internally generated bit clock frequency is Fsys/(2 x 8 x 256)=Fsys/4096. (Do not use the combination RPSR=1 and RPM7-RPM0=$00, which causes synchronization problems when using the internal Core clock as source (RHCKD=1 or RCKD=1))
7-0 RPM [7:0]	RCCR Receiver Prescale Modulus Select. The RPM7–RPM0 bits specify the divide ratio of the prescale divider in the ESAI receiver clock generator. A divide ratio from 1 to 256 (RPM[7:0]=$00 to $FF) may be selected. The bit clock output is available at the receiver serial bit clock (SCKR) pin. The bit clock output is also available internally for use as the bit clock to shift the receive shift registers. The ESAI receive clock generator functional diagram is shown in Figure 22-23.

Table 22-28. Receiver High Frequency Clock Divider

RFP3–RFP0	Divide Ratio
$0	1
$1	2
$2	3
$3	4
...	...
$F	16

Table 22-29. SCKR Pin Definition Table

Control Bits		SCKR PIN
SYN	RCKD	
0	0	SCKR input
0	1	SCKR output
1	0	IF0
1	1	OF0

Table 22-30. FSR Pin Definition Table

Control Bits			FSR Pin
SYN	TEBE	RFSD	
0	X	0	FSR input
0	X	1	FSR output
1	0	0	IF1
1	0	1	OF1
1	1	0	reserved
1	1	1	Transmitter Buffer Enable

Table 22-31. HCKR Pin Definition Table

Control Bits		HCKR PIN
SYN	RHCKD	
0	0	HCKR input
0	1	HCKR output
1	0	IF2
1	1	OF2

22.3.3.20 ESAI Transmit Slot Mask Register A (TSMA)

The Transmit Slot Mask Register A together with Transmit Slot Mask Register B (TSMA and TSMB) are two read/write registers used by the transmitters in network mode to determine for each slot whether to transmit a data word and generate a transmitter empty condition (TDE=1), or to tri-state the transmitter data pins. TSMA and TSMB should each be considered as containing half a 32-bit register TSM. Bit number N in TSM (TS**) is the enable/disable control bit for transmission in slot number N.

Offset 0x40E4 (TSMA) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 22-27. ESAI Transmit Slot Mask Register A

Table 22-32. ESAI Transmit Slot Mask Register A Field Descriptions

Field	Description
31–16	Reserved
15-0 TS [15:0]	When bit number N in TSMA is cleared, all the transmit data pins of the enabled transmitters are tri-stated during transmit time slot number N. The data is still transferred from the transmit data registers to the transmit shift registers but neither the TDE nor the TUE flags are set. This means that during a disabled slot, no transmitter empty interrupt is generated. The Core is interrupted only for enabled slots. Data that is written to the transmit data registers when servicing this request is transmitted in the next enabled transmit time slot. When bit number N in TSMA register is set, the transmit sequence is as usual: data is transferred from the TX registers to the shift registers and transmitted during slot number N, and the TDE flag is set. Using the slot mask in TSMA does not conflict with using TSR. Even if a slot is enabled in TSMA, the user may chose to write to TSR instead of writing to the transmit data registers TXx. This causes all the transmit data pins of the enabled transmitters to be tri-stated during the next slot. Data written to the TSMA affects the next frame transmission. The frame being transmitted is not affected by this data and would comply to the last TSMA setting. Data read from TSMA returns the last written data. After hardware or software reset, the TSMA register is preset to $0000FFFF, which means that all 16 possible slots are enabled for data transmission. When operating in normal mode, bit 0 of the TSMA register must be set, otherwise no output is generated.

22.3.3.21 ESAI Transmit Slot Mask Register B (TSMB)

The Transmit Slot Mask Register B together with Transmit Slot Mask Register A (TSMA and TSMB) are two read/write registers used by the transmitters in network mode to determine for each slot whether to transmit a data word and generate a transmitter empty condition (TDE=1), or to tri-state the transmitter data pins. TSMA and TSMB should each be considered as containing half a 32-bit register TSM. Bit number N in TSM (TS**) is the enable/disable control bit for transmission in slot number N.

Offset 0x40E8 (TSMB) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS[31:16]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 22-28. ESAI Transmit Slot Mask Register B

Table 22-33. ESAI Transmit Slot Mask Register B Field Descriptions

Field	Description
31–16	Reserved
15-0 TS [31:16]	When bit number N in TSMB is cleared, all the transmit data pins of the enabled transmitters are tri-stated during transmit time slot number N. The data is still transferred from the transmit data registers to the transmit shift registers but neither the TDE nor the TUE flags are set. This means that during a disabled slot, no transmitter empty interrupt is generated. The Core is interrupted only for enabled slots. Data that is written to the transmit data registers when servicing this request is transmitted in the next enabled transmit time slot. When bit number N in TSMB register is set, the transmit sequence is as usual: data is transferred from the TX registers to the shift registers and transmitted during slot number N, and the TDE flag is set. Using the slot mask in TSMB does not conflict with using TSR. Even if a slot is enabled in TSMB, the user may chose to write to TSR instead of writing to the transmit data registers TXx. This causes all the transmit data pins of the enabled transmitters to be tri-stated during the next slot. Data written to the TSMB affects the next frame transmission. The frame being transmitted is not affected by this data and would comply to the last TSMB setting. Data read from TSMB returns the last written data. After hardware or software reset, the TSMB register is preset to $0000FFFF, which means that all 16 possible slots are enabled for data transmission.

22.3.3.22 ESAI Receive Slot Mask Register A (RSMA)

The Receive Slot Mask Register A together with Receive Slot Mask Register B (RSMA and RSMB) are two read/write registers used by the receiver in network mode to determine for each slot whether to receive a data word and generate a receiver full condition (RDF=1), or to ignore the received data. RSMA and RSMB should be considered as each containing half of a 32-bit register RSM. See Bit number N in RSM (RS**) is an enable/disable control bit for receiving data in slot number N.

Offset 0x40EC (RSMA) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RS[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 22-29. ESAI Receive Slot Mask Register A

Table 22-34. ESAI Receive Slot Mask Register A Field Descriptions

Field	Description
31–16	Reserved
15-0 RS [15:0]	When bit number N in the RSMA register is cleared, the data from the enabled receivers input pins are shifted into their receive shift registers during slot number N. The data is not transferred from the receive shift registers to the receive data registers, and neither the RDF nor the ROE flag is set. This means that during a disabled slot, no receiver full interrupt is generated. The Core is interrupted only for enabled slots. When bit number N in the RSMA is set, the receive sequence is as usual: data which is shifted into the enabled receivers shift registers is transferred to the receive data registers and the RDF flag is set. Data written to the RSMA affects the next received frame. The frame being received is not affected by this data and would comply to the last RSMA setting. Data read from RSMA returns the last written data. After hardware or software reset, the RSMA register is preset to $0000FFFF, which means that all 16 possible slots are enabled for data reception. When operating in normal mode, bit 0 of the RSMA register must be set to one, otherwise no input is received.

22.3.3.23 ESAI Receive Slot Mask Register B (RSMB)

The Receive Slot Mask Register B together with Receive Slot Mask Register A (RSMA and RSMB) are two read/write registers used by the receiver in network mode to determine for each slot whether to receive a data word and generate a receiver full condition (RDF=1), or to ignore the received data. RSMA and RSMB should be considered as each containing half of a 32-bit register RSM. See Bit number N in RSM (RS**) is an enable/disable control bit for receiving data in slot number N.

Offset 0x40F0 (RSMB) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RS[31:16]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 22-30. ESAI Receive Slot Mask Register B

Table 22-35. ESAI Receive Slot Mask Register B Field Descriptions

Field	Description
31–16	Reserved
15-0 RS [31:16]	When bit number N in the RSMB register is cleared, the data from the enabled receivers input pins are shifted into their receive shift registers during slot number N. The data is not transferred from the receive shift registers to the receive data registers, and neither the RDF nor the ROE flag is set. This means that during a disabled slot, no receiver full interrupt is generated. The Core is interrupted only for enabled slots. When bit number N in the RSMB is set, the receive sequence is as usual: data which is shifted into the enabled receivers shift registers is transferred to the receive data registers and the RDF flag is set. Data written to the RSMB affects the next received frame. The frame being received is not affected by this data and would comply to the last RSMB setting. Data read from RSMB returns the last written data. After hardware or software reset, the RSMB register is preset to $0000FFFF, which means that all 16 possible slots are enabled for data reception.

22.3.3.24 ESAI Personal Reset Registers

There are two registers to control the ESAI personal reset status - Port C Direction Register (PRRC) and Port C Control Register (PCRC). Their register bits are described in the following paragraphs.

22.3.3.25 Port C Direction Register (PRRC)

The read/write 32-bit Port C Direction Register (PRRC) in conjunction with the Port C Control Register (PCRC) controls the functionality of the ESAI personal reset state. Table 22-38 provides the port pin configurations. Hardware and software reset clear all PRRC bits.

Offset 0x40F8 (PRRC) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PDC[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-31. Port C Direction Register

Table 22-36. Port C Direction Register Field Descriptions

Field	Description
31–12	Reserved
11-0 PDC [11:0]	See Table 22-38.

22.3.3.26 Port C Control Register (PCRC)

The read/write 32-bit Port C Control Register (PCRC) in conjunction with the Port C Direction Register (PRRC) controls the functionality of the ESAI personal reset state. Each of the PC(11:0) bits controls the functionality of the corresponding port pin. Table 22-38 provides the port pin configurations. Hardware and software reset clear all PCRC bits.

Offset 0x40FC (PCRC) Access: User read/write only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PC[11:0]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-32. Port C Control Register

Table 22-37. Port C Control Register Field Descriptions

Field	Description
31–12	Reserved
11-0 PC [11:0]	See Table 22-38.

Table 22-38. PCRC and PRRC Bits Functionality

PDC[i]	PC[i]	Port Pin[i] Function
0	0	Disconnected
1	1	ESAI

22.4 Functional Description

22.4.1 ESAI After Reset

Hardware or software reset clears the port control register bits and the port direction control register bits, which configure all ESAI I/O pins as disconnected and both ESAI FIFOs are also in reset state. The ESAI is in personal reset state while all ESAI pins are programmed as disconnected, and it is active only if at least one of the ESAI I/O pins is programmed as an ESAI pin.

22.4.2 ESAI Interrupt Requests

The ESAI can generate eight different interrupt requests (ordered from the highest to the lowest priority):

1. **ESAI Receive Data with Exception Status**:
   Occurs when the receive exception interrupt is enabled (REIE=1 in the RCR register), at least one of the enabled receive data registers is full (RDF=1) and a receiver overrun error has occurred (ROE=1 in the SAISR register). ROE is cleared by first reading the SAISR and then reading all the enabled receive data registers.
2. **ESAI Receive Even Data**:
   Occurs when the receive even slot data interrupt is enabled (REDIE=1), at least one of the enabled receive data registers is full (RDF=1), the data is from an even slot (REDF=1) and no exception has occurred (ROE=0 or REIE=0).
   Reading all enabled receiver data registers clears RDF and REDF.
3. **ESAI Receive Data**:
   Occurs when the receive interrupt is enabled (RIE=1), at least one of the enabled receive data registers is full (RDF=1), no exception has occurred (ROE=0 or REIE=0) and no even slot interrupt has occurred (REDF=0 or REDIE=0).
   Reading all enabled receiver data registers clears RDF.
4. **ESAI Receive Last Slot Interrupt**:
   Occurs, if enabled (RLIE=1), after the last slot of the frame ended (in network mode only) regardless of the receive mask register setting. The receive last slot interrupt may be used for resetting the receive mask slot register, reconfiguring the DMA channels and reassigning data memory pointers. Using the receive last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame is serviced with the new setting without synchronization problems. Note that the maximum receive last slot interrupt service time should not exceed N-1 ESAI bits service time (where N is the number of bits in a slot).
5. **ESAI Transmit Data with Exception Status**:
   Occurs when the transmit exception interrupt is enabled (TEIE=1), at least one transmit data register of the enabled transmitters is empty (TDE=1) and a transmitter underrun error has occurred (TUE=1). TUE is cleared by first reading the SAISR and then writing to all the enabled transmit data registers, or to the TSR register.
6. **ESAI Transmit Last Slot Interrupt**:
   Occurs, if enabled (TLIE=1), at the start of the last slot of the frame in network mode regardless of the transmit mask register setting. The transmit last slot interrupt may be used for resetting the transmit mask slot register, reconfiguring the DMA channels and reassigning data memory pointers. Using the transmit last slot interrupt guarantees that the previous frame was serviced with the previous setting and the new frame is serviced with the new setting without synchronization problems. Note that the maximum transmit last slot interrupt service time should not exceed N-1 ESAI bits service time (where N is the number of bits in a slot).
7. **ESAI Transmit Even Data**:
   Occurs when the transmit even slot data interrupt is enabled (TEDIE=1), at least one of the enabled transmit data registers is empty (TDE=1), the slot is an even slot (TEDE=1) and no exception has occurred (TUE=0 or TEIE=0).
   Writing to all the TX registers of the enabled transmitters or to TSR clears this interrupt request.

8. **ESAI Transmit Data**:
   Occurs when the transmit interrupt is enabled (TIE=1), at least one of the enabled transmit data registers is empty (TDE=1), no exception has occurred (TUE=0 or TEIE=0) and no even slot interrupt has occurred (TEDE=0 or TEDIE=0).
   Writing to all the TX registers of the enabled transmitters, or to the TSR clears this interrupt request.

22.4.3 ESAI DMA Requests from the FIFOs

The ESAI can generate two different DMA requests:

1. ESAI Transmit FIFO Empty
   Asserts when the number of empty slots in the ESAI transmit FIFO exceeds the threshold programmed in the ESAI Transmit FIFO Configuration Register (TFCR). Automatically negates when the number of empty slots is less than the threshold programmed in the ESAI Transmit FIFO Configuration Register.
2. ESAI Receive FIFO Full
   Asserts when the number of data words in the ESAI receive FIFO exceeds the threshold programmed in the ESAI Receive FIFO Configuration Register (RFCR). Automatically negates when the number of words is less than the threshold programmed in the ESAI Receive FIFO Configuration Register.

22.5 Initialization Information

22.5.1 ESAI Initialization

The correct way to initialize the ESAI is as follows:

1. Enable the ESAI logic clock by asserting bit 0 of ESAI Control Register(ECR[0]).
2. Hardware, software, ESAI individual reset. (Asserting bit 1 of ESAI Control Register only resets the ESAI core logic (including configuration registers), but not the ESAI FIFOs)
3. Reset ESAI FIFOs by asserting bit 1 of TFCR and RFCR.
4. Program ESAI control and time slot registers.(The transmit/receive enable bits of TCR/RCR should not be set)
5. Program ESAI FIFOs using TFCR and RFCR. (enable Transmit/Receive FIFO, enable transmitters/receivers, transmit initialization and set Transmit FIFO/Receive FIFO watermark)
6. Write initial words to ESAI Transmit Data Register (ETDR) (at least one word per enabled transmitter but as many as you want) if Transmit FIFO is used. If not, just write initial words to all the enabled transmitters.
7. Remove ESAI personal reset by configuring PCRC and PRRC.
8. Enabled Transmitters/Receivers in TCR/RCR.

During program execution, all ESAI pins may be defined disconnected, causing the ESAI to stop serial activity and enter the individual reset state. All status bits of the interface are set to their reset state; however, the control bits are not affected. This procedure allows the programmer to reset the ESAI

separately from the other internal peripherals. During individual reset, internal DMA accesses to the data registers of the ESAI are not valid and data read is undefined.

The programmer must use an individual ESAI reset when changing the ESAI control registers (except for TEIE, REIE, TLIE, RLIE, TIE, RIE, TE0-TE5, RE0-RE3) to ensure proper operation of the interface.

NOTE

If the ESAI receiver section is already operating with some of the receivers, enabling additional receivers on the fly, that is, without first putting the ESAI receiver in the personal reset state, by setting their REx control bits will result in erroneous data being received as the first data word for the newly enabled receivers.

22.5.2 ESAI Initialization Examples

22.5.2.1 Initializing the ESAI Using Personal Reset

1. Enable the ESAI logic clock by setting bit 0 of ESAI Control Register(ECR[0]).
2. The ESAI should be in its personal reset state (PCRC = $000 and PRRC = $000). In the personal reset state, both the transmitter and receiver sections of the ESAI are simultaneously reset. The TPR bit in the TCR register may be used to reset just the transmitter section. The RPR bit in the RCR register may be used to reset just the receiver section.
3. Configure the control registers (TCCR, TCR, RCCR, RCR) and ESAI FIFOs configuration Registers (TFCR, RFCR) according to the operating mode, but do not enable transmitters (TE5–TE0 = $0) or receivers (RE3–RE0 = $0). It is possible to set the interrupt enable bits which are in use during the operation (no interrupt occurs).
4. Enable the ESAI by setting the PCRC and PRRC register bits according to pins which are in use during operation.
5. Write initial words to ESAI Transmit Data Register (ETDR) (at least one word per enabled transmitter but as many as you want) if Transmit FIFO is used. If not, just write initial words to all the enabled transmitters which are in use during operation.
   This step is needed even if DMA is used to service the transmitters.
6. Enable the transmitters and receivers.
7. From now on ESAI can be serviced either by polling, interrupts, or DMA.

Perform the operation as follows:

- For internally generated clock and frame sync, these signals are active immediately after ESAI is enabled (step 4 above).
- Data is received only when one of the receive enable (REx) bits is set and after the occurrence of frame sync signal (either internally or externally generated).
- Data is transmitted only when the transmitter enable (TEx) bit is set and after the occurrence of frame sync signal (either internally or externally generated). The transmitter outputs remain tri-stated after TEx bit is set until the frame sync occurs.

22.5.2.2 Initializing Just the ESAI Transmitter Section

- It is assumed that the ESAI is operational; that is, at least one pin is defined as an ESAI pin.
- Enable the ESAI logic clock by setting bit 0 of ESAI Control Register(ECR[0])
- The transmitter section should be in its individual reset state (TPR = 1) and also reset the ESAI Transmit FIFO (TFCR[1] = 1).
- Configure the control registers TCCR and TCR according to the operating mode, configure the Transmit FIFO Configuration Register (bring transmit FIFO out of reset, enable Transmit FIFO, enable transmitters, transmit initialization and set watermark). Make sure to clear the transmitter enable bits (TE0–TE5). TPR must remain set.
- Take the transmitter section out of the individual reset state by clearing TPR.
- Write initial words to ESAI Transmit Data Register (ETDR) (at least one word per enabled transmitter but as many as you want) if Transmit FIFO is used. If not, just write first words to all the enabled transmitters which are in use during operation.
  This step is needed even if DMA is used to service the transmitters.
- Enable the transmitters by setting their TE bits.
- Data is transmitted only when the transmitter enable (TEx) bit is set and after the occurrence of frame sync signal (either internally or externally generated). The transmitter outputs remain tri-stated after TEx bit is set until the frame sync occurs.
- From now on the transmitters are operating and can be serviced either by polling, interrupts, or DMA.

22.5.2.3 Initializing Just the ESAI Receiver Section

- It is assumed that the ESAI is operational; that is, at least one pin is defined as an ESAI pin.
- Enable the ESAI logic clock by setting bit 0 of ESAI Control Register(ECR[0])
- The receiver section should be in its individual reset state (RPR = 1) and also reset the ESAI Receive FIFO (RFCR[1] = 1).
- Configure the control registers RCCR and RCR according to the operating mode, configure the Receive FIFO Configuration Register (bring receive FIFO out of reset, enable Receive FIFO, receivers, and set watermark). Making sure to clear the receiver enable bits (RE0–RE3). RPR must remain set.
- Take the receiver section out of the individual reset state by clearing RPR.
- Enable the receivers by setting their RE bits.
- From now on the receivers are operating and can be serviced either by polling, interrupts, or DMA.

Chapter 23
Enhanced Secured Digital Host Controller (eSDHC)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

23.1 Overview

The enhanced Secured Digital Host Controller (eSDHC) provides the interface between the host system and MMC/SD/SDIO/CE-ATA cards, including cards with reduced size or mini cards.

Figure 23-1 shows the eSDHC and its connections within the device. The eSDHC acts as a bridge, passing host bus transactions to the MMC/SD/SDIO/CE-ATA cards by sending commands and performing data accesses to and from the cards. It handles the MMC/SD/SDIO/CE-ATA protocols at the transmission level.

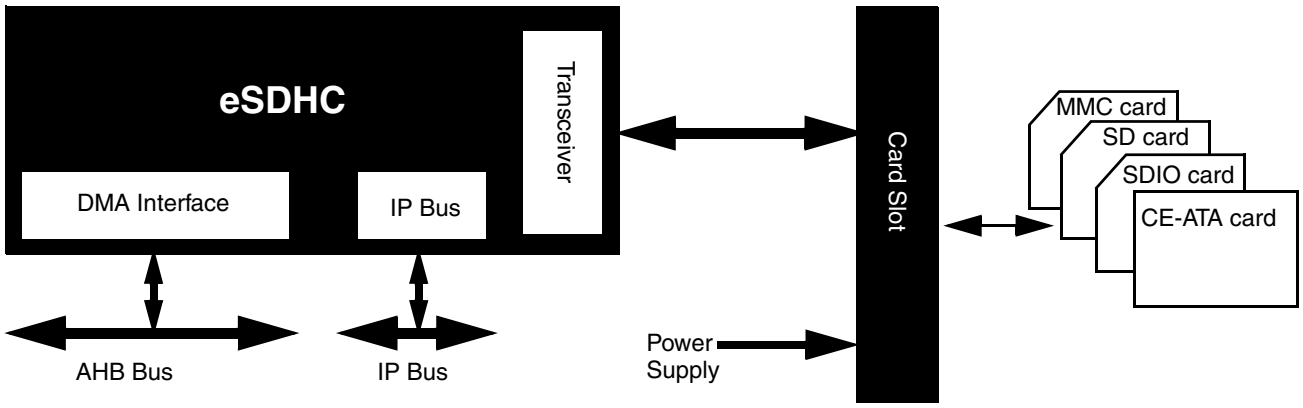


Figure 23-1. eSDHC System Connection

The different cards supported by the eSDHC are described as follows:

- The MultiMediaCard (MMC) is a universal low-cost data storage and communication media that is designed to cover a wide area of applications including mobile video and gaming both on pre-loaded MMC cards and downloadable from cellular phone, WLAN or other wireless networks. Old MMC cards are based on 7-pin serial bus with a single data pin, while the newer high-speed MMC communication is based on an advanced 11-pin serial bus designed to operate at lower voltage.
- The Secure Digital (SD) card is an evolution of earlier MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment

and data transfer protocol are forward compatible with MMC, with some additions. Under the SD protocol, an SD card can be categorized as memory card, I/O card, or combo card (having both memory and I/O functions). The memory card invokes a copyright protection mechanism that is compatible with the SDMI security standard. The I/O card provides high-speed data I/O with low power consumption for mobile electronic devices. For the sake of simplicity, Figure 23-1 does not show cards with reduced size or mini cards.

- Consumer electronics ATA (CE-ATA) is a hard drive interface that is optimized for embedded applications. The device is layered on the top of the MMC protocol stack using the same physical interface. The interface electrical and signaling definition follows the MMC specification. For more details, see the CE-ATA Specification.

23.1.1 .Features

The features of the eSDHC module include the following:

- Designed to work with MMC, MMC plus, MMC RS, SD memory, miniSD memory, SDIO, miniSDIO, SD Combo, and CE-ATA cards. Compatible with the following specifications:
  - MMC System Specification Version 4.2
  - SD Host Controller Standard Specification Version 2.0, including test event register support
  - SD Memory Card Specification Version 2.0: supports High-Capacity SD Memory Cards
  - SDIO Card Specification Version 2.0
  - CE-ATA Card Specification Version 1.0
- Supports 1-, 4-, or 8-bit MMC modes, 1- or 4-bit SD and SDIO modes, and 1-, 4-, or 8-bit CE-ATA devices
  - Card bus clock frequency up to 52 MHz
  - Up to 416 Mbps of data transfer for MMC cards in 8-bit mode
  - Up to 200 Mbps of data transfer for SD/SDIO cards in 4-bit mode
  - Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes, also supports interrupt period
- Supports single-block and multi-block read and write
  - Block sizes of 1–4096 bytes
  - Supports pause during the data transfer at block gap
  - Supports Auto CMD12 for multi-block transfer
- Supports read/write features:
  - Write protection switch for write operations
  - SDIO read wait and suspend/resume operations
  - Includes a fully configurable 128x32-bit FIFO for read/write data
- Supports internal and external DMA capabilities
  -
- Supports both synchronous and asynchronous abort
- Host can initiate non-data transfer command while data transfer is in progress
- Support voltage selection by configuring vendor-specific register bit

23.1.2 Modes of Operation

23.1.2.1 Data Transfer Modes

The eSDHC can select the following modes for data transfer:

- MMC 1-, 4-, or 8-bit transfers in full-speed mode (up to 20 MHz) or high-speed mode (up to 52 MHz)
- SD 1- or 4-bit transfers in full-speed mode (up to 25 MHz) or high-speed mode (up to 50 MHz)
- CE-ATA 1-, 4-, or 8-bit transfers
- Identification Mode (up to 400 kHz)

23.2 External Signals

23.2.1 Overview

Figure 23-2 shows the eSDHC I/O signals:

- SD_CLK is an internally-generated clock used to drive the MMC, SD, SDIO, or CE-ATA cards.
- CMD I/O is used to send commands and receive responses to/from the card.
- Eight data lines (DAT7–DAT0) are used to perform data transfers between the eSDHC and the card.
- SD_CD# and SD_WP are card-detection and write-protection signals directly routed from the socket. A low on SD_CD# indicates that a card is inserted, and a high on SD_WP indicates that the write-protect switch is active.
- SD_OD is an output signal generated at the SoC level outside eSDHC, and is used to select the external open-drain resistor.
- SD_LCTL is an output signal used to drive an external LED to indicate that the SD interface is busy.
- SD_VS is used to control the voltage at the SD_ pins listed above. When asserted, voltage is high (around 3.0V); when negated, voltage is low (around 1.8V).

SD_CD#, SD_WP, SD_OD, SD_LCTL, and SD_VS are all optional for system implementation. If the eSDHC is desired to support a 4-bit data transfer, DAT7–DAT4 are optional and can be tied to high.

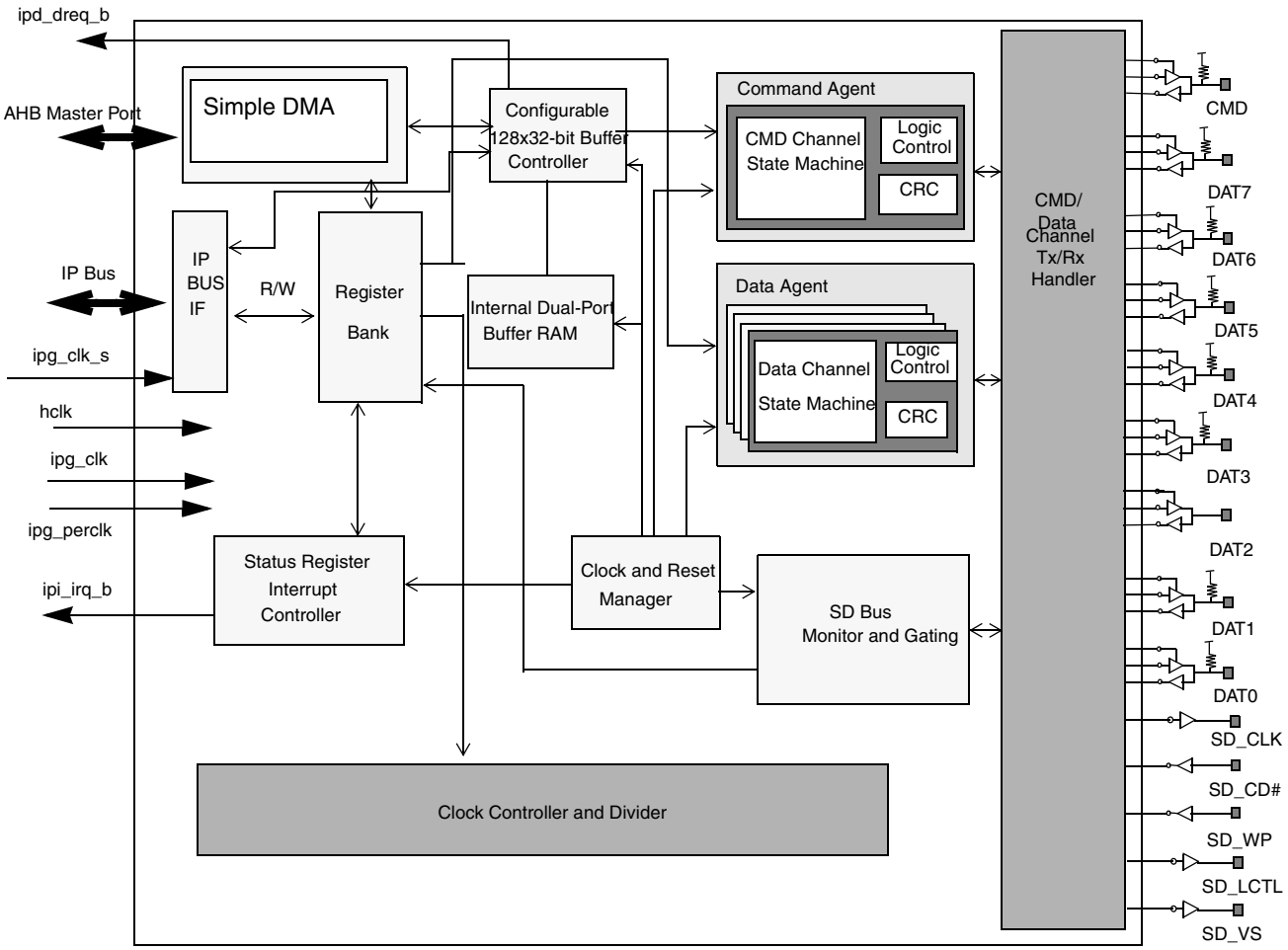


Figure 23-2. enhanced Secure Digital Host Controller (eSDHC) Block Diagram

23.2.2 Signal Descriptions

Table 23-1 shows properties of the I/O signals.

Table 23-1. Properties of I/O Signals

Name	Port	Function	Reset State	Pull up
SD_CLK	O	Clock for MMC/SD/SDIO card	0	N/A
SD_CMD	I/O	CMD line connected to card	1	Pull up
SD_DAT7	I/O	DAT7 line (MSB) in 8-bit mode Not used in other modes	1	Pull up
SD_DAT6	I/O	DAT6 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT5	I/O	DAT5 line in 8-bit mode Not used in other modes	1	Pull up

Table 23-1. Properties of I/O Signals (continued)

Name	Port	Function	Reset State	Pull up
SD_DAT4	I/O	DAT4 line in 8-bit mode Not used in other modes	1	Pull up
SD_DAT3	I/O	DAT3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	0	Pull-up/pull-down configurable
SD_DAT2	I/O	DAT2 line or read wait in 4-bit mode Read Wait in 1-bit mode	1	Pull up
SD_DAT1	I/O	DAT1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	1	Pull up
SD_DAT0	I/O	DAT0 (LSB) line in all modes Also used to detect busy state	1	Pull up
SD_CD#	I	Card detection signal If not used tie high	N/A	N/A
SD_WP	I	Card write protect detect If not used tie low	N/A	N/A
SD_OD	O	Open drain select (not generated within the eSDHC). Optional output	N/A	N/A
SD_LCTL	O	LED control used to drive an external LED Active high Fully controlled by the driver Optional output	0	N/A
SD_VS	O	Controls the voltage at SD_ pins. When asserted, voltage is high (around 3.0V); when negated, voltage is low (around 1.8V) Optional output	0	N/A

23.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers.

23.3.1 Memory Map

Table 23-2 shows the eSDHC memory map. For the base address of a particular module instantiation, see the system memory map.

All registers support 32-bit accesses only. Addresses greater than 0x0044, except for 0x0050, and 0x00FC, are reserved and read as all zeros. Writes to these registers are ignored.

Table 23-2. eSDHCv2 Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Section
0x0000 (DSADDR)	DMA system address register	R/W	23.3.3.1/23-11
0x0004 (BLKATTR)	Block attributes register	R/W	23.3.3.2/23-12
0x0008 (CMDARG)	Command argument register	R/W	23.3.3.3/23-13
0x000C (XFERTYP)	Transfer type register	R/W	23.3.3.4/23-13
0x0010 (CMDRSP0)	Command response 0 register	R	23.3.3.5/23-17
0x0014 (CMDRSP1)	Command response 1 register	R	23.3.3.5/23-17
0x0018 (CMDRSP2)	Command response 2 register	R	23.3.3.5/23-17
0x001C (CMDRSP3)	Command response 3 register	R	23.3.3.5/23-17
0x0020 (DATPORT)	Data buffer access port register	R/W	23.3.3.6/23-18
0x0024 (PRSSTAT)	Present state register	R	23.3.3.7/23-19
0x0028 (PROCTL)	Protocol control register	R/W	23.3.3.8/23-23
0x002C (SYSCTL)	System control register	R/W	23.3.3.9/23-26
0x0030 (IRQSTAT)	Interrupt status register	R	23.3.3.10/23-29
0x0034 (IRQSTATEN)	Interrupt status enable register	R/W	23.3.3.11/23-34
0x0038 (IRQSIGEN)	Interrupt signal enable register	R	23.3.3.12/23-36
0x003C (AUTOC12ERR)	Auto CMD12 status register	R	23.3.3.13/23-38
0x0040 (HOSTCAPBLT)	Host controller capabilities register	R	23.3.3.14/23-40
0x0044 (WML)	Watermark level register	R/W	23.3.3.15/23-41
0x0050 (FEVT)	Force event register	W	23.3.3.16/23-42
0x00FC (HOSTVER)	Host controller version register	R	23.3.3.17/23-44

23.3.2 Register Summary

Table 23-3 summarizes the eSDHC registers.

Table 23-3. eSDHC Register Summary

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (DSADDR)	R	DS_ADDR[31:16]															
	W																
	R	DS_ADDR[15:2]														0	0
	W																
0x0004 (BLKATTR)	R	BLKCNT[15:0]															
	W																
	R	0	0	0	BLKSIZE[12:0]												
	W																
0x0008 (CMDARG)	R	CMDARG[31:16]															
	W																
	R	CMDARG[15:0]															
	W																
0x000C (XFERTYP)	R	0	0	CMDINX[5:0]						CMDTYP [1:0]		DPS EL	CICE N	CCC EN	0	RSPTYP [1:0]	
	W																
	R	0	0	0	0	0	0	0	0	0	0	MSB SEL	DTD SEL	0	AC12 EN	BCE N	DMA EN
	W																
0x0010 (CMDRSP0)	R	CMDRSP0[31:16]															
	W																
	R	CMDRSP0[15:0]															
	W																
0x0014 (CMDRSP1)	R	CMDRSP1[31:16]															
	W																
	R	CMDRSP1[15:0]															
	W																
0x0018 (CMDRSP2)	R	CMDRSP2[31:16]															
	W																
	R	CMDRSP2[15:0]															
	W																

Table 23-3. eSDHC Register Summary (continued)

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x001C (CMDRSP3)	R	CMDRSP3[31:16]															
	W																
	R	CMDRSP3[15:0]															
	W																
0x0020 (DATPORT)	R	DATCONT[31:16]															
	W																
	R	DATCONT[15:0]															
	W																
0x0024 (PRSSTAT)	R	DLSL[7:0]								CLSL	0	0	0	WPSPL	CDPL	0	CINS
	W																
	R	0	0	0	0	BREN	BWEN	RTA	WTA	SDOFF	PEROFF	HCKOFF	IPGOFF	SDSTB	DLA	CDIHB	CIHB
	W																
0x0028 (PROCTL)	R	0	0	0	0	0	WECRM	WECINS	WECINT	0	0	0	0	IABG	RWCTL	CREQ	SABGREQ
	W																
	R	0	0	0	0	0	0	DMAS[1:0]		CDSS	CDTL	EMODE[1:0]		D3CD	DTW[1:0]		LCTL
	W																
0x002C (SYSCTL)	R	0	0	0	0	INITA	0	0	0	0	0	0	0	DTOCV[3:0]			
	W						RSTD	RSTC	RSTA								
	R	SDCLKFS[7:0]								DVS[3:0]				SDCLKEN	PEREN	HCKEN	IPGEN
	W																
0x0030 (IRQSTAT)	R	0	0	0	DMAE	0	0	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
	W				w1c				w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	0	0	0	0	0	0	0	CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
	W								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x0034 (IRQSTATEN)	R	0	0	0	DMAESEN	0	0	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
	W																
	R	0	0	0	0	0	0	0	CINTSEN	CRMSEN	CINSSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
	W																

Table 23-3. eSDHC Register Summary (continued)

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0038 (IRQSIGEN)	R	0	0	0	DMAEIEN	0	0	0	AC12EIEN	0	DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
	W																
	R	0	0	0	0	0	0	0	CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
	W																
0x003C (AUTOC12ERR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CNIBAC12E	0	0	AC12IE	AC12CE	AC12EBE	AC12TOE	AC12NE
	W																
0x0040 (HOSTCAPBLT)	R	0	0	0	0	0	VS18	VS30	VS33	SRS	DMAS	HSS	0	0	MAXBL[1:0]		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x0044 (WML)	R	0	0	0	WR_BRST_LEN[3:0]					WR_WML[7:0]							
	W																
	R	0	0	0	RD_BRST_LEN[3:0]					RD_WML[7:0]							
	W																
0x0050 (FEVT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W	FEVTCINT			FEVTDMAE				FEVTAC12E		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTCTOE
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W									FEVTCNIBAC12			FEVTAC12IE	FEVTAC12EBE	FEVTAC12CE	FEVTAC12TOE	FEVTAC12NE
0x00FC (HOSTVER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	VVN[7:0]							SVN[7:0]								
	W																

23.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 23-3 and Table 23-4 explain conventions used in register diagrams and tables.

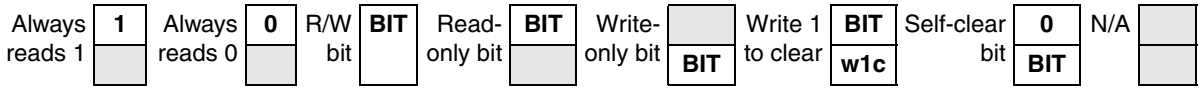


Figure 23-3. Register Field Conventions

Table 23-4. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that can be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

NOTE

The term reset refers to power-on-reset (POR). The reset values given are POR reset values: these may differ from software reset values.

23.3.3.1 DMA System Address Register (DSADDR)

The DSADDR register contains the physical system memory address used for DMA transfers. Figure 23-4 shows the register. Table 23-5 describes the register fields.

Offset 0x0000 (DSADDR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_ADDR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DS_ADDR[15:2]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 23-4. DMA System Address Register Diagram (DSADDR)

Table 23-5. DMA System Address Register Field Descriptions

Field	Description
31–2 DS_ADDR[31:2]	DMA system address. This register contains the 32-bit system memory address for a DMA transfer. Since the address must be (4-byte) word-aligned, the last 2 bits are always read as 0. When the eSDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, writes to this register are ignored. The host driver waits until the DLA bit in the present state register is cleared before writing to this register. The eSDHC internal DMA only supports continuous physical memory access, and does not support a virtual memory system. Due to AHB burst limitations, if a burst of type SEQ crosses the 1-Kbyte boundary, eSDHC I automatically changes the burst type to NSEQ. This register supports dynamic address reflection: when TC bit is set, it automatically alters the value of internal address counter. Software cannot change this register when TC bit is set.
1–0	Reserved.

23.3.3.2 Block Attributes Register (BLKATTR)

This register is used to configure the number of data blocks and the number of bytes in each block.

Figure 23-5 shows the register. Table 23-6 describes the register fields.

Offset 0x0004 (BLKATTR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BLKCNT[15:0][1]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BLKSIZE[12:0]												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-5. Block Attributes Register (BLKATTR)

Table 23-6. Block Attributes Register Field Descriptions

Field	Description
31–16 BLKCNT	Block count for current transfer. This field is enabled when the block count enable (BCEN) bit in the transfer mode register is set to 1, and is valid only for multi-block transfers. For single block transfers, the register always reads as 1. For multi-block transfers, the host driver sets this register to a value between 1 and the maximum block count. The eSDHC decrements the block count after each block transfer, and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred. This register should be accessed only when no transaction is executing (that is, after transactions are stopped). Read operations on this register during data transfers can return an invalid value, and write operations are ignored. When saving transfer content as a result of a Suspend command, the number of blocks yet to be transferred can be determined by reading this register after transfer is paused by a stop at block gap operation, and before sending the Suspend command. After the Suspend command is sent the eSDHC regards the current transfer as aborted and changes the BLKCNT register back to its original value. When restoring transfer content prior to issuing a Resume command, the host driver is responsible to restore the previously-saved block count. 0x0000Stop count 0x0001 1 block 0x0002 2 blocks ..... 0xFFFF 65535 blocks

Table 23-6. Block Attributes Register Field Descriptions (continued)

Field	Description
15–13	Reserved.
12–0 BLKSIZE[12:0]	Transfer block size. This register specifies the block size (in bytes) for block data transfers. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations are ignored. 0x000) No data transfer 0x001) 1 Byte 0x002) 2 Bytes … 0x1000) 4096 Bytes 0x1001–0xFFFF Reserved

23.3.3.3 Command Argument Register (CMDARG)

Figure 23-6 shows the register. Table 23-7 describes the register fields.

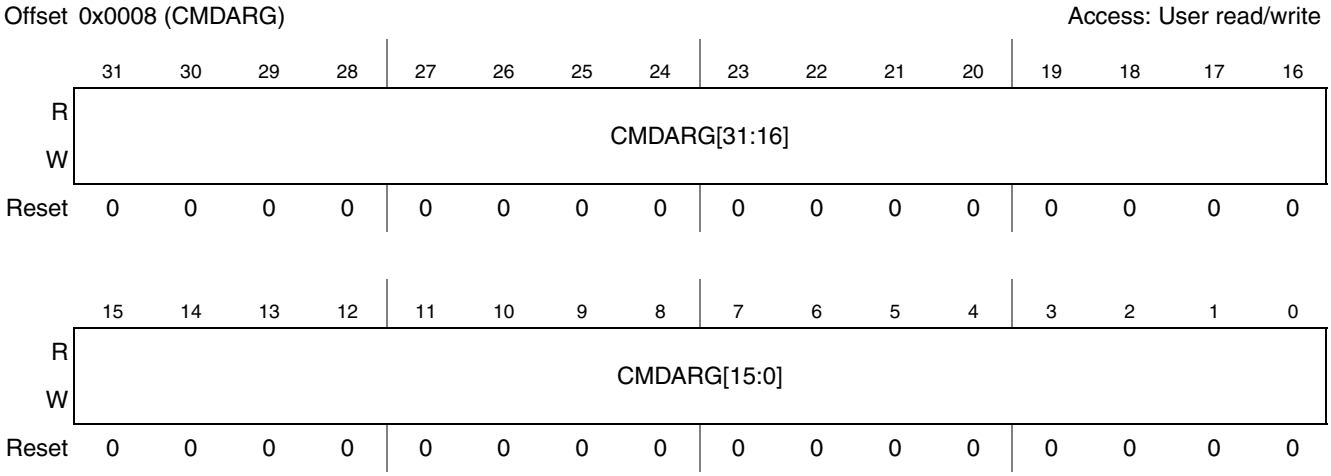


Figure 23-6. Command Argument Register (CMDARG)

Table 23-7. Command Argument Register Field Descriptions

Field	Description
31–0 CMDARG[31:0]	Command argument. The SD/MMC command argument is specified as bits 39-8 of the command format in the SD and MMC specifications.This register is write-protected when the CIHB bit is set in the present state register.

23.3.3.4 Transfer Type Register (XFERTYP)

The XFERTFYP register is used to control the operation of data transfers. The host driver is responsible to set this register before issuing a command followed by a data transfer, or before issuing a Resume command. During data transfers, to prevent data loss the eSDHC prevents writing to the following bits: DPSEL, MBSEL, DTDSEL, AC12EN, BCEN and DMAEN.

The host driver is responsible to check the command inhibit bits (CDIHB and CIHB) in the present state register before writing to this register. When the CDIHB bit is set, any attempt to send a command with data transfer by writing to this register is ignored; when the CIHB bit is set, any write to this register is ignored.

When sending commands followed by data transfer, the command is ignored unless the following conditions are met:

- Block size is nonzero (specified by the BLKSIZE field in the BLKATTR register).
- At least one of the following conditions holds:
  - Block count is nonzero (specified by the BLKCNT field in the BLKATTR register).
  - Single-block transfer is indicated (MSBSEL bit in XFERTYP register is cleared)
  - Block count is disabled (BCEN bit in XFERTYP register is cleared)
- (Write commands only) Write protect switch is inactive (WPSPL bit in present state register is set to 1).

After the command followed by data transfer is sent, if no response is received within 64 clock cycles (the response timeout period) then the eSDHC aborts the data transfer. Response timeout occurs for instance in the following cases:

- The card responds to the command, but eSDHC does not receive the response
- An internal DMA read operation occurs, so that external system memory is overwritten by the internal DMA with data sent back from the card.

If the command times out, the driver is responsible to reissue the command.

Figure 23-7 shows the register. Table 23-8 describes the register fields.

Offset 0x000C (XFERTYP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	CMDINX[5:0]						CMDTYP [1:0]		DPSEL	CICEN	CCCEN		RSPTYP [1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	MSBSEL	DTDSEL	0	AC12EN	BCEN	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-7. Transfer Type Register (XFERTYP)

Table 23-8. Transfer Type Register Field Descriptions

Field	Description
31–30	Reserved.
29–24 CMDINX[5:0]	Command index. These bits are set to the command number that is specified in bits 45-40 of the command format in the SD Memory Card Physical Layer Specification and the SDIO Card Specification.
23–22 CMDTYP[1:0]	Command type. There are three types of special commands: Suspend, Resume and Abort (for all other commands, CMDTYP is set to 0b00). The three special commands are described as follows: • Suspend command: If a Suspend command is sent, the eSDHC assumes that the card bus has been released and that it can issue the next command which uses the DAT line. However, the eSDHC does not monitor the content of command response, so it does not know if the Suspend command succeeds or not. It is the host driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform the eSDHC that a Suspend command was successfully issued. Refer to Section 23.5.3.3.1, "Suspend Operation" for more details. After the end bit of the command is sent, the eSDHC negates read wait for read transactions and stops checking busy for write transactions. In 4-bit mode, the interrupt cycle starts. If the Suspend command fails, the eSDHC maintains its current state, and the host driver restarts the transfer by setting the continue request bit in the protocol control register. • Resume command: The host driver restarts the data transfer by restoring the registers saved before sending the Suspend command and then sends the Resume command. The eSDHC checks for a pending busy state before starting write transfers. • Abort command: If this command is set when executing a read transfer, the eSDHC stops reads to the buffer. If this command is set when executing a write transfer, the eSDHC stops driving the DAT line. After issuing the Abort command, the host driver should issue a software reset (Abort transaction). 00 Normal— all other commands 01 Suspend CMD52 for writing bus suspend in CCCR 10 Resume CMD52 for writing function select in CCCR 11 Abort CMD12, CMD52 for writing I/O Abort in CCCR
21 DPSEL	Data present select. This bit is set to 1 to indicate that data is ready to be transferred using the DAT line. It is set to 0 in the following cases: • Commands using only the CMD line (such as CMD52). • Commands that transfer no data but use the busy signal on DAT[0] line (R1b or R5b, such as CMD38) 0 No data present 1 Data present **Note:** • The Resume command requires this bit be set to 1, while the other XFERTYP fields (except for CMDTYP) are the same as when the transfer was initially launched. • When the write protect switch is on, (WPSPL bit is cleared in the present state register), all writes to the transfer type register with DTDSEL = 0 and DPSEL = 1 are ignored.
20 CICEN	Command index check enable. When this bit is set to 1, the eSDHC checks the index field in the response to see if it has the same value as the command index. If not, the eSDHC reports a command index error. The Index field is not checked if the bit is cleared. 0 Disable command index check 1 Enable command index check
19 CCCEN	Command CRC check enable. If this bit is set to 1, the eSDHC checks the CRC field in the response. If an error is detected, the eSDHC reports a command CRC error. The CRC field is not checked if this bit is cleared. The number of bits checked by the CRC field value changes according to the length of the response. (Refer to RSPTYP[1:0] and Table 23-10.) 0 Disable command CRC check 1 Enable command CRC check
18	Reserved

Table 23-8. Transfer Type Register Field Descriptions (continued)

Field	Description
17–16 RSPTYP[1:0]	Response type select: 00 No response 01 Response length 136 10 Response length 48 11 Response length 48, check busy after response
15–6	Reserved.
5 MSBSEL	Multi / single-block select: This bit enables multi-block DAT line data transfers. For any other commands, this bit is set to 0. If this bit is 0, it is not necessary to set the block count register. Table 23-9 shows MSBSEL settings corresponding to different transfer types. 0 Single-block transfer 1 Multi-block transfer
4 DTDSEL	Data transfer direction select. This bit defines the direction of DAT line data transfers. The host driver sets this bit to 1 to transfer data from the SD card to the eSDHC. This bit is cleared for all other commands. 0 Write (host to card) 1 Read (card to host)
3	Reserved.
2 AC12EN	Auto CMD12 enable. Multi-block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the eSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver does not set this bit to issue commands that do not require CMD12 to stop a multi-block data transfer. In particular, secure commands defined in the file security specification do not require CMD12. In single-block transfers, the eSDHC ignores this bit. 0 Disable auto CMD12 1 Enable auto CMD12
1 BCEN	Block count enable. This bit enables the block count register, which is only relevant for multi-block transfers. When this bit is cleared, the internal block counter is disabled: this setting is used for infinite transfers. Table 23-9 shows BCEN settings corresponding to different transfer types. 0 Disable block count register 1 Enable block count register
0 DMAEN	DMA enable. This bit enables internal DMA functionality. If this bit is set to 1, an internal DMA operation begins when the host driver sets the DPSEL bit of this register. 0 Disable DMA 1 Enable DMA

Table 23-9 shows multi/single-block select (MSBSEL) and block count enable (BCEN) settings for different data transfer types.

Table 23-9. MSBSEL and BCEN Bit Settings for Different Transfer Types

Multi/Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

Table 23-10 shows the settings for responses type select (RSPTYP), index check enable (CICEN), and command CRC check enable (CCCEN) fields in the XFERTYP register corresponding to different response types.

Table 23-10. RSPTYP, CICEN, and CCCEN Settings for Different Response Types

Response Type Select (RSPTYP)	Index Check Enable (CICEN)	CRC Check Enable (CCCEN)	Response Type
00	0	0	No response
01	0	1	R2
10	0	0[1]	R3,R4
10	1	1	R1,R5,R6
11	1	1	R1b,R5b[2]

1 The CRC field for R3 and R4 is expected to be all 1's. The CRC check is disabled for these response types.

2 The SDIO Specification does not distinguish between R5 and R5b. The notation R5b here indicates the case where eSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command is used with R5b.

23.3.3.5 Command Response *n* Register (CMDRSP0–CMDRSP3)

The CMDRSP*n* registers are used to store parts *n* of the response bits from the card (*n* = 0,1,2,3).

Figure 23-8 shows the register. Table 23-11 describes the register fields.

Offset 0x0010 (CMDRSP0)
0x0014 (CMDRSP1)
0x0018 (CMDRSP2)
0x001C (CMDRSP3)

Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP*n*[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP*n*[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-8. Command Response *n* Register (CMDRSP0-CMDRSP3)

Table 23-11. Command Response *n* Register Field Description

Field	Description
31–0 CMDRSP*n*[31:0]	Command response *n*. Refer to Table 23-12 for the mapping of command responses from the SD bus to this register for each response type.

Table 23-12 describes the mapping of command responses from the SD bus to command response registers for each response type. In the table, R[*x*:*y*] refers to a bit range within the response data as transmitted on the SD bus.

Table 23-12. Response Field Definitions for Each Response Type

Response Type	Meaning of Response	Response Field	Response Register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0
R1b (auto CMD12 response)	Card status for auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O and others	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (Publish RCA)	New Published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

The eSDHC only stores part of the response data in the command response registers. This enables the host driver to efficiently read 32 bits of response data in one read cycle on a 32-bit bus system.The eSDHC checks the response data's index field and the CRC (as specified by the command index check enable and the command CRC check enable bits in the transfer type register) and generates an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the eSDHC checks R[47:1], and if the response length is 136 the eSDHC checks R[119:1].

Table 23-12 shows that the auto CMD12 and CMD_wo_DAT responses are stored in the CMDRSP3 and CMDRSP0 registers, respectively. This prevents overwriting of the auto CMD12 response with the CMD_wo_DAT (or vice versa) when the corresponding commands are executed concurrently. When the eSDHC modifies part of the command response registers as shown in Table 23-12, it preserves the unmodified bits.

23.3.3.6 Buffer Data Port Register (DATPORT)

This register contains the buffer data port. Figure 23-9 shows the register. Table 23-13 describes the register fields. The 32-bit DATPORT register is used for CPU or external DMA access the internal buffer. When the internal DMA is enabled, writes to this register is ignored, and the register is read as 0.

Offset 0x0020 (DATPORT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATCONT[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-9. Buffer Data Port Register (DATPORT)

Table 23-13. Buffer Data Port Register Field Descriptions

Field	Description
31–0 DATCONT[31:0]	Data content. When the internal DMA is enabled, writes to this register are ignored, and the register is read as 0.

23.3.3.7 Present State Register (PRSSTAT)

The 32-bit read-only PRSSTAT register provides the host driver with the eSDHC status.

Figure 23-10 shows the register. Table 23-14 describes the register fields.

Offset 0x0024 (PRSSTAT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DLSL[7:0]								CLSL	0	0	0	WPSPL	CDPL	0	CINS
W																
Reset	1	1	1	1	*n*[1]	1	1	1	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BREN	BWEN	RTA	WTA	SDOFF	PEROFF	HCKOFF	SDSTB	IPGOFF	DLA	CDIHB	CIHB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[1] The reset value of DLSL [3] depends on the pull-up/pull-down configuration of DAT[3] (see Table 23-1). If DAT[3] is pulled down, then DLSL[3] resets to 0; if DAT[3] is pulled up, then DLSL[3] resets to 1.

Figure 23-10. Present State Register (PRSSTAT)

Table 23-14. Present State Register Field Descriptions

Field	Description
31–24 DLSL[7:0]	DAT[7:0] line signal level. These status bits are used to check the DAT line levels to recover from errors, and for debugging.DLSL[0] is especially useful for detecting the busy signal level. The reset value is affected by the external pull-up / pull-down resistors. These bits all reset to 1 except for DLSL[3], whose reset value depends on the pull-up / pull-down configuration of DAT[3]. If DAT[3] is pulled down, then DLSL[3] resets to 0; if DAT[3] is pulled up, then DLSL[3] resets to 1. DAT[*n*]: Data *n* line signal level (*n* = 7…0)
23 CLSL	CMD line signal level. This status bit is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up/pull-down resistor—by default, the read value of this bit after reset is 0b1.
22–20	Reserved.
19 WPSPL	Write protect switch pin level. The write protect switch is supported for memory and combo cards.This bit reflects the inverted value of the SD_WP pin of the card socket. A software reset does not affect this bit. The reset value is effected by the external write protect switch. If the SD_WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 0 Write protected (SD_WP=1) 1 Write enabled (SD_WP=0)
18 CDPL	Card detect pin level. This bit reflects the inverse value of the SD_CD# signal for the card socket: for instance, when a card is inserted in the socket the SD_CD# input signal is negated, and the CDPL bit is set to 1. Debouncing is not performed on this bit. This bit may be valid, but is not guaranteed because of propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not effect this bit. A write to the force event register does not effect this bit. The reset value is effected by the external card detection pin. 0 No card present (SD_CD# = 1) 1 Card present (SD_CD# = 0)
17	Reserved.
16 CINS	Card inserted. This bit indicates whether a card has been inserted. The eSDHC debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from a 0 to 1 generates a card insertion interrupt in the interrupt status register. Changing from a 1 to 0 generates a card removal interrupt in the interrupt status register. A write to the force event register does not effect this bit. Setting the software reset for all (RSTA) bit in the system control register does not effect this bit. A software reset does not effect this bit. 0 Power on reset or no card 1 Card inserted
15–12	Reserved.
11 BREN	Buffer read enable. This read-only flag indicates that valid data exists in the host-side buffer.When this bit is set to 1, valid data in the buffer meets or exceeds the read watermark level. This bit changes from 1 to 0 whenever data is read from the buffer. This bit changes from 0 to1 when valid data in the buffer meets or exceeds the watermark level, and the buffer read ready interrupt has been generated and enabled. 0 Read disable 1 Read enable This status bit is used for non-DMA read transfers. The eSDHC may implement multiple buffers to transfer data efficiently.

Table 23-14. Present State Register Field Descriptions (continued)

Field	Description
10 BWEN	Buffer write enable. This read-only flag indicates that space is available for write data. If this bit is set to 1, available space in the buffer meets or exceeds the write watermark level]. This bit changes from 1 to 0 when data is written to the buffer. This bit changes from 0 to 1 when the available space in the buffer meets or exceeds the write watermark level, and the buffer write ready interrupt is generated and enabled. 0 Write disable 1 Write enable This status bit is used for non-DMA write transfers. The eSDHC can implement multiple buffers to transfer data efficiently.
9 RTA	Read transfer active. This status bit is used to detect completion of a read transfer. This bit is set in either of the following two cases: • After the end bit of the read command. • When the continue request bit in the protocol control register is set to 1 to restart a read transfer. A transfer complete interrupt is generated when the RTA bit is cleared. The bit is cleared in either of the following two cases: • When the last data block as specified by the block size is transferred to the system, so that all data are read from the eSDHC internal buffer. • When all valid data blocks have been transferred from eSDHC internal buffer to the system and no current block transfers are being sent because the stop at block gap request is set to 1. 0 No valid data 1 Transferring data
8 WTA	Write transfer active. This status bit indicates a write transfer is active. If this bit is cleared, it means no valid write data exists in the eSDHC. This bit is set in either of the following cases: • After the end bit of the write command. • When the continue request bit in the protocol control register is set to 1 to restart a write transfer. This bit is cleared in either of the following cases: • After receiving the CRC status of the last data block as specified by the transfer count (single-block or multi-block). • After receiving the CRC status of the current block, when data transmission is about to be stopped by a stop at block gap request. A block gap event interrupt is generated when this bit is cleared as result of a stop at block gap request during a write transaction. The host driver can use this status bit to determine when to issue commands during write busy state. 0 No valid data 1 Transferring data
7 SDOFF	SD clock gated off internally. This status bit indicates that the SD clock is internally gated off due to one of the following causes: • Buffer overrun or underrun • Read pause without read wait assertion • The driver has cleared the SDCLKEN bit to stop the SD clock. The host driver can use this bit to debug data transactions on the SD bus. 0 SD clock is active. 1 SD clock is gated off.
6 PEROFF	Peripheral source clock (ipg_perclk) gated off internally. This status bit indicates that the ipg_perclk is internally gated off. The host driver can use this bit for debug purposes during transactions on the SD bus. 0 ipg_perclk is active. 1 ipg_perclk is gated off.

Table 23-14. Present State Register Field Descriptions (continued)

Field	Description
5 HCKOFF	Master clock (hclk) gated off internally. This status bit indicates that the hclk is internally gated off. The host driver can use this bit for debug purposes during data transfers. 0 hclk is active. 1 hclk is gated off.
4 SDSTB	SD clock stable. This status bit indicates that the internal card clock is stable. This bit is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear SDCLKEN bit in system control register to remove glitches on the card clock when the frequency is changing. 0 Clock is changing frequency and not stable. 1 Clock is stable.
3 IPGOFF	ipg_clk is gated off internally. This status bit indicates that the ipg_clk is internally gated off. The host driver can use this bit for debug purposes. 0 ipg_clk is active. 1 ipg_clk is gated off.
2 DLA	Data line active. This status bit indicates whether one of the DAT lines on the SD bus is in use. In the case of read transactions: This bit indicates if a read transfer is executing on the SD bus. A change in this bit from 1 to 0 between data blocks generates a block gap event interrupt in the interrupt status register. This bit is set in either of the following cases: • After the end bit of the read command. • When writing a 1 to the continue request bit in the protocol control register to restart a read transfer. This bit is cleared in either of the following cases: • When the end bit of the last data block is sent from the SD bus to the eSDHC. • When the read wait state is stopped by a suspend command and the DAT2 line is released. The eSDHC waits at the next block gap by driving a read wait signal at the start of the interrupt cycle. If the read wait signal is already driven (indicating that the data buffer cannot receive data), the eSDHC can wait for a current block gap by continuing to drive the read wait signal. It is necessary to support read wait in order to use the suspend / resume function. This bit remains set to 1 during read wait. In the case of write transactions: This bit indicates that a write transfer is executing on the SD bus. Changes in this bit from 1 to 0 generate a transfer complete interrupt in the interrupt status register. This bit is set in either of the following cases: • After the end bit of the write command. • When writing to 1 to the continue request bit in the protocol control register to continue a write transfer. This bit is cleared in either of the following cases: • When the SD card releases write busy of the last data block, the eSDHC also detects if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, the eSDHC assumes the card drives the not busy signal. • When the SD card releases write busy, prior to waiting for write transfer, and as a result of a stop at block gap request. In the case of command with busy pending: This status bit indicates that a busy state follows the command and the data line is in use. This bit is cleared when the DAT0 line is released. 0 DAT line inactive 1 DAT line active

Table 23-14. Present State Register Field Descriptions (continued)

Field	Description
1 CDIHB	Command inhibit (DAT). This status bit is generated if either the DAT line active (DLA) bit or the read transfer active (RTA) bit in this register is set to 1. If this bit is cleared, it indicates that the eSDHC can issue the next SD/MMC command. Commands with a busy signal (for example, R1b, R5b type) belong to command inhibit (DAT). Except in the case when the command busy is finished, changing this bit from 1 to 0 generates a transfer complete interrupt in the interrupt status register. **Note:** The SD host driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 0 Can issue command which uses the DAT line 1 Cannot issue command which uses the DAT line
0 CIHB	Command inhibit (CMD). This status bit is cleared when the CMD line is not in use, indicating that the eSDHC can issue a SD/MMC command using the CMD line. This bit is set immediately after the transfer type register is written. This bit is cleared when the command response is received. Changing this bit from 1 to 0 sets the command complete (CC) bit in the interrupt status register, which generates an interrupt if the CCIEN bit is set in the interrupt signal enable register. Commands using only the CMD line can be issued if this bit is cleared, even when CDIHB (bit 1 of this register) is set. Examples of commands using only the CMD line include CMD0, CMD12, CMD13 (for memory cards) and CMD52 (for SDIO). This bit is remains set (and the CC bit in the interrupt status register remains cleared) in either of the following two cases: • If the eSDHC detects a CMD line conflict when the command is issued (eSDHC drives the CMD line to 1, but detects a 0 at the next SD_CLK edge). In this case, command CRC error and command timeout error bits in the interrupt status register are set to 1. • If the command is not issued due to an auto CMD12 error. 0 Can issue command using only the CMD line 1 Cannot issue command

23.3.3.8 Protocol Control Register (PROCTL)

Figure 23-11 shows the register. Table 23-15 describes the register fields.

Offset 0x0028 (PROCTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	WECRM	WECINS	WECINT					IABG	RWCTL	CREQ	SABGREQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0		CDSS	CDTL	EMODE[1:0]		D3CD	DTW[1:0]		LCTL
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 23-11. Protocol Control Register (PROCTL)

Table 23-15. Protocol Control Register Field Descriptions

Field	Description
31–27	Reserved.
26 WECRM	Wake-up event enable on SD card removal. This bit enables a wake-up event (using a card removal) in the interrupt status register. FN_WUS (wake-up support) in CIS does not effect this bit. When this bit is set, the card removal status and the eSDHC interrupt can be asserted without SD_CLK toggling. When the wake-up feature is not enabled, the SD_CLK must be active in order to assert the card removal status and assert the eSDHC interrupt. 0 Disable 1 Enable
25 WECINS	Wake-up event enable on SD card insertion. This bit enables a wake-up event, using a card insertion, in the interrupt status register. FN_WUS (wake-up support) in CIS does not effect this bit. When this bit is set, the card insertion status and the eSDHC interrupt can be asserted without SD_CLK toggling. When the wake-up feature is not enabled, the SD_CLK must be active in order to assert the card insertion status and assert the eSDHC interrupt. 0 Disable 1 Enable
24 WECINT	Wake-up event enable on card interrupt. This bit enables a wake-up event, using a card interrupt, in the interrupt status register. This bit can be set to 1 if FN_WUS (wake-up support) in CIS is set to 1. When this bit is set, the card interrupt status and the eSDHC interrupt can be asserted without SD_CLK toggling. When the wake-up feature is not enabled, the SD_CLK must be active in order to assert the card interrupt status and assert the eSDHC interrupt. 0 Disable 1 Enable
23–20	Reserved.
19 IABG	Interrupt at block gap. This bit is only valid for SDIO cards in 4-bit mode. It selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multi-block transfer. Clearing the bit disables interrupt detection during a multi-block transfer. If the SDIO card cannot signal an interrupt during a multi-block transfer, this bit should be cleared to avoid an inadvertent interrupt. When the host driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. 0 Disabled 1 Enabled
18 RWCTL	Read wait control. The read wait function is optional for SDIO cards. If the card supports read wait, setting this bit enables the read wait protocol to stop read data using the DAT[2] line. Otherwise the eSDHC has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it sets this bit according to the CCCR of the card. If the card does not support read wait, this bit is never set to 1—otherwise DAT line conflicts may occur. If this bit is cleared and the stop at block gap request (SABGREQ) bit is set, the eSDHC stops the SD clock to pause the read operation. 0 Disable read wait control, and stop SD Clock at block gap when SABGREQ bit is set 1 Enable read wait control, and assert read wait without stopping SD clock at block gap when SABGREQ bit is set
17 CREQ	Continue request. When a Suspend operation is not accepted by the card, setting this bit restarts the paused transfer. This bit is also used to restart a transaction that was stopped using the stop at block gap request (SABGREQ). To cancel the stop at block gap, the host driver clears the SABGREQ bit and sets this bit to 1 to restart the transfer. If both the SABGREQ bit and this bit are set to 1, the continue request is ignored. The eSDHC automatically clears this bit: the host driver does not need to clear it. 0 No effect 1 Restart

Table 23-15. Protocol Control Register Field Descriptions (continued)

Field	Description
16 SABGREQ	Stop at block gap request. This bit is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. In the case of read transfers, setting the SABREQ bit stops the transaction at the next block gap only if the SDIO card supports read wait, and the read wait control bit (RWCTL) in this register is set to 1. Otherwise, the read operation can be paused at the next block gap by stopping the SD bus clock. In the case of write transfers in which the host driver writes data to the data port register, the host driver sets this bit after all block data is written. If this bit is set to 1, the host driver does not write data to the data port register after a block is sent. If the host driver clears this bit before the transfer complete bit in interrupt status register is set:, then the eSDHC's behavior is unpredictable. Clearing both SABGREQ and CREQ bits does not cause the transaction to restart. This bit affects the read transfer active, write transfer active, DAT line active and command inhibit (DAT) bits in the present state register. 0 Transfer 1 Stop
15–10	Reserved.
9–8	Reserved.
7 CDSS	Card detect signal selection. This bit selects the source for the card detection. 0 Card detection level is selected (for normal purposes) 1 Card detection test level is selected (for test purposes)
6 CDTL	Card detect test level. This is bit is enabled while the card detection signal selection is set to 1 and it indicates card insertion. 0 Card detect test level is 0, no card inserted 1 Card detect test level is 1, card inserted
5–4 EMODE	Endian mode. The eSDHC supports all four endian modes in data transfer. Refer to Section 23.4.1, "Data Buffer" for more details. 00 Big-endian mode 01 Halfword big-endian mode 10 Little-endian mode 11 Reserved
3 D3CD	DAT3 as card detection pin. If this bit is set, DAT3 should be pulled down to act as a card detection pin. 0 DAT3 does not monitor card insertion 1 DAT3 as card detection pin **Note:** Since DAT3 is also a chip select for SPI mode, a pull-down on this pin and CMD0 may set the card into the SPI mode. eSDHC does not support SPI mode.
2–1 DTW[1:0]	Data transfer width. This bit selects the data width of the SD bus for a data transfer. The host driver sets it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits. 00 1-bit mode 01 4-bit mode 10 8-bit mode 11 Reserved
0 LCTL	LED control. This bit, fully controlled by the host driver, is used to caution the user not to remove the card while the card is being accessed. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands. 0 LED off 1 LED on

There are three ways to restart the transfer after stop at the block gap, depending on the status of the Suspend command.

1. If the host driver does not issue a Suspend command, the continue request is used to restart the transfer.
2. If the host driver issues a Suspend command and the SD card accepts it, a Resume command is used to restart the transfer.
3. If the host driver issues a Suspend command and the SD card does not accept it, the continue request is used to restart the transfer.

When a stop at block gap request stops the data transfer, the host driver waits for the transfer complete bit (interrupt status register) to be set before attempting to restart the transfer. When restarting the data transfer by continue request, the host driver clears the stop at block gap request before, or simultaneously with, issuing the continue request.

23.3.3.9 System Control Register (SYSCTL)

Figure 23-12 shows the register. Table 23-16 describes the register fields.

Offset 0x002C (SYSCTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	INITA	0	0	0	0	0	0	0	DTOCV[3:0]			
W						RSTD	RSTC	RSTA								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SDCLKFS[7:0]								DVS[3:0]				SDCLKEN	PEREN	HCKEN	IPGEN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-12. System Control Register (SYSCTL)

Table 23-16. System Control Register Field Descriptions

Field	Description
31–28	Reserved.
27 INITA	Initialization active. When this bit is set, 80 SD clocks are sent to the card. After the 80 clocks are sent, this bit self clears. This bit is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this bit is already 1 has no effect. Writing 0 to this bit at any time has no effect. When either of the CIHB and CDIHB bits in the present state register are set, writes to this bit is ignored (when the command or data lines are active, writes to this bit is not allowed). However, when this bit is set to 1 (that is, during the initialization active period), it is still possible to issue a command—the command bit stream appears on the CMD signal after the 80 clock cycles are completed. In this way, the driver can ensure that 80 clock cycles have completed before the command is sent out. This is a useful feature in the case where the driver needs to send 80 cycles to the card and does not want to wait until this bit is self-cleared.
26 RSTD	Software reset for DAT line. Resets part of the data circuit and the DMA circuit. Setting this bit clears and initializes the buffer, and also clears the following bits: • Data port register: all bits • Present state register: buffer read enable, buffer write enable, read transfer active, write transfer active, data line active, command inhibit (DAT) • Protocol control register: Continue request, Stop at block gap request • Interrupt status register: buffer read ready, buffer write ready, DMA interrupt, block gap event, transfer complete 0 No reset 1 Reset
25 RSTC	Software reset for CMD line. Resets part of the command circuit. The following bits are cleared by this bit: • Command inhibit (CMD) bit in present state register • Command complete (CC) bit in interrupt status register 0 No reset 1 Reset
24 RSTA	Software reset for all. This reset effects the entire host controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared. During initialization, the host driver sets this bit to 1 to reset the eSDHC. The eSDHC resets this bit to 0 when the capabilities registers are valid and the host driver can read them. Additional use of software reset for all does not affect the value of the capabilities registers. After this bit is set, it is recommended that the host driver reset and reinitialize the external card. 0 No Reset 1 Reset
23–20	Reserved.

Table 23-16. System Control Register Field Descriptions (continued)

Field	Description
19–16 DTOCV	Data timeout counter value. This value determines the interval by which DAT line timeouts are detected. Refer to the data timeout error bit in the interrupt status register for information on factors that dictate time-out generation. Time-out clock frequency is generated by dividing the base clock SD_CLK frequency by this value. The host driver can clear the data timeout error status enable (in the interrupt status enable register) to prevent inadvertent time-out events. 0000 SD_CLK x 2^{13} 0001 SD_CLK x 2^{14} ..... 1110 SD_CLK x 2^{27} 1111 Reserved
15–8 SDCLKFS	SD clock (SD_CLK) frequency prescale select. This register is used to select the prescaler of the SD_CLK signal. The frequency of the SD clock is set based on SD_CLKF and DVS settings according to the following formula: SD clock frequency = (peripheral source clock frequency) / (SDCLKF x DVS) See Section 23.4.6, “SD Clock Generator” for more information about SD clock generation. Multiple bits in this field must not be set, or the behavior of the prescaler is undefined. 0x00 Base clock (10–63 MHz) divided by 1 0x01 Base clock divided by 2 0x02 Base clock divided by 4 0x04 Base clock divided by 8 0x08 Base clock divided by 16 0x10 Base clock divided by 32 0x20 Base clock divided by 64 0x40 Base clock divided by 128 0x80 Base clock divided by 256 (reset value) Other settings Reserved, not allowed
7–4 DVS[3:0]	Frequency divisor. The frequency of the SD clock is set based on SDCLKF and DVS settings according to the following formula: SD clock frequency = (peripheral source clock frequency) / (SDCLKF x DVS) DVS provides a finer granularity for the available frequencies. Odd divisors are supported without deterioration of the duty cycle. See Section 23.4.6, “SD Clock Generator” for more information about SD clock generation. 0x0 Divide by 1 0x1 Divide by 2 ... 0xE Divide by 15 0xF Divide by 16
3 SDCLKEN	SD clock (SD_CLK) enable. The host controller stops SD_CLK when this bit is cleared. The SD_CLK frequency can only be changed when this bit is cleared. If the card inserted bit in the present state register is cleared, the host driver saves power by clearing this bit. 0 SD clock is stopped 1 SD clock is enabled

Table 23-16. System Control Register Field Descriptions (continued)

Field	Description
2 PEREN	Peripheral source clock (ipg_perclk) enable. When this bit is set, ipg_perclk is always active and no automatic gating is applied. In this case SD_CLK is active except during auto gating-off in case of buffer danger (pending underrun or overrun).When this bit is cleared, the ipg_perclk is automatically gated off when there is no transaction on the SD bus, and when none of the following conditions are met: • The cmd part is reset • Data part is reset • A soft reset • The cmd is about to be sent • Clock divisor is just updated • Continue request is just set • Card insertion is detected • Card removal is detected • Card external interrupt is detected • 80 clocks for initialization phase is ongoing Since this bit is only a feature-enabling bit, clearing this bit does not stop SD_CLK immediately. 0 ipg_perclk is internally gated off. 1 ipg_perclk is not automatically gated off and is active.
1 HCKEN	Master clock (hclk) enable. If this bit is set, hclk is always active and no automatic gating is applied. When this bit is cleared, hclk is automatically off when no data transfer is active on the SD bus. 0 hclk is internally gated off 1 hclk is not automatically gated off
0 IPGEN	ipg_clk enable. If this bit is set, ipg_clk is always active and no automatic gating is applied. When this bit is cleared, the ipg_clk is internally gated off if none of the following conditions is met: • The cmd part is reset • Data part is reset • Soft reset • The cmd is about to be sent • Clock divisor is just updated • Continue request is just set • Card insertion is detected • Card removal is detected • Card external interrupt is detected • The ipg_perclk is not gated off (thus clearing this bit has no effect unless the PEREN bit is also cleared) 0 ipg_clk is internally gated off 1 ipg_clk is not automatically gated off

23.3.3.10 Interrupt Status Register (IRQSTAT)

An interrupt is generated when at least one of the status bits in this register is set to 1, and the corresponding interrupt enable bit is set in the interrupt signal enable register. All bits are write 1 to clear: writing zeros has no effect. More than one status bit can be cleared with a single register write.

Figure 23-13 shows the register. Table 23-17 describes the register fields.

Offset 0x0030 (IRQSTAT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMAE	0	0	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				w1c				w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-13. Interrupt Status Register (IRQSTAT)

Table 23-17. Interrupt Status Register Field Descriptions

Field	Description
31–29	Reserved.
28 DMAE	DMA error. This bit is set to 1, when some error occurs in the internal DMA data transfer. The value in DMA system address register is the next fetch address where the error occurs. Since any error corrupts the whole data block, the host driver re-starts the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 0 No DMA error 1 DMA error
27–25	Reserved.
24 AC12E	Auto CMD12 error. This bit is set to 1 when the eSDHC detects that one of the bits in the auto CMD12 error status register has changed from 0 to 1. This bit is set either when the errors in auto CMD12 occur, or when the auto CMD12 is not executed due to an error in the previous command. 0 No auto CMD12 error 1 Auto CMD12 error
23	Reserved.
22 DEBE	Data end bit error. Occurs either when the eSDHC detects 0 at the end bit position of read data, which uses the DAT line, or at the end bit position of the CRC. 0 No data end bit error 1 Data end bit error
21 DCE	Data CRC error. Occurs when detecting a CRC error when transferring read data, which uses the DAT line, or when detecting the write CRC status having a value other than 010. 0 No data CRC error 1 Data CRC error

Table 23-17. Interrupt Status Register Field Descriptions (continued)

Field	Description
20 DTOE	Data timeout error. Occurs when detecting one of following time-out conditions. • Busy time-out for R1b,R5b response types • Busy time-out after write CRC status • Read data time-out. 0 No data timeout error 1 Data timeout error
19 CIE	Command index error. Occurs if a command index error occurs in the command response. 0 No command index error 1 Command index error
18 CEBE	Command end bit error. Occurs when detecting that the end bit of a command response is 0. 0 No command end error 1 End bit error generated
17 CCE	Command CRC error. A command CRC error is generated in two cases. • If a CRC error is detected in the command response. In this case, the command timeout error bit in this register remains cleared (indicating no timeout) • If the eSDHC detects a CMD line conflict when the command is issued. This occurs when the eSDHC drives the CMD line to 1, but detects a 0 on the CMD line at the next SD_CLK edge. In this case, the eSDHC aborts the command (stops driving the CMD line), and sets the command timeout error bit in this register to 1. 0 No command CRC error 1 Command CRC error generated.
16 CTOE	Command timeout error. This bit is set if no response is returned within 64 SD_CLK cycles from the end bit of the command. If the eSDHC detects a CMD line conflict this bit is set without waiting for 64 SD_CLK cycles (the command CRC error bit (CCE) bit is also set, as shown in Table 23-20). 0 No command timeout error 1 Command timeout error
15–9	Reserved.
8 CINT	Card interrupt. This status bit is set when an interrupt signal is detected from the external card. Wake-up is supported: in 1-bit mode, the eSDHC detects the card interrupt without the SD clock. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, which may introducing some delay between the interrupt signal from the SDIO card and the interrupt to the host system. Writing 1 to this bit clears it, but if the interrupt factor from the SDIO card is not cleared then the bit is immediately set again. When CINT has been set to 1, it is the host driver's responsibility to clear the card interrupt signal enable bit in the interrupt signal enable register, so that the eSDHC stops driving the interrupt while it is being serviced. After the card interrupt service is completed, and the interrupt factors in the SDIO card have been reset, then the host driver can write 1 to clear this bit and set the card interrupt signal enable to 1. This causes the eSDHC to restart sampling the interrupt signal. 0 No card interrupt 1 Generate card interrupt
7 CRM	Card removal. This status bit is set if the card inserted bit in the present state register changes from 1 to 0. When writing 1 to this bit to clear it, the host driver is responsible to confirm the value of the card inserted bit in the present state register. This is necessary because it is possible for the card state to change while the host driver clears this bit, and the interrupt event may not be generated. When the CRM bit is cleared, if no card is inserted it is immediately set again: this can be prevented by clearing the card removal status enable bit in interrupt status enable register. 0 Card state unstable or inserted 1 Card removed

Table 23-17. Interrupt Status Register Field Descriptions (continued)

Field	Description
6 CINS	Card insertion. This status bit is set when the card inserted bit in the present state register changes from 0 to 1. When writing 1 to this bit to clear it, the host driver is responsible to confirm the value of the card inserted bit in the present state register. This is necessary because it is possible for the card state to change while the host driver clears this bit, and the interrupt event may not be generated. When the CIN bit is cleared, if a card is inserted it is immediately set again: this can be prevented by clearing the card inserted status enable bit in interrupt status enable register. 0 Card state unstable or removed 1 Card inserted
5 BRR	Buffer read ready. This status bit is set when the buffer read enable bit changes from 0 to 1. Refer to the buffer read enable bit in the present state register description for additional information. 0 Not ready to read buffer 1 Ready to read buffer
4 BWR	Buffer write ready. This status bit is set when the buffer write enable bit changes from 0 to 1. Refer to the buffer write enable bit in the present state register description for additional information. 0 Not ready to write buffer 1 Ready to write buffer:
3 DINT	DMA interrupt. Occurs only when the internal DMA finishes the data transfer successfully. When errors occur during data transfer, this bit is not set: the DMAE bit is set instead. 0 No DMA interrupt 1 DMA interrupt is generated
2 BGE	Block gap event. If the stop at block gap request bit in the protocol control register is set to 1, this bit is set when a read or write transaction is stopped at a block gap. If the stop at block gap request is not set to 1, this bit is not set to 1. For read transactions, this bit is set at the falling edge of the DAT line active status signal (when the transaction is stopped at SD bus timing). Read wait must be supported in order to use this function. For write transactions, this bit is set at the falling edge of the write transfer active status signal (after getting CRC status at SD bus timing). 0 No block gap event 1 Transaction stopped at block gap

Table 23-17. Interrupt Status Register Field Descriptions (continued)

Field	Description
1 TC	Transfer complete. This bit is set when a read or write transfer is completed. For read transactions, this bit is set at the falling edge of the read transfer active status signal. There are two cases in which the transfer complete interrupt is generated: • When a data transfer is completed as specified by the data length (after the last data has been read to the host system). • When data has stopped at the block gap and completed the data transfer by setting the stop at block gap request bit in the Protocol Control register (after valid data has been read to the host system). For write transactions, this bit is set at the falling edge of the DAT line active status signal. There are two cases in which this interrupt is generated: • When the last data is written to the SD card as specified by the data length and the busy signal is released. • When data transfers are stopped at the block gap, by setting the stop at block gap request bit in the protocol control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). 0 Transfer not complete 1 Transfer complete
0 CC	Command complete. This bit is set when the end bit of the command response is received (except auto CMD12). This bit is set to 1 when the command inhibit (CMD) bit in the present state register is cleared. 0 Command not complete 1 Command complete

Table 23-18 shows the eSDHC status associated with different settings of the command timeout error and command complete bits.

Table 23-18. eSDHC Status for Command Timeout Error/Command Complete Bit Combinations

Command Complete Bit	Command Timeout Error Bit	eSDHC Status
0	0	—
0 or 1	1	Response not received within 64 SD_CLK cycles
1	0	Response received

Table 23-19 shows the eSDHC status associated with different settings of the transfer complete and the data timeout error bits.

Table 23-19. eSDHC Status for Data Timeout Error/Transfer Complete Bit Combinations

Transfer Complete Bit	Data Timeout Error Bit	eSDHC Status
0	0	—
0	1	Timeout occurred during transfer
1	0 or 1	Data transfer complete

Table 23-20 shows the eSDHC status associated with different settings of the command CRC error and command timeout error bits.

Table 23-20. eSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command CCR Error Bit	Command Timeout Error Bit	eSDHC Status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

23.3.3.11 Interrupt Status Enable Register (IRQSTATEN)

Setting bits in this register to 1 enables the corresponding interrupt status register bits to be set by the specified event. If any bit is cleared, the corresponding interrupt status bit is cleared and remains 0.

Figure 23-14 shows the register. Table 23-21 describes the register fields.

Offset 0x0034 (IRQSTATEN) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMAESEN	0	0	0	AC12ESEN	0	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CINTSEN	CRMSEN	CINSSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1

Figure 23-14. Interrupt Status Enable Register (IRQSTATEN)

Table 23-21. Interrupt Status Enable Register Field Descriptions

Field	Description
31–29	Reserved.
28 DMAESEN	DMA error status enable 0 Masked 1 Enabled
27–25	Reserved.
24 AC12ESEN	Auto CMD12 error status enable 0 Masked 1 Enabled
23	Reserved.

Table 23-21. Interrupt Status Enable Register Field Descriptions (continued)

Field	Description
22 DEBESEN	Data end bit error status enable 0 Masked 1 Enabled
21 DCESEN	Data CRC error status enable 0 Masked 1 Enabled
20 DTOESEN	Data timeout error status enable 0 Masked 1 Enabled
19 CIESEN	Command index error status enable 0 Masked 1 Enabled
18 CEBESEN	Command end bit error status enable 0 Masked 1 Enabled
17 CCESEN	Command CRC error status enable 0 Masked 1 Enabled
16 CTOESEN	Command timeout error status enable 0 Masked 1 Enabled
15–9	Reserved.
8 CINTSEN	Card interrupt status enable. If this bit is set to 0, the eSDHC clears the interrupt request to the system. the card interrupt detection is stopped when this bit is cleared, and restarted when this bit is set to 1. The host driver is responsible to clear this bit before servicing interrupts, to prevent inadvertent interrupts. After servicing the interrupt, the host driver is responsible to set this bit to 1. 0 Masked 1 Enabled
7 CRMSEN	Card removal status enable 0 Masked 1 Enabled
6 CINSEN	Card insertion status enable 0 Masked 1 Enabled
5 BRRSEN	Buffer read ready status enable 0 Masked 1 Enabled
4 BWRSEN	Buffer write ready status enable 0 Masked 1 Enabled
3 DINTSEN	DMA interrupt status enable 0 Masked 1 Enabled

Table 23-21. Interrupt Status Enable Register Field Descriptions (continued)

Field	Description
2 BGESEN	Block gap event status enable 0 Masked 1 Enabled
1 TCSEN	Transfer complete status enable 0 Masked 1 Enabled
0 CCSEN	Command complete status enable 0 Masked 1 Enabled

NOTE

Depending on the setting of the IABG bit in the protocol control register, eSDHC may be programmed to sample the card interrupt signal during the interrupt period and hold this value in the flip-flop. This can cause a delay between the assertion of the card interrupt signal and notification of the host system.

To detect a CMD line conflict, the host driver sets both the command timeout error status enable bit and the command CRC error status enable bit to 1.

23.3.3.12 Interrupt Signal Enable Register (IRQSIGEN)

This register is used to select the events which are signaled to the host system as interrupts (these status bits all share the same interrupt line). When a bit in this register is set to 1, then setting the corresponding interrupt status register bit to 1 generates an interrupt.

Figure 23-15 shows the register. Table 23-22 describes the register fields.

Offset 0x0038 (IRQSIGEN) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	DMA EIEN	0	0	0	AC12 EIEN	0	DEBE IEN	DCEI EN	DTOE IEN	CIEIE N	CEBE IEN	CCEIE N	CTO EIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CINTI EN	CRMI EN	CINSI EN	BRRI EN	BWRI EN	DINTI EN	BGEI EN	TCIEN	CCI EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-15. Interrupt Signal Enable Register (IRQSIGEN)

Table 23-22. Interrupt Signal Enable Register Field Descriptions

Field	Description
31–29	Reserved.
28 DMAEIEN	DMA error interrupt enable 0 Masked 1 Enable
27–25	Reserved.
24 AC12EIEN	Auto CMD12 error interrupt enable 0 Masked 1 Enabled
23	Reserved.
22 DEBEIEN	Data end bit error interrupt enable 0 Masked 1 Enabled
21 DCEIEN	Data CRC error interrupt enable 0 Masked 1 Enabled
20 DTOEIEN	Data timeout error interrupt enable 0 Masked 1 Enabled
19 CIEIEN	Command index error interrupt enable 0 Masked 1 Enabled
18 CEBEIEN	Command end bit error interrupt enable 0 Masked 1 Enabled
17 CCEIEN	Command CRC error interrupt enable 0 Masked 1 Enabled
16 CTOEIEN	Command timeout error interrupt enable 0 Masked 1 Enabled
15–9	Reserved.
8 CINTIEN	Card interrupt interrupt enable 0 Masked 1 Enabled
7 CRMIEN	Card removal interrupt enable 0 Masked 1 Enabled
6 CINIEN	Card insertion interrupt enable 0 Masked 1 Enabled
5 BRRIEN	Buffer read ready interrupt enable 0 Masked 1 Enabled

Table 23-22. Interrupt Signal Enable Register Field Descriptions (continued)

Field	Description
4 BWRIEN	Buffer write ready interrupt enable 0 Masked 1 Enabled
3 DINTIEN	DMA interrupt enable 0 Masked 1 Enabled
2 BGEIEN	Block gap event interrupt enable 0 Masked 1 Enabled
1 TCIEN	Transfer complete interrupt enable 0 Masked 1 Enabled
0 CCIEN	Command complete interrupt enable 0 Masked 1 Enabled

23.3.3.13 Auto CMD12 Error Status Register (AUTOC12ERR)

When the auto CMD12 error status bit in the status register is set to 1, the host driver checks this register to identify the source of auto CMD12 errors. This register is valid only when the auto CMD12 error status bit in the status register is set to 1.

Figure 23-16 shows the register. Table 23-23 describes the register fields.

Offset 0x003C (AUTOC12ERR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CNIBAC12E	0	0	AC12IE	AC12CE	AC12EBE	AC12TOE	AC12NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-16. Auto CMD12 Error Status Register (AUTOC12ERR)

Table 23-23. Auto CMD12 Error Status Register Field Descriptions

Field	Description
31–8	Reserved.
7 CNIBAC12E	Command not issued by auto CMD12 error. This bit is set to 1 when CMD_wo_DAT is not executed due to an auto CMD12 error (D04–D01) in this register. 0 No error 1 Not issued
6–5	Reserved.
4 AC12IE	Auto CMD12 index error. Indicates that a command index error occurred in response to a command. 0 No error 1 Error, the CMD index in response is not CMD12
3 AC12CE	Auto CMD12 CRC error. Indicates a CRC error in the command response. 0 No CRC error 1 CRC Error Met in auto CMD12 Response
2 AC12EBE	Auto CMD12 end bit error. indicates the end bit of command response is 0 which should be 1. 0 No error 1 End Bit Error Generated
1 AC12TOE	Auto CMD12 timeout error. Indicates that no response is returned within 64 SD_CLK cycles after the end bit of the command. If this bit is set to1, then the other error status bits (2–4) are not valid. 0 No error 1 Time out
0 AC12NE	Auto CMD12 not executed. This bit is set to 1 when the eSDHC cannot issue the auto CMD12 to stop a memory multi-block data transfer due to some error. In case the memory multi-block data transfer is not started due to a command error, this bit is not set because it is not necessary to issue an auto CMD12. If this bit is set to 1, other error status bits (1-4) have no meaning. 0 Executed 1 Not executed

Table 23-24 shows error types associated with different auto CMD12 CRC error and auto CMD12 command timeout error bit settings.

Table 23-24. Command CRC Error and Command Timeout Error Bit Settings and Error Types

Auto CMD12 CRC Error Bit	Auto CMD12 Timeout Error Bit	Type of Error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

When issuing auto CMD12 commands, the eSDHC sets bits in the auto CMD12 error status register according to the following steps:

1. Before issuing an auto CMD12 command, the eSDHC sets the auto CMD12 not executed bit (bit 0) of the auto CMD12 error status register if the auto CMD12 cannot be issued due to an error in the previous command.
2. If the auto CMD12 is issued, then the auto CMD12 not executed bit (bit 0) is cleared.

3. At the end bit of an auto CMD12 response, the eSDHC checks errors corresponding to bits 0–4 (AC12IE, AC12CE, AC12EBE, and AC12TOE) and sets the bit if the corresponding error is detected.
4. Before reading the command not issued by auto CMD12 error bit (bit 7) the eSDHC sets bit 7 to 1 if there is a command that cannot be issued, and clears bit 7 if there is no command to issue.
5. Auto CMD12 errors and writes to the command register are asynchronous. After an auto CMD12 command, bit 7 is sampled when the driver is not writing to the command register. The driver can avoid problems by setting the AC12E bit in the interrupt status register before reading this register. An auto CMD12 error interrupt is generated when one of the error bits (0-4) is set to 1. The command not issued by auto CMD12 error does not generate an interrupt.

23.3.3.14 Host Controller Capabilities Register (HOSTCAPBLT)

This register provides the host driver with information specific to the eSDHC implementation. The value in this register is the same as at power-on reset, and does not change during a software reset. Any write to this register is ignored.

Figure 23-17 shows the register. Table 23-25 describes the register fields.

Offset 0x0040 (HOSTCAPBLT) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	VS18	VS30	VS33	SRS	DMAS	HSS	0	0	MBL[2:0]		
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-17. Host Controller Capabilities Register (HOSTCAPBLT)

Table 23-25. Host Capabilities Register Field Descriptions

Field	Description
31–27	Reserved.
26 VS18	Voltage support 1.8V. This bit's setting depends on the host system ability. 0 1.8V not supported 1 1.8V supported
25 VS30	Voltage support 3.0 V. This bit's setting depends on the host system ability. 0 3.0 V not supported 1 3.0 V supported

Table 23-25. Host Capabilities Register Field Descriptions (continued)

Field	Description
24 VS33	Voltage support 3.3 V. This bit's setting depends on the host system ability. 0 3.3 V not supported 1 3.3 V supported
23 SRS	Suspend/Resume support. This bit indicates whether the eSDHC supports Suspend/Resume functionality. If this bit is 0, the Suspend and Resume mechanism, as well as the Read Wait, are not supported, and the host driver does not issue either Suspend or Resume commands. 0 Not supported 1 Supported
22 DMAS	DMA support. This bit indicates whether the eSDHC is capable of using the internal DMA to transfer data between system memory and the data buffer directly. 0 DMA not supported 1 DMA supported
21 HSS	High-speed mode support. This bit indicates whether the eSDHC supports High-speed mode and the Host System can supply a SD clock frequency from 25 MHz to 50 MHz. 0 High-speed not supported 1 High-speed supported
20	Reserved
19	Reserved.
18–16 MBL[2:0]	Maximum block length. This value indicates the maximum block size that the host driver can read and write to the eSDHC's buffer. The buffer transfers blocks up to this size without wait cycles. 000512 bytes 0011024 bytes 0102048 bytes 0114096 bytes
15–0	Reserved.

23.3.3.15 Watermark Level Register (WML)

This register configures watermark levels (FIFO thresholds) and burst lengths for write and read. Watermark levels can range from 1–128 words; burst lengths can range from 1–31 words.

Figure 23-18 shows the register. Table 23-26 describes the register fields.

Offset 0x0044 (WML) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	WR_BRST_LEN[3:0]					WR_WML[7:0]							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	RD_BRST_LEN[3:0]					RD_WML[7:0]							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

Figure 23-18. Watermark Level Register (WML)

Table 23-26. Watermark Level Register Field Descriptions

Field	Description
31–29	Reserved.
28–24 WR_BRST_LEN[4:0]	Write burst length. The number of words the eSDHC writes in a single burst. The write burst length must not exceed the write watermark level, and all bursts within a watermark level transfer are in back-to-back mode. This field resets to 01000. The field cannot be cleared: writing 0 to this field results in the reset value 01000. Due to system restrictions, the actual burst length does not exceed 16.
23–16 WR_WML[7:0]	Write watermark level. The number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also the number of words in a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 127. The reset write watermark level is 16.
15–13	Reserved.
12–8 RD_BRST_LEN[4:0]	Read burst length. The number of words the eSDHC reads in a single burst. The read burst length must not exceed the read watermark level, and all bursts within a watermark level transfer are in back-to-back mode. This field resets to 01000. The field cannot be cleared: writing 0 to this field results in the reset value 01000. Due to system restrictions, the actual burst length does not exceed 16.
7–0 RD_WML[7:0]	Read watermark level. The number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also the number of words in a sequence of read bursts in back-to-back mode. The maximum legal value for the read watermark level is 127. The reset read watermark level is 16.

23.3.3.16 Force Event Register (FEVT)

The force event register is not a physically-implemented register, but rather an address where the interrupt status register can be written if the corresponding bit of the interrupt status enable register is set. Writing 1’s to this register sets the corresponding bits in the interrupt status register. This register is a write-only register: reads always return zero. Writing zero to the register has no effect.

When writing to this register to change status bits in the interrupt status register, the driver is responsible to ensure that the ipg_clk is active by setting the IPGEN bit in the system control register.

Figure 23-19 shows the register. Table 23-27 describes the register fields.

Offset 0x0050 (FEVT) Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	FEVTCINT			FEVTDMAE				FEVTAC12E		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTTCTOE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W									FEVTCNIBAC12E			FEVTAC12IE	FEVTAC12EBE	FEVTAC12CE	FEVTAC12TOE	FEVTAC12NE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23-19. Force Event Register (FEVT)

Table 23-27. Force Event Register Field Descriptions

Field	Description
31 FEVTCINT	Force event card interrupt. Writing 1 to this bit generates a short low-level pulse on the internal DAT[1] line, as if a self clearing interrupt was received from the external card. If enabled, the CINT bit is set and the interrupt service routine responds to this interrupt as a normal interrupt from the external card. It is not necessary for he interrupt service routine to poll the card interrupt factor, since the interrupt is self cleared.
30–29	Reserved.
28 FEVTDMAE	Force Event DMA Error. Forces the DMAE bit of interrupt status register to be set
27–25	Reserved.
24 FEVTAC12E	Force event auto CMD12 error. Forces the AC12E bit of interrupt status register to be set
23	Reserved.
22 FEVTDEBE	Force event data end bit error. Forces the DEBE bit of interrupt status register to be set
21 FEVTDCE	Force event data CRC error. Forces the DCE bit of interrupt status register to be set
20 FEVTDTOE	Force event data time out error. Force the DTOE bit of interrupt status register to be set

Table 23-27. Force Event Register Field Descriptions (continued)

Field	Description
19 FEVTCIE	Force event command index error. Forces the CCE bit of interrupt status register to be set
18 FEVTCEBE	Force event command end bit error. Forces the CEBE bit of interrupt status register to be set
17 FEVTCCE	Force event command CRC error. Forces the CCE bit of interrupt status register to be set
16 FEVTCCE	Force event command time out error. Forces the CTOE bit of interrupt status register to be set
15–8	Reserved.
7 FEVTCNIBAC12E	Force event command not executed by auto CMD12 error. Forces the CNIBAC12E bit in the Auto CMD12 Error Status Register to be set.
6–5	Reserved.
4 FEVTAC12IE	Force event auto CMD12 Index error. Forces the AC12IE bit in the auto CMD12 error status register to be set.
3 FEVTAC12EBE	Force event auto CMD12 end bit error. Forces the AC12EBE bit in the auto CMD12 error status register to be set.
2 FEVTAC12CE	Force event auto CMD12 CRC error. Forces the AC12CE bit in the auto CMD12 error status register to be set.
1 FEVTAC12TOE	Force event auto CMD12 time out error. Forces the AC12TOE bit in the auto CMD12 error status register to be set.
0 FEVTAC12NE	Force event auto CMD12 not executed. Forces the AC12NE bit in the auto CMD12 error status register to be set.

23.3.3.17 Host Controller Version (HOSTVER)

This register contains the vendor host controller version information. All bits are read-only with the same value as at power-on reset.

Figure 23-20 shows the register. Table 23-28 describes the register fields.

Offset 0x00FC (HOSTVER) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VVN								SVN							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Figure 23-20. Host Controller Version Register (HOSTVER)

Table 23-28. Host Controller Version Register Field Descriptions

Field	Description
31–16	Reserved.
15–8 VVN	Vendor version number. These status bits are reserved for the vendor version number. The host driver does not use this status. 0x00 Freescale eSDHC Version 1.0 0x10 Freescale eSDHC Version 2.0 0x11 Freescale eSDHC Version 2.1 0x12 Freescale eSDHC Version 2.2 others) Reserved
7–0 SVN	Specification version number. These status bits indicate the Host Controller Specification version. 0x00 SD Host Specification Version 1.0 All others) Reserved

23.4 Functional Description

The following subsections provide a brief functional description of the major system blocks, including the data buffer, DMA AHB interface, register bank and IP bus interface, dual-port memory wrapper, data/command controller, clock and reset manager and clock generator.

23.4.1 Data Buffer

The eSDHC uses a configurable data buffer to optimize data transfer between the system bus (IP Bus or AHB bus) in the master clock (hclk) domain, and the SD card in the IP peripheral source clock (ipg_perclk) domain.

Figure 23-21 shows the buffer scheme. The buffer is used as temporary storage for data being transferred between the host system and the card. The watermark levels for read and write are both configurable, from

1–128 words inclusive. The burst lengths for read and write are also configurable, from 1–16 words inclusive (the burst length field of the watermark level register can range from 1–31, but actual burst lengths greater than 16 are not supported).

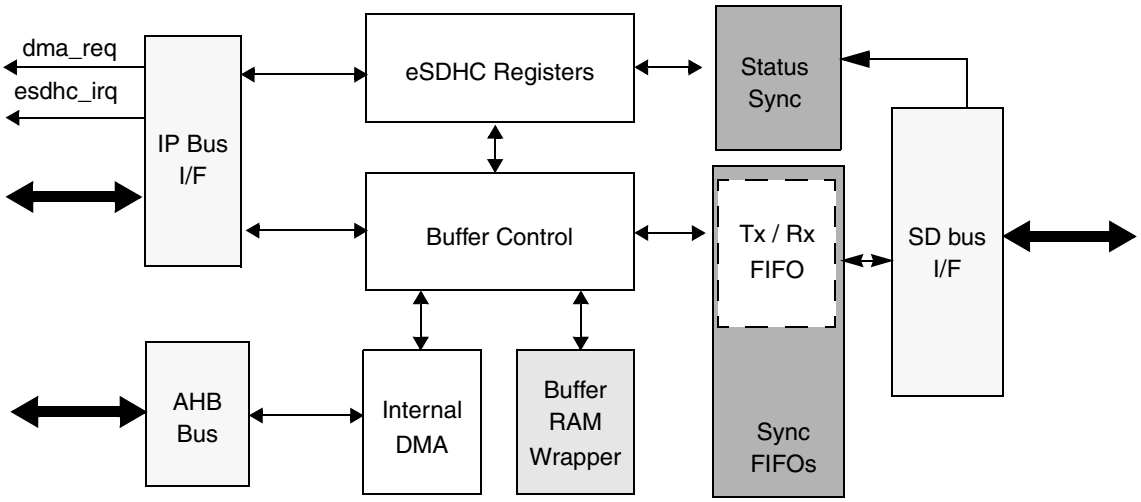


Figure 23-21. eSDHC Buffer Scheme

23.4.1.1 Data Buffer Transfer Modes

There are three transfer modes to access the data buffer:

- External DMA mode
- CPU polling mode
- Internal DMA mode

These transfer modes are explained in the following subsections

23.4.1.1.1 External DMA Mode

External DMA uses the IP bus. Using external DMA requires that DMAEN bit in the transfer type register be cleared when the DMA request is sent.

For read operations, the eSDHC sends an external DMA request (by asserting the signal esdhc_dreq_b to 0) when the number of words received in the buffer meets or exceeds the read watermark value (RD_WML in the watermark level register). The request is immediately negated when there is an access on the buffer data port register. If the number of words in the buffer after the current burst still meets or exceeds RD_WML value, the DMA request is reasserted, with one idle cycle between successive requests.

Write operations in external DMA mode are similar to read operations, except the write watermark value WR_WML is used instead of RD_WML to determine whether there are sufficiently many empty words in the buffer before the eSDHC sends the external DMA request.

23.4.1.1.2 CPU Polling Mode

CPU polling mode is similar to external DMA mode in that it uses the IP bus, and requires the DMAEN bit in the transfer type register be cleared.

CPU polling mode requires that the BRRIEN bit be set to 1 in the interrupt signal enable register. It differs from external DMA mode in that rather than relying on an external DMA request the host driver polls the BRR bit in the interrupt status register, as follows:

For read operations, when the number of words received in the buffer meets or exceeds the read watermark value (RD_WML field in the watermark level register) the eSDHC sets the BRR bit in the interrupt status register to 1 (this occurs simultaneously with sending the external DMA request). The host driver is responsible to poll the BRR bit, and when it detects that BRR is set it reads the buffer data port register to fetch RD_WML words from the buffer.

Write operation in CPU polling mode requires setting the BWRIEN to 1 in the interrupt signal enable register. Write operations are similar to read operations, except that the write watermark level WR_WML is used instead of RD_WML to determine whether there are sufficiently many empty words in the buffer, and the BWR bit is polled instead of BRR.

23.4.1.1.3 Internal DMA Mode

Internal DMA accesses utilize the AHB bus, and requires that the DMAEN bit in the transfer type register be set to 1 when the DMA request is sent. Unlike external DMA or CPU polling mode, the external DMA request is never sent out.

Internal DMA Read Operations

For internal DMA read operations, when the number of words in the buffer meets or exceeds the watermark level (RD_WML field in the watermark level register) the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8 (for burst lengths 4 and 8, respectively), the burst type is always INCR mode, and the burst length depends on the shortest of following factors:

- Burst length, configured in the burst length field of the watermark level register
- Watermark level boundary
- Remaining number of words in the current block
- 1-Kbyte address boundary defined in the AHB protocol

When internal DMA is used, if no error is encountered the eSDHC does not inform the system before the required number of bytes are transferred. When an error occurs during the data transfer, the eSDHC aborts the data transfer and abandon the current block. The host driver is responsible to read the contents of the DMA system address register to find the starting address of the abandoned data block. If the current data transfer is in multi block mode, the eSDHC does not automatically send CMD12, even though the AC12EN bit in the transfer type register is set. Instead, it is the host drivers responsibility to send CMD12 and restart the read operation from that address. It is recommended that a software reset for data be applied before the transfer is restarted after error recovery.

The eSDHC does not start data transmission until the number of words in the buffer meets or exceeds the read watermark level RD_WML. If the buffer is full and the host system does not read data in time, the eSDHC stops SD_CLK to avoid a data buffer overrun.

Internal DMA Write Operations

Write operations in internal DMA mode are similar to read operations, except that the write watermark level WR_WML is used to determine whether there are sufficiently many empty words in the buffer.The eSDHC does not start data transmission until the number of words set in the WR_WML register can be held in the buffer. If the buffer is empty and the host system does not write data in time, the eSDHC stops SD_CLK to avoid a data buffer under-run situation.

23.4.1.2 Data Addressing and Byte Ordering

Sequential and contiguous access is necessary to update the pointer address value correctly. Random or skipped access is not possible. On reset, byte ordering is set to little endian mode. The actually byte order is swapped inside the buffer according to the endian mode configured by software, as shown in Figure 23-22 and Figure 23-23. For a host write operation, byte order is swapped after data is fetched from the buffer and before it is sent to the SD bus. For a host read operation, byte order is swapped before the data is stored into the buffer.

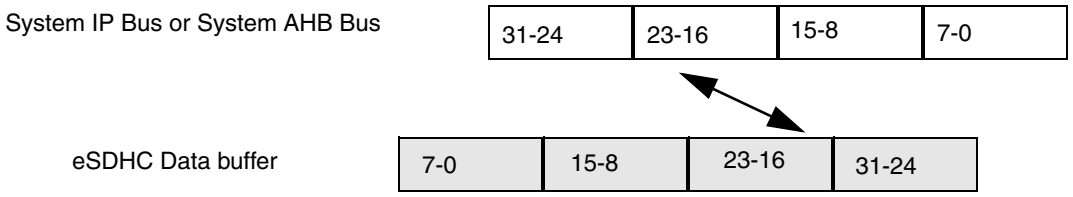


Figure 23-22. Data Swap between System Bus and eSDHC Data Buffer in Byte Little Endian Mode

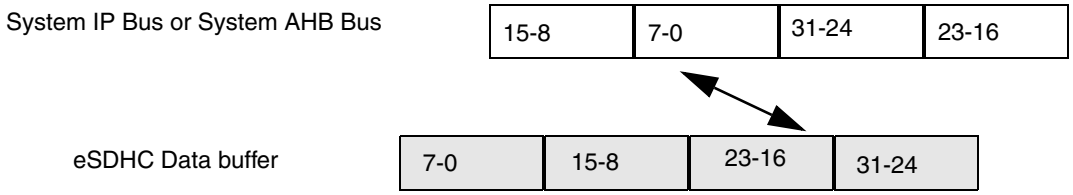


Figure 23-23. Data Swap Between System Bus and eSDHC Data Buffer in Half Word Big Endian Mode

23.4.1.3 Data Transfer Parameter Settings

In the eSDHC, the data buffer holds up to 127 4-byte words. The watermark levels for write and read can be configured from 1–127 words inclusive. For both read and write, the burst length, can be from 1–16 words inclusive (the burst length field in the watermark level ranges from 1–31, but actual burst lengths greater than 16 are not supported). The host driver is responsible to configure these values according to the system situation and requirements.

During a multi-block data transfer, the block size may be set to any value between 1 and 4096 bytes. However, additional restrictions may be imposed by the external card, which may not support large block sizes and/or partial block accesses.

For block size that are not multiples of four (that is, not word-aligned), the eSDHC requires stuff bytes at the end of each block (this is because the eSDHC treats each block individually). For example, if the block size is 7 bytes and the write is 12 blocks, the system side does 2 writes for each block, and the last byte for each block is discarded by eSDHC since it only sends 7 bytes to the card, then picks data from the following system write. Thus in this example there are 24 total beats required for the write.

23.4.1.3.1 Dividing Large Data Transfers

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multi-block transfer must be an integer multiple of the block size. If the total data length is not a multiple of the block size then there are two ways to transfer the data, depending on the function and the card design:

- The host driver splits the transaction, and fractional block portion of data is transferred using a second (single block) transfer.
- Dummy data is added to fill the last block. In this case, the card must manage the removal of the dummy data.

Figure 23-24 shows an example of a large data transfer using a WLAN SDIO card that only supports block sizes up to 64 bytes. In this case, 544 bytes are divided into eight 64-byte blocks plus 32 bytes, which are transferred using two separate CMD53 commands.

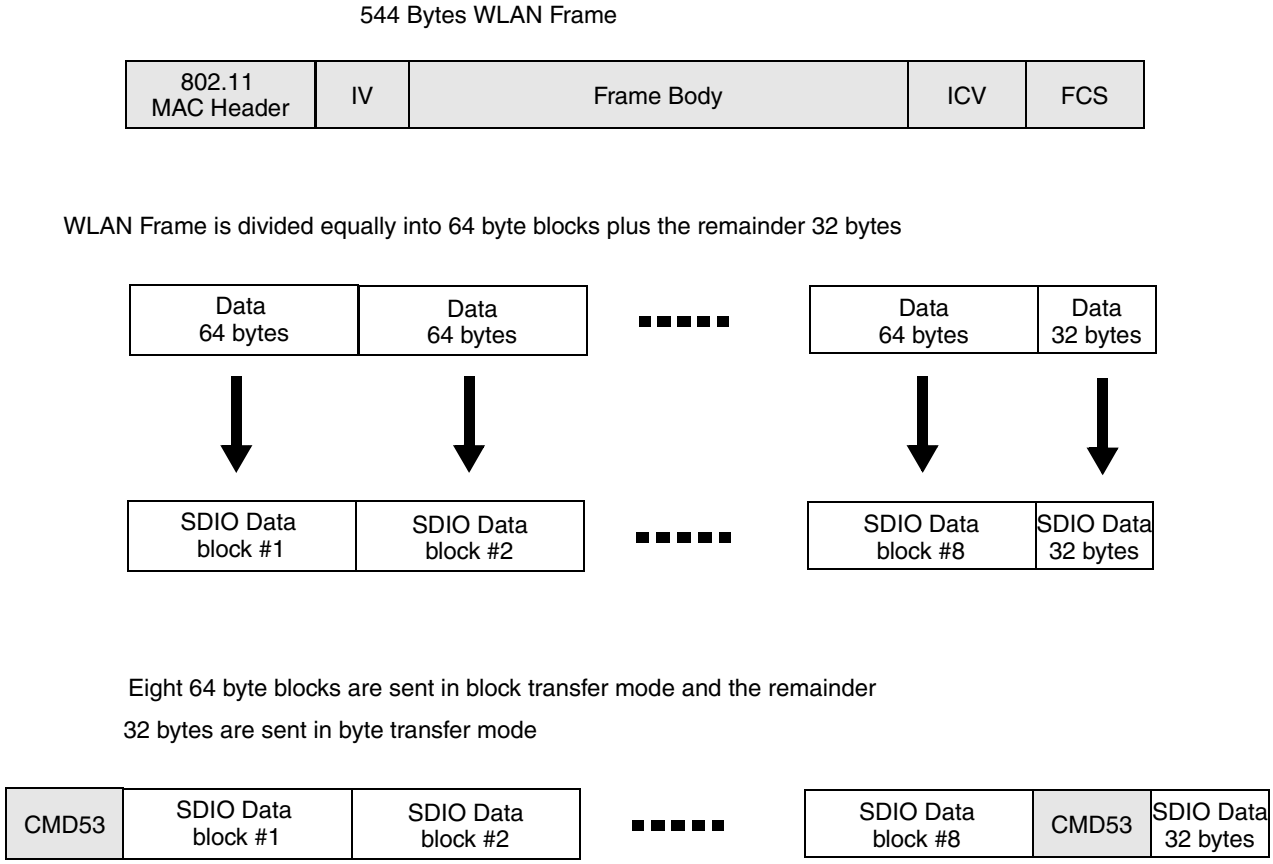


Figure 23-24. Example for Dividing Large Data Transfers

23.4.1.4 Data Transfer Parameters for External DMA Requests

[In external DMA transfer mode, since the DMA burst length cannot change during a data transfer it is necessary that the watermark level (read or write) must be a divisor of the block size (in word units): otherwise, transferring of the block may cause buffer underrun (for reads) or overrun (for writes). This implies for example that a block size of 512 bytes (128 words) requires that read and write watermark levels must be a power of two between 1 and 128 inclusive.

In CPU polling mode there are no such restrictions on the watermark levels, because the last access in the block transfer can be controlled by software. The watermark levels can even be larger than the block size (but no greater than 128 words). The actual number of bytes in each transfer is controlled by software, and does not exceed the block size.

Non-word–aligned block sizes are supported in CPU polling mode, as long as the card supports that block size. For example, the block size can be set at 31 bytes (if supported by the card), and the watermark level and burst length can take any allowed value. This is because the software transfers 8 words, and the eSDHC also sets the BRR or BWR bits (for reads or writes, respectively) when the remaining data satisfies watermark level conditions. Even though 8 words are transferred using the data port register, the eSDHC transfers only 31 bytes over the SD bus, as required by the BLKSIZE bits. In data transfers with non-word aligned block sizes, the endian mode should be set carefully, or invalid data can be transferred to/from the card.

23.4.1.5 Data Transfer Parameters for Internal DMA Requests

Internal DMA data transfers are in block units, and the host driver is responsible to set the watermark level as the minimum remaining number of words remaining in a block following a series of burst transfers. For instance, consider a multi-block read with block size 31 bytes and burst length set to 6 (4-byte) words. After the first burst transfer, there are 7 (= 31 – 4*6) bytes remaining to be read from the first block: the remaining data occupies 2 words. The host driver sets the read watermark level as 2 words, so that another DMA request is sent if there are 2 or more words in the buffer (which might contain some data from the next block). The eSDHC reads 2 words out of the buffer, which include 7 valid bytes and 1 stuff byte.

The DMA burst length for the internal DMA engine can be from 1 to 16 words inclusive, just as in CPU polling mode. The actual burst length for the DMA depends on the smaller of the configured burst length and the remaining words of the current block. In the above example with block size 31 bytes and burst length set to 6, the actual burst lengths alternate between 6 and 2 words. The host driver is responsible to take this variable burst length into account. One option is to configure the burst length as a divisor of the block size, so that the burst length does not need to vary.

23.4.2 DMA AHB Interface

Figure 23-25 shows the DMA AHB interface block.The internal DMA AHB interface includes a DMA engine and the AHB master.

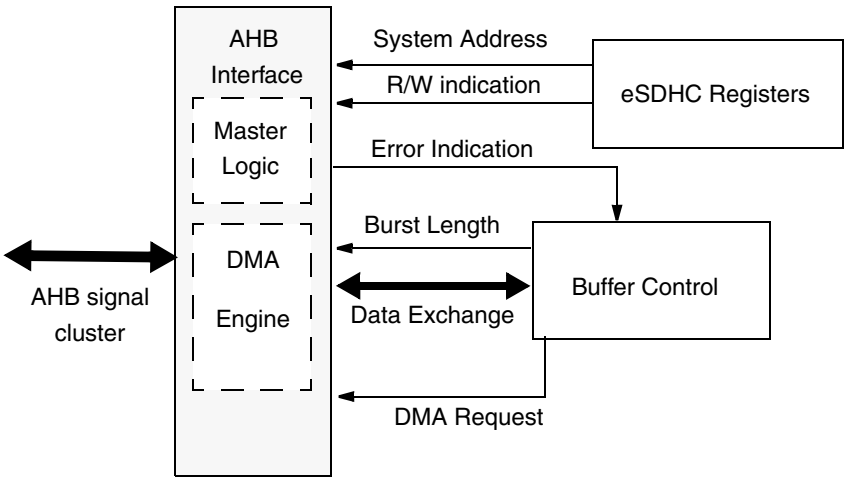


Figure 23-25. DMA AHB Interface Block

23.4.2.1 Internal DMA Requests

Internal DMA transfers are enabled by setting the DMAEN bit in the transfer type register. The external DMA request signal (esdhc_dreq_b) is disabled: however, the BRR and BWR bits in the interrupt status register are still valid, as long as the BRRSEN and BWRSEN bits have been set in the interrupt status enable register. See Section 23.4.1.1.3, “Internal DMA Mode,” for more details about internal DMA transfers.

If the watermark level requirement is met, the data buffer block sends a DMA request to the AHB interface. There is a delay in the internal DMA engine’s response that depends on the system AHB bus loading and the priority assigned to the eSDHC. The DMA engine does not respond to a request during burst transfers, but is ready to serve as soon as the burst is over. The data buffer negates the request after the buffer is accessed. Upon access to the buffer by internal DMA, the data buffer updates its internal buffer pointer, and if the watermark level requirement is satisfied another DMA request is sent.

23.4.2.2 AHB Master Interface

If the internal AHB DMA engine fails during the data transfer, the following actions occur:

- The DMA engine stops the transfer and goes to the idle state
- The internal data buffer stops accepting incoming data.
- The DMAE bit in the interrupt status register is set to inform the driver.

After the DMAE interrupt is received, the software sends a CMD12 to abort the current transfer and reads the DS_ADDR bits of the DMA system address register to get the starting address of the corrupted block. After the DMA error is fixed, the software applies a data reset and restarts the transfer from this address to recover the corrupted block.

23.4.3 Register Bank Access using IP Bus Interface

The IP bus-accessible registers are contained in the register bank. Figure 23-26 is a block diagram of the register bank. Only 32-bit accesses are allowed, and no partial read/writes are supported. All accesses are word aligned (the lowest two address bits are tied to 0).

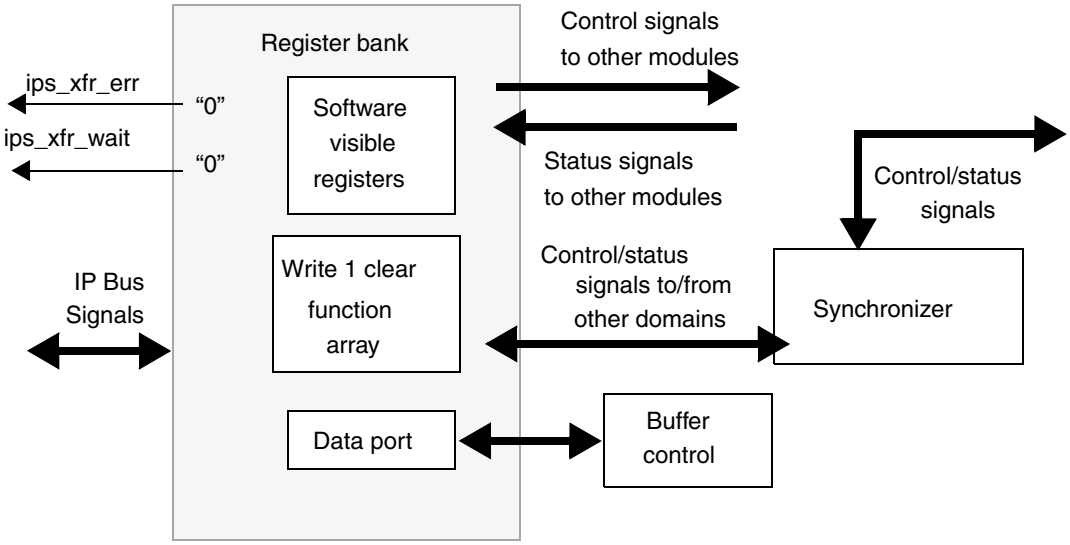


Figure 23-26. Register Bank Diagram

23.4.4 SD Protocol Unit

The SD protocol unit deals with all SD protocol affairs, and performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the DAT[0] line
- Monitors interrupts from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when the buffer announces danger status
- Detects the write-protect state

The SD protocol unit consists of four submodules:

- SD transceiver
- SD clock and monitor
- Command agent
- Data agent

23.4.4.1 SD Transceiver

The transceiver is the main control submodule in the SD protocol unit. It consists of a finite state machine (FSM) and a control module, from which the control signals for the other three submodules are generated.

23.4.4.2 SD Clock and Monitor

This submodule monitors the signal level on all 8 data lines and the command lines, and directly routes the level values into the register bank. The driver can use these values for debug purposes.

This submodule detects the card detection (CD) line as well as the DAT[3] line. The transceiver reports the card insertion state according to the CD state, or the signal level on the DAT[3] line, when the D3CD bit in the protocol control register is set.

In addition this submodule detects the write protect (WP) line, and informs the register bank when the WP switch is on. When the WP switch is on, the register bank ignores commands involving a write operation.

If the internal data buffer is in danger of overrun or underrun, this submodule asserts the gates off the SD clock. The SD clock is gated on again when the system access of the buffer catches up.

This submodule also drives the LED control output signal (SD_LCTL) when the LCTL bit is set by the driver.

23.4.4.3 Command Agent

The command agent deals with the transactions on the CMD line. Figure 23-27 shows the command CRC shift register.

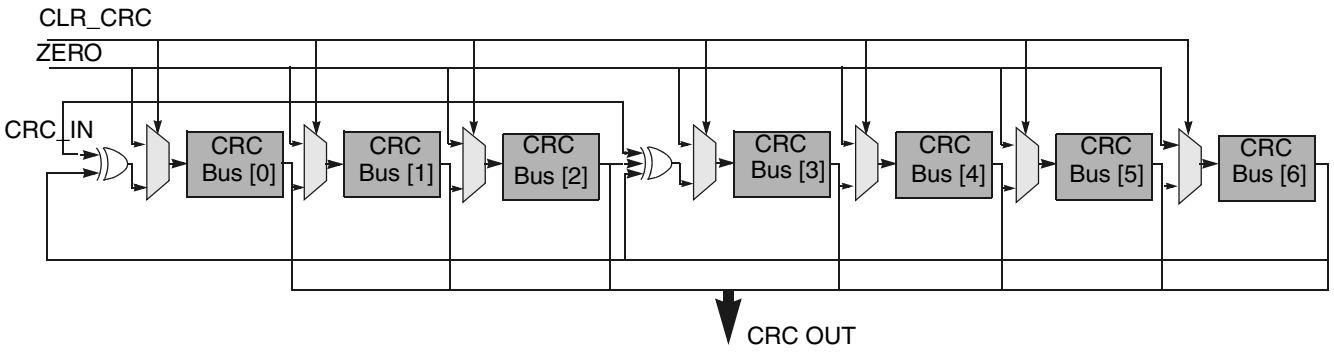


Figure 23-27. Command CRC Shift Register

The CRC polynomial for CMD is computed as follows:

$$\text{Generator polynomial: } G(x) = x^7 + x^3 + 1$$
$$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit}) * x^0$$
$$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$$

23.4.4.4 Data Agent

The data agent deals with the transactions on the eight data lines. This module also detects the busy state from the DAT[0] line, and generates the read wait state when requested by the transceiver.

The CRC polynomials for DAT are computed as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$
$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

23.4.5 Clock and Reset Manager Submodule (CRM)

The CRM controls all the reset signals within the eSDHC, which can be classified as four types:

- Hardware reset
- Software reset for all
- Software reset for the data part
- Software reset for the command part

Reset signals are fed into the CRM, and stable signals are generated inside the CRM to reset all other modules.

The CRM also monitors the activities of all other eSDHC submodules, supplies the clocks for them and (when enabled) automatically gates off the corresponding clocks. Clocks used by the eSDHC include IPG clock (ipg_clk), peripheral source clock (ipg_perclk), and master clock (hclk).

23.4.6 SD Clock Generator

The SD clock generator generates the SD clock (SD_CLK) signal from the peripheral source clock by dividing in two stages. Refer to Figure 23-28 for the structure of the divider.

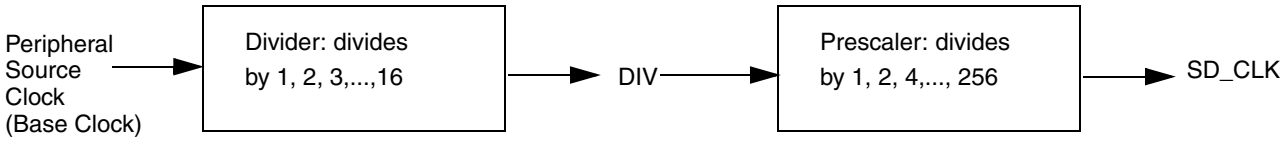


Figure 23-28. Two Stages of the Clock Divider

The SD_CLK signal which is output from the clock divider drives the clocks for all submodules of the SD protocol unit, and for the sync FIFOs to synchronize with the data rate in the internal data buffer (see Figure 23-21).

The divider and prescaler values are determined by the DVS and the SDCLKF bits in the system control register. For example, if the peripheral source clock frequency (ipg_perclk) is 96 MHz, and the target SD_CLK frequency is 25 MHz, then choosing SDCLKF as 0x01 (divide by 2) and DVS as 0x1 (divide by 2) yields an SD_CLK frequency 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to obtain a SD_CLK frequency of 400 kHz, SDCLKF = 0x08 and DVS = 0xE yields the exact clock value of 400 kHz. The highest SD_CLK frequency is equal to the base clock's (peripheral source clock); the second highest frequency is base/2; the lowest frequency is base/4096. The ipg_perclk is about 96 MHz: the reset values of SDCLKFS and DVS correspond to divides of 256 and 1 respectively, so the SD clock frequency after reset is 375 kHz.

According to the SD Physical Specification Version 1.1 and the SDIO Card Specification Version 1.2, the maximum SD clock frequency is 50 MHz and must never exceed this limit.

If the peripheral source clock has duty cycle of 50% (which is usually true), then the duty cycle of SD_CLK is also 50%.

23.4.7 SDIO Card Interrupts

23.4.7.1 Interrupts in 1-Bit Mode

In 1-bit mode the DAT[1] signal is dedicated to providing the interrupt function. An interrupt is asserted by pulling DAT[1] low from the SDIO card, until the interrupt service routine clears the interrupt.

23.4.7.2 Interrupts in 4-Bit Mode

In 4-bit mode the interrupt and data line 1 share pin 8. To accommodate this sharing, the eSDHC treats pin 8 as an interrupt only during specified time intervals known as interrupt periods. At all other times, the level on pin 8, is treated as a data signal.

The definition of the interrupt period is different for single-block and multi-block data transfers:

- For single-block transfers, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until the card has received the end bit of the next command that has a data block transfer associated with it.
- For multi-block data transfers, the interrupt period is limited to two clock cycles, due to the limited time between blocks. The interrupt period begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DAT[1] line is held low for one clock cycle with the last clock cycle pulling DAT[1] high. On completion of the interrupt period, the card releases the DAT[1] line into the high Z state. The eSDHC samples the DAT[1] during the interrupt period when the IABG bit in the protocol control register is set.

Refer to SDIO Card Specification v1.10f for further information about SDIO card interrupts.

23.4.7.3 Card Interrupt Handling

Before servicing a card interrupt, the host driver is responsible to clear the card interrupt interrupt enable (CINTIEN) bit in the interrupt signal enable register. This prevents inadvertent interrupts from occurring while the interrupt is being serviced. After all interrupt requests from the card are cleared, it is the host driver’s responsibility to reset this bit to 1.

The SDIO status bit is cleared by resetting the SDIO interrupt. Writing to this bit would have no effects. In 1-bit mode, the eSDHC will detect the SDIO interrupt with or without the SD clock (to support wake-up). In 4-bit mode, the interrupt signal is sampled during the interrupt period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status has been set, and the host driver needs to start the interrupt service routine, the SDIO bit in the Interrupt Control Register of the SDIO card is cleared. This is required to clear the SDIO interrupt status latched in the eSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The host driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO interrupt enable bit is set to 1, and the eSDHC starts sampling the interrupt signal again.

Figure 23-29 shows the system's interrupt-handling features, and also shows a flowchart that summarizes the interrupt detection and handling procedure.

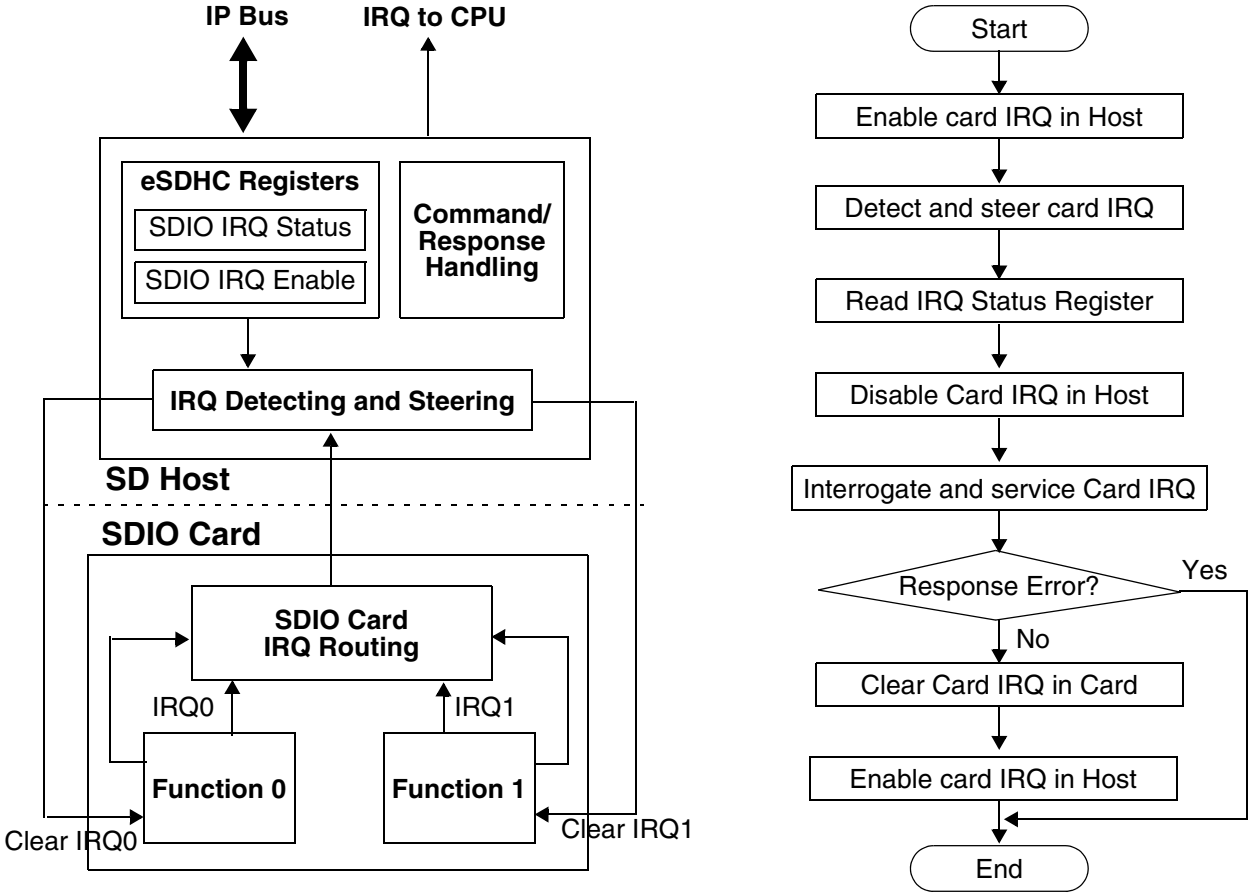


Figure 23-29. Card Interrupt-Handling Features and Card Interrupt-Detection and Handling Procedure

23.4.8 Card Insertion and Removal Detection

The eSDHC uses either the DAT[3] or the SD_CD signal to detect card insertion or removal. When there is no card on the MMC/SD bus, the DAT[3] is pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the eSDHC detects the logic value changes on the DAT[3] pin and generates an interrupt. Whether DAT[3] signal is used for card detection or not, the SD_CD pin must be connected for card detection. It may be implemented by a GPIO. The SD_CD pin is always used as a reference for card detection. If either the DAT[3] or SD_CD signal reports card inserted, the eSDHC informs the host system that a card is inserted and an interrupt is sent if it is enabled.

23.4.9 Power Management and Wake-up Events

The eSDHC offers a power management feature: by clearing the PEREN, HCKEN or IPGEN bits in the system control register, the peripheral source clock, master clock, or IPG clocks are gated in the low position to the eSDHC. For maximum power saving, the user can disable all the clocks to the eSDHC when no operation is in progress.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- The internal data buffer is empty

Even when the system is in low power mode and the clocks to the eSDHC are disabled, there are some events for which the clocks must be enabled to handle the event. There are three events (denoted as wake-up events or wake-up interrupts) that can be used to wake up the eSDHC by re-enabling the clocks, even if there are no clocks enabled. These three wake-up interrupts are:

- Card removal interrupt
- Card insertion interrupt
- Interrupt from the SDIO card

In order to enable these interrupts to wake up the eSDHC, the corresponding wake-up enabled bits need to be set in the protocol control register, before the CPU enters sleep mode. See Section 23.3.3.8, “Protocol Control Register (PROCTL)” for more details.

23.5 Initialization/Application Information

All communication between system and cards is controlled by the host. See Section 23.6, “MMC/SD/SDIO/CE-ATA Card Commands for a complete list of the commands that the host can issue. There are two types of commands:

- Broadcast commands are intended for all cards: examples include “GO_IDLE_STATE”, “SEND_OP_COND”, “ALL_SEND_CID” and so on. In broadcast mode, all cards are in open-drain mode to avoid bus contention.
- Addressed commands can be issued after the broadcast command CMD3 has been sent, causing the cards to enter standby mode. In standby mode, the CMD and DAT I/O pads are in push-pull mode, which provides the driving capability for maximum frequency operation.

23.5.1 Command Send and Response Receive Basic Operation

The following pseudocode indicates the procedure for sending a command to the card(s). The data type WORD here denotes unsigned 32-bit integer.

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into transfer type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) >> 24; // set the first 8 bits as ‘00’+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
        set MSBSEL bit;
        if (finite block number) {
                set BCEN bit;
```

```
                if (auto12 command is to use) set AC12EN bit;
        }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set transfer type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
}
```

In this example the function wait_for_response is implemented by means of polling for the sake of simplicity. For an effective and formal way, the response is checked after the command complete interrupt is received. By doing this, the corresponding interrupt status bits are verified.

In some scenarios, a response timeout is expected after a command is issued. For example, after the host issues a CMD3 command and all cards enter the standby state, then when CMD2 is sent the cards send no response to the host. The host driver is responsible to deal with these ‘fake’ errors.

23.5.2 Card Identification Mode

When a card is inserted into the socket, or when the card is reset by the host, the host is responsible for performing a sequence of operations on the card including the following:

- Card detection (when the card is inserted) or card reset (when the card is reset by the host)
- Voltage validation (also determines whether the card is MMC, SD, SDIO, or CE-ATA)
- Card registration (including card identification and determination of relative card address (RCA))

23.5.2.1 Card Detection

Figure 23-30 shows a flowchart for the detection of MMC, SD and SDIO cards using the eSDHC.

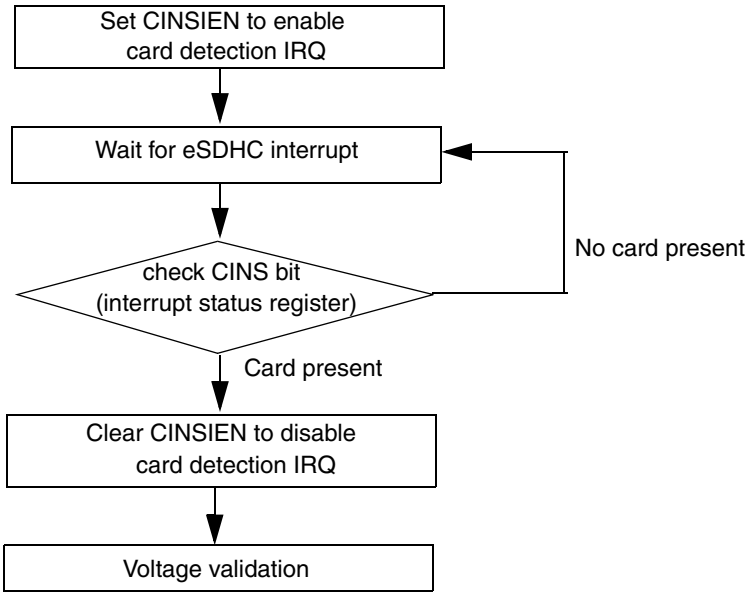


Figure 23-30. Flow Diagram for Card Detection

23.5.2.2 Reset

There are three types of resets involving the eSDHC:

- Hardware reset (card and host), which is driven by POR (power on reset)
- Software reset (host only). There are three types of software reset, which are effected by writes to bits in the system controller register as follows:
  - Setting the RSTD bit to 1 resets the data part of the eSDHC
  - Setting the RSTC bit to 1 resets the command part of the eSDHC
  - Setting the RSTA bit to 1 resets all parts of the eSDHC.
- Card reset (Card Only). The command, “Go_Idle_State” (CMD0), is the software reset command for all types of MMC, SD Memory, and CE-ATA cards. This command sets each card into the idle state regardless of the current card state. For SD I/O Cards, CMD52 is used to write an I/O reset in the CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host is responsible to validate the voltage range of the card. See Figure 23-31 for pseudocode to reset both the eSDHC and the card.

For CE-ATA devices that support ATA mode, two CMD39 commands should be issued back-to-back to the ATA control register before issuing CMD0 to reset the MMC layer. The first CMD39 has the SRST bit set to one, and the second has the SRST bit cleared.

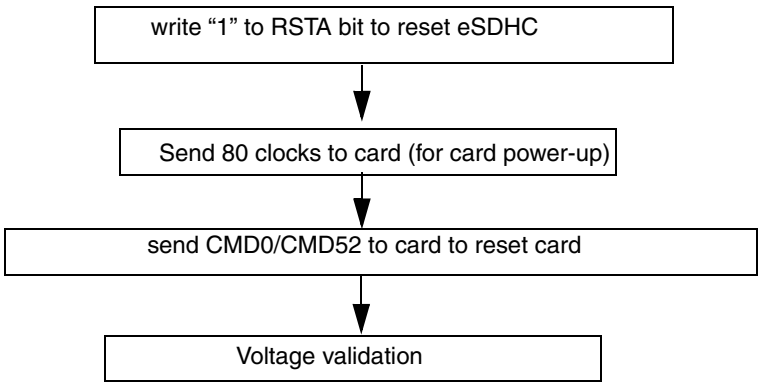


Figure 23-31. Flowchart for Reset of the eSDHC and SD I/O Card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the host
set DTOCV and SDCLKFS bit fields to get the SD_CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

23.5.2.3 Voltage Validation

The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the V_{dd} range(s) desired by the host.

A card’s supported minimum and maximum values for V_{dd} are defined in the card’s operating conditions register (OCR).The supported range may not cover the maximum allowed voltage range specified in the card specification. Cards that store the card identification (CID) and card-specific data (CSD) in preloaded memory are only able to communicate this information under data transfer V_{dd} conditions: so if the host and card have non-overlapping V_{dd} ranges, then the card is not able to complete the identification cycle, nor is it be able to send CSD data.

Voltage validation makes use of the special commands listed in Table 23-29 as follows:

- The host can send a command in Table 23-29 with the desired V_{dd} voltage window as operand. Then cards of the corresponding type that cannot perform the data transfer in the specified range disqualify themselves from further bus operations, and go into the inactive state.
- Alternatively, the host can send a command in Table 23-29 and omits the voltage range in the command. By this means the host can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query can be used to enable the host to select a common voltage range, or to enable notification of the system when an unusable card is detected in the stack.

Table 23-29. Voltage Validation Commands for Different Card Types

Card Type	Command Name	CMD #
MMC, CE-ATA	Send_Op_Cont	CMD1
SD	SD_Send_Op_Cont	ACMD41
SDIO	IO_Send_Op_Cont	CMD5

The following pseudocode outlines the voltage validation procedure when a card is inserted:

```
voltage_validation(voltage_range_argument)
{
label the card as UNKNOWN;
send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
operating voltage, command argument is zero
if (RESP_TIMEOUT != wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted
        if (0 < number of IO functions) {
                label the card as SDIO;
                IORDY = 0;
                while (!(IORDY in IO OCR response)) { // set voltage range for each IO function
                        send_command(IO_SEND_OP_COND, <voltage range>, <other parameter>);
                        wait_for_response(IO_SEND_OP_COND);
                } // end of while ...
        } // end of if (0 < ...
        if (memory part is present inside SDIO card) Label the card as SDCombo; // this is an
SD-Combo card
} // end of if (RESP_TIMEOUT ...
if (the card is labelled as SDIO card) return; // card type is identified and voltage range is
set, so exit the function;
send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application specific CMD
prefix
if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55 is accepted
        send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set voltage range
for memory part or SD card
        wait_for_response(SD_APP_OP_COND); // voltage range is set
        if (card type is UNKNOWN) label the card as SD;
        return; //
} // end of if (no error ...
else if (errors other than time-out occur) { // command/response pair is corrupted
        deal with it by program specific manner;
} // of else if (response time-out
else { // CMD55 is refused, it must be MMC card or CE-ATA card
        if (card is already labelled as SDCombo) { // change label
                re-label the card as SDIO;
                ignore the error or report it;
                return; // card is identified as SDIO card
        } // of if (card is ...
        send_command(SEND_OP_COND, <voltage range>, <...>);
        if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted, either
                label the card as UNKNOWN;
                return;
        } // of if (RESP_TIMEOUT ...
        if (check for CE-ATA signature succeeded) { // the card is CE-ATA
                store CE-ATA specific info from the signature;
                label the card as CE-ATA;
```

```
        } // of if (check for CE-ATA ...
        else label the card as MMC;
} // of else
}
```

The host detects a CE-ATA device by issuing a CMD 60 to check for a CE-ATA signature, after completing all other MMC voltage validation and identification steps. If the device responds to CMD 60 with the CE-ATA signature, then a CE-ATA device has been found.

It is possible that the CE-ATA device does not support the ATA mode, so the driver cannot issue ATA command to the device. In order to check this, the driver queries the EXT_CSD register byte 504 (S_CMD_SET) in the MMC register space. If the ATA bit (bit 4) is set, then ATA commands are supported, and the driver sets the ATA bit (bit 4) of the EXT_CSD register byte 191 (CMD_SET) to activate the ATA command set for use. To choose the ATA command set, the driver issues CMD6.

23.5.2.4 Card Identification and Registration

The card identification and registration process establishes a relative card address (RCA), which is used to address the card for future data transfer operations. The registration processes for different card types are described in Section 23.5.2.4.1, “Card Identification and Registration for MMC Cards,” Section 23.5.2.4.2, “Card Registration for SD, SDIO, and SD Combo Cards,” and Section 23.5.2.4.3, “Card Identification and Registration for CE-ATA Cards,” respectively. Section 23.5.2.4.4, “Card Registration Pseudocode,” outlines the entire registration process.

23.5.2.4.1 Card Identification and Registration for MMC Cards

For MMC cards, the relative card address (RCA) is set by the host. The procedure for identifying the cards and setting their RCAs is described below.

Card identification mode for MMC cards uses a clock frequency of less than 400 kHz and a power voltage greater than 2.7 V. The process begins in open-drain mode: the open-drain driver stages on the CMD line allow parallel card operation during card identification. The host sends the following sequence of commands:

1. After the bus is activated, the host broadcasts a CMD1 command requesting the cards to send their valid operating conditions. The response to CMD1 is the “wired OR” operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the inactive state.
2. The host then broadcasts an All_Send_CID (CMD2) command, asking all cards for their unique card identification (CID) number. All unidentified cards (in ready state) simultaneously start sending their CID numbers serially, while bitwise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately, and must wait for the next identification cycle. Since the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into the identification state.
3. Next, the host issues a point-to-point Set_Relative_Addr command (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received, the card state changes to the standby state, and the card does not participate in further identification cycles. The card’s output driver switches from open-drain to push-pull.

4. The host repeats steps 2 and 3 above until the host sends a CMD2 which times out, signifying the completion of the identification process.

23.5.2.4.2 Card Registration for SD, SDIO, and SD Combo Cards

For SD, SDIO, and SD Combo cards, the RCA is published by the cards at the host's request. The procedure for identifying the cards and setting their RCAs is described below.

Card identification mode for SD cards uses a clock frequency of less than 400 kHz and a power voltage greater than 2.7 V (as defined in the SD card specification). The CMD line output drivers are in push-pull mode. The host sends the following sequence of commands:

1. After the bus is activated, the host broadcasts a ACMD41 (for SD) or CMD5 (for SDIO) command to all new cards in the system, requesting them to send their valid operating conditions. The card responds by sending the contents of its operating conditions register. Incompatible cards are put into the inactive state.
2. The host then broadcasts an All_Send_CID (CMD2) command, asking all cards for their unique card identification (CID) number. All unidentified cards (in ready state) simultaneously start sending their CID numbers serially, while bitwise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately, and must wait for the next identification cycle. Since the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into the identification state.
3. Next, the host issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. Once the RCA is received, the card changes its state to the standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.
4. The host repeats steps 2 and 3 above until the host sends a CMD2 which times out, signifying the completion of the identification process.

23.5.2.4.3 Card Identification and Registration for CE-ATA Cards

CE-ATA operation has the same interface as MMC cards. However, CE-ATA devices are connected in a point-to-point manner, and no RCA is needed. All data communications in card identification mode use the command line (CMD) only. See CE-ATA Digital Protocol, Revision 1.1 for more details.

CE-ATA enters the tran state after reset is completed.

23.5.2.4.4 Card Registration Pseudocode

The following pseudocode outlines the card registration process for all cards (corresponding to steps 2 through 4 in the procedures shown in Section 23.5.2.4.1, "Card Identification and Registration for MMC Cards," and Section 23.5.2.4.2, "Card Registration for SD, SDIO, and SD Combo Cards").

```
card_registry()
{
do { // decide RCA for each card until response time-out
        if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
```

```
                send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to publish its
RCA
                retrieve RCA from response;
        } // end if (card is labelled as SDCombo ...
        else if (card is labelled as SD) { // for SD card
                send_command(ALL_SEND_CID, <...>);
                if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                send_command(SET_RELATIVE_ADDR, <...>);
                retrieve RCA from response;
        } // else if (card is labelled as SD ...
        else if (card is labelled as MMC or CE-ATA) { // treat CE-ATA as MMC
                send_command(ALL_SEND_CID, <...>);
                rca = 0x1; // arbitrarily set RCA, 1 here for example, this RCA is also the
relative address to access the CE-ATA card
                send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper 16
bits
        } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

23.5.3 Card Accesses

This section describes write, read, suspend or resume, and ADMA card accesses.

23.5.3.1 Block Write

Normal writes and writes with pause are described in Section 23.5.3.1.1, “Normal Write,” and Section 23.5.3.1.2, “Write with Pause,” respectively.

23.5.3.1.1 Normal Write

During a block write (CMD24–27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card indicates the failure on the DAT line. The transferred data is discarded and not written, and all further transmitted blocks (in multi-block write mode) is ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set, and the CE-ATA card does not support partial block write, either), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, while ignoring all further data transfer, waits in the Receive-data-State for a stop command. For a CE-ATA card, check the CE-ATA card specification for its behavior in block misalignment. The write operation is also aborted if the host tries to write over a write protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block size setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and not change any register contents.

Different cards may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DAT line low if its

write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO and CE-ATA cards at any time, and the card responds with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby State and release the DAT line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling DAT to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   a) For SD/MMC cards, use SET_BLOCKLEN (CMD16)
   b) For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1–7)
   c) For CE-ATA cards, configure bits 1~0 in the scrControl register
3. Set the eSDHC block size register to be the same as the block size set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the eSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the transfer complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

23.5.3.1.2 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the SD_CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition during a write operation to the data blocks, a write to all types of cards can be paused in this way, and if the DAT0 line is not required to negate to release the busy state, no suspend command is needed.

Like in the flow described in Section 23.5.3.1.1, “Normal Write,” the write with pause is shown with the same kind of write operation:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a) For SD/MMC, use SET_BLOCKLEN (CMD16)

b) For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1–7)

c) For CE-ATA cards, configure bits 1:0 in the scrControl register

3. Set the eSDHC block size register to be the same as the block size set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the eSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the transfer complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the transfer complete interrupt.
12. Check the status bit to see if a write CRC error occurred, or some another error, that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. The data transfer and the setting of the SABGREQ bit are concurrent, and the delay of the register read and writing, the actual number of blocks left may not be exactly the value read earlier. The driver reads the value of BLKCNT after the transfer is paused and the transfer complete interrupt is received.

It is also possible the last block has begun when the stop at block gap request is sent to the buffer. In this case, the next block gap is actually the end of the transfer. These types of requests are ignored and the driver should treat this as a non-pause transfer and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the Host System is not stopped, and the transfer is active until the data buffer is full. Because of this (if not needed), it is recommended to avoid using the Suspend Command for the SDIO card. This is because when such a command is sent, the eSDHC thinks the System switches to another function on the SDIO card, and flush the data buffer. The eSDHC takes the Resume Command as a normal command with data transfer, and it is left for the driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the transfer type register are set as well as the AC12EN bit. However, the eSDHC automatically sends a CMD12 to mark the end of the multi-block transfer.

23.5.3.2 Block Read

Normal reads and reads with pause are described in Section 23.5.3.2.1, “Normal Read,” and Section 23.5.3.2.2, “Read with Pause,” respectively.

23.5.3.2.1 Normal Read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data

transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer State. For multi-block read, data blocks is continuously transferred until a stop command is issued.

The software flow to read from a card incorporates the internal DMA, and the read operation is a multi-block read with the Auto CMD12 enabled. For the other two methods (by means of external DMA or CPU polling status) with different transfers, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a) For SD/MMC, use SET_BLOCKLEN (CMD16)
   b) For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1–7)
   c) For CE-ATA cards, configure bits 1:0 in the scrControl register
3. Set the eSDHC block size register to be the same as the block size set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the eSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set:
6. Wait for the transfer complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

23.5.3.2.2 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the read wait feature can pause during the read operation. If the SDIO card supports read wait (SRW bit in CCCR register is 1), the driver can set the SABGREQ bit in the protocol control register to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the RWCTL bit in the protocol control register is set, otherwise the eSDHC does not assert the read wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit once the Read Wait capability of the SDIO card is recognized.

As in the flow described in Section 23.5.3.2.1, “Normal Read,” the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm the card supports read wait.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   a) For SD/MMC, use SET_BLOCKLEN (CMD16)

b) For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1–7)

c) For CE-ATA cards, configure bits 1~0 in the scrControl register

5. Set the eSDHC block size register to be the same as the block size set for the card in Step 2.
6. Set the eSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the eSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set
8. Set the SABGREQ bit.
9. Wait for the transfer complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.
12. Set the CREQ bit to continue the read operation.
13. Wait for the transfer complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or some another error, occurred during the auto12 command sending and response receiving.

Like the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the eSDHC ignores the stop at block gap request and treat it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause is transferred to the host system. Whether the Suspend command is sent or not, the internal data buffer is not flushed.

If the Suspend command is sent and the transfer is later resumed by means of a Resume command, the eSDHC takes the command as a normal one accompanied with data transfer. It is left for the driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the transfer type register are set, as well as the AC12EN bit. However, the eSDHC automatically sends the CMD12 to mark the end of multi-block transfer.

23.5.3.3 Suspend/Resume Operations

The eSDHC supports suspend and resume operations for SDIO cards, although the implementation of suspend is slightly different than that suggested in the SDIO Card Specification.

23.5.3.3.1 Suspend Operation

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of the SDIO card. The eSDHC does not monitor the content of the response, so it doesn't know if the Suspend command succeeded or not. Accordingly, it does not negate read wait for a read pause. To solve this problem, the driver first sends the command as if it were a normal command (CMDTYP = 01). After the command succeeds, and the BS bit is set to 1 in the response, the driver then sends another command marked as Suspend to inform the eSDHC that the current transfer is suspended. The following sequence is used for the Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.

2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field is set as 0b00.
3. Check the BS bit of the CCCR in the response. If it is set, repeat this step until the BS bit is cleared or abandon the Suspend operation according to the driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 0b01, to inform the eSDHC that the paused operation has successfully suspended. If the paused transfer is a read operation, the eSDHC stops driving DAT2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA system address register (for internal DMA operation), and the block attribute register.
6. Begin operation for another function on the SDIO card.

23.5.3.3.2 Resume Operation

Data transfer is resumed following a Suspend command according to the following procedure:

1. Resume the suspended function by restoring the context register with the value saved in step #5 of the Suspend operation (see Section 23.5.3.3.1, “Suspend Operation”).
2. Send the Resume command. In the transfer type register, all bit fields are set to the value as if this were another ordinary data transfer, instead of a transfer resume (except the CMDTYP field is set to 0b10).
3. If the Resume command has responded, the data transfer is resumed.

23.5.3.4 Handling of Transfer Errors

23.5.3.4.1 CRC Error

It is possible at the end of a block transfer, that a write CRC status error or read CRC error occurs. For this type of error the last block received is discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one. For a multi-block transfer, the host driver issues a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even if AC12EN and BCEND bits are set, the eSDHC does not automatically send a CMD12 because the last block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 is sent by the eSDHC. In this case, the driver resends or reobtains the last block with a single-block transfer.

23.5.3.4.2 Internal DMA Error

During the data transfer with internal simple DMA, if the DMA engine encounters some error on the AHB bus, the DMA operation is aborted and DMA error interrupt is sent to the Host System. When acknowledged by such an interrupt, the driver calculates the start address of data block in which the error occurs. The start address can be calculated by either of the following methods:

1. Read the DMA system address register. The error occurs during the previous burst. Taking the block size, the previous burst length and the start address of the next burst transfer into account, it is straightforward to obtain the start address of the corrupted block.

2. Read the BLKCNT field of the block attribute register. By the number of blocks left, the total number of blocks to transfer, the start address of transfer, and the size of each block, the start address of the corrupted block can be obtained. When the BCEN bit is not set, the contents of the block attribute register do not change, so this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover from the error.

23.5.3.4.3 Auto CMD12 Error

If the AC12EN bit is set when a multi-block data transfer is initiated by the data command, then the eSDHC automatically sends a CMD12 to the card to stop the transfer after the last block of the transfer is sent or received. It is recommended that the driver respond to errors in the auto CMD12 command as follows:

1. If the error is an auto CMD12 response timeout (indicated by bit 1 of the auto CMD12 error status register (AUTOC12ERR), then it is not certain whether the command has been accepted by the card or not. In this case, the driver can clear the AUTOC12ERR status bits and resend the CMD12 until it is accepted by the card.
2. Auto CMD12 response CRC error (indicated by bit 3 of AUTOC12ERR). Since the card responds to CMD12, the card aborts the transfer. In this case, the driver can ignore the error and clear the error status bit.
3. Auto CMD12 conflict error or not sent. The CMD12 command is not sent. In this case, the driver can send a CMD12 manually.

23.5.3.5 Card Interrupts

The external cards can inform the host controller by means of some special signals. For the SDIO card, it can be the low level on the DAT[1] line during some special period. For the CE-ATA card, it can be a pulse on the CMD line to inform the host controller that the command and its response is finished, and it is possible that some additional external interrupt behaviors are defined. The eSDHC only monitors the DAT[1] line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the eSDHC, and the host system is informed by the eSDHC asserting the eSDHC interrupt line, the interrupt service from the host driver is called.

As the interrupt factor is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared by writing 1. See Section 23.4.7.3, “Card Interrupt Handling,” for the card interrupt handling flow.

23.5.4 Switch Function

MMC cards transferring data at bus widths other than 1-bit is a new feature added to the MMC specification. High-speed timing mode for all card devices, is also a recent addition to the various card

specifications. To enable these new features in cards of different types the host driver issues commands as follows:

- For MMC cards (and CE-ATA over MMC interface), the high-speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol SWITCH). 4-bit and 8-bit MMC bus widths are also enabled by SWITCH commands, but with a different argument.
- For SD cards, the high-speed mode is queried and enabled by a CMD6 (with the mnemonic symbol SWITCH_FUNC).
- For SDIO cards, the high-speed mode is enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed.

These new functions can be disabled by a software reset. For SDIO cards this can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

The following subsections provides pseudocode for enabling or disabling high-speed mode for various card types, and for setting MMC bus width. For the sake of simplicity, the pseudocode does not show current capability check, which is recommended in the function-switch process.

23.5.4.1 Query, Enable and Disable SDIO High-Speed Mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is ‘0’) report the SDIO card does not support high-speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is cleared;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

23.5.4.2 Query, Enable and Disable SD High-Speed Mode

```
enable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFF1 and read 64 bytes of data accompanying the R1 response;
wait data transfer done bit is set;
check if the bit 401 of received 512 bit is set;
if (bit 401 is ‘0’) report the SD card does not support high-speed mode and return;
send CMD6, with argument 0x80FFFFF1 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 50MHz;
```

```
(data transactions like normal peers)
}
disable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

23.5.4.3 Query, Enable and Disable MMC High-Speed Mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high-speed mode and return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the ‘high-speed mode’ in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high-speed mode failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

23.5.4.4 Set MMC Bus Width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and return;
send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

23.6 MMC/SD/SDIO/CE-ATA Card Commands

There are four kinds of commands defined to control MultiMediaCards (MMC):

- Broadcast commands (bc), which elicit no response from the cards
- Broadcast commands with response (bcr), which elicit responses from all cards simultaneously
- Addressed (point-to-point) commands (ac), which do not involve data transfer on the DAT lines
- Addressed (point-to-point) data transfer commands (adtc), which involve data transfer

Table 23-30 lists commands for MMC/SD/SDIO/CE-ATA cards. Refer to the corresponding specifications for more details about these commands.

Table 23-30. Commands for MMC/SD/SDIO/CE-ATA Cards

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	—	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	—	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operating conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to “SD Physical Specification V1.1” for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to “The MultiMediaCard System Specification Version 4.0 Final draft 2” for more details.

Table 23-30. Commands for MMC/SD/SDIO/CE-ATA Cards (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Toggles a card between the standby and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	—	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.

Table 23-30. Commands for MMC/SD/SDIO/CE-ATA Cards (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command is issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.

Table 23-30. Commands for MMC/SD/SDIO/CE-ATA Cards (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather that a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57~59	Reserved				

Table 23-30. Commands for MMC/SD/SDIO/CE-ATA Cards (continued)

CMD INDEX	Type	Argument	Response	Abbreviation	Description
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	CE-ATA devices contain a set of Status and Control registers that begin at register offset 80h. These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are word (32-bit) in size and are word aligned. CMD60 is used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command.
CMD62~63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width (‘00’=1bit or ‘10’=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	—	R1	SET_WR_BLK_ERASE_COUNT	—
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating conditions register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	—	R1	SET_CLR_CARD_DETECT	—
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1 CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).

2 CMD6 differs completely between high-speed MMC cards and high-speed SD cards. Command SWITCH_FUNC is for high-speed SD cards.

3 Command SWITCH is for high-speed MMC cards as well as for CE-ATA cards over the MMC interface. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 23-31.

[4] ACMDs is preceded with the APP_CMD command. (Commands listed are used for SD only, other SD commands not listed are not supported on this module).

The Access Bits for the EXT_CSD Access Modes are shown in Table 23-31.

Table 23-31. EXT_CSD Access Modes

Bits	Access Name	Operation
00	Command Set	The command set is changed according to the Cmd Set field of the argument
01	Set Bits	The bits in the pointed byte are set, according to the 1 bits in the Value field.
10	Clear Bits	The bits in the pointed byte are cleared, according to the 1 bits in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

23.7 Software Restrictions

23.7.1 Initialization Active

The driver cannot set the INITA bit in the system control register when either of the command line or data lines is active, so the driver is responsible to ensure both CDIHB and CIHB bits are cleared. In order to auto clear the INITA bit, the SDCLKEN bit must be 1, otherwise no clock signal goes out to the card and INITA never clears.

23.7.1.1 Software Polling Procedure

For polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as the values set in the watermark level register; moreover, if the block size is not the times of the value in watermark level register (read and write respectively), the software must access exactly the remained number of words at the end of each block. For example, for read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

23.7.1.2 Suspend Operation

In order to suspend the data transfer, the software must inform eSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as ‘01’) to inform eSDHC that the transfer is suspended.

If the software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remained number of blocks before sending the normal command marked as suspend, otherwise on sending such ‘suspend’ command, eSDHC regards that the current transfer is aborted and changes BLKCNT register to its original value, instead of keeping the remained number of blocks.

23.7.1.3 DMA Address Setting

To configure DMA address register, when TC bit is set, the register always updates itself with the internal address value to support dynamic address synchronization, so software must make sure TC bit is cleared prior to configuring DMA address register.

23.7.1.4 Data Port Access

Data Port does not support parallel access. For example, during an external DMA access, it is not allowed to write any data to the Data Port by CPU; or during a CPU read operation, it is also prohibited to write any data to the Data Port, by either CPU or external DMA. Otherwise the data would be corrupted inside the eSDHC buffer.

23.7.1.5 Change Clock Frequency

eSDHC does not automatically gates off the card clock when the host driver changes the clock frequency. To remove possible glitch on the card clock, clear SDCLKEN bit when changing clock divisor value and set SDCLKEN bit to ‘1’ after SDSTB bit is ‘1’ again.

Chapter 24
Enhanced SDRAM Controller (ESDRAMC)

The enhanced synchronous dynamic RAM controller (ESDRAMC) provides interface and control for synchronous DRAM (SDRAM) memories for the system. SDRAM memories use a synchronous interface with all signals registered on a clock edge. A command protocol is used for initialization, read, write, and refresh operations to the SDRAM and is generated on the signals by the controller when required due to external or internal requests. It has support for both single- and double-data rate SDRAMs. It supports 64-,128-, 256-, or 512-Mbit and 1-Gbit 4-bank SDRAM per chip select.

Figure 24-1 is the enhanced SDRAM controller top-level diagram, showing the functional organization of the module.

NOTE

The ESDRAMC and ESDCTL mnemonics are equivalent. For historical reasons they are alternately used throughout the document.

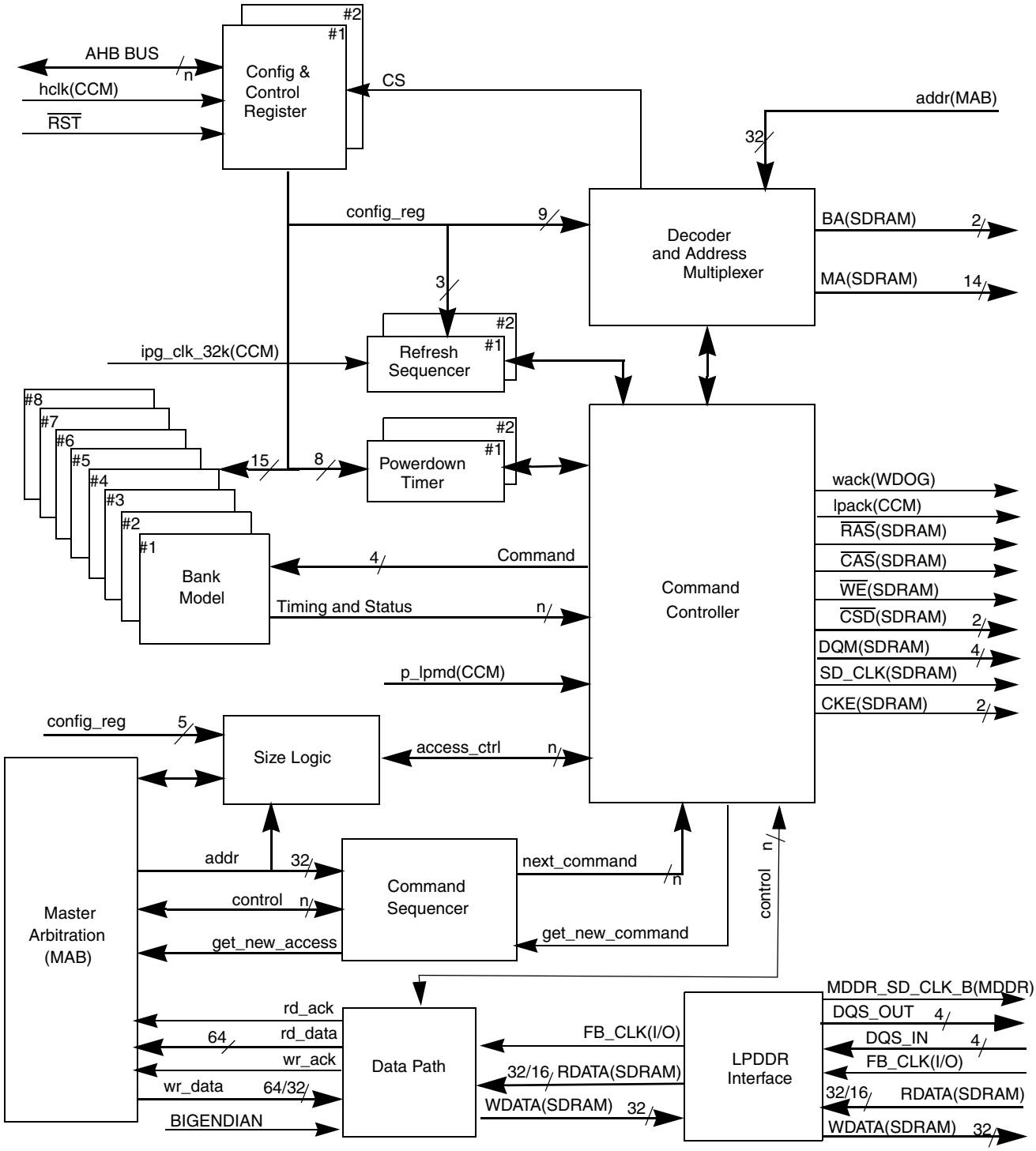


Figure 24-1. Enhanced SDR/LPDDR SDRAM Controller Module Top-Level Diagram

24.1 Overview

The enhanced SDRAM controller consists of nine major submodules:

- SDRAM command controller
- 8 Bank models (page and bank address comparators)
- Row/Column address multiplexer,
- Control and configuration registers
- Refresh request counter
- Command sequencer
- Size logic (splitting access)
- Data aligner/multiplexer (data path)
- LPDDR interface
- Power-down timer

These submodules are described in the following subsections.

24.1.1 SDRAM Command Controller

This submodule controls most of the actions within the ESDRAMC, and includes 12 flip-flops indicating if the bus to the memory is busy for the next 12 cycles. All commands to the memory are executed through this submodule.

24.1.2 Bank Models

There are a total of 8 address comparators, one comparator for each of the 4 banks within each of the two chip select regions. The comparators are used to determine if a requested access falls within the address range of a currently active SDRAM page. The bank model also includes all timing parameters for the comparators.

24.1.3 Row/Column Address Multiplexer

All synchronous SDRAMs incorporate a multiplexed address bus, although the number of bytes per bank and the address folding points vary according to memory density, number of data I/O, and the processor data bus width. The enhanced SDRAM controller takes these variables into account and provides the proper alignment of the multiplexed address through the combination of the row/column address multiplexer, non-multiplexed address pins, and the connections between the controller and the memory devices.

24.1.4 ESDRAMC Control and Configuration Registers

Control and configuration registers determine the operating mode of the ESDRAMC. Configurable parameters include memory device density and bus width, number of memory devices, CAS latency, row to column delay, and burst length. Enable bits are provided for refresh and the power-down timer. Mode

bits provide a mechanism for software initiated SDRAM initialization, setting the device mode register, precharge, and auto-refresh cycles.

24.1.5 Refresh Request Counter

SDRAM memories require periodic refresh to retain data. The refresh request counter generates requests to the SDRAM command controller to perform those refresh cycles. Requests are scheduled according to a 32-kHz clock input. One, 2, 4, 8, or 16 refresh cycles are generated per a 32-kHz clock.

24.1.6 Command Sequencer

The command sequencer submodule sends the commands be executed (including precharge-all, precharge-bank, active, read, write, and burst terminate commands) to the command controller, after taking into account the bank's state of the access, the command controller execution signal and the command busy situation.

24.1.7 Size Logic

The size logic submodule gets the following inputs:

- From master arbitration: address, access size, and wrap/incr access
- From the config control register: configuration values (burst length and SDRAM memory data width)

In case of a misaligned access, the size logic splits the access into multiple accesses as a function of the inputs to the submodule.

24.1.8 Mobile/Low Power DDR (LPDDR) Interface

The LPDDR interface provides the device's interface with LPDDR SDRAM. It converts the double–data rate signal that is synchronized to both positive and negative edges of the data strobe (DQS) signals to a pass through a double-width data bus synchronized to the positive edge of the controller internal clock (which is derived from HCLK).

The interface uses a read FIFO which samples the data with delayed DQS in read cycles. Two read FIFOs are used for positive and negative clock edges, respectively. Delay lines are used to generate delayed versions of DQS input signals in order to sample the data at the middle of the data valid window. In write cycles DQS are output signals that are generated using a delay line. The data at the input to the interface has a double width from the memory so it is divided into the memory width but with double frequency. For this purpose a multiplexer is used to pass the upper half and lower half of the data. In read cycles, double-rate data is received with a DQS signal that is edge-aligned with read data; and in write cycles double-rate data is created with a centered DQS signal.

DQS delay lines are based on three functional units:

1. The measurement unit measures one cycle time between successive positive edges. The output of this unit is the number of small delay intervals needed to delay one positive edge of the clock to overlap the following positive edge.

2. The division unit divides the result of the measurement by 4.
3. The delay unit takes the result and selects the correct delay tap.

The delay unit is duplicated 5 times: 4 units for read (one for each byte lane's DQS signal) and one unit for write (to delay overall data sampling).

24.1.8.1 Power-Down Timer

The power-down timer detects periods of inactivity to the SDRAM and disables the clock when the inactive period surpasses the selected timeout. Data is retained during the power-down state. Subsequent requests to the SDRAM incur only a minimal added start-up delay (beyond the normal access time, as specified in Table 24-19). The power-down timer can be disabled, or can be programmed to expire under any one of the following conditions:

- Any time the controller is not actively reading/writing the memory,
- After 64 clocks of inactivity
- After 128 clocks of inactivity

24.1.9 Features

The ESDRAMC includes the following features:

- Optimizes consecutive memory accesses through memory command anticipation (latency hiding)
  - Hides latency by optimization the commands to both chip selects (command anticipation)
  - Keeps track of open memory pages
  - Bankwise memory address mapping
  - SDRAM burst length configuration of 4 or 8 (for 16-bit memory burst length 4 is not supported) or full-page mode
  - LPDDR/DDR2 burst length configuration of 8
  - Support of different internal burst lengths (1/4/8 words) by using burst truncate commands
  - ARM AMBA AHB-lite-compatible
  - Shared address and command bus to SDRAM/LPDDR
- Supports 64-, 128-, 256-, or 512-Mbit and 1-Gbit sizes of 4-bank, single data rate, synchronous SDRAM, LPDDR and non-mobile DDR1 devices. Limited support for DDR2 devices (4 banks only, no ODT control signal)
  - Two independent chip selects
  - Up to 128 Mbytes per chip select
  - Up to four banks active simultaneously per chip select
  - JEDEC standard pinout/operation
- Support for 16-bit LPDDR/DDR2 devices
- PC133-compliant interface
  - 133 MHz system clock achievable with "–7" option for PC133-compatible memories
  - Single fixed-length (4/8-word) burst or full-page access

  - Access time of 9-1-1-1-1-1-1-1 at 133 MHz (for read access when memory bus is available, row is open and CAS latency configured to 3 cycles). The access time includes the M3IF delay (assuming no arbitration penalty).
- Software-configurable for different system and memory devices requirements
  - 16-bit memory data bus width (equal in both chip selects)
  - Number of row and column addresses
  - Row cycle delay (t_{RC})
  - Row precharge delay (t_{RP})
  - Row to column delay (t_{RCD})
  - Column to data delay (CAS Latency)
  - Load mode register to active command (t_{MRD})
  - Write to precharge (t_{WR})
  - Write to read (t_{WTR}) for LPDDR memories only
  - LPDDR exit power-down to next valid command delay (t_{XS})
  - Active to precharge (t_{RAS})
  - Bank active to active (t_{RRD})
- Built- in auto-refresh timer and state machine
- Hardware- and software-supported self-refresh entry and exit
  - Data remains valid during system reset and low power modes
  - Auto–power-down timer (one per chip select)
  - Auto-precharge timer (one per bank in each chip select)

24.1.10 Modes of Operation

This section provides a high-level description of the ESDRAMC operating modes: see Section 24.4, “Functional Description,” for more detailed descriptions.

The ESDRAMC’s different operating modes for each chip select ($\overline{CSD}n$, n = 0,1) are defined by the 3-bit SMODE field in the corresponding ESDCTL*n* register. In addition to the normal operating mode, the controller is capable of operating in three alternate operating modes primarily used for SDRAM/LPDDR initialization, as follows:

- Normal read/write mode

  This is the normal operating mode used to read/write (single or burst accesses) from/to external SDRAM/LPDDR devices. The ESDRAMC automatically drives the precharge/active/burst/terminate commands during the normal operating mode.
- Precharge mode

  The manual precharge command is used to manually deactivate the open (active) row in a particular bank or the open row in all banks. The bank(s) are available for a subsequent row access at a specified time (tRP) after the precharge command is issued. External memory device input A10 determines whether one or all banks are to be precharged, and in the case that only one bank is to

be precharged, inputs BA0 and BA1 select the bank. The manual precharge command is used during SDRAM initialization, load mode register command, and manual refresh cycles

- Auto-refresh mode

  The auto-refresh command is used to retain data in the SDRAM memory devices. This command is non persistent, so it must be issued each time a refresh is required. The ESDRAMC has a refresh counter for each CSD_B (external memory region), and it automatically handles the refresh commands to the memory. The manual auto-refresh command is used during SDRAM initialization or in case the refresh counters are not enabled.

- Load mode register mode

  The mode register is used to define the specific mode of operation of the SDRAM device. This definition includes the selection of a burst length, a burst type, a CAS latency, an operating mode and a write burst mode. The mode register is programmed using the load mode register command and retains the stored information until it is programmed again or the external memory device loses power. The mode register can only be loaded when all banks are idle (after precharge-all). The ESDRAMC waits a specified period of time (t_{MRD})as configured in the ESDCFG0 or ESDCFG1 register. Violating either of these requirements by an incorrect controller register configuration results in unspecified operation.

Any access to the SDRAM/LPDDR memory space while in one of the alternate modes results in the corresponding special cycle being run. Moving from normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitioning out of normal read/write mode. Reset initializes the operating mode to normal read/write mode.

24.2 External Signal Descriptions

This section discusses input and output signals between the enhanced SDRAM controller and the external memory devices. Other than the chip select outputs ($\overline{CSD}0$ and $\overline{CSD}1$) and clock enables (CKE0 and CKE1), all signals are shared between the two chip select regions.

Interconnect and timing diagrams are included as part of the detailed discussion on controller operation in Section 24.4, “Functional Description.”

24.2.1 Overview of Signals

Table 24-1 summarizes the interface signals.

Table 24-1. ESDRAMC Signal Properties

Name	Port	Function	Reset State	Direction
SDCLK	—	Clock to SDRAM (up to 133MHz)	1	Output
SDCKE0	—	Clock to SDRAM to SDRAM 0	0	Output
SDCKE1	—	Clock to SDRAM to SDRAM 1	0	Output
$\overline{CSD}$[0]	—	Chip select to SDRAM Array 0	1	Output
$\overline{CSD}$[1]	—	Chip select to SDRAM Array 1	1	Output

Table 24-1. ESDRAMC Signal Properties (continued)

Name	Port	Function	Reset State	Direction
SDATA[15:0]	—	Data bus	0	I/O
MA[13:0]	—	Multiplexed address	0	Output
SDBA[1:0]	—	Bank address	0	Output
DQM1	—	Data qualifier mask byte 1(D[15:8])	0	Output
DQM0	—	Data Qualifier mask byte 0 (D[7:0])	0	Output
$\overline{\text{SWE}}$	—	Write enable	1	Output
$\overline{\text{RAS}}$	—	Row address strobe	1	Output
$\overline{\text{CAS}}$	—	Column address strobe	1	Output
Mobile LPDDR signals				
SDCLK_B	—	Inverted clock to LPDDR SDRAM	0	Output
SDDQS1	—	Data strobe byte 1 (D[15:8])	0	Input
SDQS0	—	Data strobe byte 0 (D[7:0])	0	Input

24.2.2 Detailed Signal Descriptions

Table 24-2 lists the signals and descriptions of all of the I/O signals that interface with the ESDRAMC.

Table 24-2. ESDRAMC Detailed Signal Description

Signal	I/O	Description
SDCLK	O	SDRAM clock. The SDCLK output provides the timing reference for the memory devices. All other SDRAM interface signals are referenced to this clock. SDCLK is synchronous to the system clock, but is gated off during low power operating modes when both SDCKE0 and SDCKE1 are negated.
SDCKE0 SDCKE1	O	SDRAM/MMDR clock enables. Clock enable outputs to the SDRAM memory devices. SDCKE0 corresponds to SDRAM/LPDDR array 0 and SDCKE1 to SDRAM/LPDDR array 1. Activates the memory's clock input when high, indicating a stable clock is being supplied. Deactivates the memory's clock input when low. SDCKEx low initiates power-down and self-refresh modes to the SDRAM.
$\overline{\text{CSD0}}$ $\overline{\text{CSD1}}$	O	SDRAM/LPDDR chip select. $\overline{\text{CSD0}}$ and $\overline{\text{CSD1}}$ are used to select SDRAM/LPDDR array 0 and SDRAM/LPDDR array 1, respectively. The chip select signals are used to indicate when a valid command is present on the other control signals and to which device the command is directed.
SDATA[15:0]	I/O	Read/write data bus. The 16 data pins are used to transfer data between the enhanced SDRAM controller and memory. Data bit 15 is the most significant bit. Bit 0 is the least significant. 16-bit memory alignment is selectable according to the setting of the SDRAM memory data width (DSIZ field in the ESDCTLn register) (See Section 24.3.3, "Register Descriptions").

Table 24-2. ESDRAMC Detailed Signal Description (continued)

Signal	I/O	Description
MA[13:0]	O	Multiplexed address bus. The multiplexed address bus specifies the SDRAM page and location within the page targeted by the current access. Connections between the enhanced SDRAM Controller and memory varies depending on the SDRAM device density. See Section 24.4.3, "Address Multiplexing" and specifically Table 24-17 and Table 24-22 for details on supported SDRAM configurations.
SDBA[1:0]	O	Bank address bus. The bank address pins specify to which bank the current command is targeted. Table 24-15 and Table 24-16 document which address pins are used as the bank address bus for given device configuration.
DQM1, DQM0	O	Data qualifier mask. During read cycles, DQMx controls the SDRAM data output buffers. DQMx asserted high disables the output buffers leaving them in a high-impedance state. DQMx low allows the data buffers to drive normally. During write cycles, DQMx controls which bytes are written in the SDRAM. DQMx driven low enables a write to the corresponding byte, while DQMx asserted high leaves the byte unchanged. DQM0 corresponds to D[7:0] and DQM1 to D[15:8]. The ESDRAMC takes care of the endian operating mode, and drives the respective DQM strobe required. **Note:** When the controller is used to interface mobile/low power DDR, DQM changes twice each cycle (like the data) and is aligned to DQS edges.
$\overline{\text{SDWE}}$	O	Write enable. Write enable is part of the three bit command field (RAS and CAS make up the other two bits) used by the SDRAM. Generally, $\overline{\text{SDWE}}$ is asserted low if a command transfers data to the memory. A detailed summary of the supported SDRAM commands is provided in Table 24-18
$\overline{\text{RAS}}$	O	Row address strobe. Row address strobe is also part of the SDRAM command field. It is generally used to indicate an operation affecting an entire bank or row. RAS is an active low signal. Table 24-20 provides details on SDRAM command encoding.
$\overline{\text{CAS}}$	O	Column address strobe. The column address strobe is the third signal comprised in the command field. It generally signifies a column oriented command. CAS is an active low signal. Table 24-20 provides details on SDRAM command encoding.
SDCLK_B	O	LPDDR SDRAM inverted clock. SDCLK and SDCLK_B are mobile/low power DDR differential clocks. All LPDDRs address and control input signals are sampled on the crossing of the positive edge of SDCLK and negative edge of SDCLK_B. Internal clock signals are derived from SDCLK/SDCLK_B.
SDQS1, SDQS0	I	Data strobes inputs outputs. During read cycles, SDQSx are generated by the memory devices and are edge aligned with read data. During write cycles, they are generated by the ESDRAMC and are centered with write data. The SDQS delay module is used to delay the SDQSx signal and center it in the data valid window.

24.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers.

All implemented bits in the ESDRAMC registers are fully readable and writable, except for the enhanced MDDR delay line cycle length debug register (ESDCDLYL) register which is read-only. Reserved bit locations are unaffected by writes and always read as zero. All register accesses must be single-word (32-bit) operations through the AHB bus protocol. Accesses of any other size have indeterminate results.

All ESDRAMC registers can be only accessed by one master at a time: multiple-master accesses to the registers cause undetermined behavior.

24.3.1 Memory Map

Table 24-3 shows the ESDRAMC memory map. For the base address of a particular module instantiation, see the system memory map.

The ESDRAMC supports memory devices on two independent chip selects. Each chip select defines a specific memory address range: Table 24-4 shows the ESDRAMC memory address ranges for each chip select.

Table 24-3. ESDRAMC Memory Map

Base Address Offset (Name Abbreviation)	Register	Access	Reset	Section/Page
0x0000 (ESDCTL0)	Enhanced SDRAM control register 0	R/W	0x0111_0080	24.3.3.1/24-13
0x0004 (ESDCFG0)	Enhanced SDRAM configuration register 0	R/W	0x0076_EB3A	24.3.3.2/24-17
0x0008 (ESDCTL1)	Enhanced SDRAM control register 1	R/W	0x0112_0080	24.3.3.1/24-13
0x000C (ESDCFG1)	Enhanced SDRAM configuration register 1	R/W	0x007A_C727	24.3.3.2/24-17
0x0010 (ESDMISC)	Enhanced SDRAM miscellaneous register	R/W	0x0000_0000	24.3.3.3/24-22
0x0020 (ESDCDLY1)	Enhanced MDDR Delay Line 1 configuration debug register	R/W	0x00F4_8000	24.3.3.4/24-23
0x0024 (ESDCDLY2)	Enhanced MDDR delay line 2 configuration debug register	R/W	0x00F4_8000	24.3.3.4/24-23
0x0028 (ESDCDLY3)	Enhanced MDDR delay line 3 configuration debug register	R/W	0x00F4_8000	24.3.3.4/24-23
0x002C (ESDCDLY4)	Enhanced MDDR delay line 4 configuration debug register	R/W	0x00F4_8000	24.3.3.4/24-23
0x0030 (ESDCDLY5)	Enhanced MDDR delay line 5 configuration debug register	R/W	0x00F4_8000	24.3.3.5/24-25
0x0034 (ESDCDLYL)	Enhanced MDDR delay line cycle length debug register	R	N/A	24.3.3.6/24-26

Table 24-4. ESDRAMC Memory Regions

Address Range	Use	Access
CS0_BASE–(CS0_BASE + 0x0FFF_FFFF)	CSD0 SDRAM or LPDDR memory region (256 Mbytes)	Read/write
CS1_BASE–(CS1_BASE + 0x0FFF_FFFF)	CSD1 SDRAM or LPDDR memory region (256 Mbytes)	Read/write

24.3.2 Register Summary

Figure 24-2 shows the key to the register fields and Table 24-5 shows the register figure conventions.

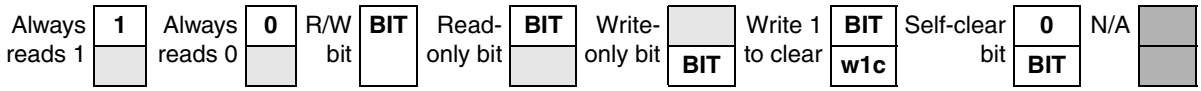


Figure 24-2. Key to Register Fields

Table 24-5. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 24-6 shows the ESDRAMC register summary. For the base address of a particular module instantiation, see the system memory map.

Table 24-6. ESDRAMC Register Summary

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (ESDCTL0)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																

Table 24-6. ESDRAMC Register Summary (continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0004 (ESDCFG0)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
	W																
0x0008 (ESDCTL1)	R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
	W																
	R	SREFR			0	PWDT		0	FP	BL	0	PRCT					
	W																
0x000C (ESDCFG1)	R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
	W																
	R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
	W																
0x0010 (ESDMISC)	R	SDRAM_RDY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	DDR2_EN	DDR_EN	FRC_MSR	MA10_SHARE	LHD	MDDR_MDIS	0	MDDR EN	0	0
	W													MDDR_DL_RST		RST	
0x0020 (ESDCDLY1)	R	SEL_DLY_REG_1	0	0	0	0	0	0	0	DLY_OFFSET_1							
	W																
	R	DLY_ABS_OFFSET_1								DLY_REG_1							
	W																
0x0024 (ESDCDLY2)	R	SEL_DLY_REG_2	0	0	0	0	0	0	0	DLY_OFFSET_2							
	W																
	R	DLY_ABS_OFFSET_2								DLY_REG_2							
	W																
0x0028 (ESDCDLY3)	R	SEL_DLY_REG_3	0	0	0	0	0	0	0	DLY_OFFSET_3							
	W																
	R	DLY_ABS_OFFSET_3								DLY_REG_3							
	W																

Table 24-6. ESDRAMC Register Summary (continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x002C (ESDCDLY4)	R	SEL_DLY_REG_4	0	0	0	0	0	0	0	DLY_OFFSET_4							
	W																
	R	DLY_ABS_OFFSET_4								DLY_REG_4							
	W																
0x0030 (ESDCDLY5)	R	SEL_DLY_REG_5	0	0	0	0	0	0	0	DLY_OFFSET_5							
	W																
	R	DLY_ABS_OFFSET_5								DLY_REG_5							
	W																
0x0034 (ESDCDLYL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
	W																
	R	0	0	0	0	0	0	0	0	QTR_CYCLE_LENGTH							
	W																

24.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

24.3.3.1 Enhanced SDRAM Control Registers (ESDCTL*n*, *n* = 0,1)

These registers configure the memory and control settings for the ESDRAMC for chip selects 0 and 1 respectively. Figure 24-3 and Figure 24-4 show the bit assignments for the registers. Note that bit field assignments are identical for the two registers, but reset values for some fields are different. Table 24-7 provides field descriptions.

Offset 0x0000 (ESDCTL0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 24-3. Enhanced SDRAM Control Register 0 (ESDCTL0)

Offset 0x0008 (ESDCTL1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDE	SMODE			SP	ROW			0	0	COL		0	0	DSIZ	
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SREFR			0	PWDT		0	FP	BL		PRCT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 24-4. Enhanced SDRAM Control Register 1 (ESDCTL1)

Table 24-7. Enhanced SDRAM Control Register (ESDCTL*n*) Field Descriptions

Field	Description
31 SDE	Enhanced SDRAM controller enable. This control bit enables/disables the enhanced SDRAM controller. Writing 1 to both control bits enables the module for both chip selects. Clearing both SDE bits disables the module, and all clocks within the module shuts off (with the exception of register accesses). The reset value of both SDE bits is 0, so that the module is disabled. 0 Disabled (reset value) 1 Enabled
30–28 SMODE	SDRAM controller operating mode. This bit field determines the operating mode of the enhanced SDRAM controller. In addition to the normal operating mode, the controller is capable of operating in several alternate operating modes, which are primarily used for SDRAM initialization. Any access to the SDRAM memory space, while in one of the alternate modes, results in the corresponding special cycle being run. Transitioning from normal to any other mode does not close (precharge) any banks that may be open (activated). Under most circumstances, software should run a precharge-all cycle when transitioning out of normal read/write mode. Operating mode details are provided in Section 24.4, “Functional Description.” Reset initializes the operating mode to “normal read/write”. 000 Normal read/write 001 Precharge command 010 Auto-refresh command 011 Load mode register command 100 Manual self-refresh 101 through 111 Reserved
27 SP	Supervisor protect. This control bit is used to restrict user accesses within the chip select region. The default at reset are that both user and supervisor accesses are allowed. 0 User mode accesses are allowed to this chip select region. 1 User mode accesses are prohibited. An attempted access to this chip select region while in user mode will result in a high HRESP[1] being returned back to the CPU. The chip select will not be asserted. An ESDRAMC error response is generated by the M3IF arbitrator.
26–24 ROW	Row address width. This control field specifies the number of row addresses used by the memory array. This number does not include the bank, column, or data qualifier addresses. Parameters affected by the programming of this field include the page-hit address comparators and the bank address bit locations. 000 11 row addresses 001 12 row addresses 010 13 row addresses 011 14 row addresses 100 through 111 Reserved
23–22	Reserved
21–20 COL	Column address width. The COL control field is used to specify the number of column addresses in the memory array and determine the break point in the address multiplexer. Column width is the number of multiplexed column addresses and does not include bank, and row addresses, or addresses used to generate the DQM signals. 00 8 column addresses 01 9 column addresses 10 10 column addresses 11 Reserved
19–18	Reserved

Table 24-7. Enhanced SDRAM Control Register (ESDCTL*n*) Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>17–16
DSIZ</td><td>SDRAM memory data width. This field defines the width of the SDRAM memory external data bus. Memories aligned to D[31:16] use DQM2 and DQM3. Memories aligned to D[15:0] use DQM0 and DQM1.
00 Reserved
01 16-bit memory width aligned to D[15:0] (reset value for CSD0)
10 Reserved (reset value for CSD1)
11 Reserved</td></tr>
<tr><td>15–13
SREFR</td><td>SDRAM refresh rate. This control bit field enables/disables SDRAM refresh cycles and controls the refresh rate. Refresh cycles are referenced to a 32 kHz clock. At each falling edge 1, 2, 4, 8 or 16 rows are refreshed as determined by this bit field. Multiple refresh cycles are separated by the row cycle delay specified in the SRC control field. Refresh is disabled by hardware reset. For an example of usage see Table 24-13.
<table>
<tr><th>SREFR[2:0]</th><th>Rows per Refresh Clock</th><th>Rows per 64 ms @ 32 kHz</th><th>Row Rate @ 32 kHz</th></tr>
<tr><td>000</td><td colspan="3">Refresh Disabled (bit field reset value)</td></tr>
<tr><td>001</td><td>1</td><td>2048</td><td>31.25 µs</td></tr>
<tr><td>010</td><td>2</td><td>4096</td><td>15.62 µs</td></tr>
<tr><td>011</td><td>4</td><td>8192</td><td>7.81 µs</td></tr>
<tr><td>100</td><td>8</td><td>16384</td><td>3.91µs</td></tr>
<tr><td>101</td><td>16</td><td>32768</td><td>1.95 µs</td></tr>
<tr><td>110</td><td colspan="3">Reserved</td></tr>
<tr><td>111</td><td colspan="3">Reserved</td></tr>
</table></td></tr>
<tr><td>12</td><td>Reserved</td></tr>
<tr><td>11–10
PWDT</td><td>Power-down timer. This field determines whether the SDRAM is placed in a power-down condition after a selectable delay from the last access. The power-down timeout can be triggered on either the absence of an active bank (PWDT=01) or a clock (HCLK) count from the last access (PWDT=10 or 11). Count-based timeouts do not force the SDRAM into an idle condition (for example, any active banks remain open). The power-down timers feature is disabled by hardware reset. See sections Section 24.4.6.3, “Power-Down Modes” and Section 24.4.6.3.2, “Active Power-Down Mode” for a comprehensive description of this operating mode.
<table>
<tr><th>PWDT[1:0]</th><th>PowerDown Timeout</th><th>Memory Device Operating Mode</th></tr>
<tr><td>00</td><td>Disabled (bit field reset value)</td><td>Run mode</td></tr>
<tr><td>01</td><td>Any time no banks are active</td><td>Precharge power-down</td></tr>
<tr><td>10</td><td>64 clocks (HCLK) after completion of last access[1]</td><td>Active power-down</td></tr>
<tr><td>11</td><td>128 clocks (HCLK) after completion of last access[1]</td><td>Active power-down</td></tr>
</table>
[1] This setting cannot be used if the PRCT (precharge timer) is enabled.
Note: When PWDT is enabled and lpmd or dvfs requests are required by system, PWDT must be disabled before the requests are granted. PWDT can only be re-enabled after these modes have been exited.</td></tr>
<tr><td>9</td><td>Reserved</td></tr>
</table>

Table 24-7. Enhanced SDRAM Control Register (ESDCTL*n*) Field Descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>8
FP</td><td>Full page. This bit should be set to 1 if the burst length of the SDRAM connected to the CSD has been configured to full page mode. This bit is needed since ESDRAMC needs to induce a BURST TERMINATE (BT) command to terminate early all accesses that are less than full page.
0 Burst length of the external memory device is not set to full page.
1 Burst length of the external memory device is set to full page.
Note: Full page mode is not supported when LPDDR external devices are used.</td></tr>
<tr><td>7
BL</td><td>Burst length. This bit configures the access burst length. For proper operation the ESDRAMC burst length configuration must match the external SDRAM/LPDDR memory device (configured using special operating load mode register). Setting this bit to 1 means that the external memory device connected to the CSD have been configured to burst length of 8. If this bit is cleared to 0, means that the external memory device connected to the CSD have been configured to burst length of 4.
BL bit settings are given in the following table:
<table>
<tr><th>BL setting</th><th>SDR SDRAM</th><th>LPDDR SDRAM</th></tr>
<tr><td>0</td><td>4 (not supported for 16-bit SDRAM devices</td><td>Reserved</td></tr>
<tr><td>1 (reset value)</td><td>8</td><td>8</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>Reserved</td></tr>
<tr><td>5–0
PRCT</td><td>Precharge timer. Banks are precharged after 2xPRCT clocks (HCLK, up to 133 MHz) of no activity.
Using the precharge command to close the last used/open row in any inactive bank within a chip select reduces the power consumption of the external memory device. The power saving is device-dependent: the user may consult the external memory device specification for more details on power consumption reduction.
If PRCT is enabled, the PRCT field specifies the approximately number of HCLK cycles of inactivity to one of the SDRAM/LPDDR banks before the precharge command is issued. The number of cycles before the precharge command is issued depends on command bus (WE, RAS, CAS and CSD) availability (means there is no active access to other bank) and the memory timing parameters.
000000 Disabled (reset value)
000001 2 clocks to precharge
000010 4 clocks to precharge
000011 6 clocks to precharge
000100 8 clocks to precharge
…
100000 64 clocks to precharge
…
111111 126 clocks to precharge</td></tr>
</table>

24.3.3.2 Enhanced SDRAM Configuration Registers (ESDCFG*n*, *n* = 0,1)

Figure 24-5 and Figure 24-6 show the bit assignments for the enhanced SDRAM configuration registers. Bit assignments for the two registers are identical, but reset values for some fields differ. Table 24-8 provides field descriptions for the registers.

NOTE

The ESDRAMC configuration registers reset values define timing parameters that meet memory device optimal timing requirements at 133 MHz. If the external memory device operates at a lower frequency, for optimal performance the user should reconfigure the timing parameters according to the device electrical characteristics and operating conditions.

Offset 0x0004 (ESDCFG0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	1	0	1	0	1	1	0	0	1	1	1	0	1	0

Figure 24-5. Enhanced SDRAM Configuration Register 0 (ESDCFG0)

Offset 0x000C (ESDCFG1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	tXP		tWTR	tRP		tMRD	
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	tWR		tRAS		tRRD		tCAS		0	tRCD			tRC			
W																
Reset	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	1

Figure 24-6. Enhanced SDRAM Configuration Register 1 (ESDCFG1)

Table 24-8. ESDCFG*n* Field Descriptions

Field	Description
31–23	Reserved
22–21 t_{XP}	LPDDR exit power-down to next valid command delay. This control field determines the minimum delay between a valid command is issued to the LPDDR after exiting power-down mode. The value programmed in t_{XP} is the number of clocks inserted after exiting power-down mode and any subsequent new valid command. An example timing diagram for t_{XP} can be found in Figure 24-22. 00 1 clock delay before new command issued to LPDDR after power-down mode exit 01 2 clock delay before new command issued to LPDDR after power-down mode exit 10 3 clock delay before new command issued to LPDDR after power-down mode exit 11 4 clock delay before new command issued to LPDDR after power-down mode exit
20 t_{WTR}	tLPDDR write to read command delay. Data for any write burst may be followed by a subsequent read command. To follow a write without truncating the write burst, t_{WTR} should be set as shown in Figure 24-11. The LPDDRC automatically induces a t_{WTR} number of idle cycles between a write followed by a read command. The t_{WTR} should be configured according to the LPDDR device being used. The t_{WTR} is referenced from the first positive clock edge after the last data-in pair. 0 1 clock 1 2 clocks
19–18 t_{RP}	SDRAM row precharge delay. This control bit determines the number of idle clocks inserted between a precharge command and the next row activate command to the same bank. Hardware reset initializes the controller to insert 3 clocks. Following a precharge command, a subsequent command to the same bank cannot be issued until t_{RP} is met. 00 1 clock 01 2 clocks 10 3 clocks 11 4 clocks
17–16 t_{MRD}	tMRD - SDRAM load mode register to active command. This control bits determines the minimum number of idle clocks required between a load mode register (LMR) command to active. Hardware reset initializes the controller to insert 2 clocks. 00 1 clock 01 2 clocks 10 3 clocks 11 4 clocks
15 t_{WR}	SDRAM write to precharge command. Data for a fixed length write burst may be followed by, or truncated with, a precharge command to the same bank (provided that auto-precharge was not activated), and a full-page write burst may be truncated with a precharge command to the same bank. The precharge command should be issued t_{WR} after the clock edge at which the last desired input data element is registered. t_{WR} control bit determines the number of idle clocks inserted between the last desired input data element and the next precharge command, as shown in Figure 24-14. **Note:** The auto-precharge mode requires a t_{WR} of at least one clock plus time, regardless of frequency.

tWR	Write to Precharge (SDRAM)	Write to Precharge (LPDDR)[1]
0	2 clock	2 clocks
1 (reset value)	3 clocks	3 clocks

[1]Applies in case that MDDR_EN bit is set, that is, the external memory device is a LPDDR.

Table 24-8. ESDCFG*n* Field Descriptions (continued)

Field	Description
14–12 t_{RAS}	SDRAM active to precharge command. These control bits determine the minimum number of clocks required between a active to precharge command to the same bank. Hardware reset initializes the controller to insert 6 clocks. Following a active command, a subsequent precharge command to the same bank cannot be issued until t_{RAS} is met. Figure 24-15 presents an example of a single read (without auto-precharge). It should be noticed that the precharge command is not allowed at T3 and at T4, since t_{RAS} is violated (for t_{RAS}= 4 clock cycles). Note that t_{RAS} also defines the minimum period of time that the SDRAM must remain in self-refresh mode, as it shown in Figure 24-16. 000 1 clock 001 2 clocks 010 3 clocks 011 4 clocks 100 5 clocks 101 6 clocks 110 7 clocks 111 8 clocks
11–10 t_{RRD}	Active bank A to active bank B command. A subsequent active command to a different row in the same bank can only be issued after the previous active row has been “closed” (precharged). A subsequent active command to another bank can be issued while the first bank is being accessed, which results in a reduction of total row-access overhead. The minimum interval between successive active commands to different banks is defined by t_{RRD} as shown in Figure 24-17 (for t_{RRD}=3). The t_{RRD} bits field encoding is listed below and it determines the number of idle clocks inserted between consecutive active commands to different banks. 00 1 clock active to active (different banks) 01 2 clocks active to active (different banks) 10 3 clocks active to active (different banks) 11 4 clocks active to active (different banks)
9–8 t_{CAS}	SDRAM CAS latency. This field determines the latency between a read command and the availability of data on the bus, as shown in Figure 24-18. This field does not affect the second and subsequent data words in a burst. This control field has no effect on write cycles. CAS latency is initialized to 3 clocks following a hardware reset. 00 3 clocks only for LPDDR SDRAM CAS latency 01 Reserved 10 2 clocks SDR and LPDDR SDRAM CAS latency 11 3 clocks SDR and LPDDR SDRAM CAS latency
7	Reserved

Table 24-8. ESDCFG*n* Field Descriptions (continued)

Field	Description
6–4 t_{RCD}	SDRAM row to column delay. This field determines the number of clocks inserted between a row activate command and a subsequent read or write command to the same bank. Hardware reset initializes the delay to 3 clocks. 000 1 clock row to column delay 001 2 clocks row to column delay 010 3 clocks row to column delay 011 4 clocks row to column delay 100 5 clocks row to column delay 101 6 clocks row to column delay 110 7 clocks row to column delay 111 8 clocks row to column delay
3–0 t_{RC}	SDRAM row cycle delay. This control field determines the minimum delay between a refresh and any subsequent refresh or read/write access. This delay corresponds to the minimum row cycle time captured in the t_{RC}/t_{RFC} memory timing specification. The value programmed in t_{RC} is the number of clocks inserted between the refresh and subsequent refresh/activate command. An example timing diagram for t_{RC} can be found in Figure 24-21. 0000 20 clocks 0001 2 clocks 0010 3 clocks 0011 4 clocks … 1111 16 clocks **Note:** The t_{RC} control field is not used to enforce t_{RC} timing for row activate to row activate within the same bank as this is implicitly guaranteed by the sum of t_{RCD} + t_{CAS} + t_{RP}.Use regular paragraphs to summarize register function, then use the following special styles to define bit and field function. **Note:** Since t_{RC} +1 is also used to determine the time between auto-refresh cycles (may be referred to in memories as t_{RFC} or t_{ARFC}) and self-refresh exit time in SDR memories (may be referred to as t_{SREX} or t_{SRFX} for some vendors), and for a specific memory these timings may be worse then t_{RC}, this parameter should be configured to the worst of the three, like so: t_{RC} in controller = max (t_{ARFC} -1, t_{SRFX} -1, t_{RC} in memory). Example: For Samsung K4M51323PC-75 mobile SDR SDRAM, t_{RC} is 72.5 ns, t_{ARFC} is 80 ns, and t_{SRFX} is 120 ns. @133MHz, cycle time is 7.5 ns, so t_{RC} = max (80/7.5 - 1, 120/7.5 - 1, 72.5/7.5) = 15 cycles.

24.3.3.3 Enhanced SDRAM Miscellaneous Register (ESDMISC)

This register configures various memory and control settings for the ESDRAMC. The bit assignments for the register are shown in Figure 24-7 and the field descriptions for the bit assignments are listed in Table 24-9.

Offset 0x0010 (ESDMISC) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDRAM RDY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DDR2_EN	DDR_EN	FRC_MSR	MA10_SHARE	LHD	MDDR_MDIS	0	MDDR_EN	0	0
W													MDDR_DL_RST		RST	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-7. ESDRAMC Miscellaneous Register (ESDMISC)

Table 24-9. Enhanced SDRAM Miscellaneous Register Field Descriptions

Field	Description
31 SDRAM_RDY	External SDRAM/LPDDR device status. This is a read-only status bit, that indicates the state of the external memory device(s). This bit is cleared at reset. This bit is set after the 200 µs SDRAM/LPDDR external memory wake-up period. After the wake-up period the software can start the external memory initialization (as described in Section 24.5.4, “SDRAM/LPDDR Initialization Sequence”). 0 SDRAM/LPDDR external device is not ready for use (reset value). 1 SDRAM/MMDR external device is ready for use.
30-10	Reserved
9 DDR2_EN	Regular (non-mobile) DDR2 device is connected. This bit is common for both chip selects. 0 DDR2 device is not used (reset value) 1 DDR2 device is used **Note:** When this bit is asserted, the MDDR_EN and DDR_EN bits must also be asserted.
8 DDR_EN	Regular (non-mobile) device is connected. This bit is common for both chip selects. 0 DDR1/DDR2 device is not being used (reset value) 1 Non-mobile DDR device is used (DDR2 or DDR1) **Note:** When this bit is asserted the MDDR_EN bit must also be asserted.
7 FRC_MSR	Force measurement. When this bit is set the measurement unit will start a new measurement over and over again till this bit is cleared.t

Table 24-9. Enhanced SDRAM Miscellaneous Register Field Descriptions (continued)

Field	Description
6 MA10_SHARE	MA10 share. Need to be enabled if MA10 address line is shared with other memory controllers address line. If enabled, the ESDRAMC will request the address line from the M3IF before issuing the precharge-all command (during auto-refresh cycles). After the precharge-all command is completed the ESDRAMC will remove the request. If MA10 share is disable, the ESDRAMC will execute the precharge-all command without requesting the MA10 address line, by assuming that MA10 address line is dedicated to ESDRAMC. 0 MA10 share disable 1 MA10 share enable
5 LHD	Latency hiding disable. This bit disables the command anticipation (latency hiding) mechanism. If this bit is set, the M3IF/ESDRAMC will operate in MIF1 (non-optimized) mode as shown in Figure 24-25 and Figure 24-26. The first memory command of a new access is sent to the memory only after the previous access is completed, For example: the last data word of a burst has been read or written. The reset value of this bit is 0, meaning that latency hiding is enabled (M3IF/ESDRAMC works in MIF2 mode) as described in Section 24.4.2, “Enhanced SDRAM Controller Optimization Strategy”). 0 Latency hiding enable 1 Latency hiding disable
4 MDDR_MDIS	LPDDR delay line measure disable. This is a read/write bit, that, if set, disables the delay line measure unit. After reset, this bit is cleared, meaning the delay line measure unit is enabled. The measure time period is estimated to be around 2000 clock cycles of the AHB HCLK. 0 LPDDR delay line measure unit is enabled. 1 LPDDR delay line measure unit is disabled.
3 MDDR_DL_RST	LPDDR delay line soft reset. This is a write only bit, that if set the delay line unit is reset. After reset the delay unit will automatically (if LPDDR_MDIS is cleared) start a new measurement. 0 LPDDR delay line is not reset. 1 LPDDR delay line is reset.
2 MDDR_EN	Enable DDR SDRAM device. 0 SDR SDRAM is used. 1 DDR SDRAM is used (either mobile or non-mobile).
1 RST	Software initiated local module reset. This bit generate local module reset to the ESDRAMC. Writing a 1 to RST bit results in a one cycle reset pulse to the controller. This bit is always read as 0. All ESDRAMC registers are not affected by the software reset, in order to keep the refresh mechanism active as initially configured, so the SDRAM/LPDDR data is not violated. A burst terminate command is issued to the memory after the soft reset (to terminate any active bursts, in order to prevent potential contention on the data pads). During the software reset an error response (HRESP[1]=1) is broadcast to all masters with active access to the ESDRAMC by the multi master memory interface (M3IF) module and the M3IF arbitration pipeline is cleared. For detailed information on error response see theM3IF module specification. **Note:** After soft reset, a precharge-all command must be issued prior to normal usage of the ESDRAMC. 0 Soft reset disabled. 1 Soft reset initiated.
0	Reserved

24.3.3.4 MDDR Delay Line *n* Configuration Debug Registers (ESDCDLY*n*, *n* = 1–4)

These debug registers controls the functionality of delay lines 1–4 (signals DQS[0–3]) used during read cycles. It enables software to override the delays of the DQS[*n*] lines that is used during read cycles of byte *n*. The delay line compensates for process variations, and produces a constant delay regardless of the

process, temperature and voltage. The bit assignments for the register are shown in Figure 24-8 and the field descriptions for the bit assignments are listed in Table 24-10.

Offset 0x0020 (ESDCDLY1)
0x0024 (ESDCDLY2)
0x0028 (ESDCDLY3)
0x002C (ESDCDLY4)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_1	0	0	0	0	0	0	0	DLY_OFFSET_n							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLY_ABS_OFFSET_n								DLY_REG_n							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-8. MDDR Delay Line *n* Configuration Debug Register

Table 24-10. Enhanced MDDR Delay Line *n* Control Register (ESDCDLY*n*) Field Descriptions

Field	Description
31 SEL_DLY_REG_*n*	SEL_DLY_REG_*n* bit selects the delay used by delay line *n*. It selects between the following options: • Delay line *n* value is a quarter of a cycle (measured) plus or minus the delay line *n* offset field (DLY_OFFSET_*n*) • Delay line *n* value is equal to the delay line *n* register value (DLY_REG_*n*). 0 Delay line *n* value is a quarter of a cycle (measured) plus or minus the delay line *n* correction factor field. 1 Bypass mode: Delay line *n* value is the value of delay line *n* register field (skipping the measurement).
30-24	Reserved
23-16 DLY_OFFSET_*n*	Delay line *n* offset value. The offset value is used only if SEL_DLY_REG_*n* is cleared. The offset value is used to compensate process-dependent misalignments between the DQS signal and the corresponding read data (in number of small delay intervals). The field represents positive and negative numbers using twos complement representation. This allows positive and negative offsets from the quarter cycle measured value. For example, to add 3 delay intervals to the measured delay, set this field to 00000011. To subtract 3 delay intervals, set it to 11111101.

Table 24-10. Enhanced MDDR Delay Line *n* Control Register (ESDCDLY*n*) Field Descriptions (continued)

Field	Description
15-8 DLY_ABS_OFFSET_*n*	Absolute delay offset. This field decides the specific delay line absolute delay in fraction of a cycle terms. The fraction is process and frequency independent. The delay line's delay is computed as (DLY_ABS_OFFSET_*n* / 512) * tCK. For example, the default value of 128 causes a quarter cycle delay. In bypass mode (SEL_DLY_REG_*n* = 1), this register has no effect on the delay. **Note:** Not all changes have an effect on the actual delay. If the requested change is smaller then the delay line resolution (1 delay interval, which varies between 20 pSec in the best case to 50 pSec in the worst case), then no change occurs. **Note:** Updating this field requires two cycles to update in the delay line.
7-0 DLY_REG_*n*	This field is the delay (in number of buffer units) that is used by delay line *n*, if the SEL_DLY_REG_*n* bit is set. Since the delay depends on process, temperature and voltage, a given value of this field produces variable actual delay values. This field contains only positive numbers.

24.3.3.5 MDDR Delay Line 5 Configuration Debug Register (ESDCDLY5)

This debug register controls delay line 5 functionality, that is data bus delay used during write cycles. It allows software to manually set the delay of the data bus, during write cycles. The bit assignments for the register are shown in Figure 24-9 and the field descriptions for the bit assignments are listed in Table 24-11.

Offset 0x0030 (ESDCDLY5) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SEL_DLY_REG_5	0	0	0	0	0	0	0	DLY_OFFSET_5							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLY_ABS_OFFSET_5								DLY_REG_5							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24-9. MDDR Delay Line 5 Configuration Debug Register

Table 24-11. Enhanced MDDR Delay Line 5 Control Register (ESDCDLY5) Field Descriptions

Field	Description
31 SEL_DLY_REG_5	SEL_DLY_REG_5 bit selects the delay used by delay line 5. It selects between a quarter of a cycle (measured) plus or minus the delay line 5 offset field (DLY_OFFSET_5) and delay line 5 register value (DLY_REG_5). 0 Delay line 5 value is a quarter of a cycle (measured) plus or minus the delay line 5 correction factor field. 1 Bypass mode: Delay line 5 value is the value of delay line 5 register field (skipping the measurement).
30-24	Reserved

Table 24-11. Enhanced MDDR Delay Line 5 Control Register (ESDCDLY5) Field Descriptions (continued)

Field	Description
23-16 DLY_OFFSET_5	This field is the delay line 5 offset value. The offset value is used only if SEL_DLY_REG_5 is cleared. The offset value is used to compensate process-dependent misalignments between the DQS signal and the corresponding read data (in number of small delay intervals). The field represents positive and negative numbers using twos complement representation. This allows positive and negative offsets from the quarter cycle measured value. For example, to add 3 delay units to the measured delay, set this field to 00000011, to subtract 3 delay units, set it to 11111101.
15-8 DLY_ABS_OFFSET_5	Absolute delay offset. This field decides the specific delay line absolute delay in fraction of a cycle terms. The fraction is process and frequency independent. The delay for the delay line would be (DLY_ABS_OFFSET_5 / 512) * tCK. So for the default value of 128 we get a quarter cycle delay. In Bypass mode (SEL_DLY_REG_5 = 1), this register has no effect on the delay. **Note:** Not all changes will have effect on the actual delay. If the requested change is smaller then the delay line resolution (1 delay unit, which varies between 20 pSec in the best case to 50 pSec in the worst case), then no change will occur. Remark: Updating this field requires two cycles to update in the delay line.
7-0 DLY_REG_5	This field is the delay (in number of buffer units) that is used by delay line 5, if the SEL_DLY_REG_5 bit is set. Since the delay depends on process, temperature and voltage, a given value of this field produces variable actual delay values. This field contains only positive numbers.

24.3.3.6 MDDR Delay Line Cycle Length Debug Register (ESDCDLYL)

This read-only register's value represents the number of inverters required to achieve a delay of one clock cycle, as a function of the IC conditions (temperature, voltage, frequency, process). The reset value is unknown, because after reset the register value is updated from the measured delay. The bit assignments for the register are shown in Figure 24-10 and the field descriptions for the bit assignments are listed in Table 24-12.

Offset 0x0034 (ESDCDLYL) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	QTR_CYCLE_LENGTH							
W																
Reset	0	0	0	0	0	0	0	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figure 24-10. MDDR Delay Line Cycle Length Debug Register

Table 24-12. Enhanced MDDR Delay Line Cycle Length Debug Register (ESDCDLY1) Field Descriptions

Field	Description
31-8	Reserved
7-0 QTR_CYCLE_LENGTH	This debug register shows the number of delay intervals that are being used to create the needed delay for Delay Line 5 (for write operations). Changes to the DLY_ABS_OFFSET_5 or DLY_OFFSET_5 fields of the ESDCDLY5 register cause this value to change, as long as the minimal or maximal delay is not reached.

24.4 Functional Description

The ESDRAMC's general operating characteristics are addressed in this section, including the following topics:

- Section 24.4.1, "ESDRAMC Configurable Timing Parameters"
- Section 24.4.2, "Enhanced SDRAM Controller Optimization Strategy"
- Section 24.4.3, "Address Multiplexing"
- Section 24.4.4, "Multiplexed Address Bus During Precharge or Load Mode Registers Modes"
- Section 24.4.5, "Refresh"
- Section 24.4.6, "Low Power Operating Modes"
- Section 24.4.7, "SDRAM (SDR and LPDDR) Command Encoding"

In Section 24.4.8, "Normal Read/Write Mode," through Section 24.4.12, "Load Mode Register Mode," the different ESDRAMC operating modes are described. The discussion includes details on basic operation, relationship to SDRAM/LPDDR operating modes, and any special precautions which need to be observed. State and timing diagrams are included where appropriate.

The enhanced SDRAM controller is designed to support a broad range of JEDEC standard SDRAM/LPDDR configurations including devices of 64-, 128-, 256-, 512-Mbit and 1-Gbit densities. The design support memory devices with data widths of 16 bits. Table 24-13 summarizes the devices supported by the design. Only 4-bank devices are supported. 133-MHz system bus operation is possible with PC133-compliant single– or double–data rate memory devices.

24.4.1 ESDRAMC Configurable Timing Parameters

Table 24-14 summarizes the enhanced SDRAM controller configurable set of timing parameters for SDRAM and LPDDR devices. These parameters are set in the eSDRAMC configuration registers (ESDFG*n*). Figure 24-11 through Figure 24-22 are timing diagrams that show the significance of these parameters.

Table 24-13. JEDEC Standard Single/Double–Data Rate SDRAMs

SDRAM Configurations—4-Bank Devices										
Size	64 MBit		128 MBit		256 MBit		512MBit[1]		1-GBit[1]	
Bus size	16	32	16	32	16	32	16	32	16	32

Table 24-13. JEDEC Standard Single/Double–Data Rate SDRAMs (continued)

SDRAM Configurations—4-Bank Devices										
Depth	4M	2M	8M	4M	16M	8M	32M	16M	64M	32M
Refresh Rows	4096	4096	4096	4096	8192	8192	8192	8192	16384	16384
Refresh rate (us)	15.6	31.25	15.6	15.6	7.81	7.81	7.81	7.81	3.91	3.91
Refresh cycles	2	1	2	2	4	4	4	4	8	8
Row Address	12	11	12	12	13	13	13	13	14	14
Col. Address	8	8	9	8	9	8	10	9	10	9

[1] Not ratified by JEDEC—row-column organization may change.

Table 24-14. Configurable SDRAM/LPDDR Timing Parameters

Symbol	Description	Relevancy	Typical Values at 133 MHz
t_{MRD}	Load mode register command to active or refresh command	Always	2 cycles
t_{WR}	Write recovery time (write to precharge)	Commands to same bank	2 cycles
t_{RAS}	Active to precharge command	Commands to same bank	6 cycles
t_{RRD}	Active bank A to active bank B command	Commands to different banks	2 cycles
t_{CAS}	Read to data out period (known as CAS latency)	Always	3 cycles
t_{RP}	Precharge command period	Commands to same bank	3 cycles
t_{RCD}	Active to read or write delay	Commands to same bank	3 cycles
t_{RC}	Active to active command period	Commands to same bank	10 cycles[1]
t_{WTR}	LPDDR read to write command delay	Commands to same bank	2 cycles
t_{XP}	LPDDR exit power-down to next valid command delay	Always	4 cycles

1. Not all of the capin or cmpout signals are implemented in all GPT instances. Users must check which signals are present in the IOMUX table.

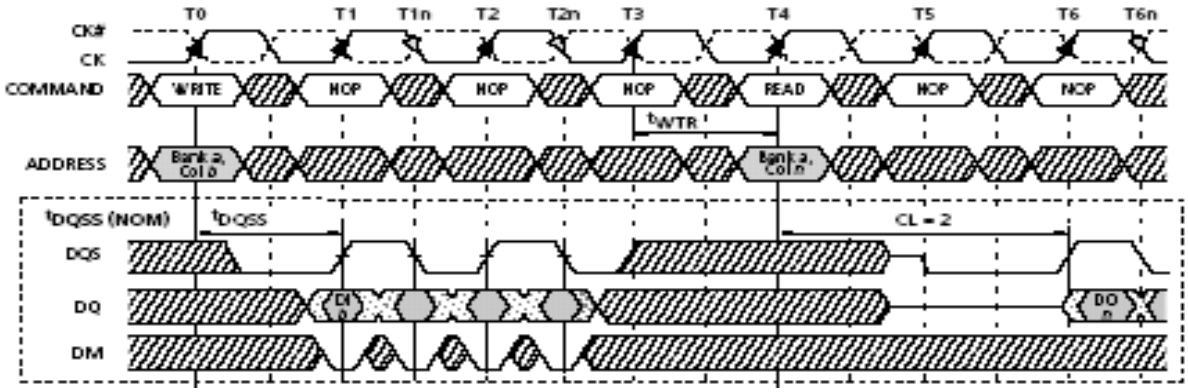


Figure 24-11. tWTR—Write-to-Read Command Delay Timing

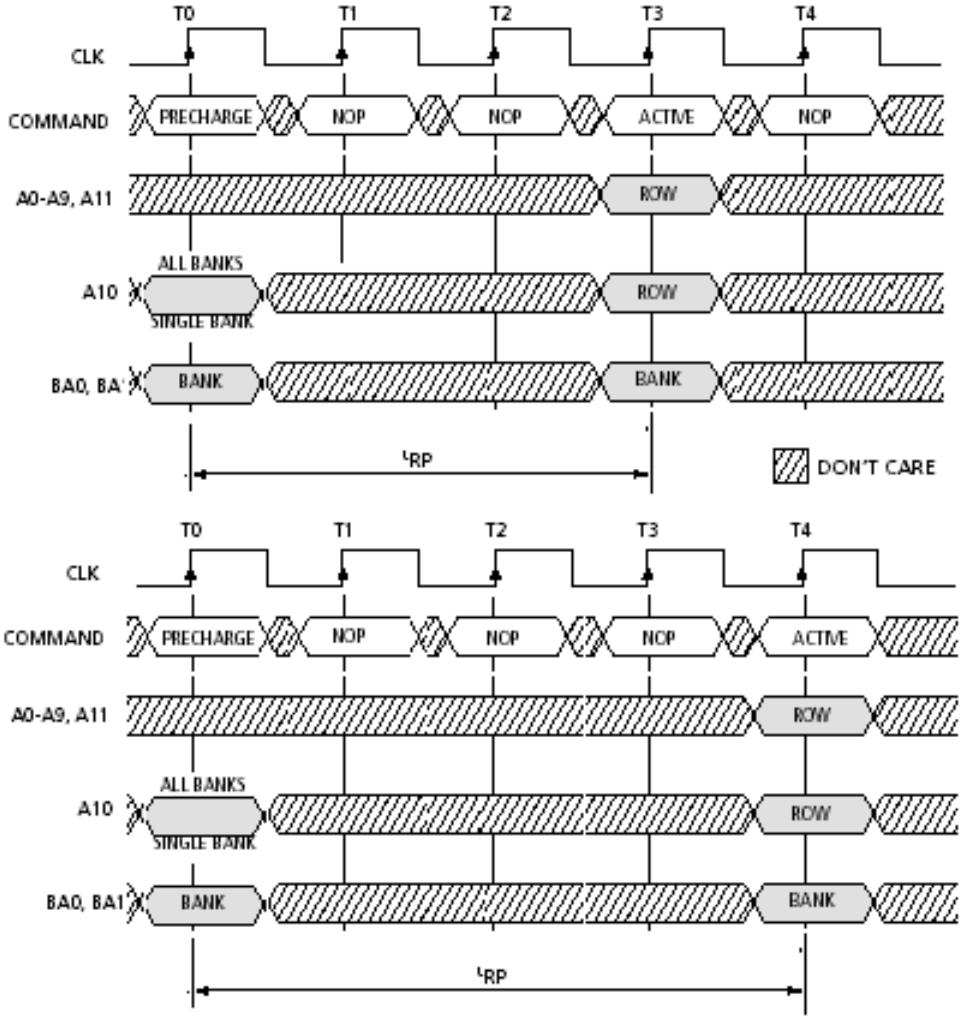


Figure 24-12. tRP—Precharge Delay Timing

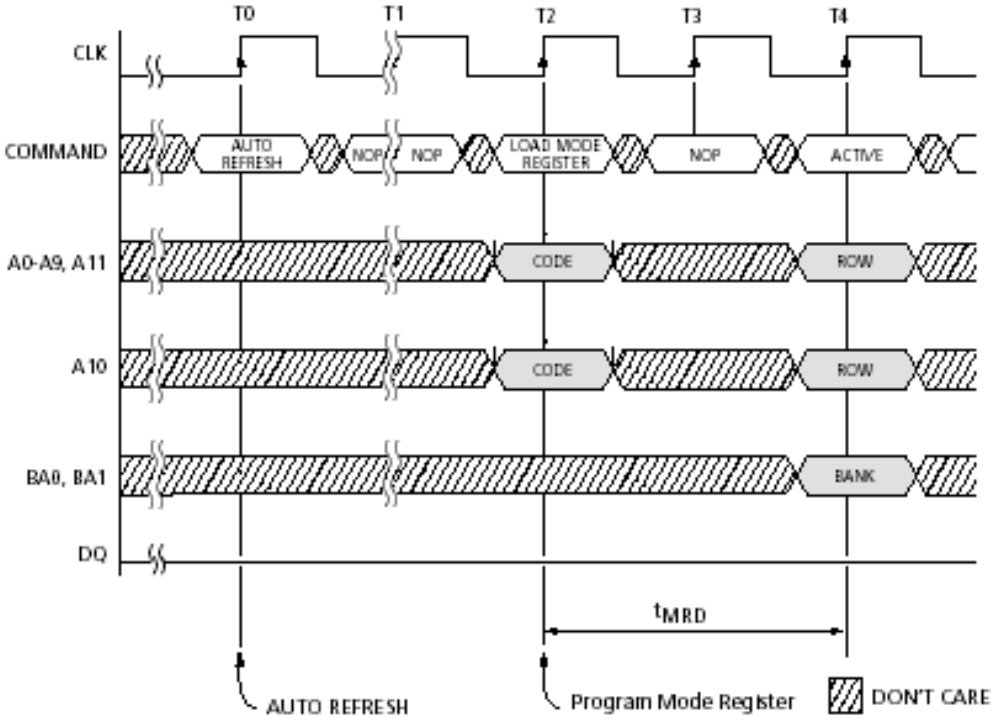


Figure 24-13. tMRD—SDRAM Load Mode Register to Active Command Timing Diagram

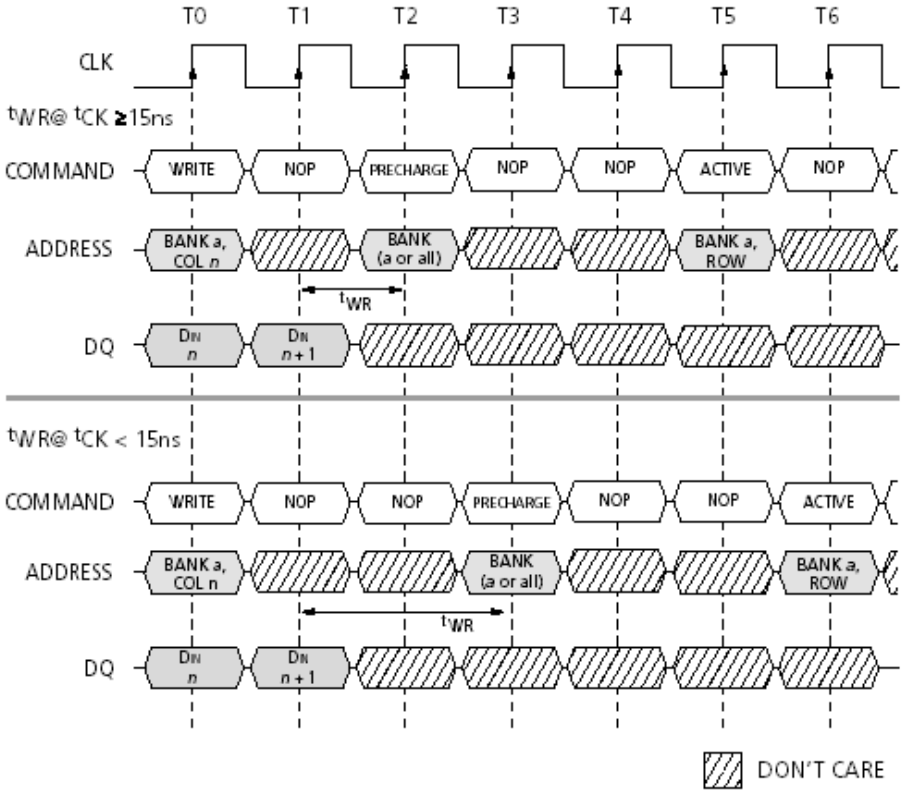


Figure 24-14. tWR—Write to Precharge Timing Diagram

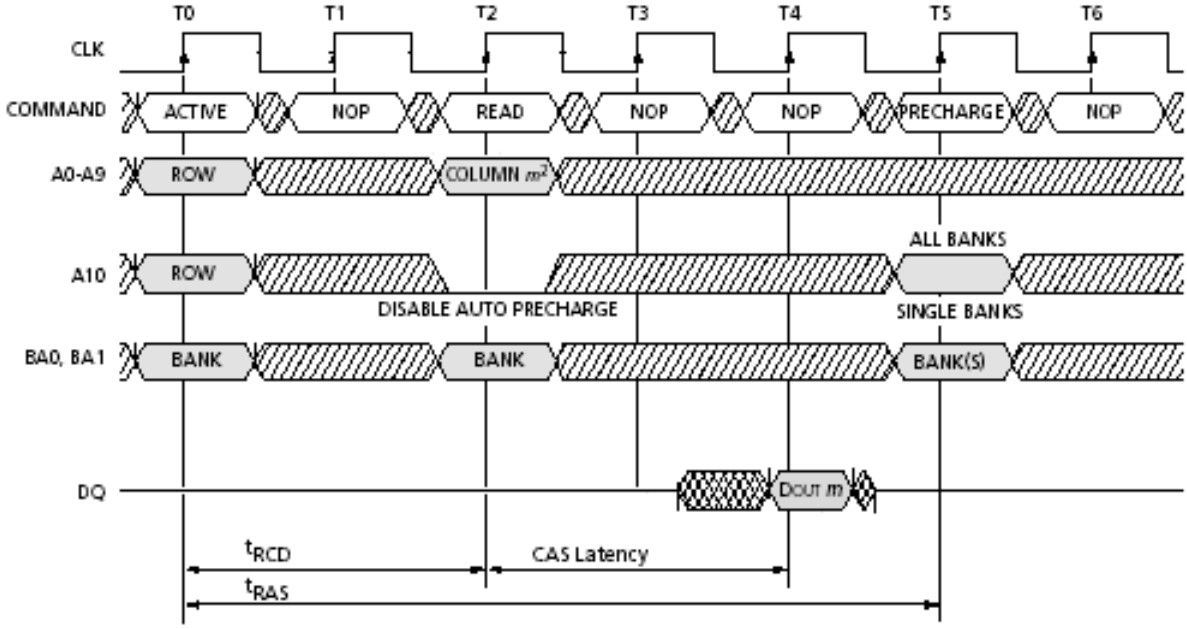


Figure 24-15. t_{RAS}—SDRAM Active to Precharge Command Timing Diagram

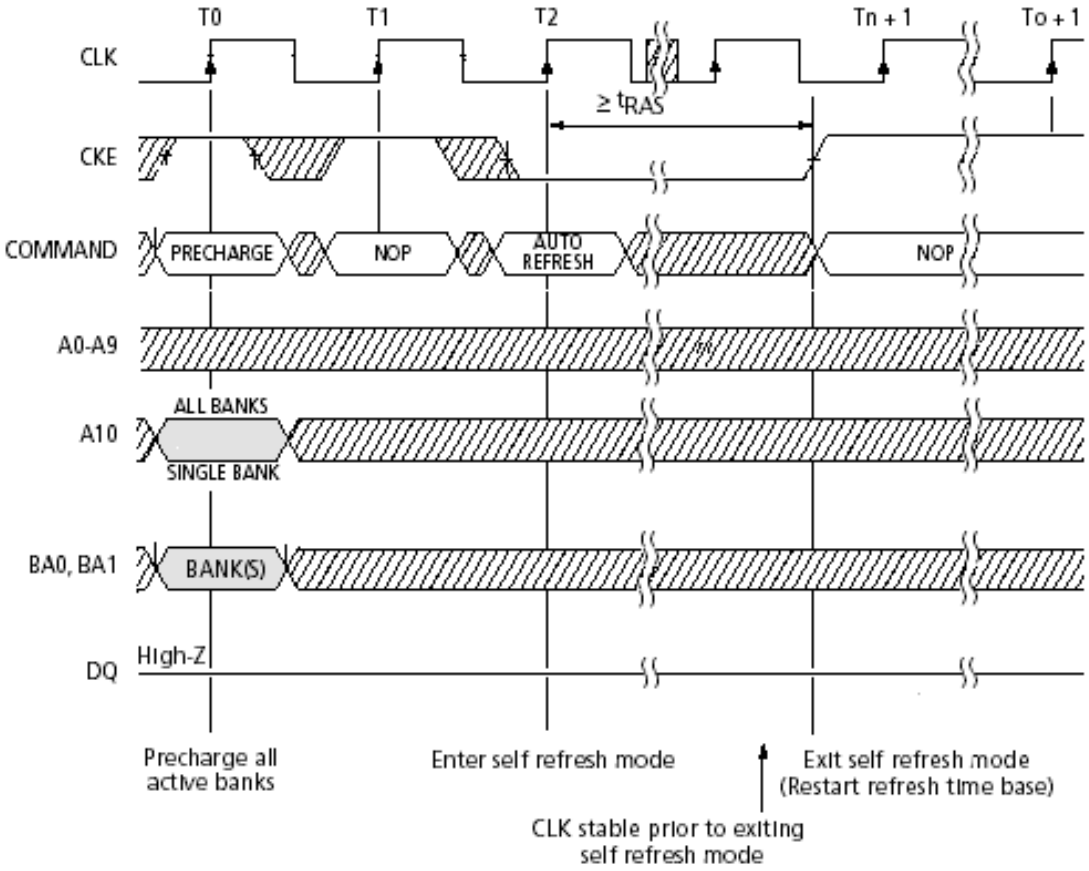


Figure 24-16. t_{RAS}—Self-Refresh Mode Minimum Time Period

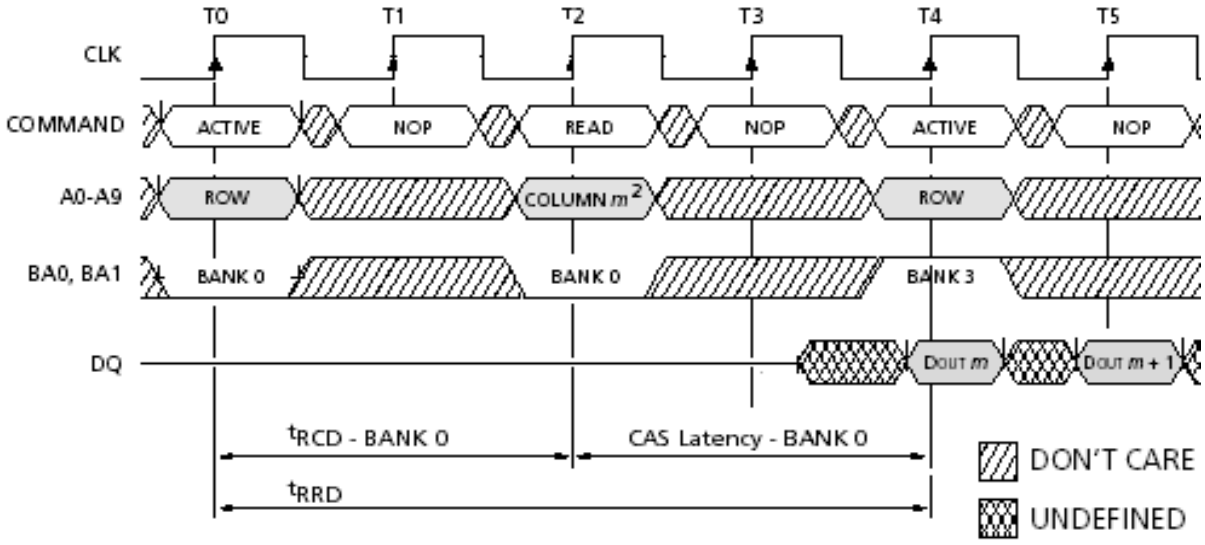


Figure 24-17. t_{RRD}—Alternating Bank Read Access

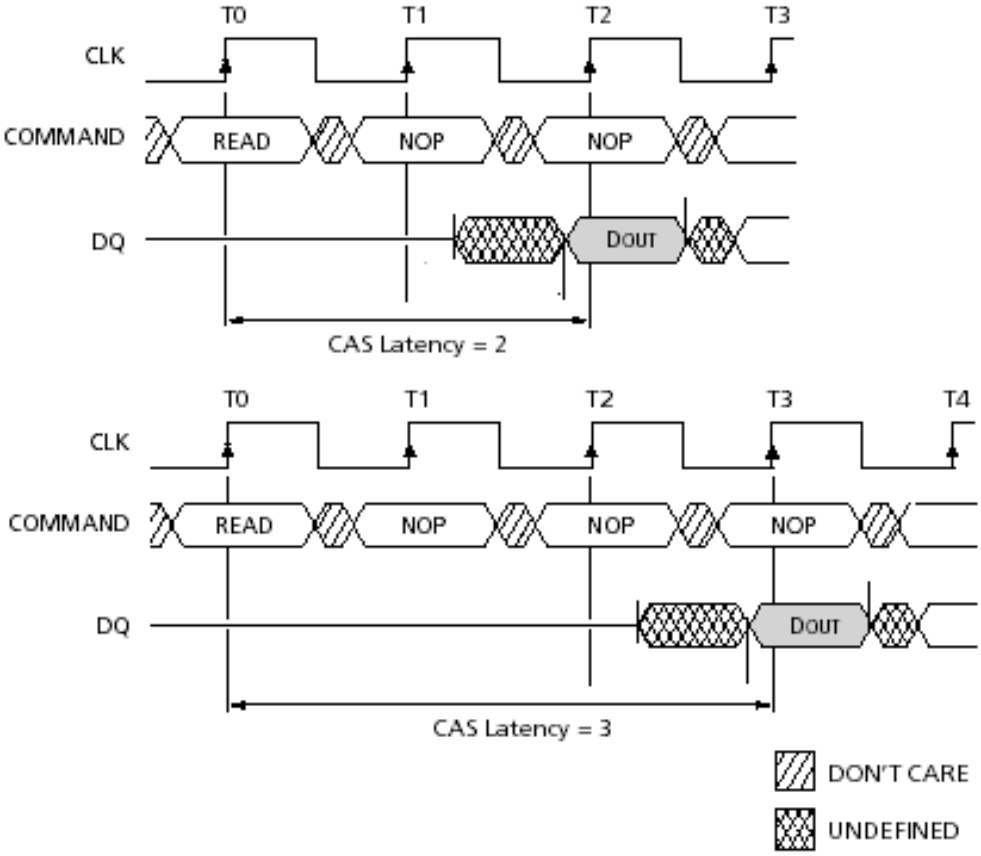


Figure 24-18. $_{SDR}$ CAS Latency Timing

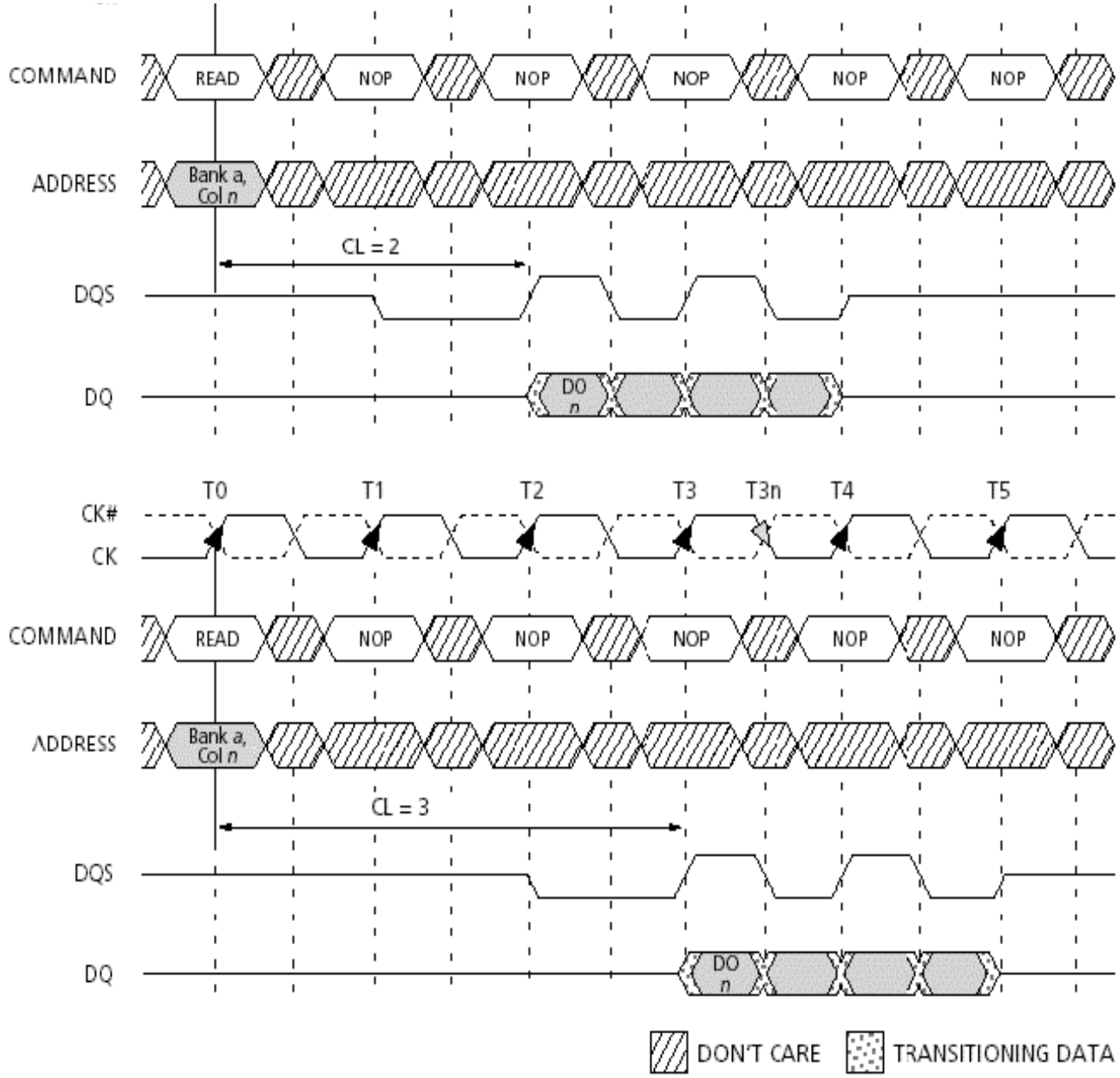


Figure 24-19. Mobile LPDDR CAS Latency Timing

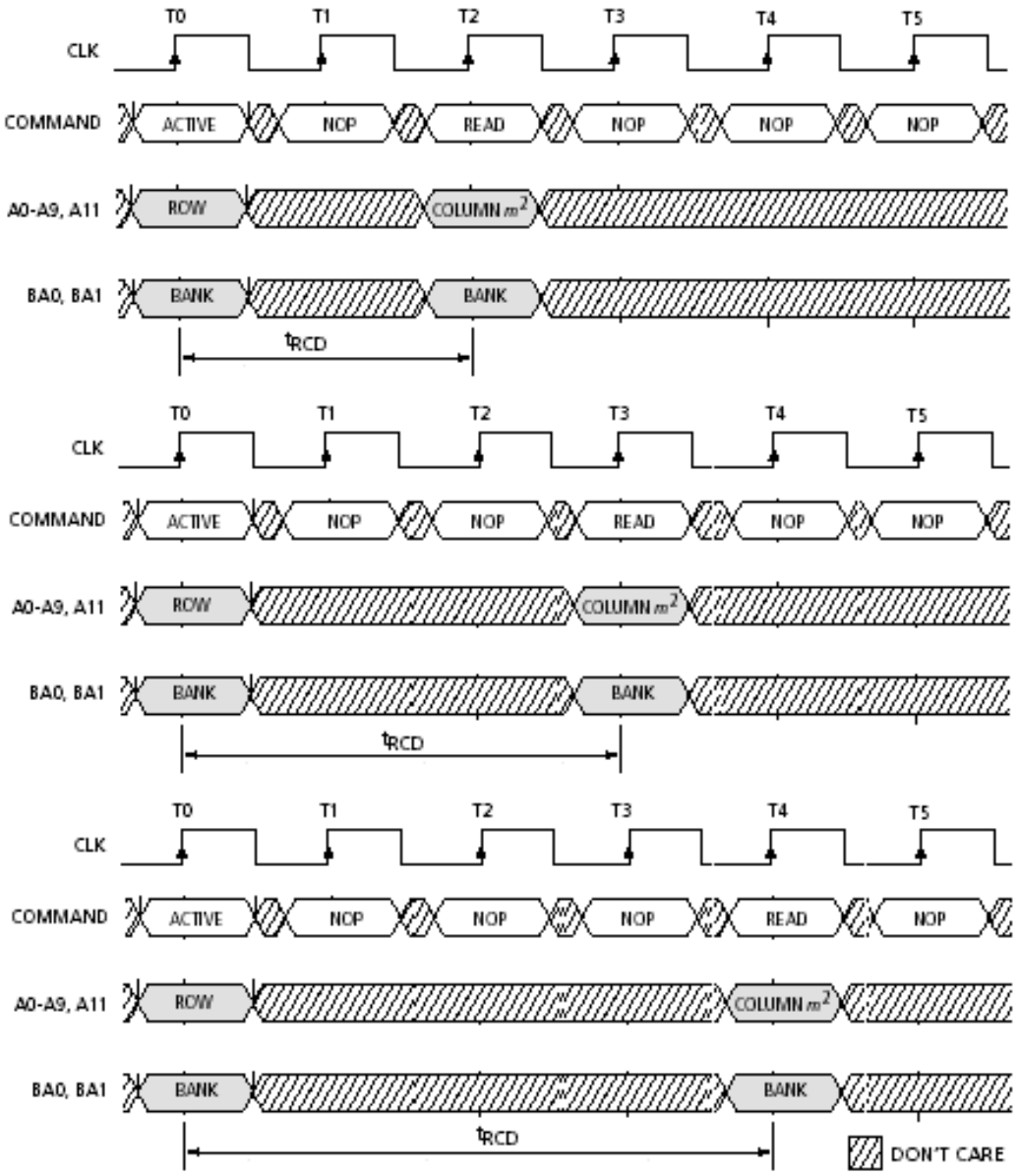


Figure 24-20. t_{RCD}—Row-to-Column Delay Timing

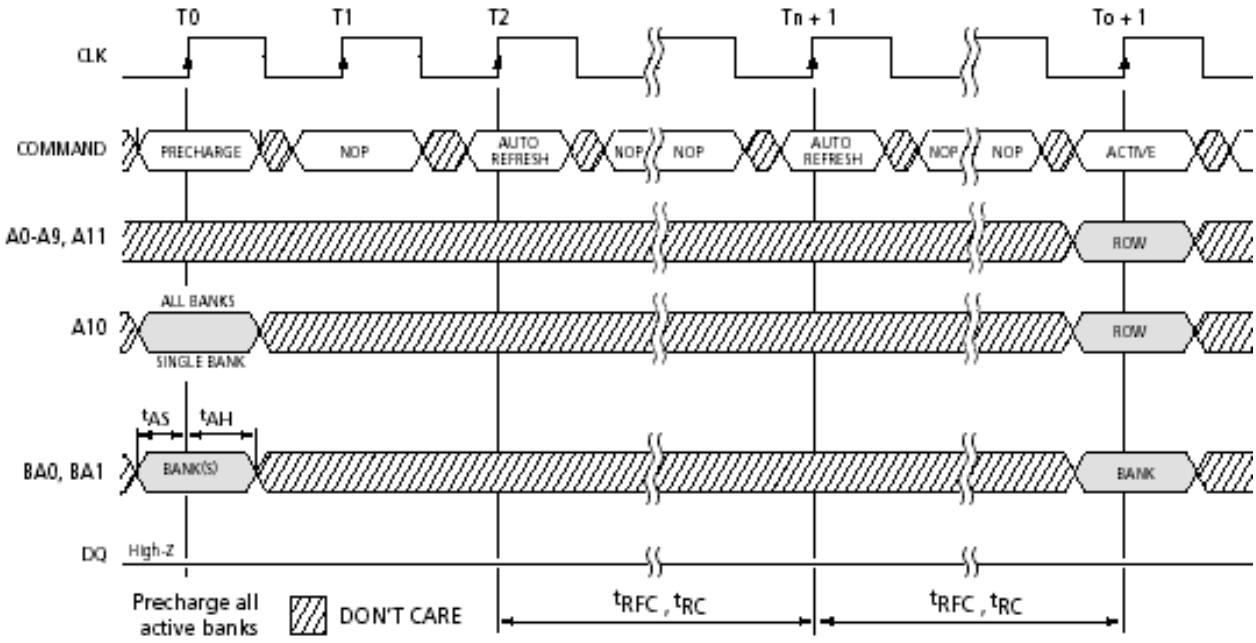


Figure 24-21. t_{RC}—Row Cycle Timing

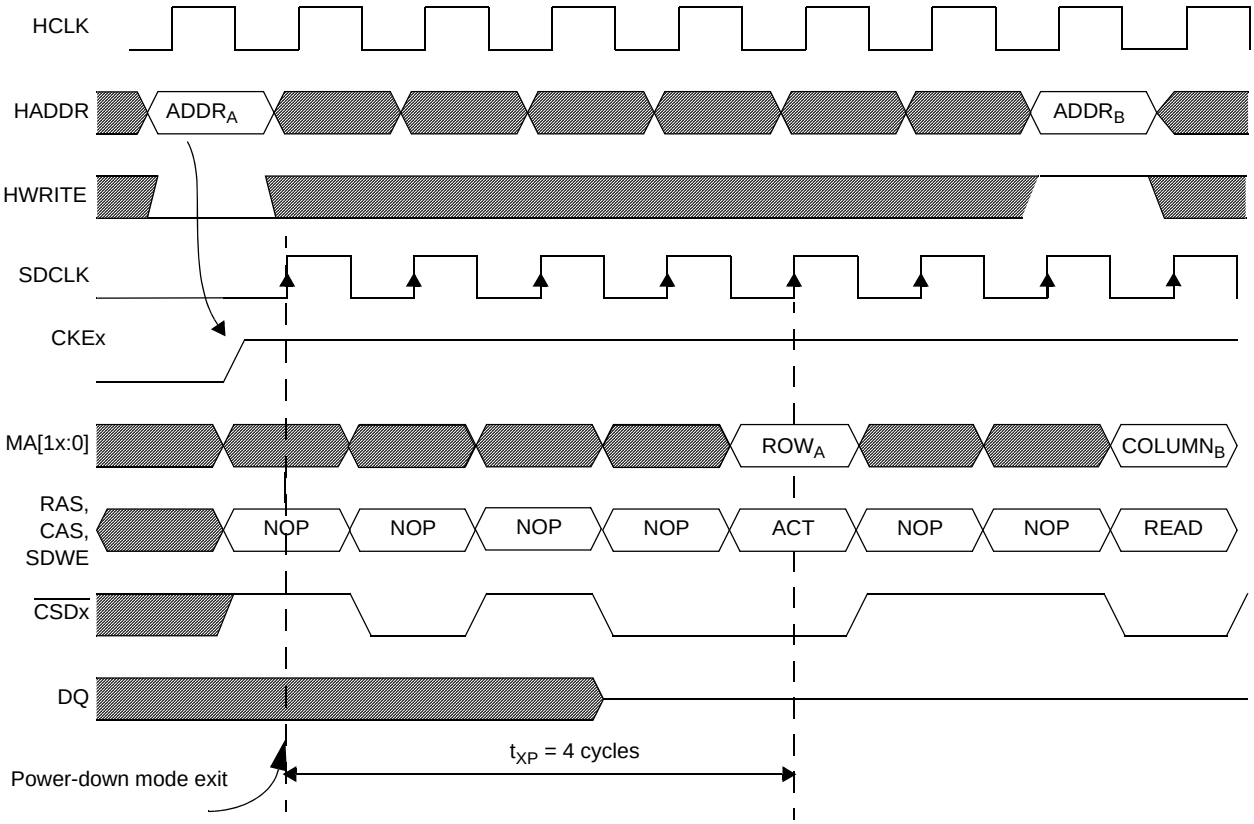


Figure 24-22. t_{XP}—New Command After Power-Down Exit (4 cycles)

24.4.2 Enhanced SDRAM Controller Optimization Strategy

SDRAM (SDR and LPDDR) memories provide high speed access by hiding the latency of consecutive memory accesses through a pipeline interface architecture. The resulting high bandwidth is achieved with a rather complex command interface and a large number of timing constraints that must be kept by the memory controller.

SDRAM, LPDDR and memories are organized in several independent banks. By issuing a row address and bank number to the memory device, the corresponding memory page is activated (opened). Consecutive READ commands (together with the column address) into this memory page (same row address) have a low latency. Accessing another page in the same bank requires closing the open page by a precharge command, followed by the activation (ACTIVE command) of the new memory page (new row address).

Figure 24-23 and Figure 24-24 show examples of SDR and LPDDR SDRAM read bursts. Initially the cell in [COL a, ROW a] is active/open. Accessing the cell position [COL b, ROW b] requires first closing the old cell in [COL a, ROW a], which is done with a PRECHARGE command (P in the figure). Then the access row address (ROW b) can be activated (A in the figure) before the column address (COL b) is passed (READ command R). Timing restrictions are as follows:

- The ACTIVE command can be issued only tRP cycles after the PRECHARGE command.
- The READ command can be issued only tRCD cycles after the ACTIVE command.
- The first data is available only tCAS cycles after the READ command has been issued.

Hence, in those examples a 4-word (8-word in LPDDR is converted to 4-word with twice data width) read burst requires 9 cycles (6 cycles latency from the PRECHARGE command to the first word on the external bus) or 6-1-1-1. A read access from an already-open row is shown as well: the read burst from target cell [ROW b, COL c] onward takes only 5 cycles (2 cycles latency from the READ command to the first word on the external bus) or 2-1-1-1.

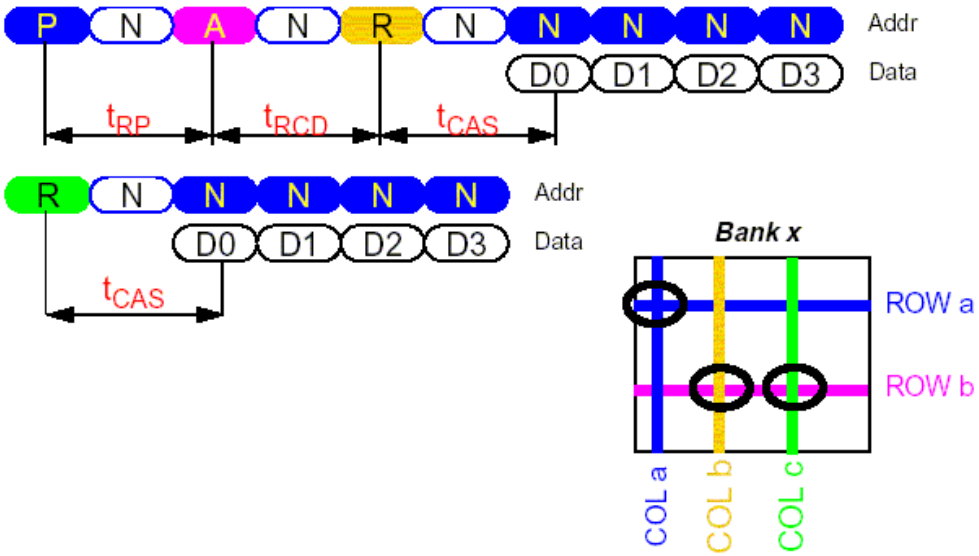


Figure 24-23. SDR SDRAM Read Burst Command Sequence Example

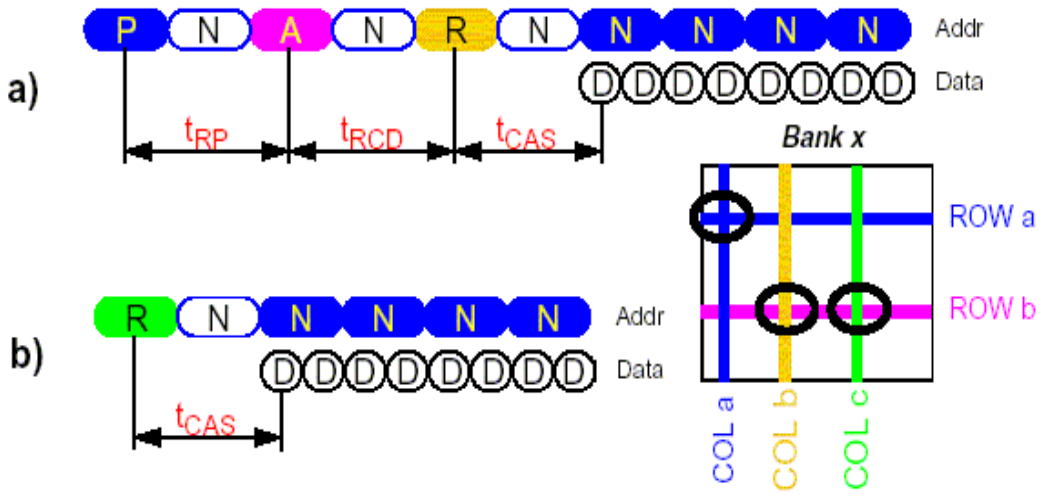


Figure 24-24. LPDDR SDRAM Read Burst Command Sequence Example

Since the ESDRAMC handles two devices that both feature 4 independent memory banks, there are opportunities for the controller to optimize the access timing of consecutive memory accesses by hiding as much as possible the latency of the first data word in a burst. Table 24-15 summarizes cases where latency hiding is possible.

Table 24-15. Possibilities for Latency Hiding

Current Burst Access	Next Burst Access
SDRAM bank *x*	SDRAM bank *y*
SDRAM bank *x* (row *y*, col *z*)	SDRAM bank *x* (row *y*, col *w*)
LPDDR SDRAM bank *x*	LPDDR SDRAM bank *y*

Figure 24-25 (SDR) and Figure 24-26 (LPDDR) show the no-optimization (MIF1) and medium-level optimization (MIF2) strategies, respectively. For SDR SDRAM each strategy, two examples for two consecutive read accesses are given:

- An 8-word burst from the SDRAM with CAS latency of 2 cycles followed by an 8-word burst from the same bank and row (different column) with CAS latency of 2 cycles.
- An 8-word burst from one bank in the SDRAM followed by an 8-word burst from a different bank to the same SDRAM.

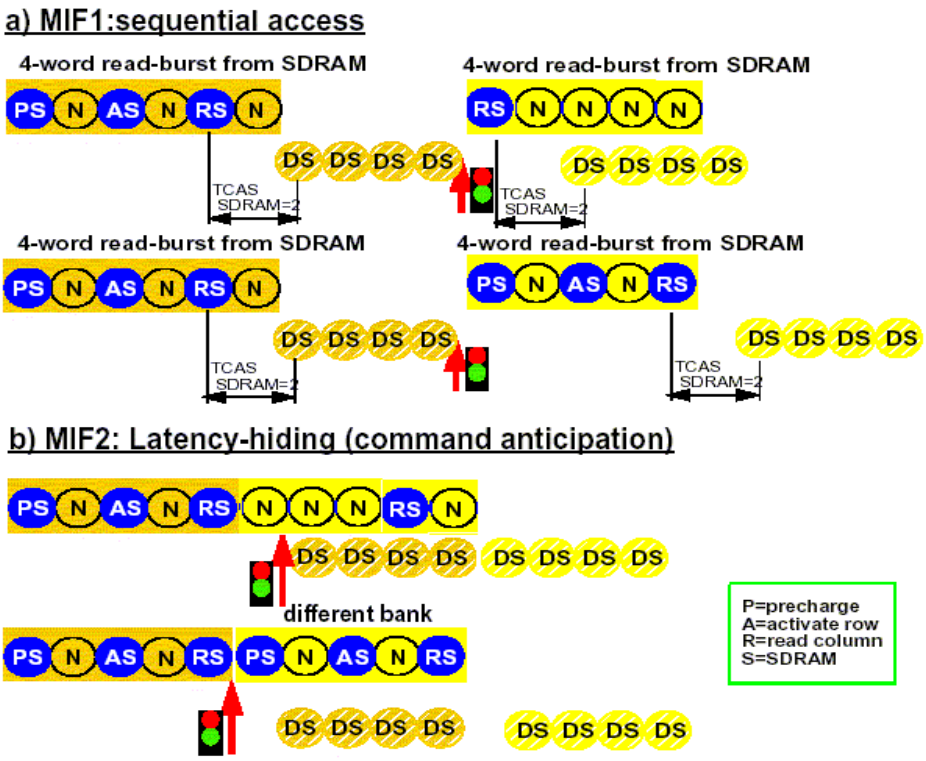


Figure 24-25. SDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

For LPDDR SDRAM each strategy, one example for two consecutive read accesses is given:

- An 8-word burst from one bank in the LPDDR SDRAM followed by an 8-word burst from a different bank to the same LPDDR SDRAM.

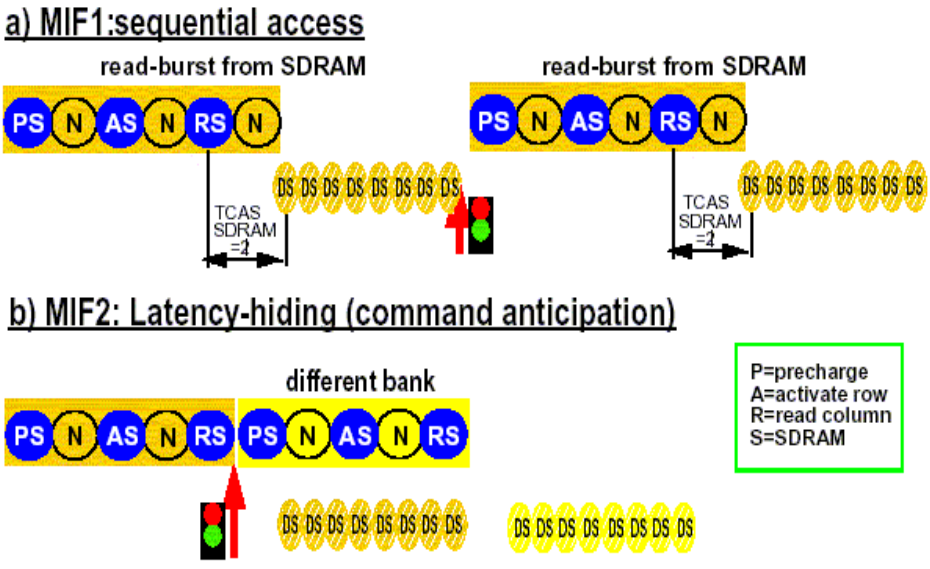


Figure 24-26. Mobile LPDDR SDRAM Optimization Strategies—MIF1 and MIF2 Examples

24.4.2.1 MIF1—No Optimization/Sequential Accesses

This is the non-optimized case shown in Figure 24-25 and Figure 24-26. The first memory command of a new access is sent to the memory only after the previous access is completed, for example, the last data word of a burst has been read or written. It can be seen in this example that although the 2 memory accesses follow with no delay between them, the bandwidth usage for the command (address) and data busses is far from being optimal. This no optimization/ sequential accesses occurs in the following cases;

- Only one master active/present in a given system - in this case only sequential commands are possible since the given master need to receive/send (read/write) all data's for one access before it can proceed to the next access, for example, command anticipation is not possible.
- Low density accesses, such as non-overlapped/consecutive/continuous requests, means that the next SDRAM request starts after the previous request is completed. In this low SDRAM utilization only sequential accesses occurs.
- Large number of SINGLE or INCR (aborted after one data) accesses instead of burst type accesses. SINGLE/INCR see the AMBA AHB bus protocol.
- The LHD (latency hiding disable) bit is set.

24.4.2.2 MIF2—Medium Level Optimization/Command Anticipation

This strategy is shown in Figure 24-25 and Figure 24-26. As soon as the address and command bus (referred to as the SDRAM control bus) is no longer used for issuing the previous memory access command, the controller can use this bus to start issuing the PRECHARGE/ACTIVE commands for the next scheduled memory access while the previous one is still active on the data bus. Two conditions limit the use of this optimization at a given time:

- Memory timing constraints must not be violated.
- The execution of the previous command should not be affected (for example, truncated).

This approach allows for hiding a part of the latency for the first data word or even the complete hiding of the latency in case that the burst length exceeds the maximal command sequence length.

24.4.2.3 Latency Hiding

The ESDRAMC optimization is based on command anticipation (MIF2), so that the next access control phase (memory address and command) is driven during the previous access data phase (data flow from/to the memory). This creates an overlap between accesses which hides part or all of the latency. The ESDRAMC module optimizes data bus utilization in accesses to memory according to the system initial configuration. Some examples for different memory/access conditions are as follows.

The timing diagrams in Figure 24-27 and Figure 24-28 show latency hiding for SDR and mobile DDR respectively in the case where a read miss is followed by a read hit. In Figure 24-27, the miss burst read from bank A in SDR memory is followed by a hit burst read request from the same chip select. The second READ command is issued during the first access data phase, so the first data of the second access (D10) is valid immediately after the last data of the first access (D4 for the SDR and D8 for the LPDDR). The CAS latency (tCAS) is set to 2 cycles. The second access latency is fully hidden during the first access data phase.

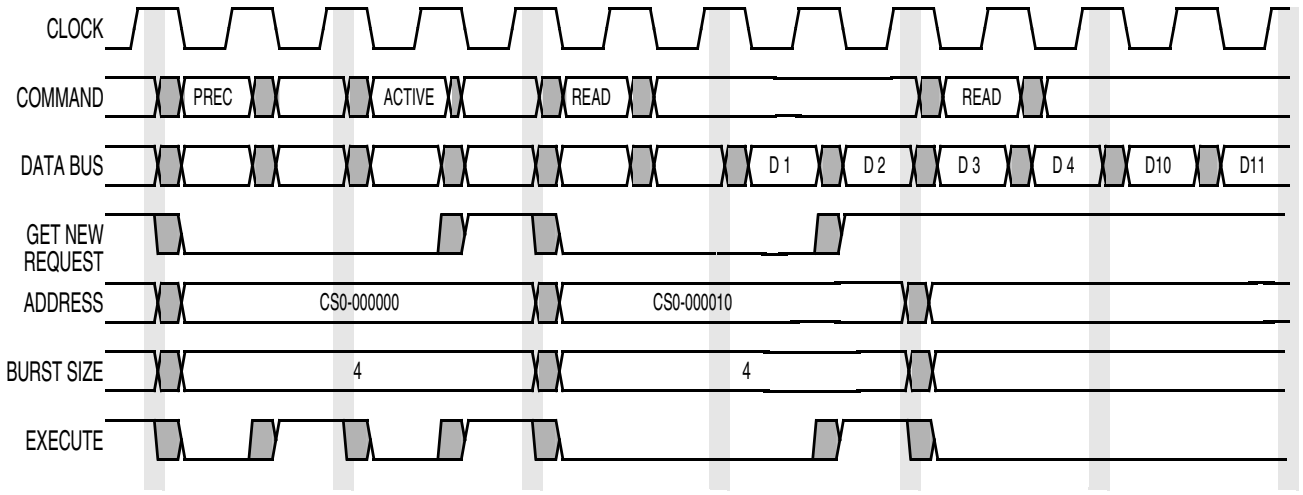


Figure 24-27. SDR Simple Read after Read Latency Hiding Timing Diagram

Figure 24-28 shows the case where a miss burst read from bank A in one mobile / low power DDR memory device is followed by a hit burst read request from another LPDDR on the other chip select. In this case, an empty cycle is needed on the data bus to prevent contention of DQS signals from two different LPDDRs on two different chip selects (contention between the last two data cycles of the first transfer and the preamble of the second transfer).

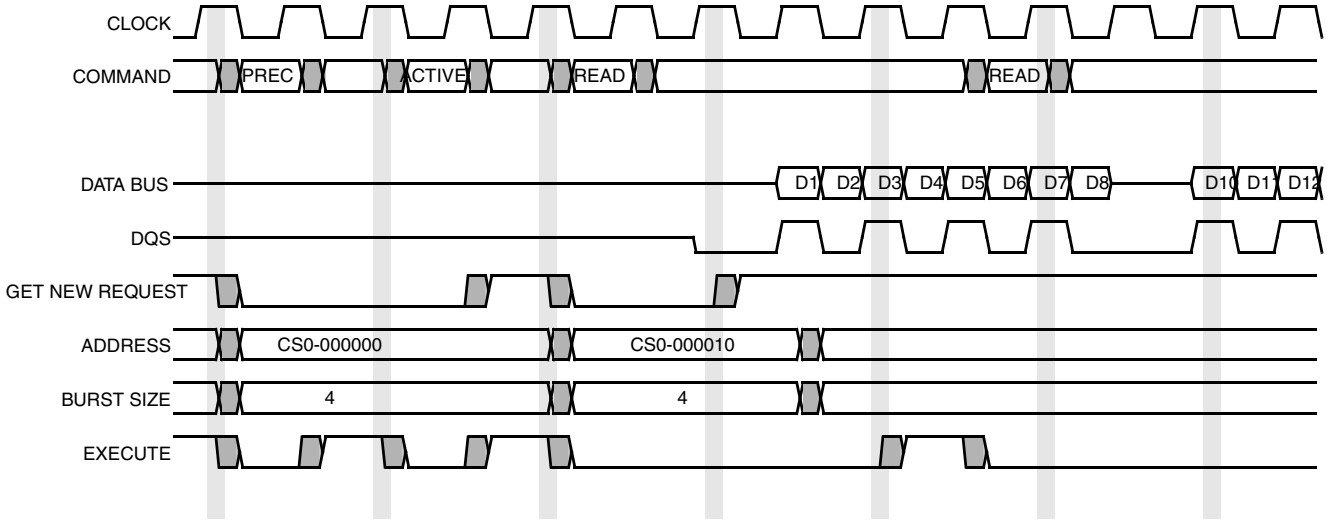


Figure 24-28. Mobile DDR Simple Read After Read Latency Hiding Timing Diagram

The timing diagrams in Figure 24-29 and Figure 24-30 show latency hiding for SDR and mobile DDR memories respectively in case a burst read to CSD0 is followed by a miss burst write access to CSD1. In the SDR case (Figure 24-29), both CSD burst lengths are 4; while in the LPDDR case the burst lengths are 8 but from the system point of view appear as burst length 4 with double bus width. Due to command anticipation the second access latency is fully hidden during the first access data phase. The first WRITE command to CSD1 (D10) is issued immediately after the last data from CSD0 (D4 for SDR and D8 for

LPDDR), even though the access to CSD1 is a miss access CSD0 CAS latency is set to 3 cycles in these examples.

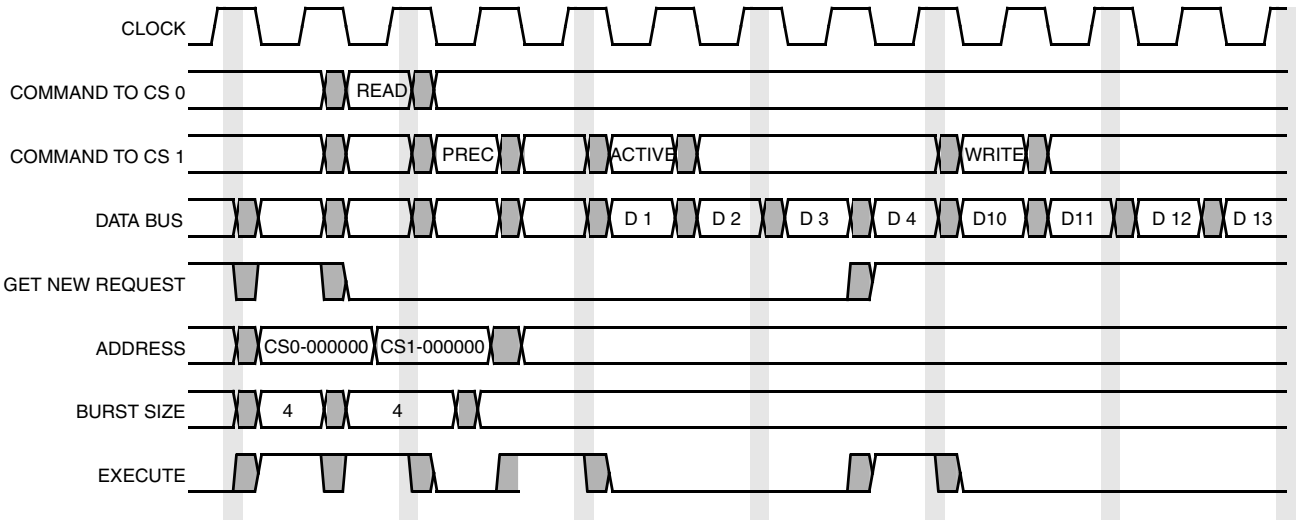


Figure 24-29. SDR Miss Write to CSD1 after Read from CSD0

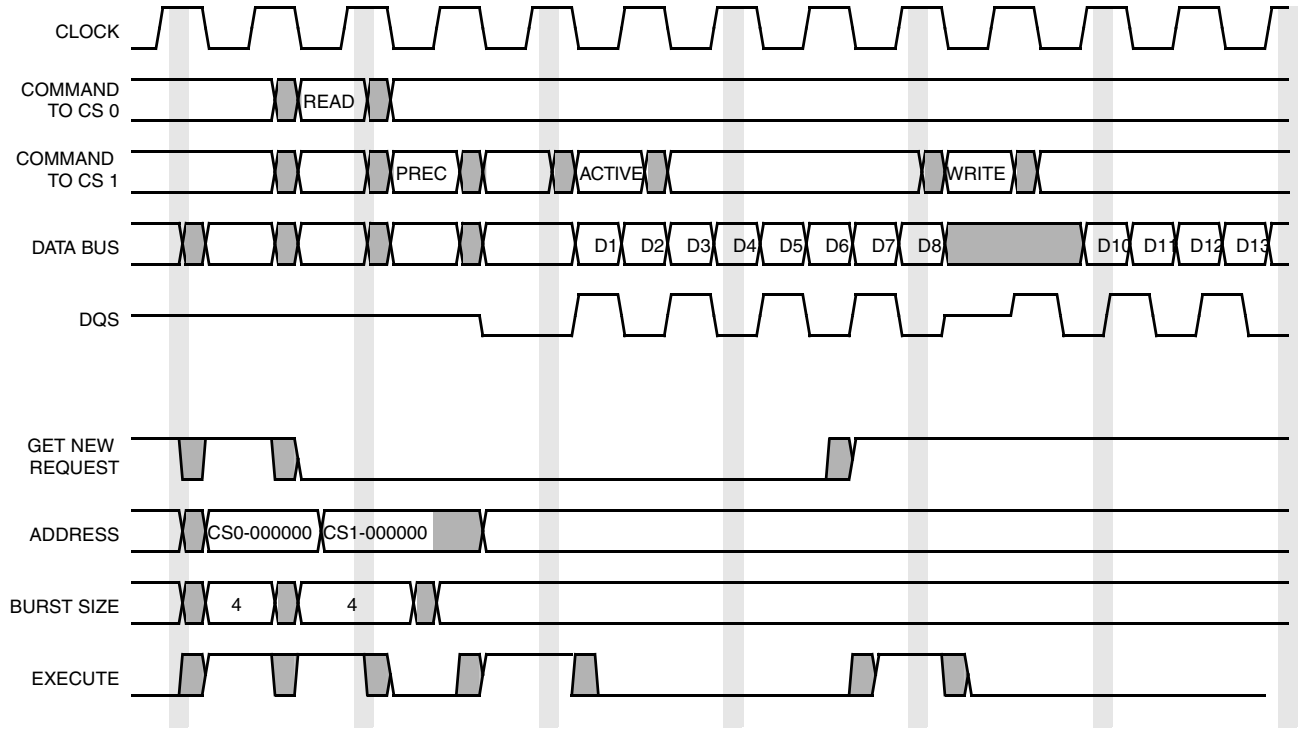


Figure 24-30. Mobile DDR Miss Write to CSD1 after Read from CSD0

24.4.3 Address Multiplexing

24.4.3.1 Multiplexed Address Bus

Table 24-16 illustrates several examples on how a CPU address is scrambled by the enhanced SDRAM controller to implement a contiguous address space.

Table 24-16. CPU to SDRAM/LPDDR Translation

CPU ADDRESS	16-bit SDRAM[1]
A25	—
A24	BA1
A23	BA0
A22	R11
A21	R10
A20	R9
A19	R8
A18	R7
A17	R6
A16	R5
A15	R4
A14	R3
A13	R2
A12	R1
A11	R0
A10	C9
A9	C8
A8	C7
A7	C6
A6	C5
A5	C4
A4	C3
A3	C2
A2	C1
A1[2]	C0
A0[3]	—

1 For the SDRAM example a memory configuration with 10 columns and 12 rows is illustrated. The address translation is based on the following concept, COLUMN - ROW - BANK.

2 CPU A1 defines how the data masks are driven—it is used as the byte enable for non-word accesses. This bit has a regular/normal use only in case of 16-bit memory, while CPU A0 defines the 2 bytes (low and high) in the 16-bit word.

3 CPU A0 defines how the data masks are driven.

The enhanced SDRAM controller multiplexed address bus is aligned to the column addresses so that address line A1 always appears on pin MA0 for 16-bit memory devices. With this alignment, the “folding point” in the multiplexer is driven solely by the number of column address bits. Column bus widths of 8 to 11 bits are supported. Table 24-17 summarizes the multiplex options supported by the controller for 16-bit devices. Column addresses through A10 are driven regardless of the multiplexer configuration, although some of the lines are unused for the smaller page sizes.

Table 24-17. Address Multiplexing by Column/Row Width for 16-bit Devices

Device Pins	ESDRAMC Pins	16-bit SDR and LPDDR SDRAM Memory Device									
		64 Mbit		128 Mbit		256 Mbit		512 Mbit		1 Gbit	
		8 col 12 row		9 col 12 row		9 col 13 row		10 col 13 row		10 col 14 row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
BA1	BA1	A22	A22	A23	A23	A24	A24	A25	A25	A26	A26
BA0	BA0	A21	A21	A22	A22	A23	A23	A24	A24	A25	A25
MA13	MA13	—	—	—	—	—	—	—	—	—	A24
MA12	MA12	—	—	—	—	—	A22	—	A23	—	A23
MA11	MA11	—	A20	—	A21	—	A21	—	A22	—	A22
MA10	MA10	—	A19	—	A20	—	A20	—	A21	—	A21
MA9	MA9	—	A18	—	A19	—	A19	A10	A20	A10	A20
MA8	MA8	—	A17	A9	A18	A9	A18	A9	A19	A9	A19
MA7	MA7	A8	A16	A8	A17	A8	A17	A8	A18	A8	A18
MA6	MA6	A7	A15	A7	A16	A7	A16	A7	A17	A7	A17
MA5	MA5	A6	A14	A6	A15	A6	A15	A6	A16	A6	A16
MA4	MA4	A5	A13	A5	A14	A5	A14	A5	A15	A5	A15
MA3	MA3	A4	A12	A4	A13	A4	A13	A4	A14	A4	A14
MA2	MA2	A3	A11	A3	A12	A3	A12	A3	A13	A3	A13

Table 24-17. Address Multiplexing by Column/Row Width for 16-bit Devices (continued)

Device Pins	ESDRAMC Pins	16-bit SDR and LPDDR SDRAM Memory Device									
		64 Mbit		128 Mbit		256 Mbit		512 Mbit		1 Gbit	
		8 col 12 row		9 col 12 row		9 col 13 row		10 col 13 row		10 col 14 row	
		Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
MA1	MA1	A2	A10	A2	A11	A2	A11	A2	A12	A2	A12
MA0	MA0	A1	A9	A1	A10	A1	A10	A1	A11	A1	A11

24.4.3.2 Bank Addresses

Bank address connections are summarized in Table 24-18. Bank addressing utilizes the most-significant addresses to specify the active bank, the actual bits being dependent on the density of the memory system. Page size and density for a number of potential configurations are documented in Table 24-18. For undocumented configurations, the Equation 24-1 and Equation 24-2 can be used to calculate page size and density.

$$\text{Page Size (bytes)} = 2^{\#\text{ Column address bits}} \times (\text{Memory width in bits} / 8) \qquad \textit{Eqn. 24-1}$$

$$\text{Density (bytes)} = 2^{(\#\text{Column Address Bits} + \#\text{ Row Address Bits})} \times (\text{Memory width in bits} / 2) \qquad \textit{Eqn. 24-2}$$

Table 24-18. Bank Address Bit Assignment

Density (Bytes)	Page Size (Bytes)	BA1	BA0
8 Mbytes	X	A22	A21
16 Mbytes		A23	A22
32 Mbytes		A24	A23
64 Mbytes		A25	A24
128 Mbytes		A26	A25
256 Mbytes		A27	A26

24.4.4 Multiplexed Address Bus During Precharge or Load Mode Registers Modes

During precharge or load mode registers modes (SMODE = 0b001 or 0b010) there is no address shifting, so that CPU address A0 is mapped to MA0 for all memory widths. For example, to drive the MA10 bit (PRECHARGE ALL command) the CPU A10 bit is required to be set. The same logic is valid for the load mode register command, as shown in the initialization routine example in Section 24.5.4.1, “SDRAM Initialization.”

Byte accesses are required during precharge and load mode register modes, since the address can be nonaligned depending on the load mode register data.

24.4.5 Refresh

The ESDRAMC hardware satisfies all SDRAM refresh requirements based on initial configuration by the user software. 0,1,2,4,8 or 16 refresh cycles are scheduled at 31.25 µs (nominal 32 kHz clock) intervals, providing 0,2048,4096,8192,16384 or 32768 refresh cycles every 64 ms. The refresh rate is programmed through the REFR field in the ESDCTL0 and ESDCTL1 registers. Each array can have a different rate, allowing a mix of SDRAM/LPDDR devices, or different SDRAM densities. Refresh is disabled by hardware reset.

A refresh request is made pending at each rising edge on the 32 kHz clock. In response to this request, the hardware gains control of the SDRAM as soon as any in-process bus cycle completes. Once it has gained control of the memory, commands are issued to precharge all banks. Following a row precharge delay (t_{RP}), an AUTO REFRESH command is issued. At t_{RC} intervals, additional auto-refresh cycles are issued until the specified number of cycles have been run. Figure 24-31 illustrates two refresh sequences. Burst transfers in progress when the refresh request arrives are allowed to complete prior to the refresh operation. SDRAM bus accesses queued after the refresh request are held off until the refresh completes.

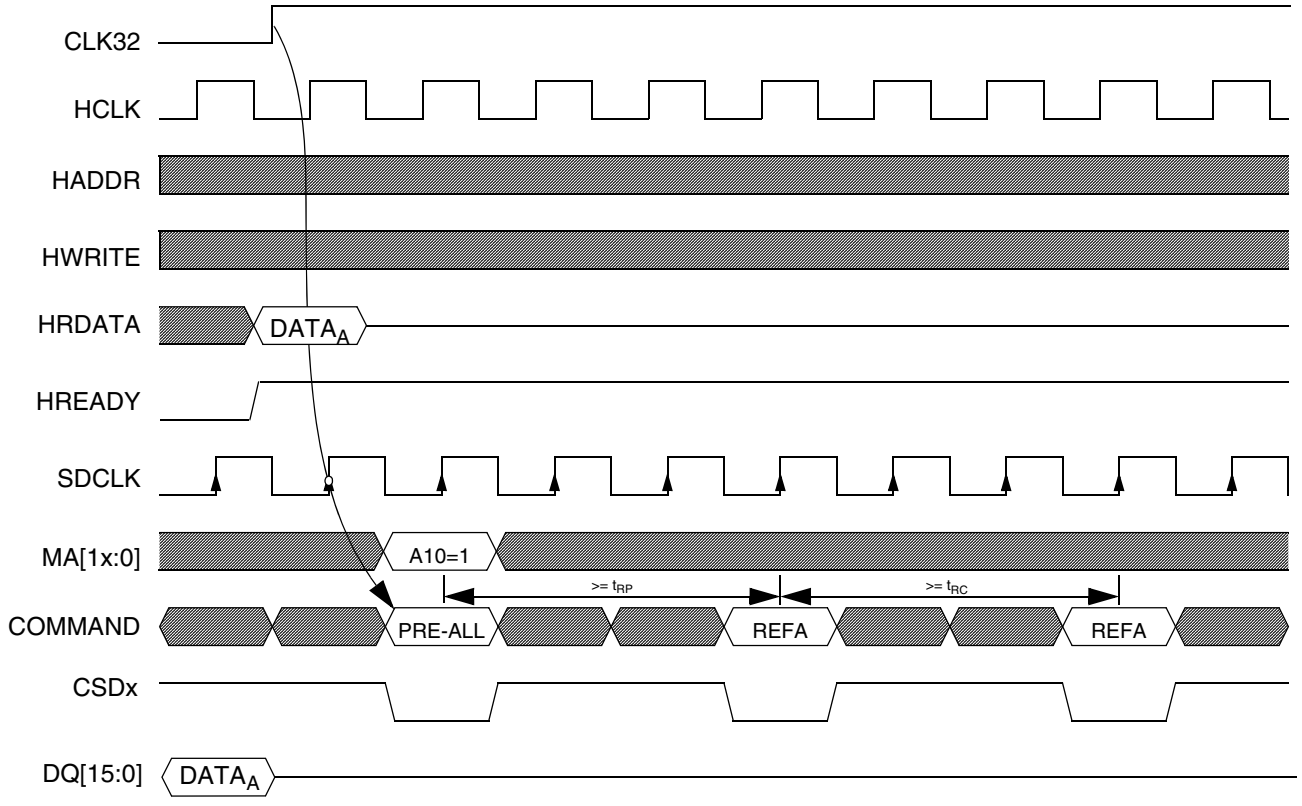


Figure 24-31. Hardware Refresh Timing Diagram

In Figure 24-32, an access is queued just as the refresh begins. This cycle is delayed until the precharge and single refresh (REFR=01) cycles are run. Bus cycles targeted to other memory or peripheral devices are allowed to progress normally while the refresh is in progress. None of the pins shared between the SDRAM and other devices are required for the refresh operation.

NOTE

Because refresh commands (requires all banks to be in idle state, achieved by precharge-all) are issued automatically by the enhanced SDRAM controller at each 32 KHz clock, address bit A10 cannot be shared with other peripherals' address bus in the system.

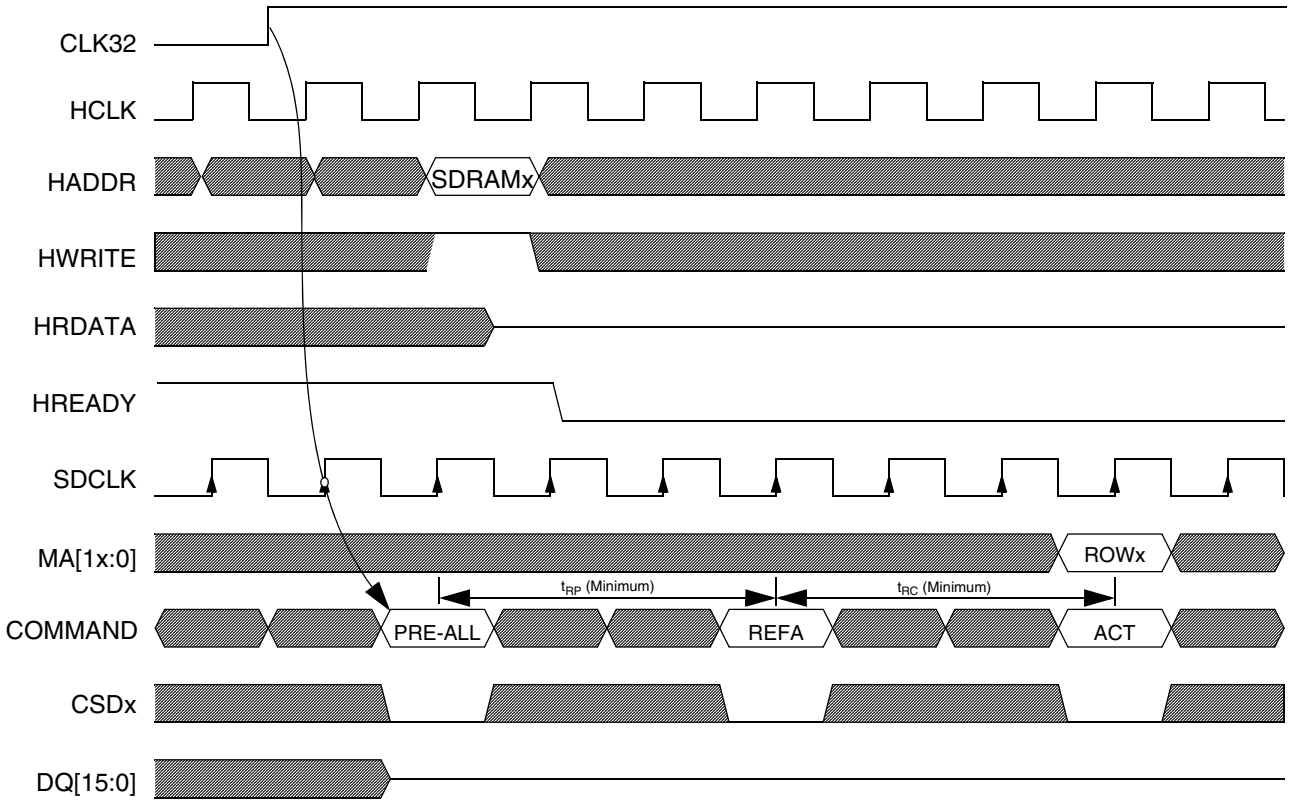


Figure 24-32. Hardware Refresh with Pending Bus Cycle Timing Diagram

24.4.6 Low Power Operating Modes

The following section describes the low power operating modes of the ESDRAMC. Table 24-19 summarizes the low power modes supported by the ESDRAMC for SDRAM and LPDDR devices.

Table 24-19. ESDRAMC Low Power Operating Modes

System Operating Mode	Memory Device Low Power Operating Mode	Wake-Up Penalty (SDRAM Device)	Wake-Up Penalty (LPDDR Device)
Run	Power-down mode	1 clock cycle	tXP
Run	Precharge bank(s)	1 clock cycle	1 clock cycle
Run	Manual self-refresh mode	2 refresh periods	tXS + 2 refresh periods
Stop	Self-refresh mode	2 refresh periods	tXS + 2 refresh periods

24.4.6.1 Self-Refresh Mode for SDRAM/LPDDR Devices

The controller puts external memory in self-refresh mode in response to a low power mode (p_lpmd) request from the clock controller module (CCM). When done, the controller sends a low power acknowledge (lpack) signal to the CCM. If the controller is busy when the low power mode (p_lpmd) request is received, it first finishes the received accesses and then puts the memory in self-refresh mode.

Figure 24-33 and Figure 24-34 show timing for entering and exiting self-refresh mode, respectively.

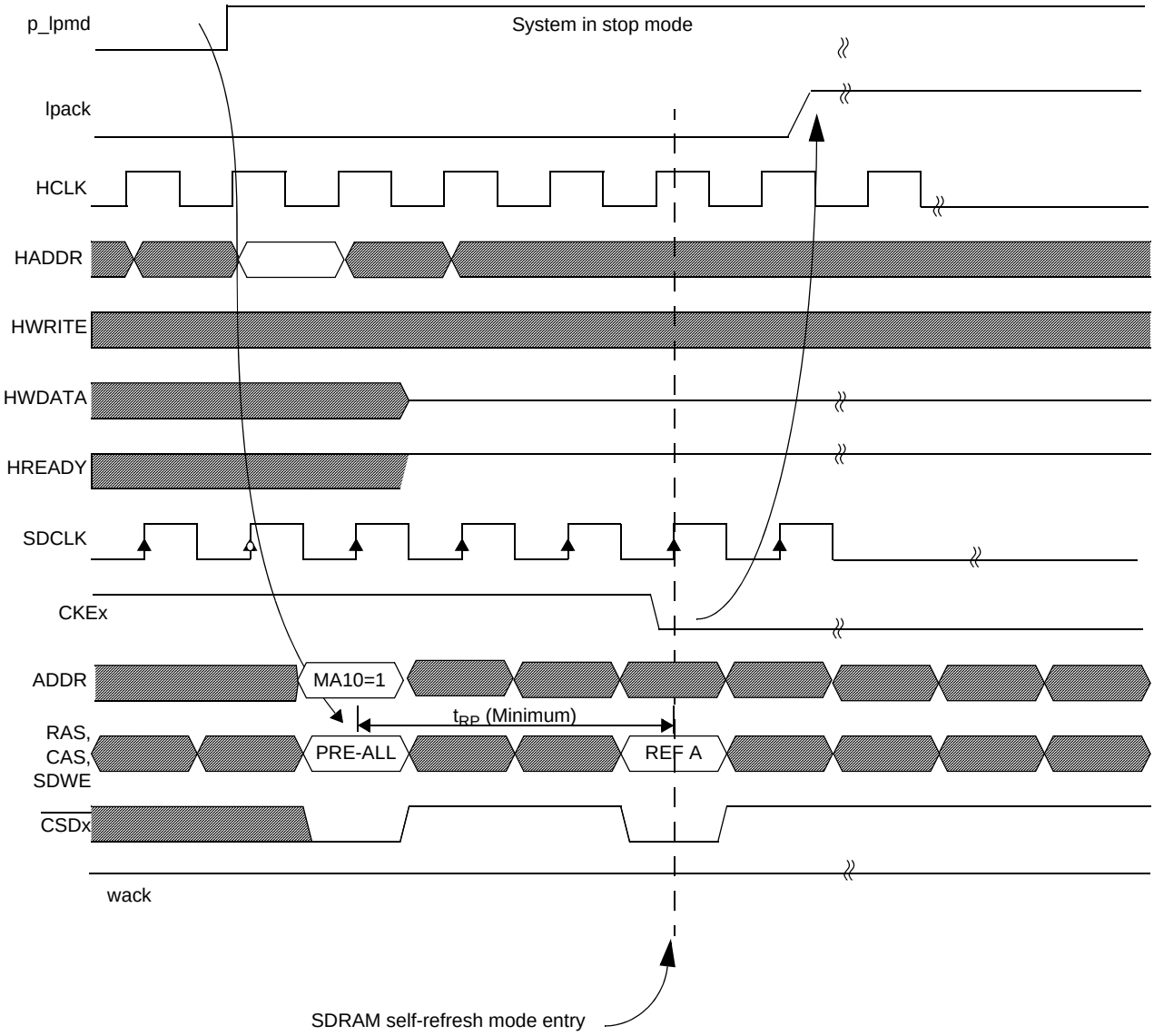


Figure 24-33. SDRAM/LPDDR Enter Self-Refresh Mode During System STOP Mode

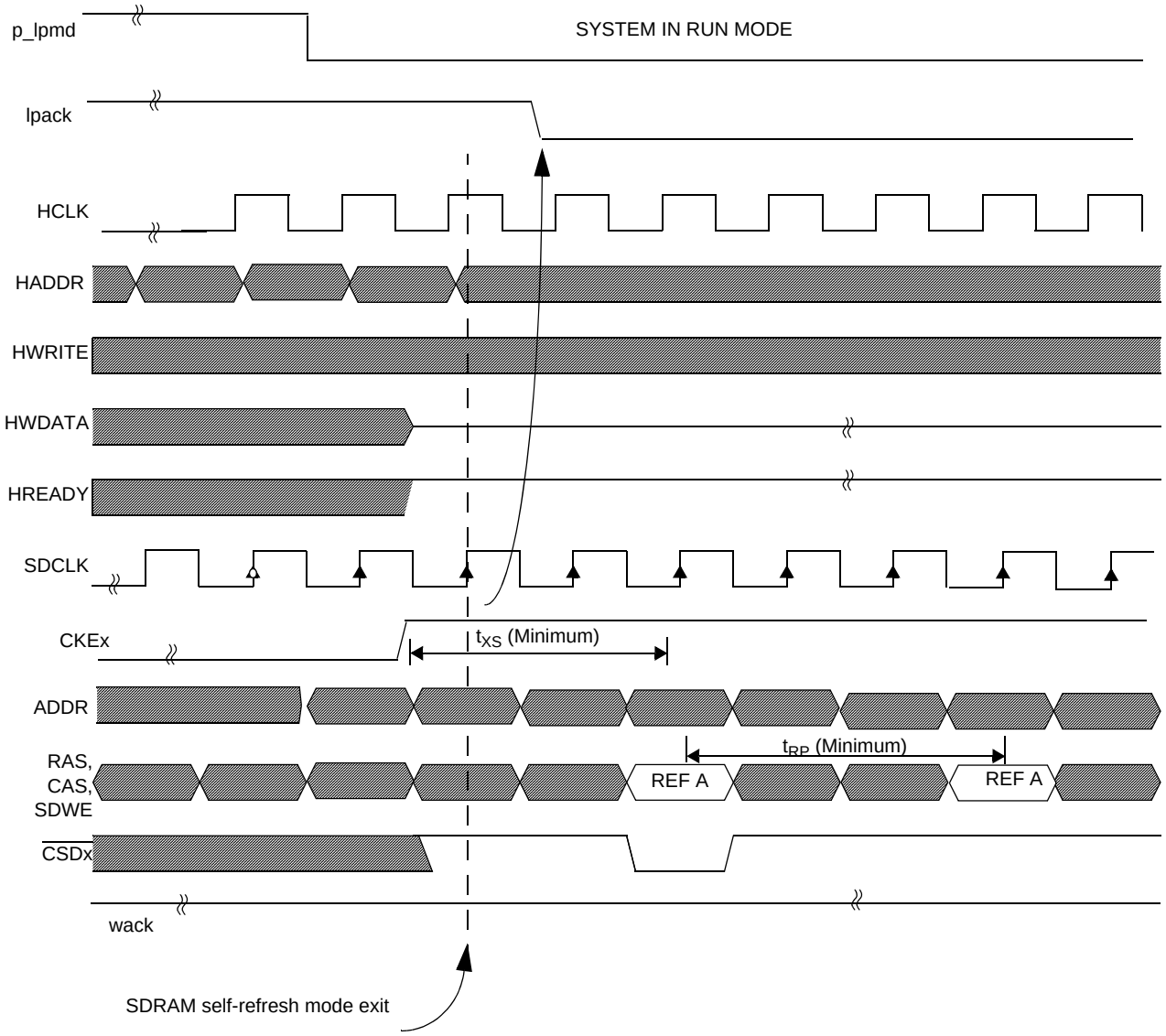


Figure 24-34. SDRAM/LPDDR Exit Self-Refresh Mode during System STOP Mode

24.4.6.2 Manual Self-Refresh Mode for SDRAM/LPDDR Devices

This operating mode allows the software to control a self-refresh mode entry of the external SDRAM/LPDDR device. When this mode is selected (SMODE=100 in the respective CSD control register) and refresh is enabled the enhanced SDRAM controller will complete any active access and a self-refresh command to the external device is issued. No access is allowed to the respective CSD during manual self-refresh mode. If refresh has not been enabled, the enhanced SDRAM controller places the memory in a low power consumption mode known as power-down.

NOTE

- A manual precharge-all should be initiated by the user before a manual self-refresh.

- SDCLK stops only if both chip selects are in manual self-refresh mode, so that one chip select can be used while the other is in manual self-refresh (in case that both chip selects are in use).

Manual self-refresh mode is exited by changing SMODE bits in the respective chip select control register to select a different operating mode. When a different mode is selected, the controller takes the SDRAM device out of self-refresh mode and begins issuing auto-refresh cycles (if refresh has been enabled).

Figure 24-35 and Figure 24-36 show timing for entry and exit from manual self-refresh mode, respectively.

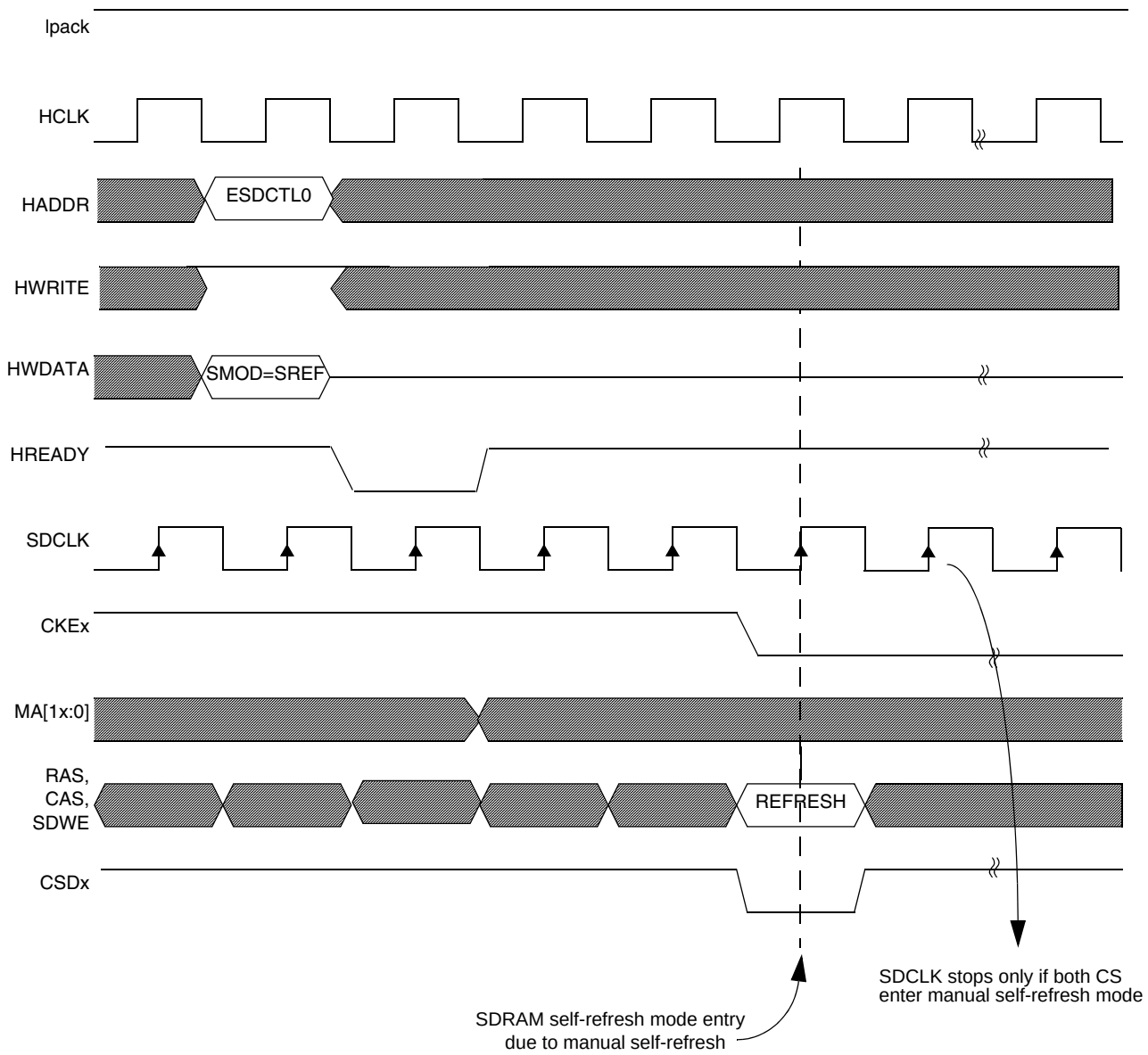


Figure 24-35. Manual Self-Refresh Entry Timing Diagram

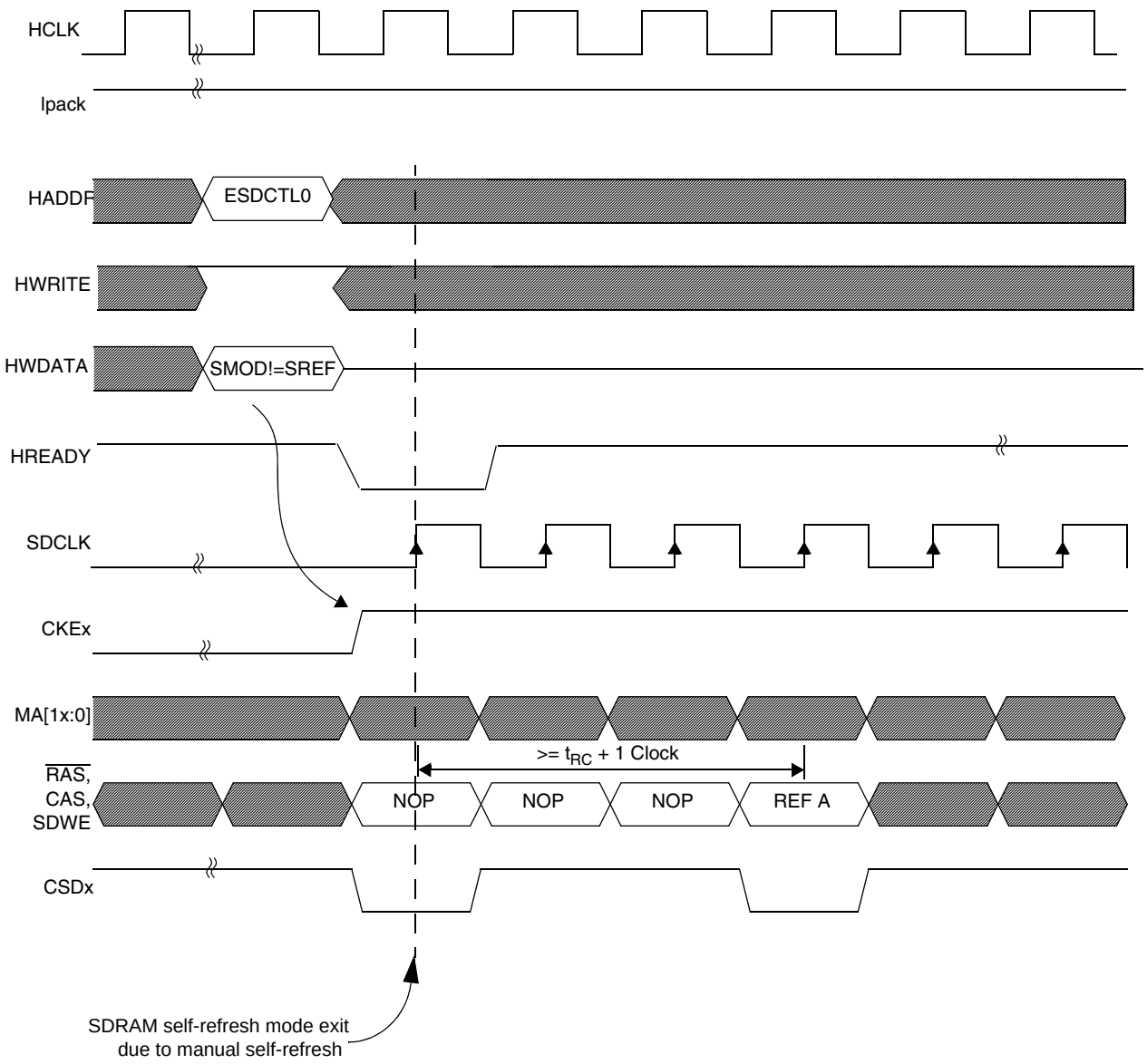


Figure 24-36. Manual Self-Refresh Exit Timing Diagram

24.4.6.3 Power-Down Modes

If the SDRAM/LPDDR memory utilization is low, the ESDRAMC can reduce power consumption by putting the SDRAM/LPDDR into power-down mode. Power-down mode for CSD*n* is activated by programming the PWDT bits in the ESDCTL*n* register (*n* = 0,1). In power-down mode the ESDRAMC automatically issues the refresh commands to the SDRAM/LPDDR memories at the rate defined by the SREFR bits in the corresponding ESDCTL*n* register.

The two power-down modes are discussed in Section 24.4.6.3.1, “Precharge Power-Down Mode,” and Section 24.4.6.3.2, “Active Power-Down Mode,” respectively. The distinguishing factor between the two

is whether banks remain active while the clock is stopped. Active power-down allows banks to remain activated, while precharge power-down does not.

24.4.6.3.1 Precharge Power-Down Mode

Programming PWDT[1:0] = 01 in the ESDCTL*n* register causes the ESDRAMC to place the memories in power-down mode any time the controller detects that no banks are active. This mode is useful in applications where a memory array is accessed infrequently, and the chances of more than one access to the same page are minimal.

Reading or writing to memory activates a page within the addressed bank. Reset, software generated precharge, and hardware initiated refresh are three ways to close an active bank. The periodically occurring refresh is the normal means that invokes the power-down mode. At each refresh interval, all banks are closed by a precharge all command, followed by the refresh operation. The controller then issues the power-down command to the memories. A few cycles of delay are incurred with the first read or write cycle in order to restart the clocks, but only on the first cycle. After that, the clocks continue to run until the next refresh operation or until any active banks are manually precharged.

Page misses on read and write cycles cause the addressed bank to be closed (precharged) and a new page opened within the bank. This operation does not cause the clocks to stop, nor does manually precharging only a single bank within the memory. All banks within the memory must be inactive before the power-down mode is invoked.

Power-down mode occurs if the CKE is registered low coincident with a NOP or COMMAND INHIBIT, when no accesses are in progress. Entering power-down will deactivate the input and output buffers (excluding CKE) of the device. The power-down mode state is exited by registering a NOP or COMMAND INHIBIT and CKE is high at the desired clock edge.

NOTE

Because the ESDRAMC does not issue auto precharge commands to the SDRAM, the software is responsible to issue a precharge all command in order to enter precharge power-down mode, to wait for precharge timer (PRCT) to close/precharge all active banks, or to wait for the next refresh cycle in order to enter this low power mode. (During the refresh cycle, the ESDRAMC automatically issues the precharge all command).

Figure 24-37 and Figure 24-38 illustrate timing for SDR SDRAM power-down mode entry and exit respectively. Figure 24-39 and Figure 24-40 illustrate timing for LPDDR SDRAM power-down mode entry and exit respectively.

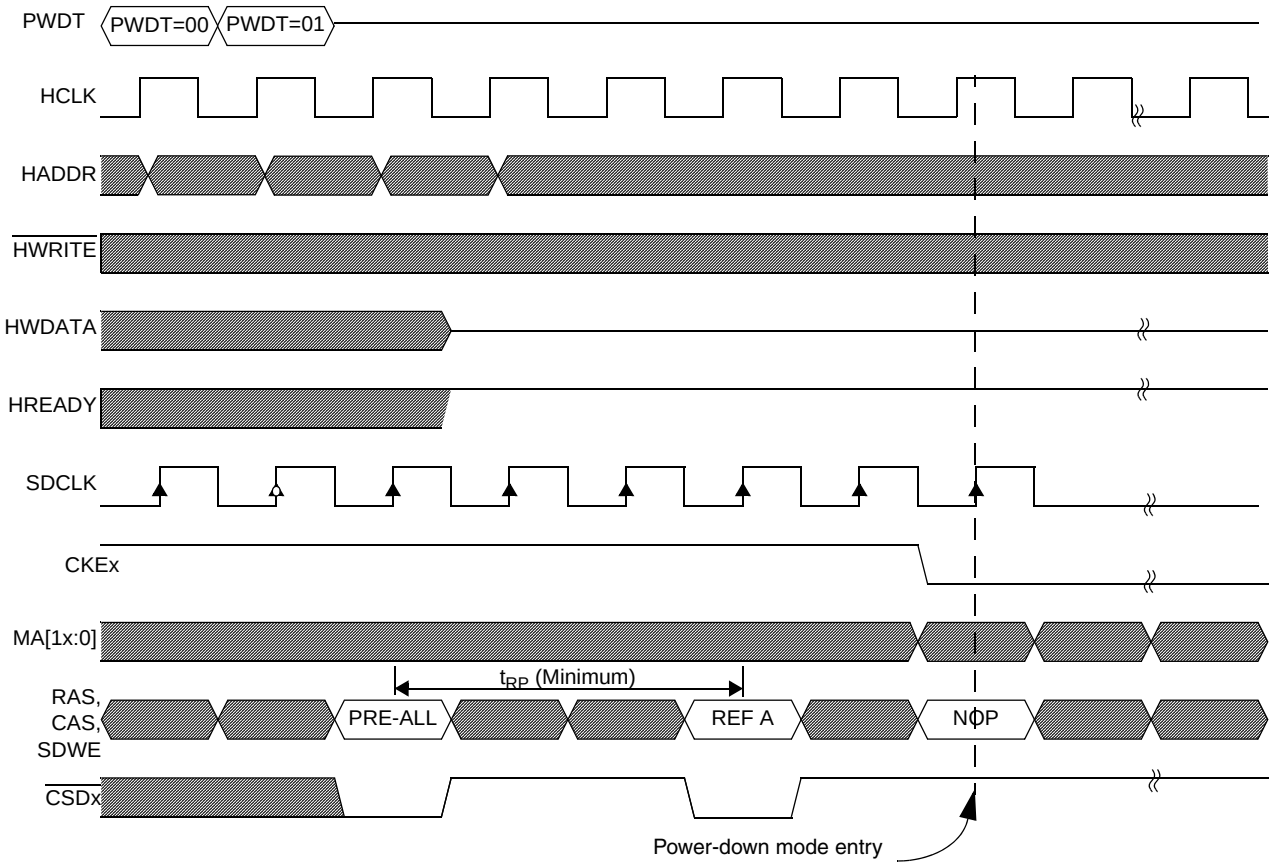


Figure 24-37. SDR SDRAM Precharge Power-Down Mode Entry Timing Diagram

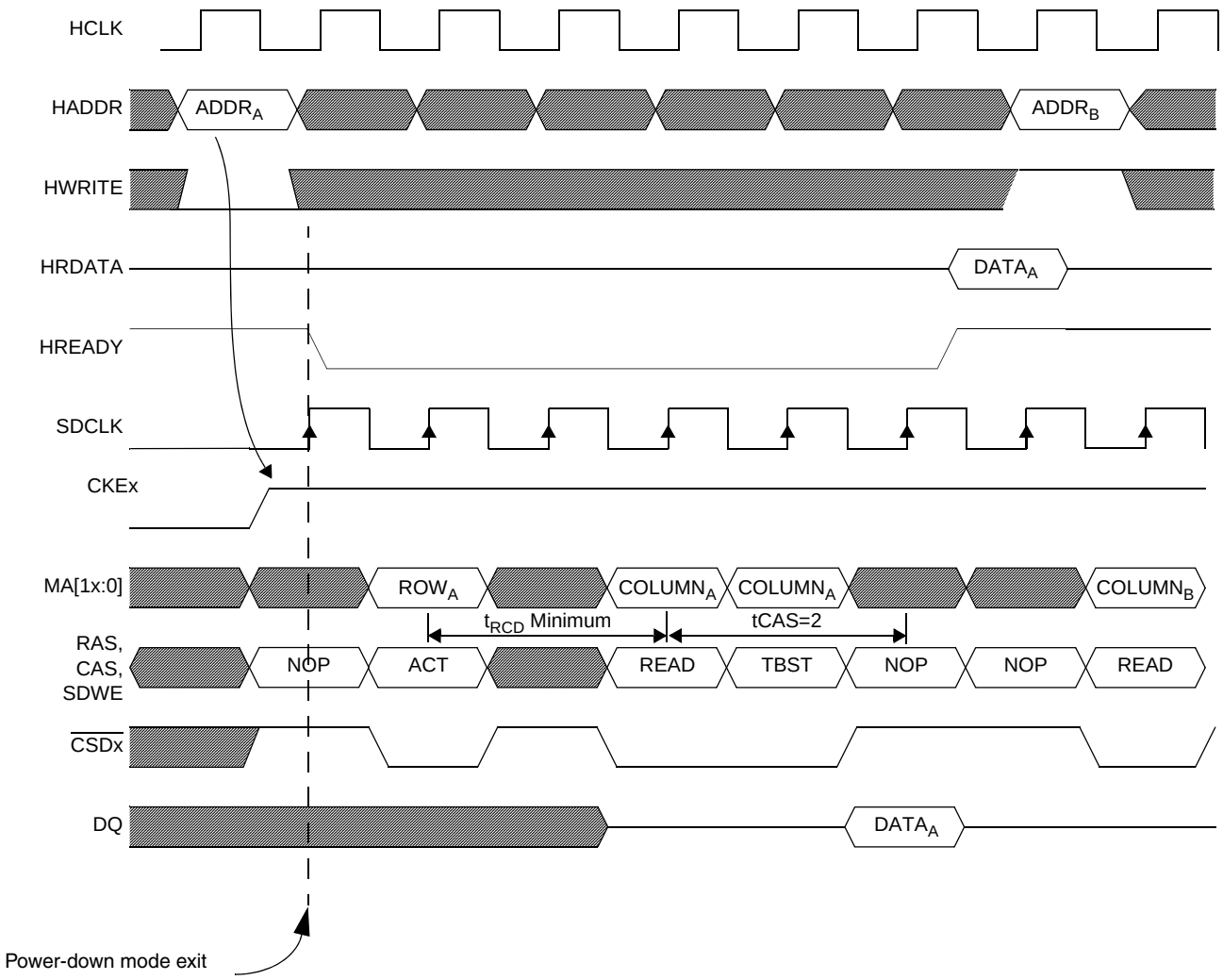


Figure 24-38. SDR SDRAM Precharge Power-Down Mode Exit Timing Diagram

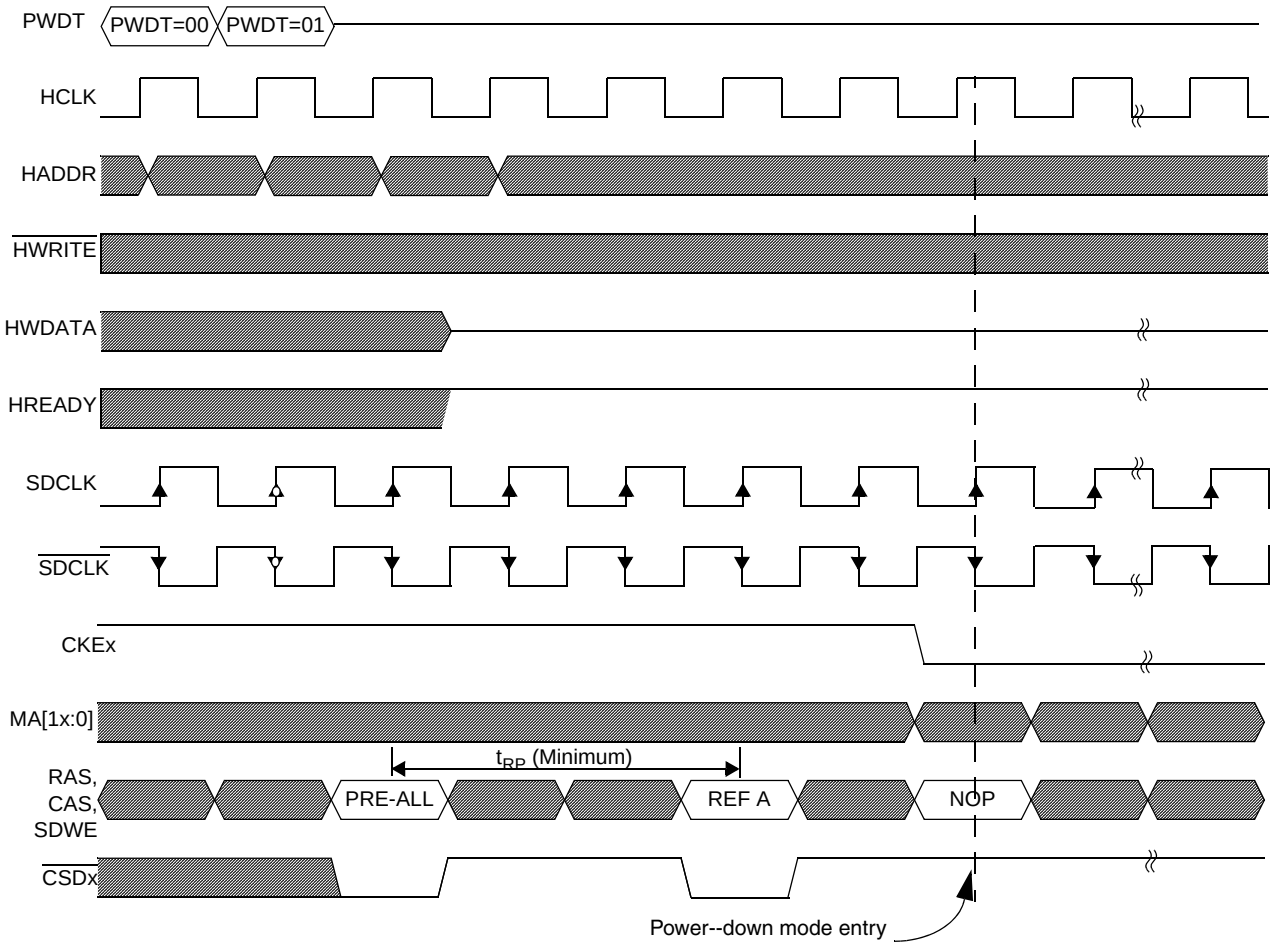


Figure 24-39. Mobile DDR SDRAM Precharge Power-Down Mode Entry Timing Diagram

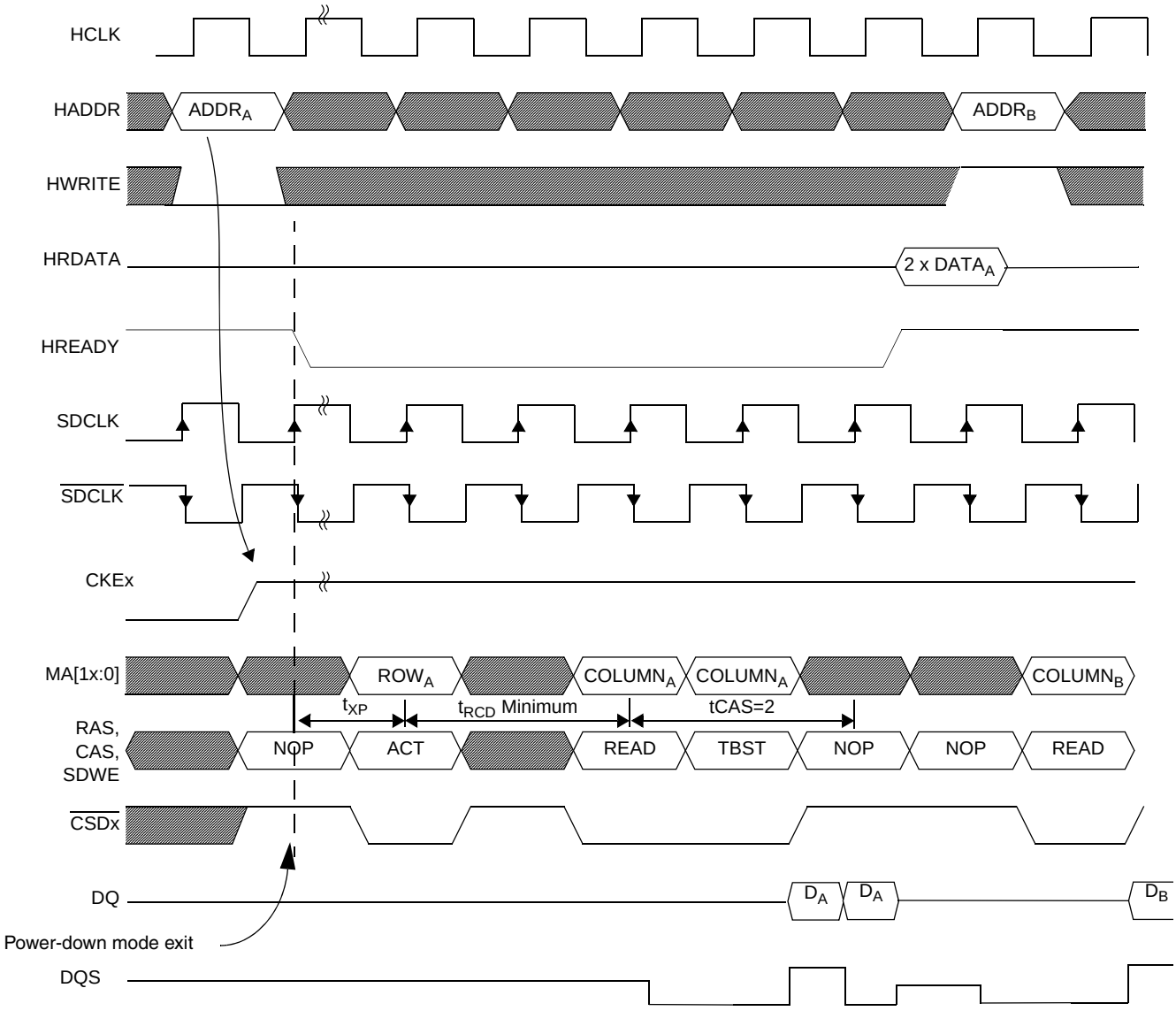


Figure 24-40. Mobile DDR SDRAM Precharge Power-Down Mode Exit Timing Diagram

24.4.6.3.2 Active Power-Down Mode

Active power-down mode is selected whenever PWDT[1:0] = 1*n*. In this mode the SDCLK is stopped after a selectable delay from the last access to the array. Active banks are not closed prior to disabling the SDRAM/LPDDR clock. Either 64 (PWDT[1:0] = 10) or 128 (PWDT[1:0] = 11) cycle delays are possible. SDRAM/LPDDR clocks are counted from the end of the last read or write access. Subsequent read accesses, write accesses, and self-refresh modes reset the counter. Auto-refresh cycles do not affect the counter: however, if the counter expires during a refresh operation the clock is disabled immediately following the refresh.

Figure 24-41 and Figure 24-42 are timing diagrams for entry and exit of SDR and LPDDR SDRAM active power-down modes.

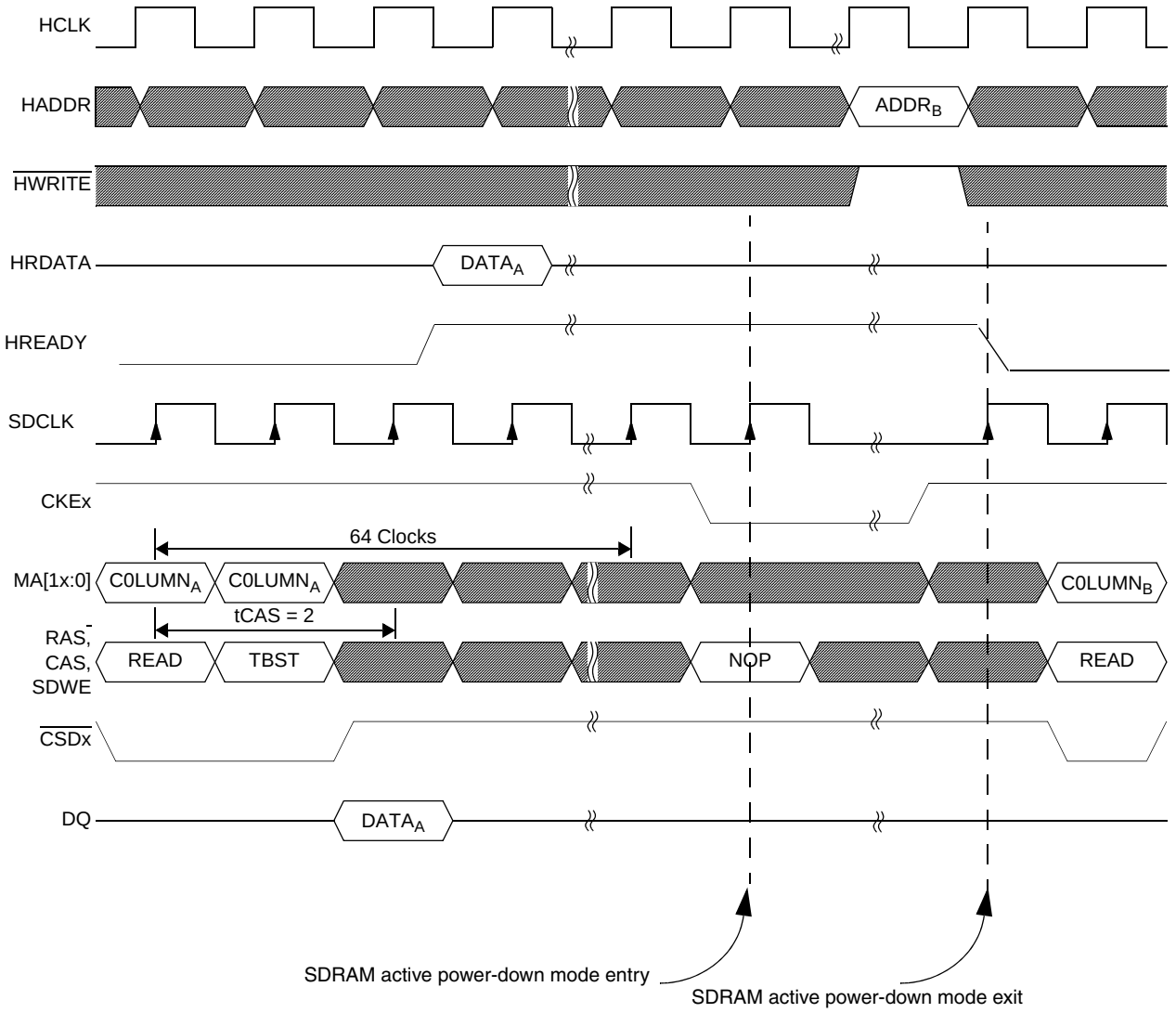


Figure 24-41. SDR SDRAM Active Power-Down Mode Timing Diagram

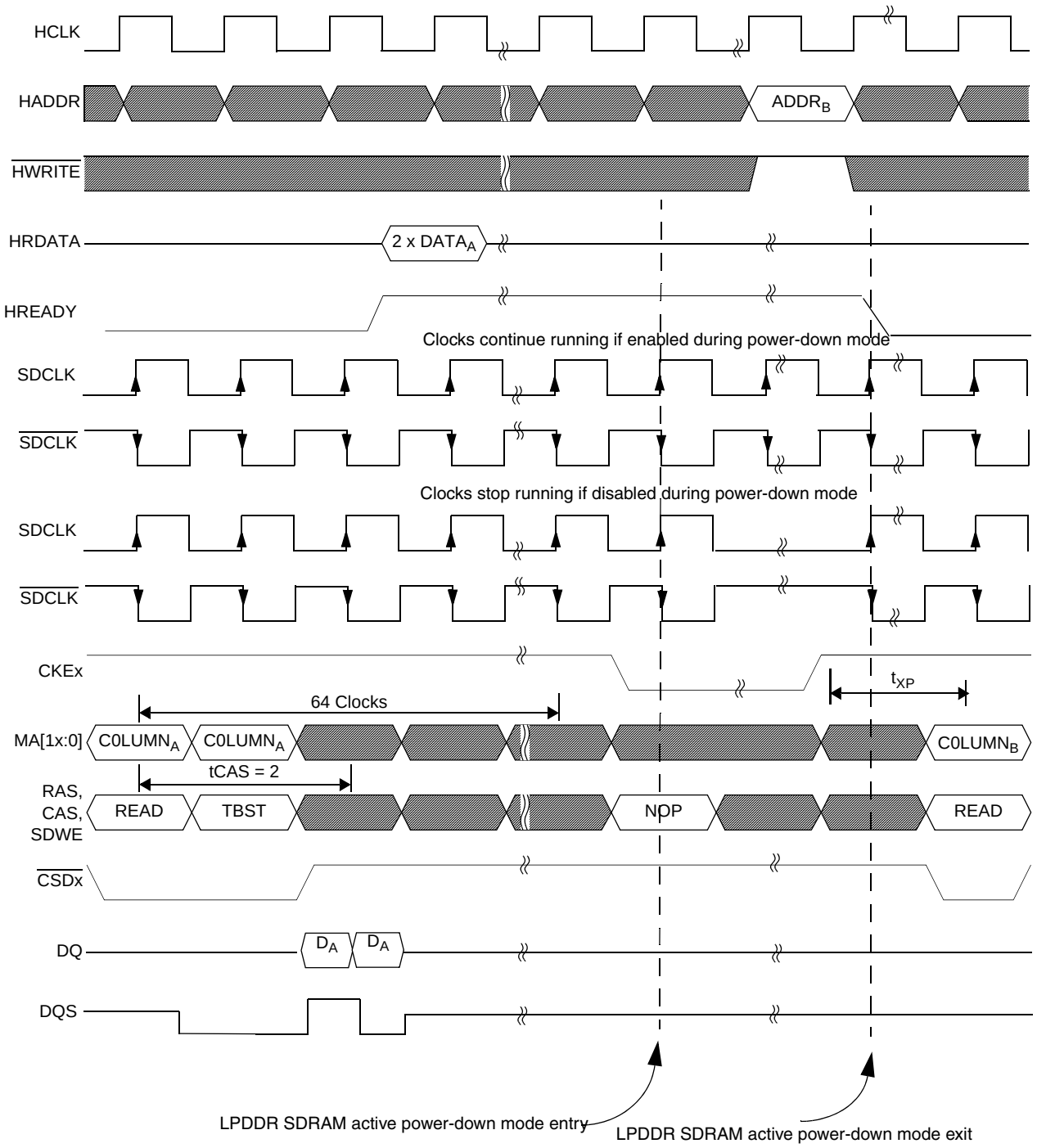


Figure 24-42. Mobile DDR SDRAM Active Power-Down Mode Timing Diagram

24.4.6.4 Precharge Bank(s)—Low Power Mode

"Closing" (due to PRECHARGE command) the last used/open row in any non active bank within a chip select reduces the power consumption of the external memory device. The power saving is device

dependent, and one should consult/examine the external memory device specification for more details on power consumption reduction. The precharge bank is activated if PRCT is enabled. A PRECHARGE command is issued after 2xPRCT HCLK cycles of no activity to one of the SDRAM/LPDDR banks. The number of cycles before the PRECHARGE command is issued depends on the command bus ($\overline{WE}$, $\overline{RAS}$, $\overline{CAS}$ and CSD) availability (meaning there is no active access to other bank) and the memory timing parameters.

24.4.6.5 LPDDR Frequency Change

To change the frequency in a LPDDR-based system the following can be used to ensure the DDRC delay line locks to the new frequency:

1. Issue PRECHARGE ALL command.
2. Put the external memories into self-refresh operating mode.
3. Change the system frequency.
4. Reset the delay line, by setting the MDDR_DL_RST bit in the enhanced SDRAMC miscellaneous register.
5. Wait about 4500 HCLK cycles for the delay line to lock on the new frequency.
6. Exit self-refresh mode.

After the above steps are performed the LPDDR is ready for normal operation at the new frequency.

24.4.7 SDRAM (SDR and LPDDR) Command Encoding

Table 24-20 summarizes the command encoding utilized by this controller. These commands represent a subset of the commands defined by the JEDEC standard JESD79-4A.

Table 24-20. SDRAM (SDR and LPDDR) Command Encoding

Function	Symbol	CKE *n*-1	CKE *n*	CS	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	A11	A10	BA[1:0]	A[13:0]
Deselect	DSEL	H	X	H	X	X	X	X	X	X	X
No Operation	NOP	H	X	L	H	H	H	X	X	X	X
Read	READ	H	X	L	H	L	H	V	L	V	V
Write	WRIT	H	X	L	H	L	L	V	L	V	V
Bank activate	ACT	H	X	L	L	H	H	V	V	V	V
Burst terminate[1]	TBST	H	X	L	H	H	L	X	X	V	X
Precharge select bank	PRE	H	X	L	L	H	L	V	L	V	X
Precharge all banks	PALL	H	X	L	L	H	L	X	H	X	X
Auto-refresh	CBR	H	X	L	L	L	H	X	X	X	X
Self-refresh entry	SLFRSH	H	L	L	L	L	H	X	X	X	X
Self-refresh exit	SLFRSHX	L	H	H	X	X	X	X	X	X	X
Power-down entry	PWRDN	H	L	X	X	X	X	X	X	X	X

Table 24-20. SDRAM (SDR and LPDDR) Command Encoding (continued)

Function	Symbol	CKE *n*-1	CKE *n*	CS	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	A11	A10	BA[1:0]	A[13:0]
Power-down exit	PWRDNX	L	H	H	X	X	X	X	X	X	X
Load mode register[2]	MRS	H	X	L	L	L	L	L	L	V	V

1 For mobile DDR, applies only to read bursts (with auto-precharge disabled).

2 BA0-BA1 select either the mode register, the extended mode register or the low power extended mode register.

24.4.7.1 Reset

Assertion of the $\overline{RST}$ signal initializes the controller into the idle state, and disables the module. While disabled, the controller remains in the idle state with the internal clocks stopped. The reset state of the control register allows for basic read/write operations sufficient to fetch the reset vector and execute the initialization code. A complete initialization of the controller should be performed as part of the start-up code sequence.

Read/write cycles, refresh and low-power mode requests, and power-down timeouts will all trigger transitions out of the idle state. As shown in the simplified enhanced SDRAM controller state diagram in Figure 24-43, state transitions due to a read or write request depend on the operating mode. Other transitions require the corresponding function to be enabled in the ESDCTL*n* registers. Some state transitions have been removed from the figure to minimize complexity and allow an easier understanding of the basic controller operation.

24.4.7.2 Warm Reset

Warm reset enables the user to reset the ESDRAMC without affecting the data in external SDR/DDR memories. The warm reset is fired by using the warm reset signal (that would be kept high during warm reset inside the ESDRAMC). This signal is latched while exiting reset, which causes ESDRAMC internal logic to keep CKE signals at low level, thus ensuring the self-refresh configuration of the outside device preserved until CS_EN is configured in the ESDCTL*n* registers.

The warm_reset signal to EMI should be driven high before reset is fired. Figure 24-43 shows the signal definitions at the EMI boundary level.

NOTE

The SDRAM_RDY status bit is cleared after warm reset, and remains cleared for 7 CKIL cycles. Access to the DDR external device is blocked during this time interval. This delay of about 220 µsec can be avoided by using regular reset along with warm reset, as shown in Figure 24-43.

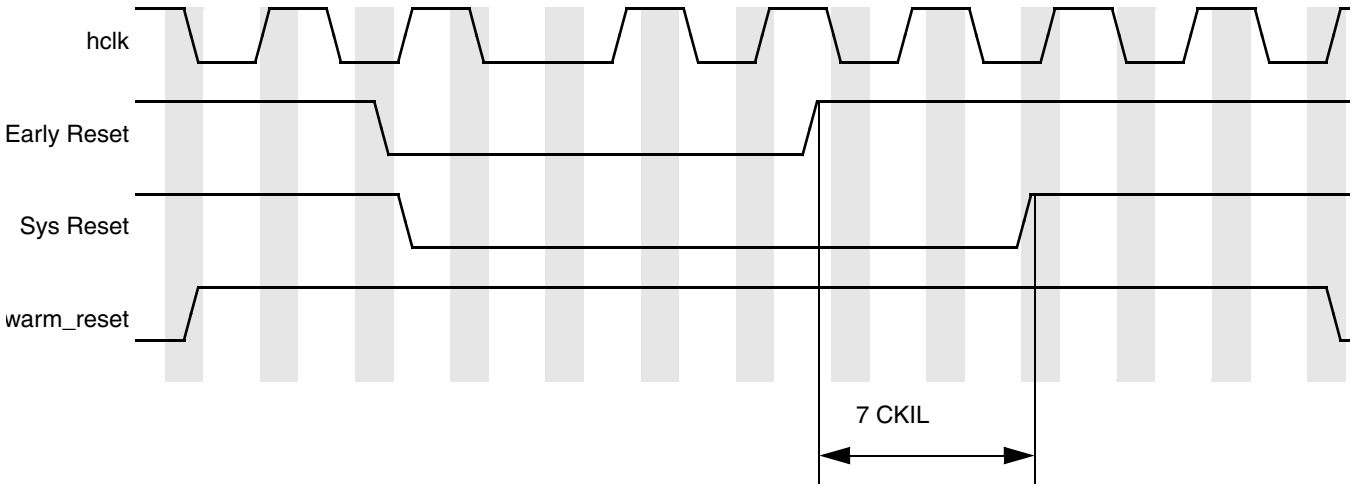


Figure 24-43. Warm Reset Timing Diagram

Figure 24-44 and Figure 24-45 describe the flow to enter self-refresh mode before and during warm reset, as well as ESDRAMC configuration when exiting warm reset.

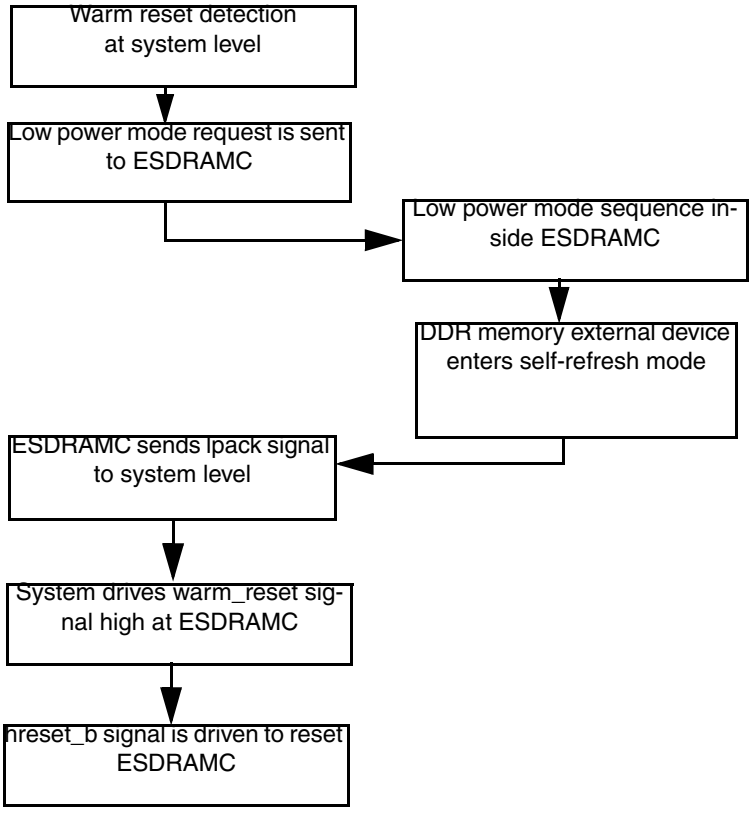


Figure 24-44. Self-Refresh Entry Before warm_reset Assertion

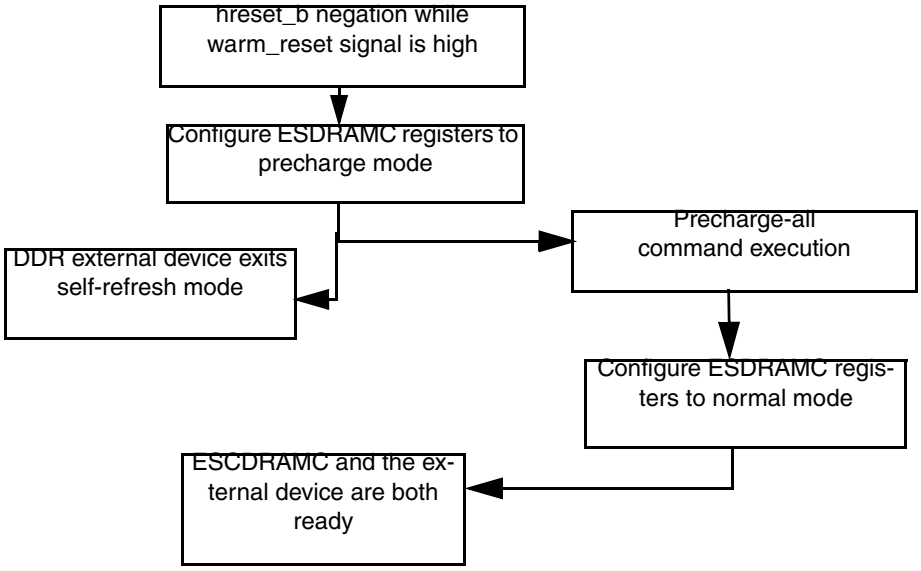


Figure 24-45. Self-Refresh Exit After Warm Reset Assertion

24.4.8 Normal Read/Write Mode

The normal read/write mode (SMODE[2:0] = 000) is used for general read and write accesses (AHB-lite-compatible) to the SDRAM/LPDDR. Single and read burst accesses are supported for both SDRAM/LPDDR memories (although bursts requests are limited as shown in Table 24-21). For SDRAM/LPDDR memories single and burst write accesses are supported as well.

Table 24-21. SDRAM/LPDDR Burst Access Support

Internal AHB Word Access (32 bit)		External Memory Device[3]			Description
		16-bit			
HBURST	TYPE	BL=4	BL=8	BL=FP	
000	SINGLE[1,2]	No	Yes	Yes	Single transfer
001	INCR	No	Yes	Yes	Incrementing burst
010	WRAP4	No	Yes	Yes	4-beat wrapping burst
011	INCR4	No	Yes	Yes	4-beat incrementing burst
100	WRAP8	No	Yes	Yes	8-beat wrapping burst
101	INCR8	No	Yes	Yes	8-beat incrementing burst
110	WRAP16	No	No	No	16-beat wrapping burst
111	INCR16	No	No	No	16-beat incrementing burst

[1] ESDRAMC only supports bursts of 32-bit (word size). Byte or half-word accesses are only supported for single transfers (HBURST type SINGLE). The ESDRAMC automatically splits the AHB bursts as a function of the external memory device according to the ESDCTL configuration register settings (size and burst length), in such a way that a continuous flow of data is obtained for both read or write bursts (see example in Table 24-22).

[2] BL = Burst Length field in device mode register; FP = Full Page (wrap at external memory device ROW boundary). For LPDDR devices only BL=8 is supported.

Read or write requests to the enhanced SDRAM controller initiate a check to see whether the page is already open. This check consists of comparing the requested address against the last row accessed within the corresponding bank. If the rows are different, a precharge has occurred since the last access, or there has never been an access to the bank, then the access must follow the “off-page” sequence. If the requested and last row match, the shorter “on-page” access is used. An off-page sequence must first activate the requested row, an operation which is analogous to a conventional DRAM $\overline{\text{RAS}}$ cycle. An activate cycle is the first operation depicted in Figure 24-46. During the activate cycle, the appropriate chip select is driven low, the row addresses are placed on the multiplexed address pins, the non-multiplexed addresses are driven to their respective values, the write enable signal is driven high, $\overline{\text{CAS}}$ is driven high, and $\overline{\text{RAS}}$ is driven low. These latter three pins form the SDRAM command word. The data bus is unused during the activate command.

Once the selected row has been activated, the read operation begins after the row to column delay (tRCD) has been met. This delay is either 2 or 3 clocks, as determined by the tRCD control field. During the read cycle, the chip select is once again asserted, the column addresses are driven onto the multiplexed address bus, the non-multiplexed addresses remain driven to the value presented during the activate cycle, the write enable signal is driven high (read), $\overline{\text{RAS}}$ is driven high, and $\overline{\text{CAS}}$ is driven low. After the CAS latency has expired, data is transferred across the data bus. CAS latency is programmable using the tCAS control field. As data is being returned across the AHB, transfer acknowledge is asserted back to the CPU indicating that the CPU should latch data. While data is still on the bus, the enhanced SDRAM controller must begin monitoring transfer request since the CPU is free to issue the next bus request on the same edge that data is being latched.

Data transfers can be either single operand or a burst (WRAP or INCR) of up to a full page. Burst requests are designated as such by the HBURST bus indicating the length of the access, When HBURST equal to 0 a single access is required, otherwise the access is a burst of HBURST words.

SDRAM memories assume that all transfers are burst transfers unless terminated early. Burst transfers can be terminated by a variety of mechanisms: another read or write cycle, a precharge operation, or through a burst terminate command. Burst terminate commands are the general mechanism used by the ESDCTL for early burst termination, The burst terminate command is subject to the CAS latency and must be pipelined similar to the read command, as shown in Figure 24-46 through Figure 24-66.

NOTE

- ESDRAMC handles the HUNALIGN accesses in the following way. The M3IF module converts the original access address to a word align address to the ESDRAMC. The HBSTRB are used by the ESDRAMC to drive the correct value on the DQM signals to the external memory.
- 32-bit examples are not relevant for i.MX25 as only the 16-bit mode is supported.

SDRAM write cycles are different than read cycles in one important aspect. Whereas read data was delayed by the CAS latency, write data has no delay and is supplied at the same time as the write command. Figure 24-54 illustrates an off-page write cycle followed by one that hits on-page. Note that the write data is driven during the same clock that the write command is issued. A burst terminate command cancels the burst operation, but again without the CAS latency.

NOTE

ESDRAMC handles the HUNALIGN accesses in the following way. The M3IF module converts the original access address to a word align address to the ESDRAMC. The HBSTRB are used by the ESDRAMC to drive the correct value on the DQM signals to the external memory.

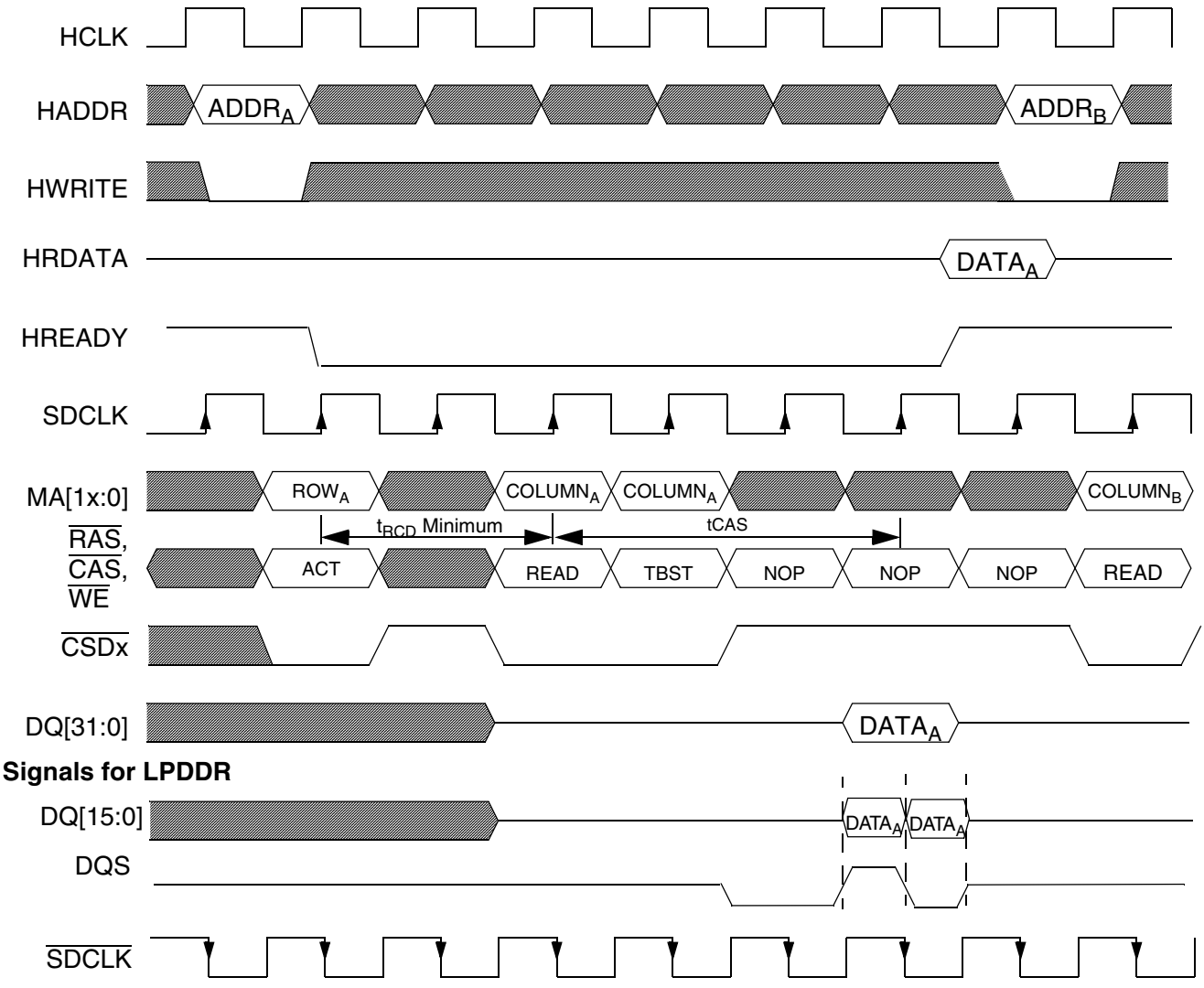


Figure 24-46. SDR and LPDDR Off-Page Single Read Timing Diagram (32-Bit SDR and 16-Bit LPDDR)

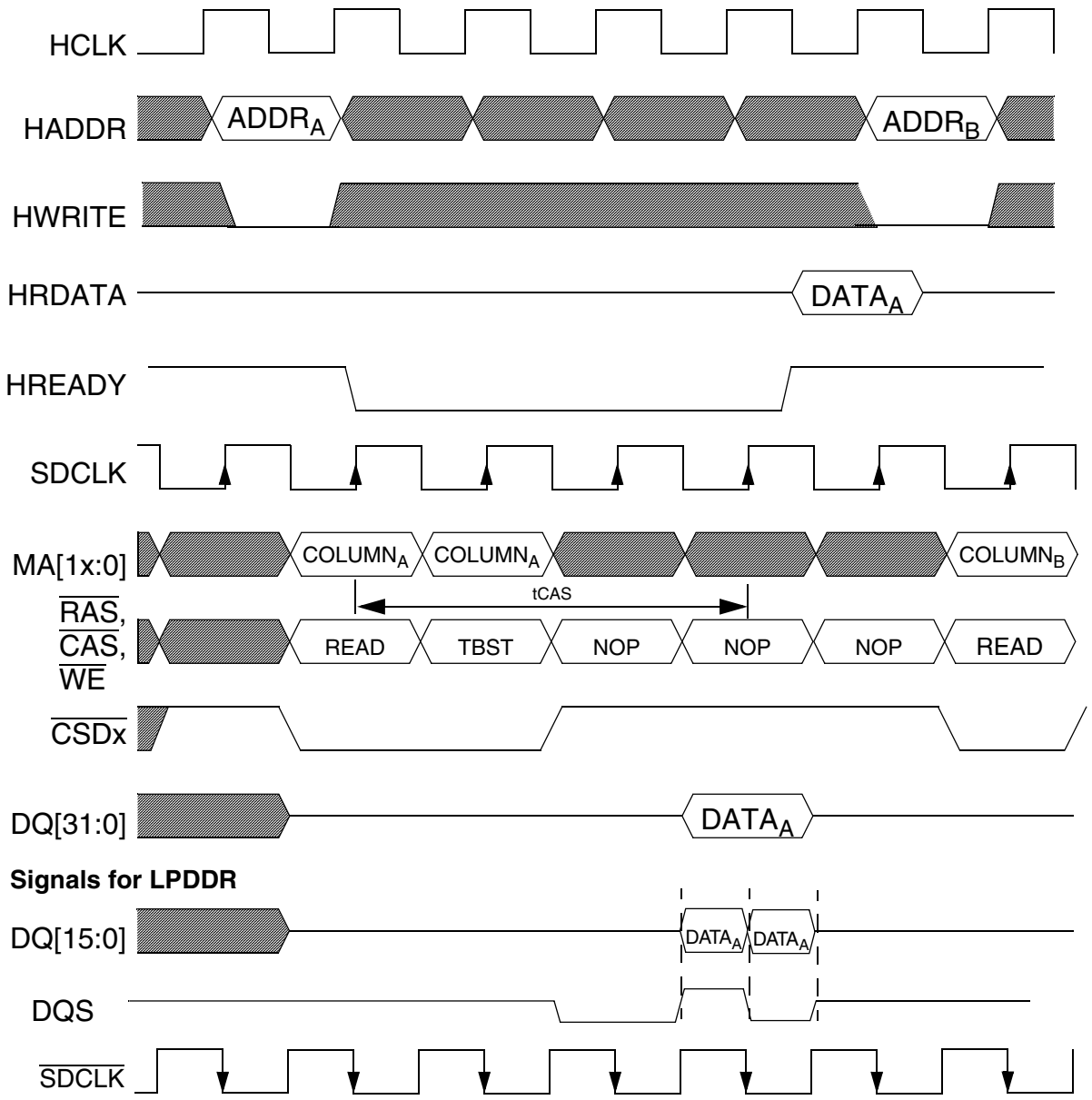


Figure 24-47. SDR and LPDDR On-Page Single Read Timing Diagram (32-Bit SDR, 16-Bit LPDDR)

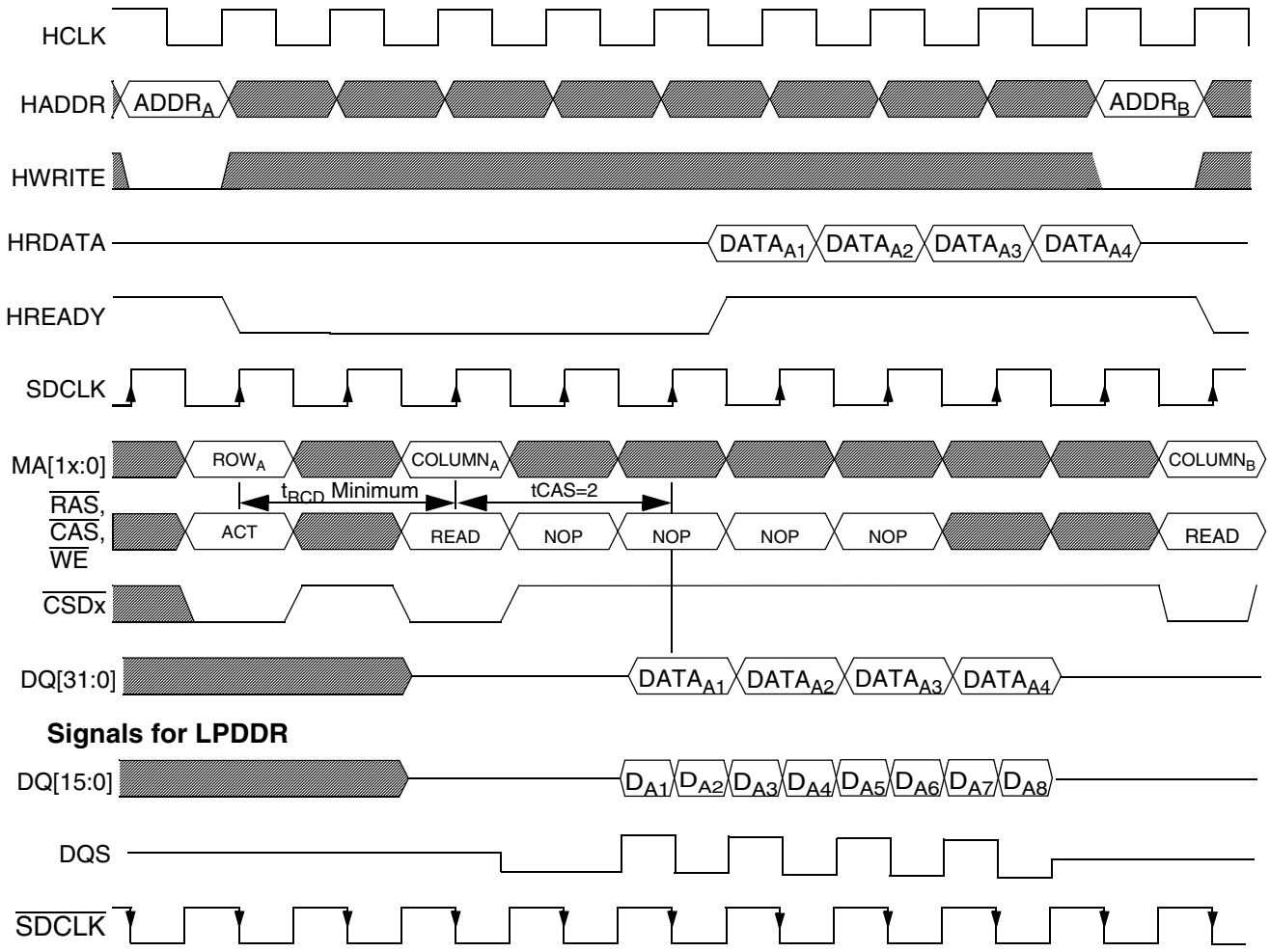


Figure 24-48. SDR and LPDDR Off-Page Burst Read Timing Diagram (32-Bit SDR, 16-Bit LPDDR)

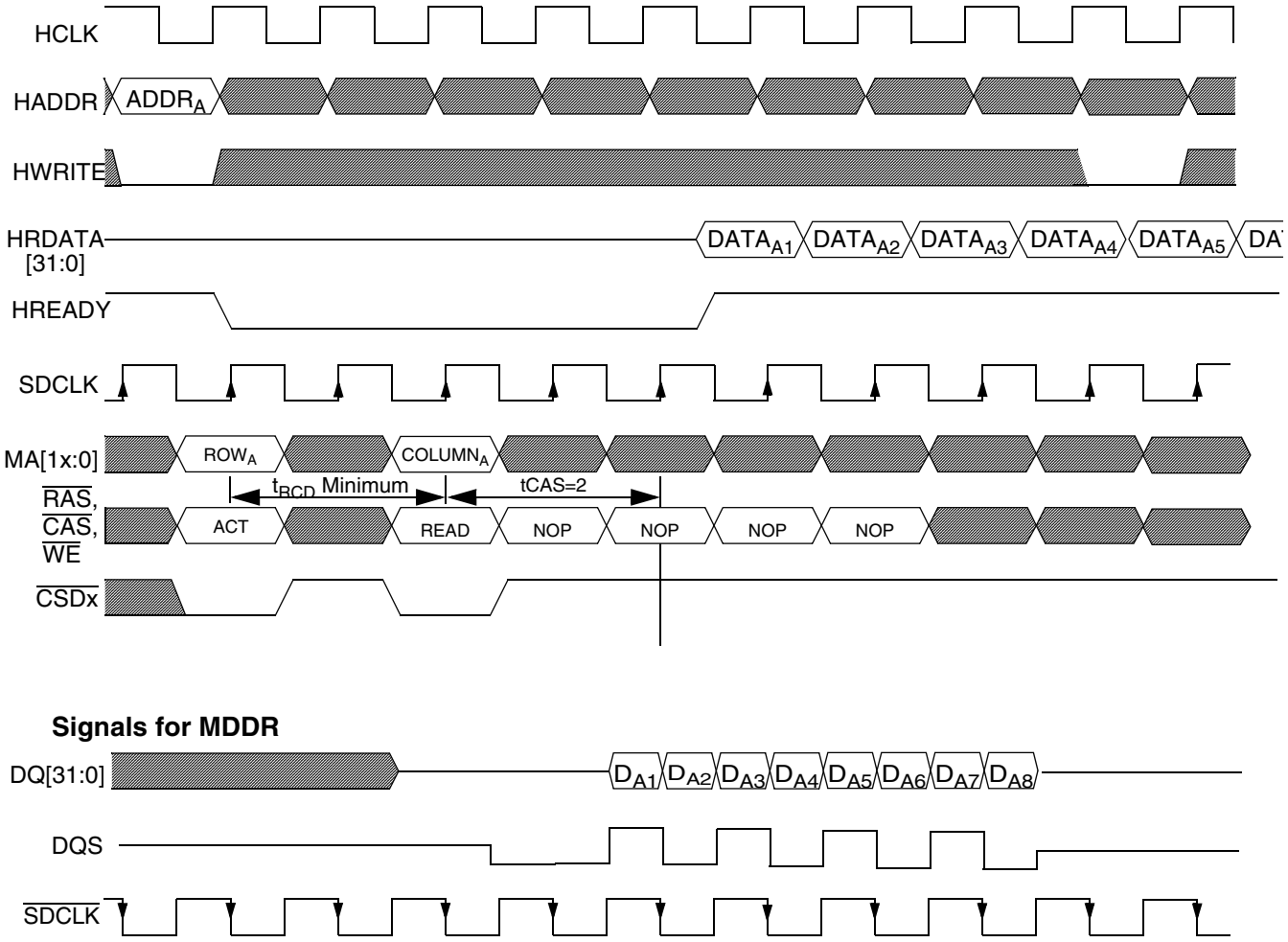


Figure 24-49. AHB 32-bit Read From LPDDR: Off-Page Burst Read Timing Diagram (32-Bit SDR and LPDDR)

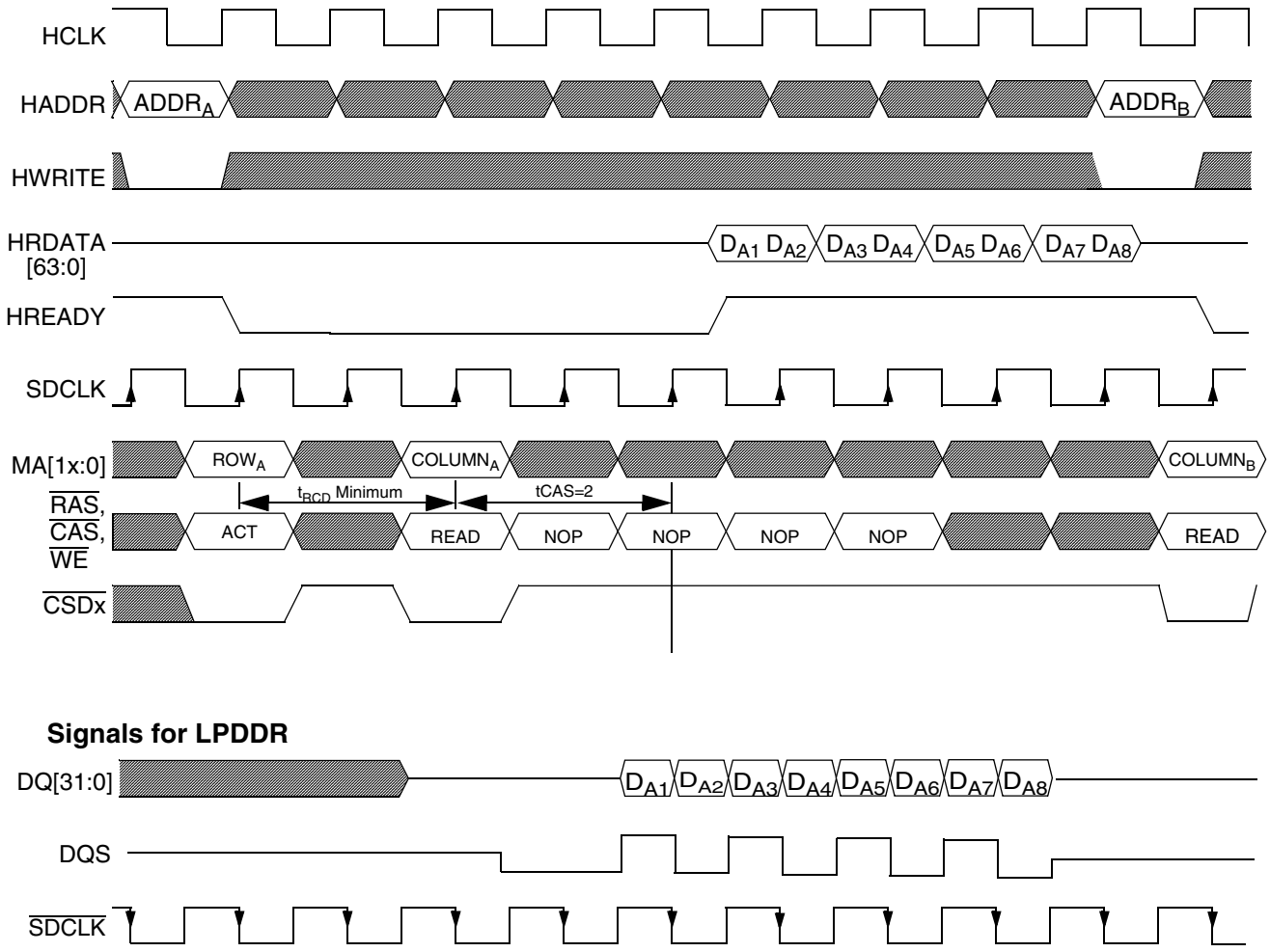


Figure 24-50. AHB 64-bit Read From LPDDR: Off-Page Burst Read Timing Diagram (32-Bit SDR and LPDDR)

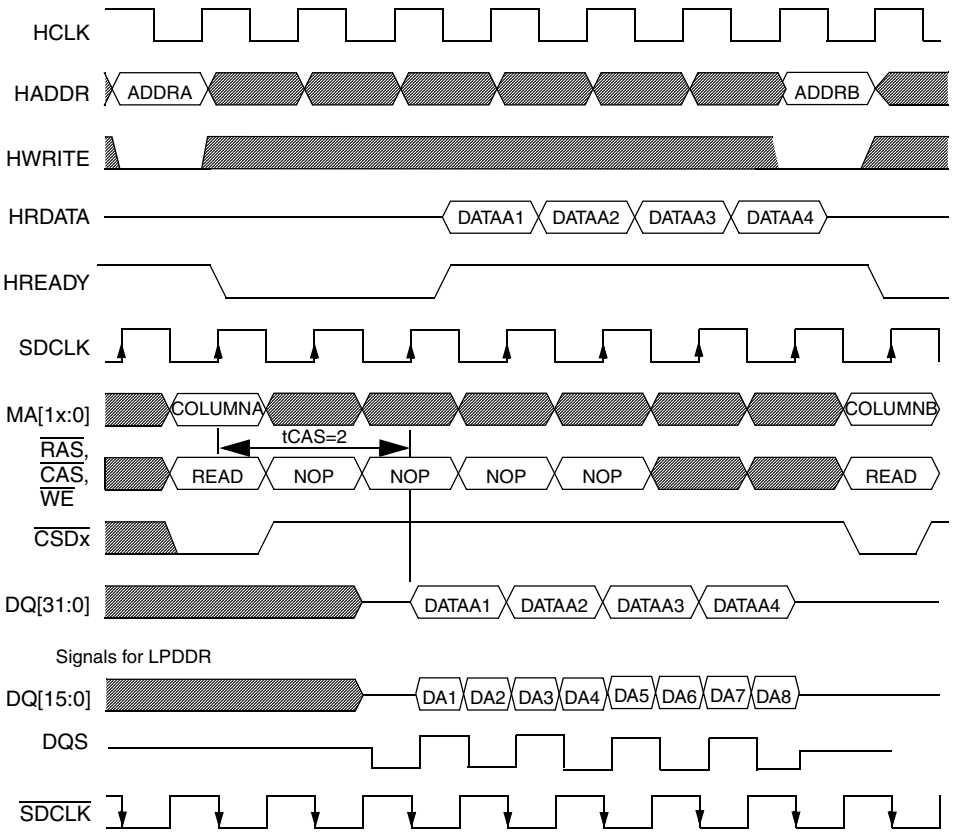


Figure 24-51. SDR and LPDDR On-Page Burst Read Timing Diagram (32-Bit SDR, 16-Bit LPDDR)

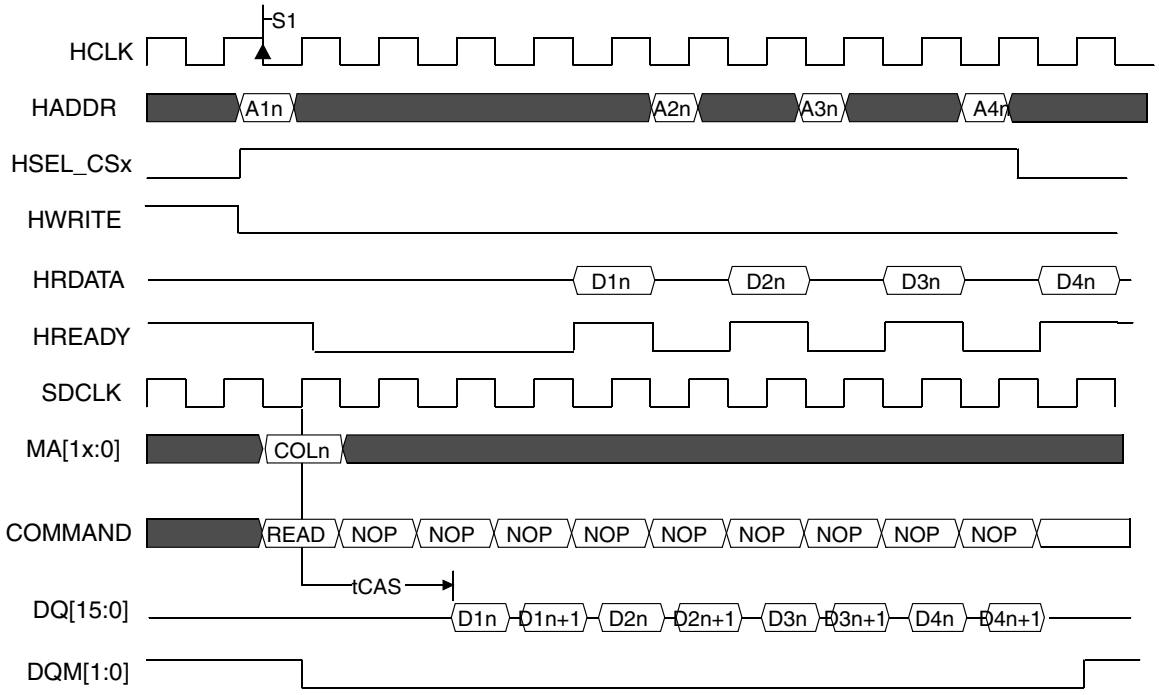


Figure 24-52. On-Page Burst Read Timing Diagram (16-Bit SDR: 8-Bit LPDDR is not Supported)

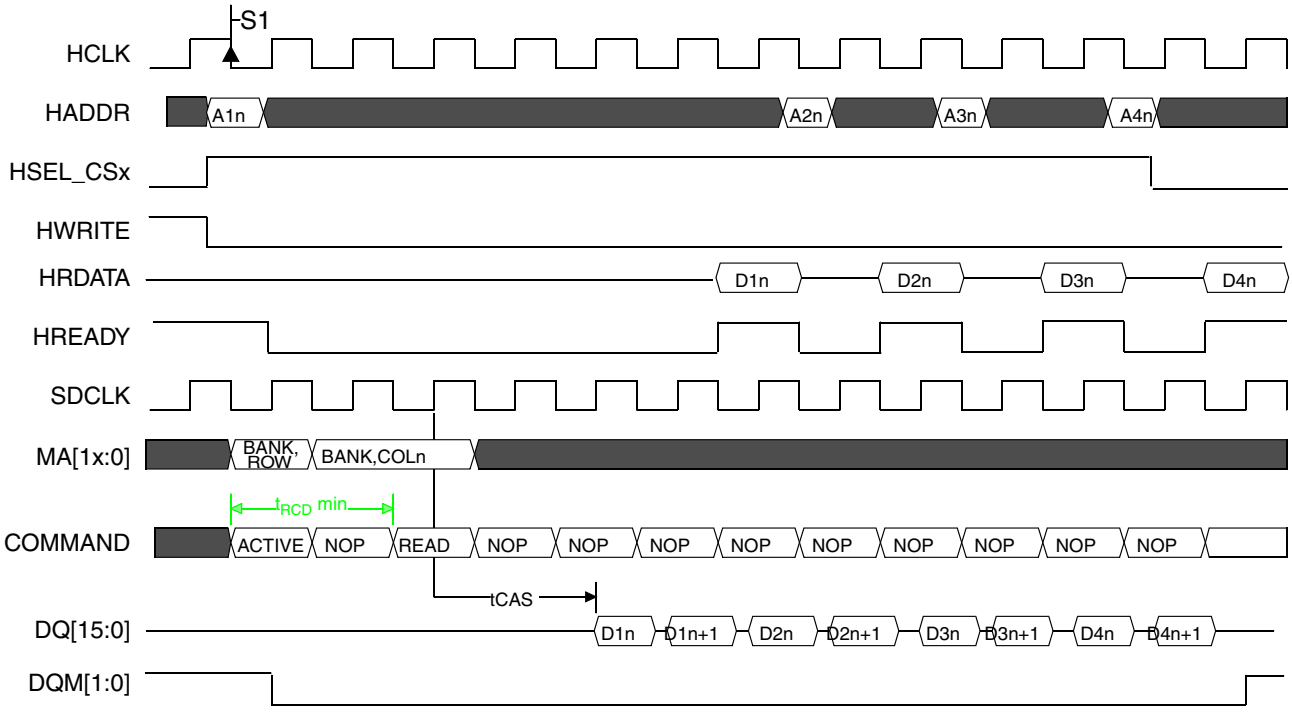


Figure 24-53. Off-Page Burst Read Timing Diagram (16-Bit SDR; 8-Bit LPDDR is not Supported)

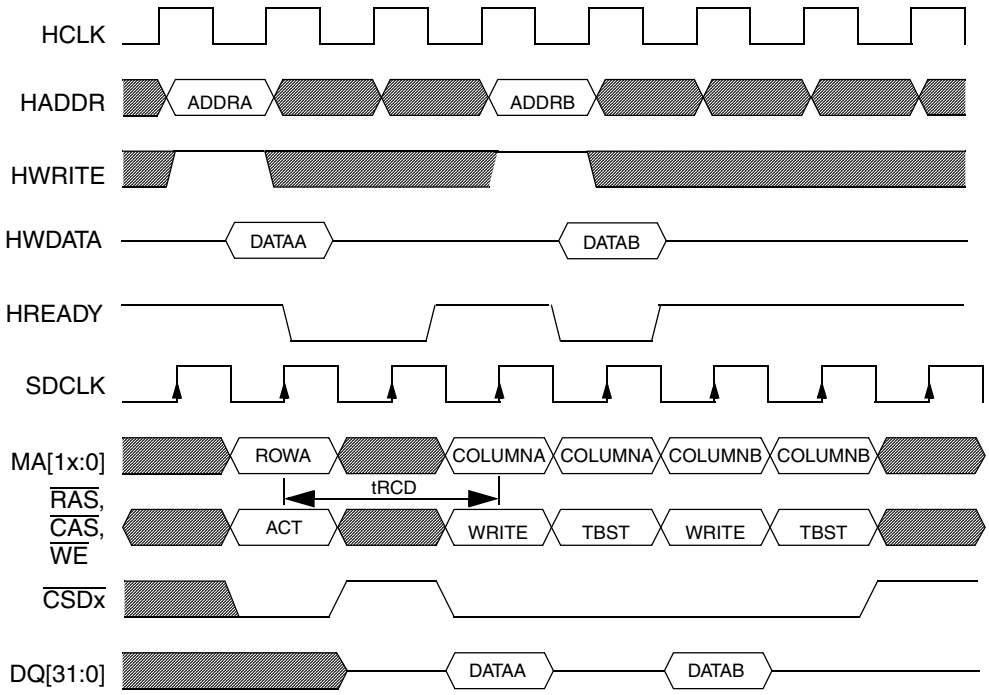
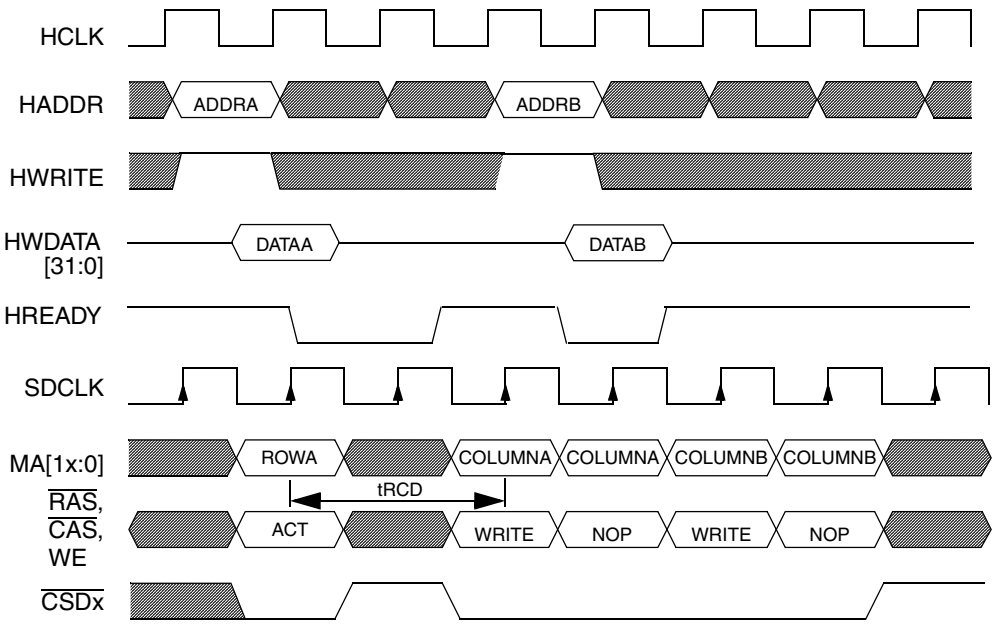


Figure 24-54. SDR Off-Page Write Followed by On-Page Write Timing Diagram



Signals for LPDDR

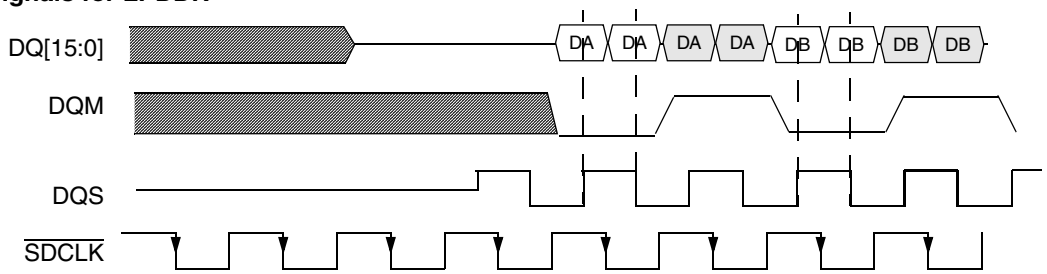


Figure 24-55. LPDDR Off-Page Write Followed by On-Page Write Timing Diagram

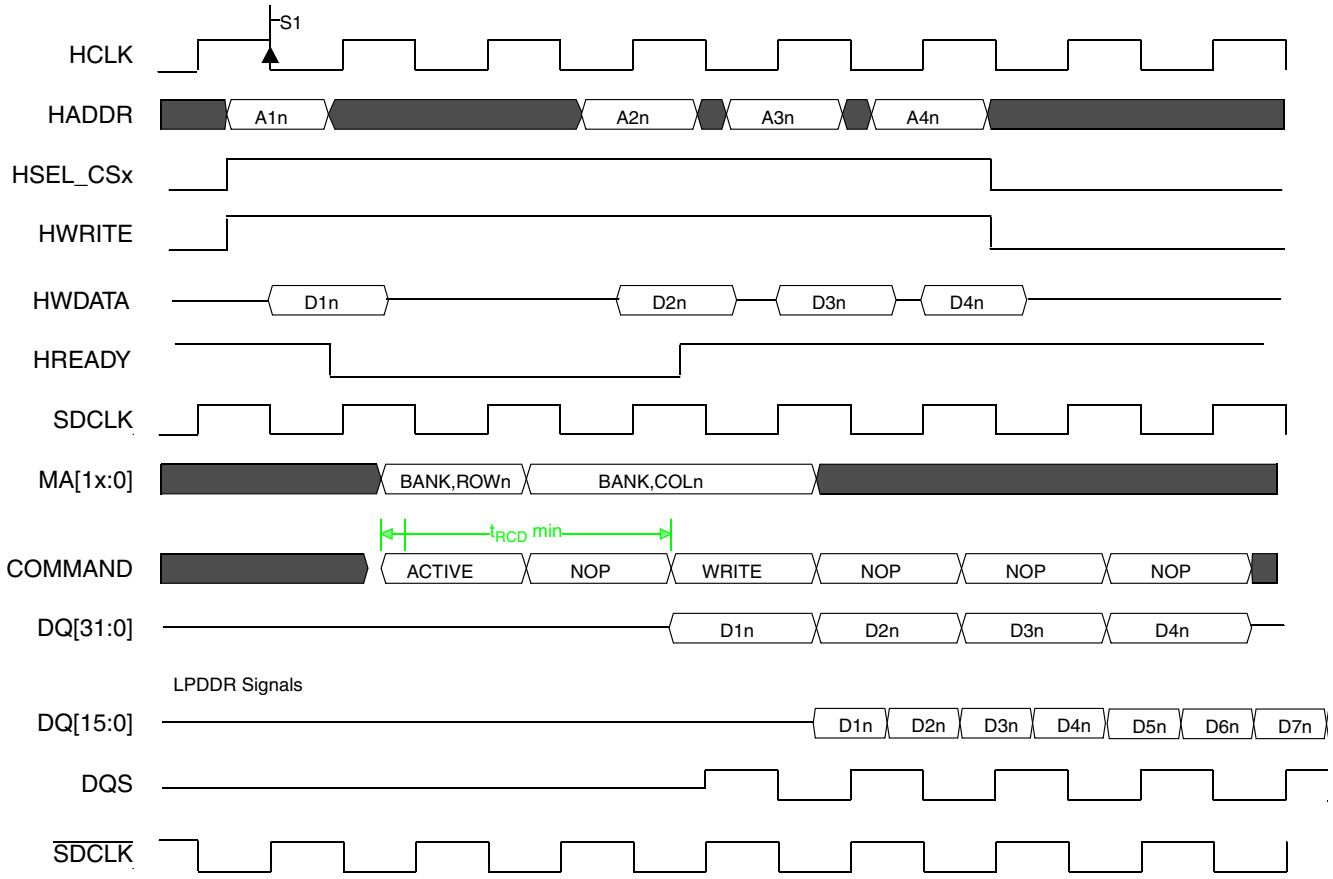


Figure 24-56. Off-Page Burst Write Timing Diagram (32-bit Memory for SDR, 16-bit for LPDDR)

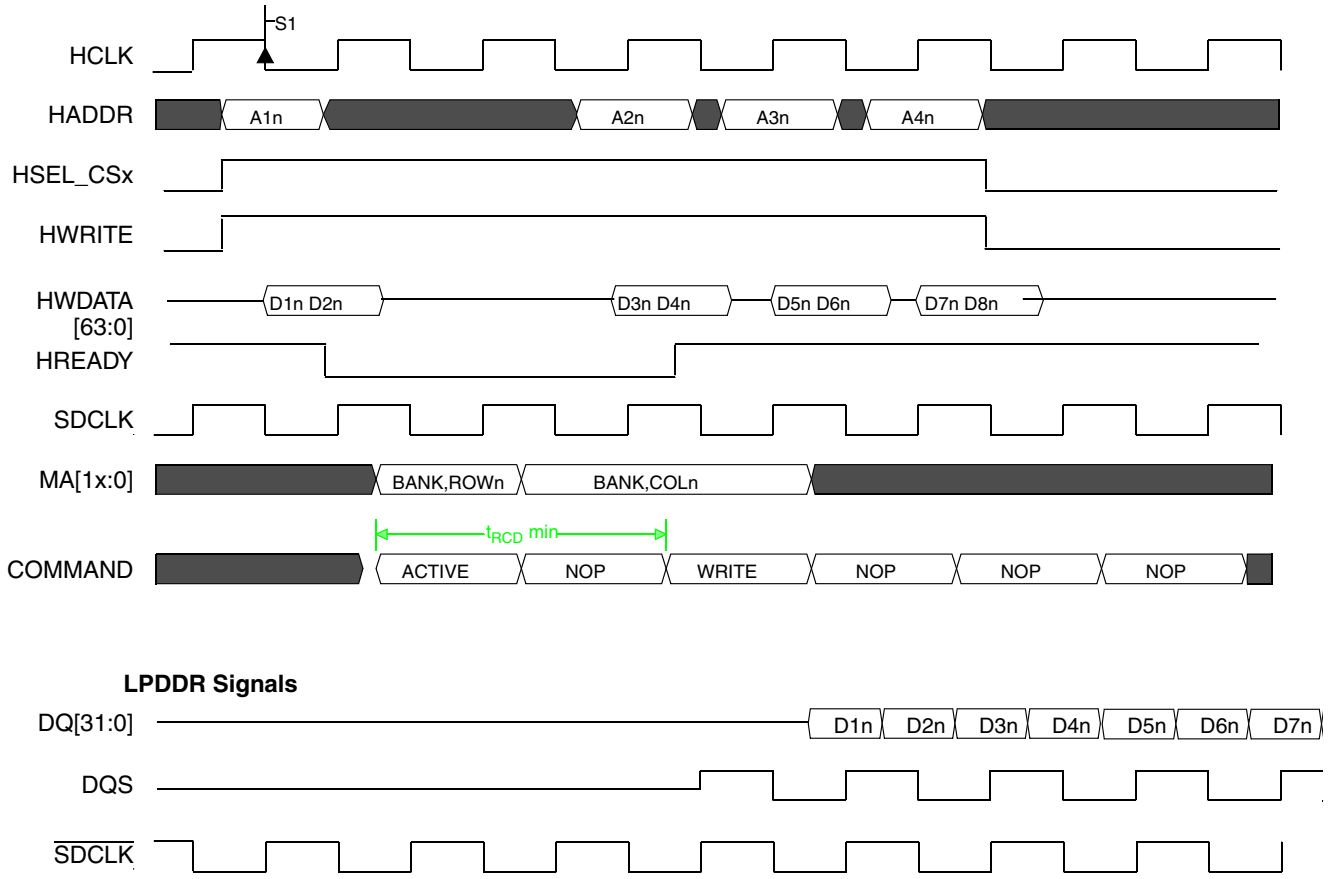


Figure 24-57. AHB 64-bit write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

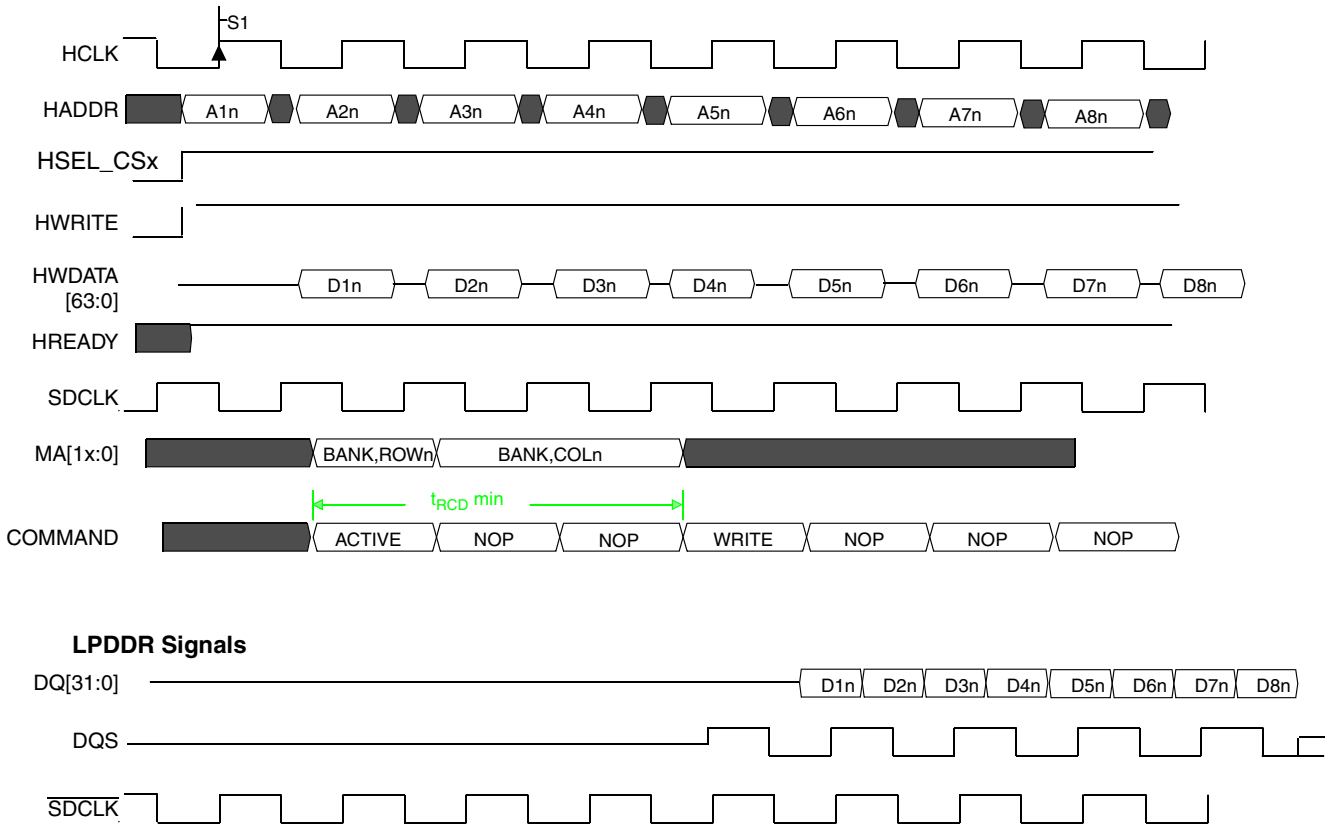


Figure 24-58. AHB 32-bit write to LPDDR: Off-Page Burst Write Timing Diagram (32-Bit Memory)

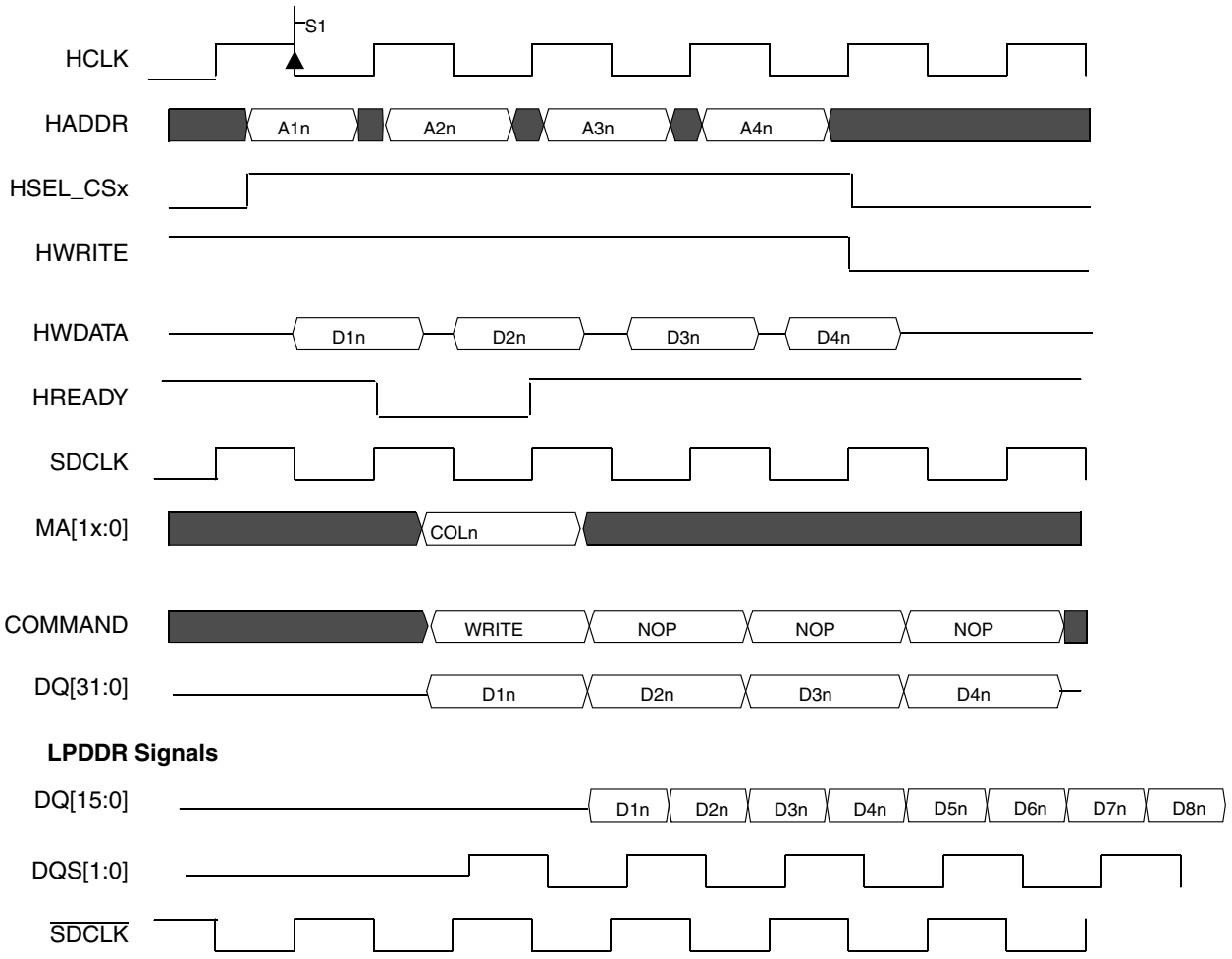


Figure 24-59. On-Page Burst Write Timing Diagram (32-Bit Memory for SDR, 16-Bit for LPDDR)

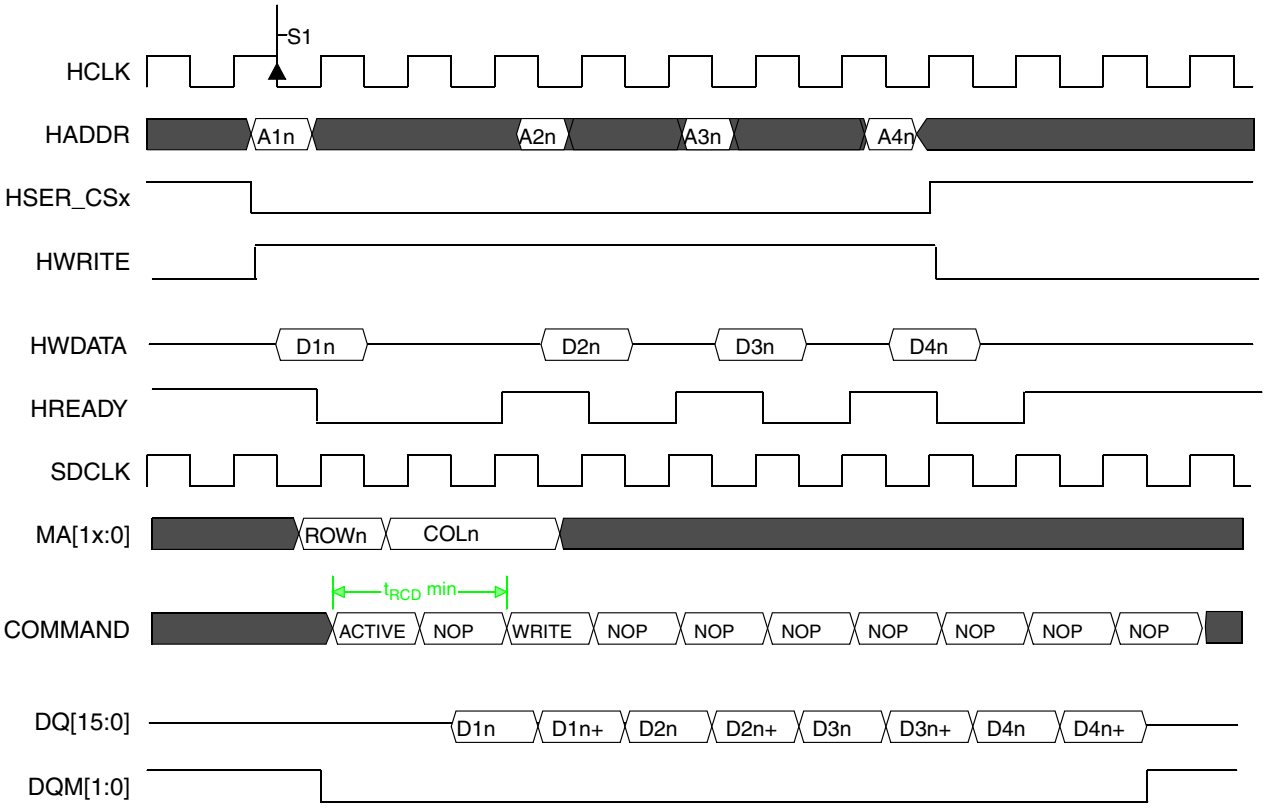


Figure 24-60. Off-Page Burst Write Timing Diagram—SDR 16-Bit Memory (LPDDR 8-Bit is not Supported)

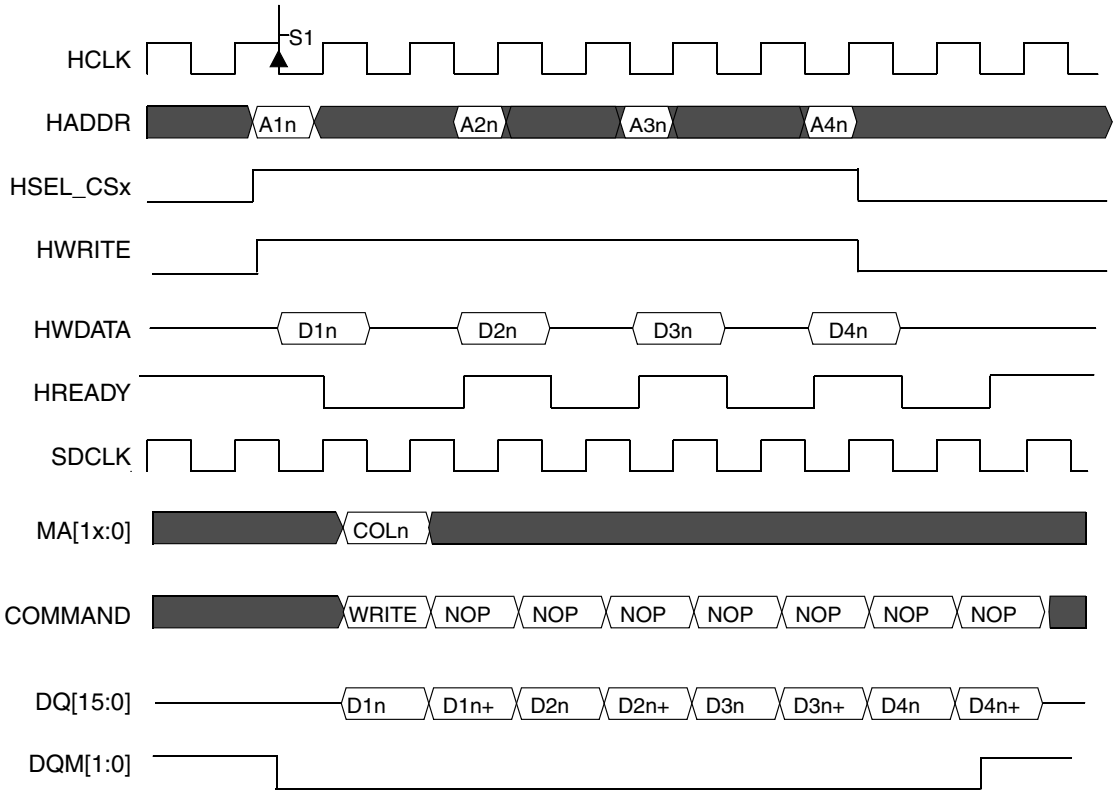


Figure 24-61. On-Page Burst Write Timing Diagram—SDR 16-Bit Memory (LPDDR 8-Bit Memory is not Supported)

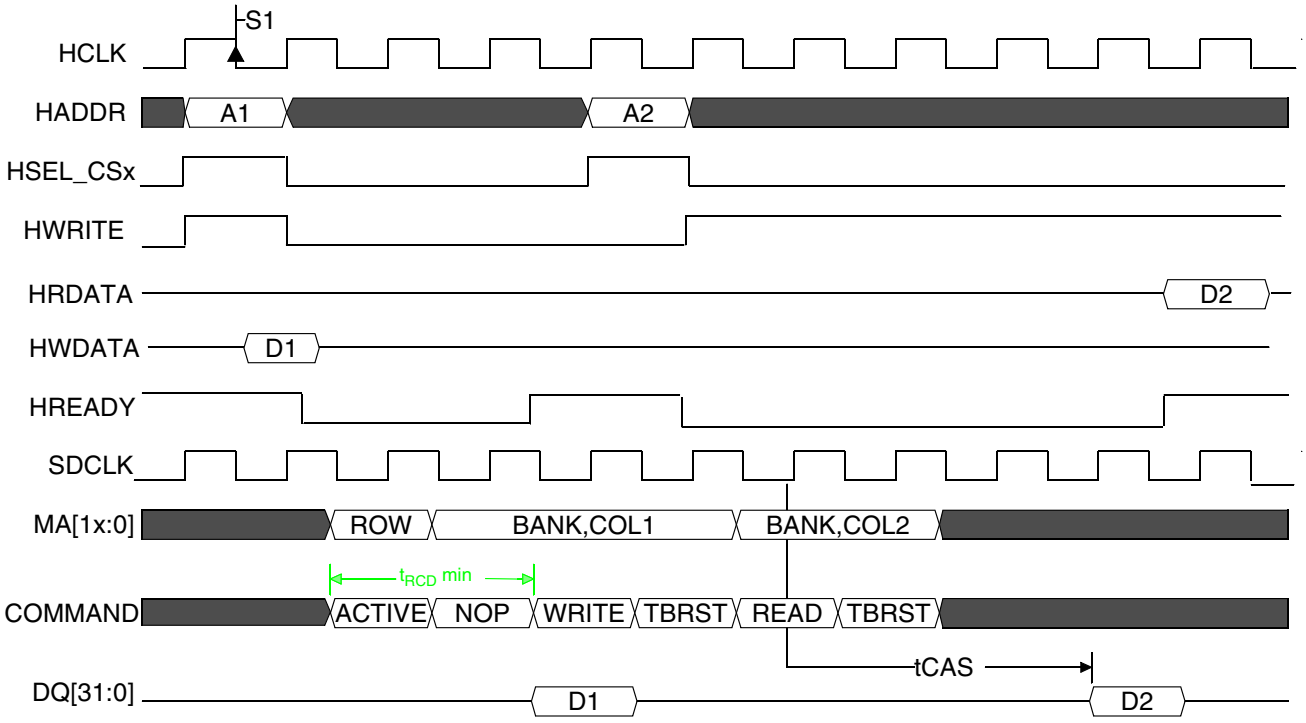


Figure 24-62. SDR Single Write Followed by On-Page Read Timing Diagram

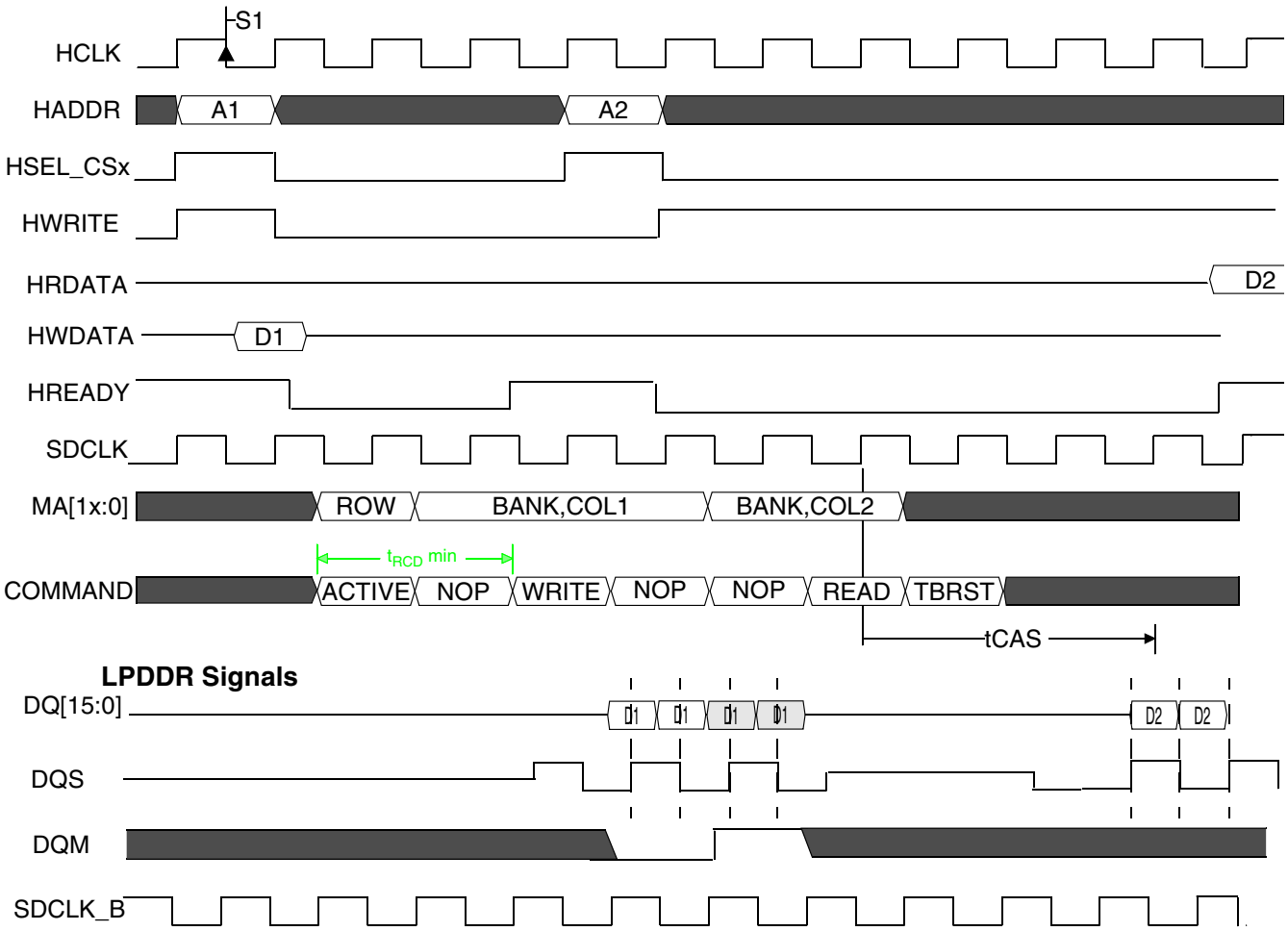


Figure 24-63. LPDDR Single Write Followed by On-Page Read Timing Diagram

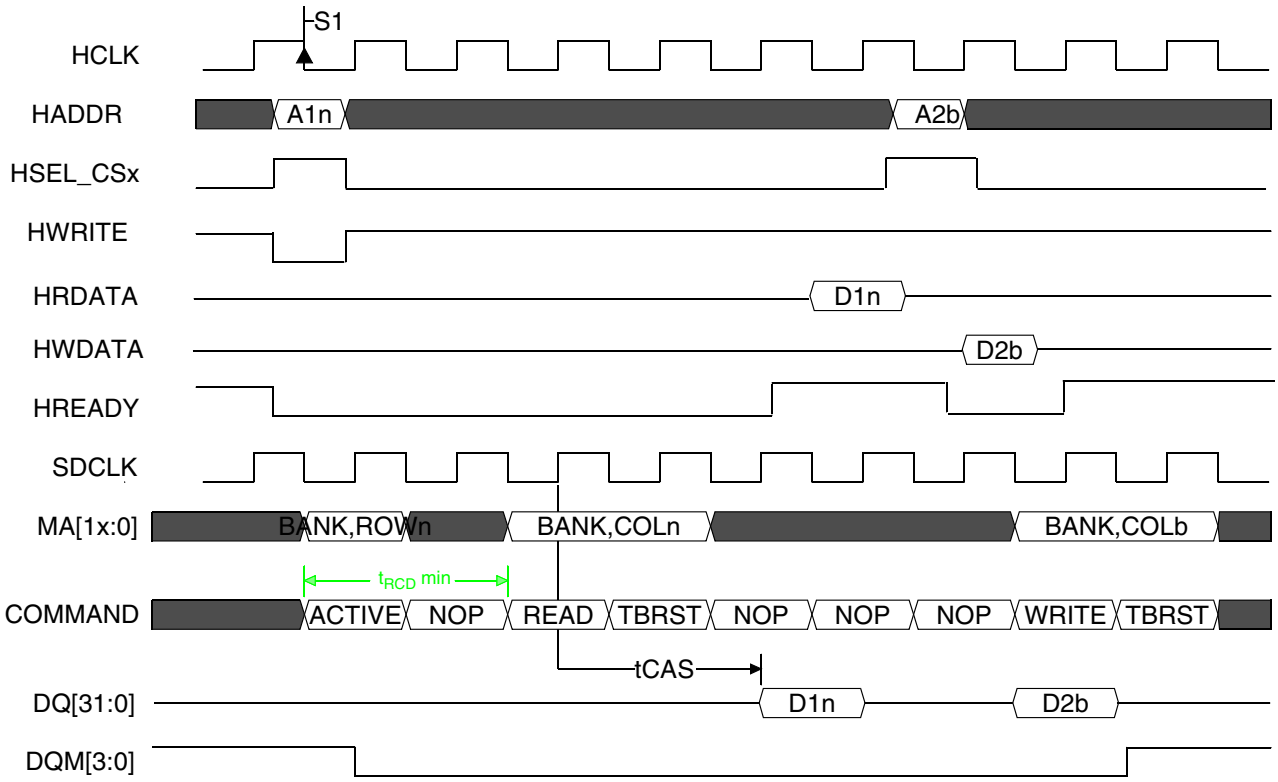


Figure 24-64. SDR Single Read Followed by On-Page Write Timing Diagram

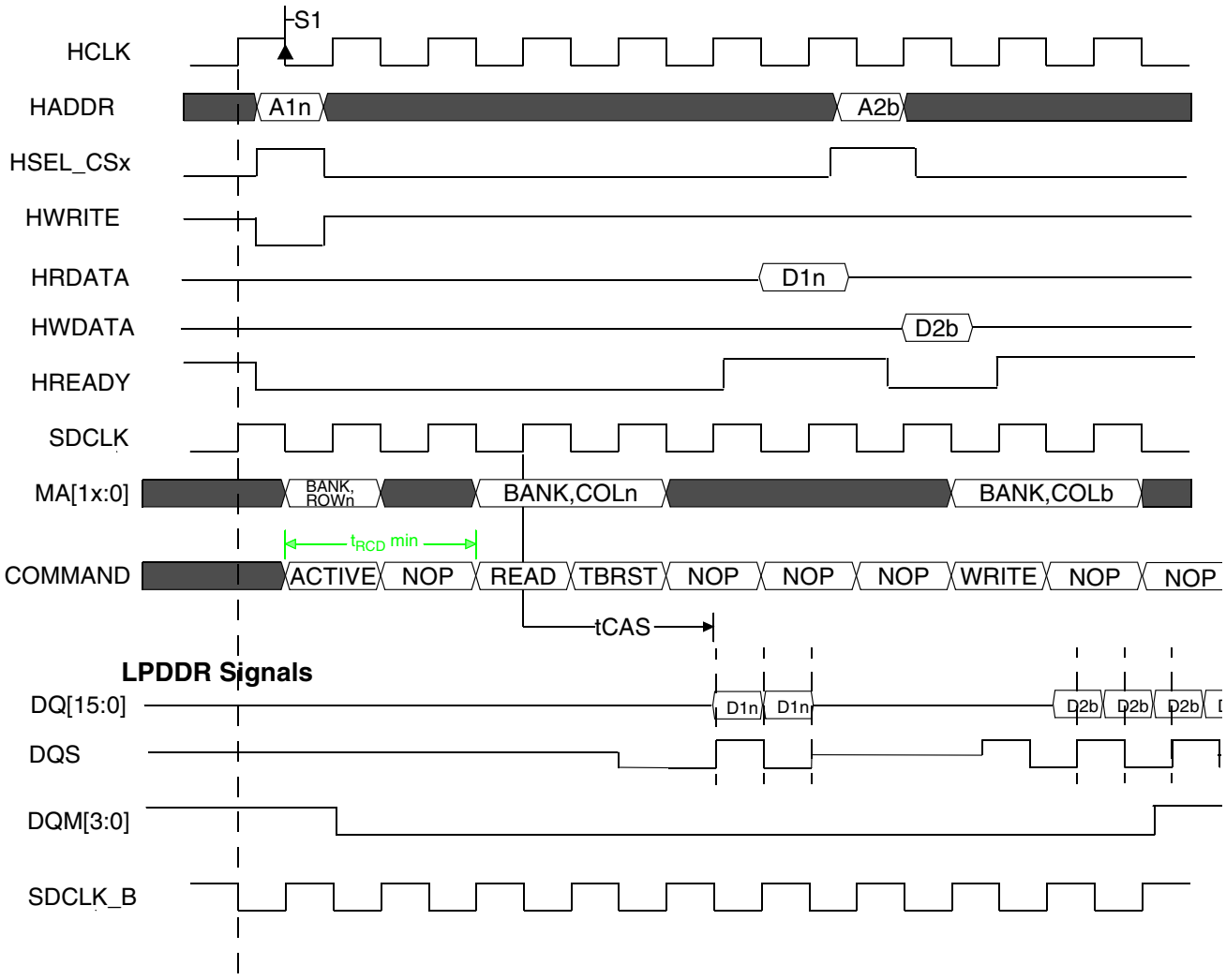


Figure 24-65. LPDDR Single Read Followed by On-Page Write Timing Diagram

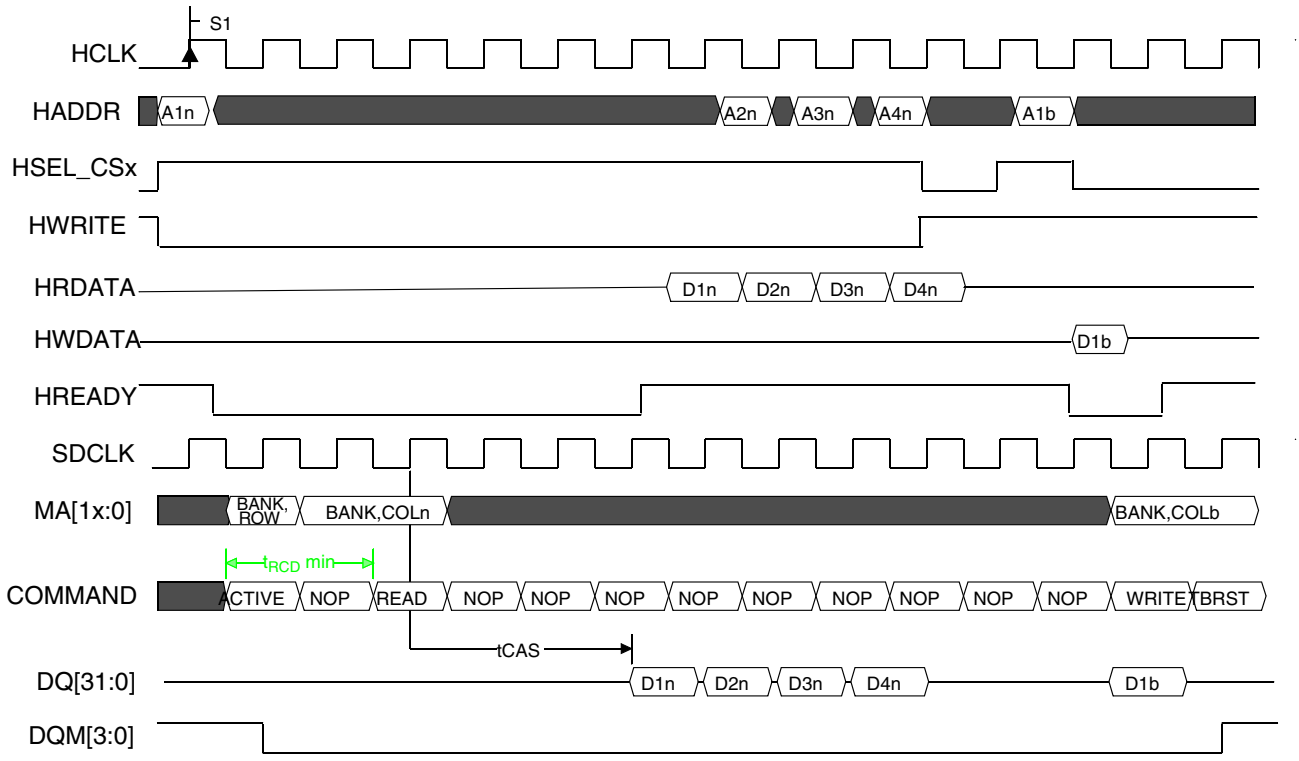


Figure 24-66. SDR Burst Read Followed by On-Page Write Timing Diagram

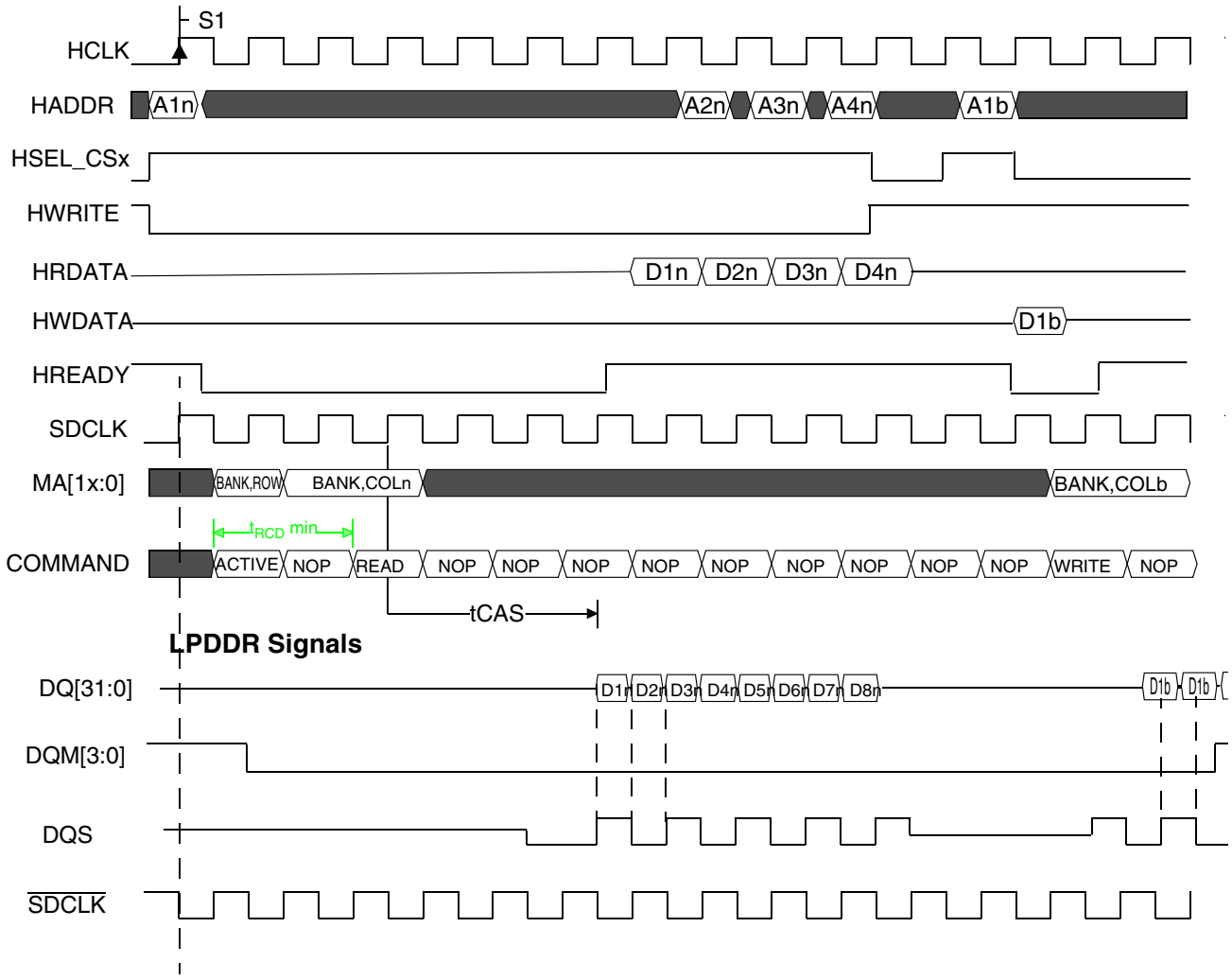


Figure 24-67. LPDDR Burst Read Followed by On-Page Write Timing Diagram

24.4.8.1 SDR Cycle Accurate Enhanced SDRAM Controller Accesses

This section provides cycle accurate timing diagrams for several ESDRAMC (AMBA AHB-Lite) supported read and write accesses to bit data width SDR memory devices from only one master. The diagrams are provided to emphasize ESDRAMC performance for single master hit (ACTIVE row) requests. The CAS latency for all diagrams is 2 cycles and burst length is set to 4 words for 16-bit memory.

24.4.8.1.1 Single Read Word Access to 16-bit Memory

The markers in Figure 24-68 marks the request access time, for example, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

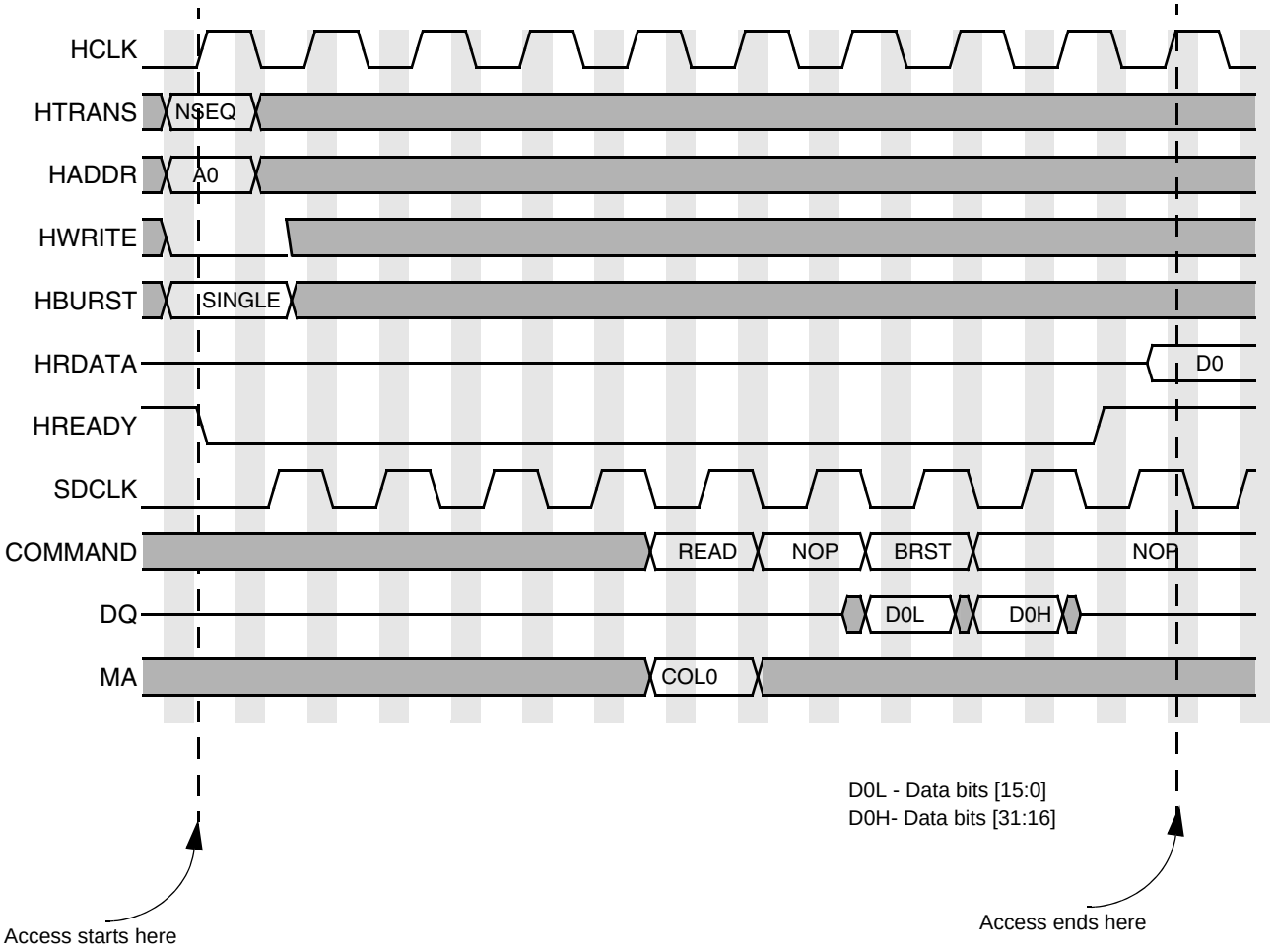


Figure 24-68. Single on Page Read—Word Access to 16-Bit Memory (Cycle Accurate)

24.4.8.1.2 Misaligned INCR4 Burst Read Access to 16-Bit Memory

The markers in Figure 24-69 marks the request access time, for example, the time period between HREADY goes LOW (the ESDRAMC starts to execute the request) and HREADY goes HIGH (the ESDRAMC request execution is completed).

Two read commands are issued to the SDRAM memory, since the misaligned read burst crosses the 16-bit SDRAM (4 words) memory boundary. The read addresses are 0x04, 0x08, 0x0C, and 0x10. Without issuing the second read the last SDRAM data will come from address 0x00. The ESDRAMC issue the second read command in such a way (at a specific timing) so continuous data flow is obtained. There is no timing difference between an aligned and a misaligned access although the second one requires more commands.

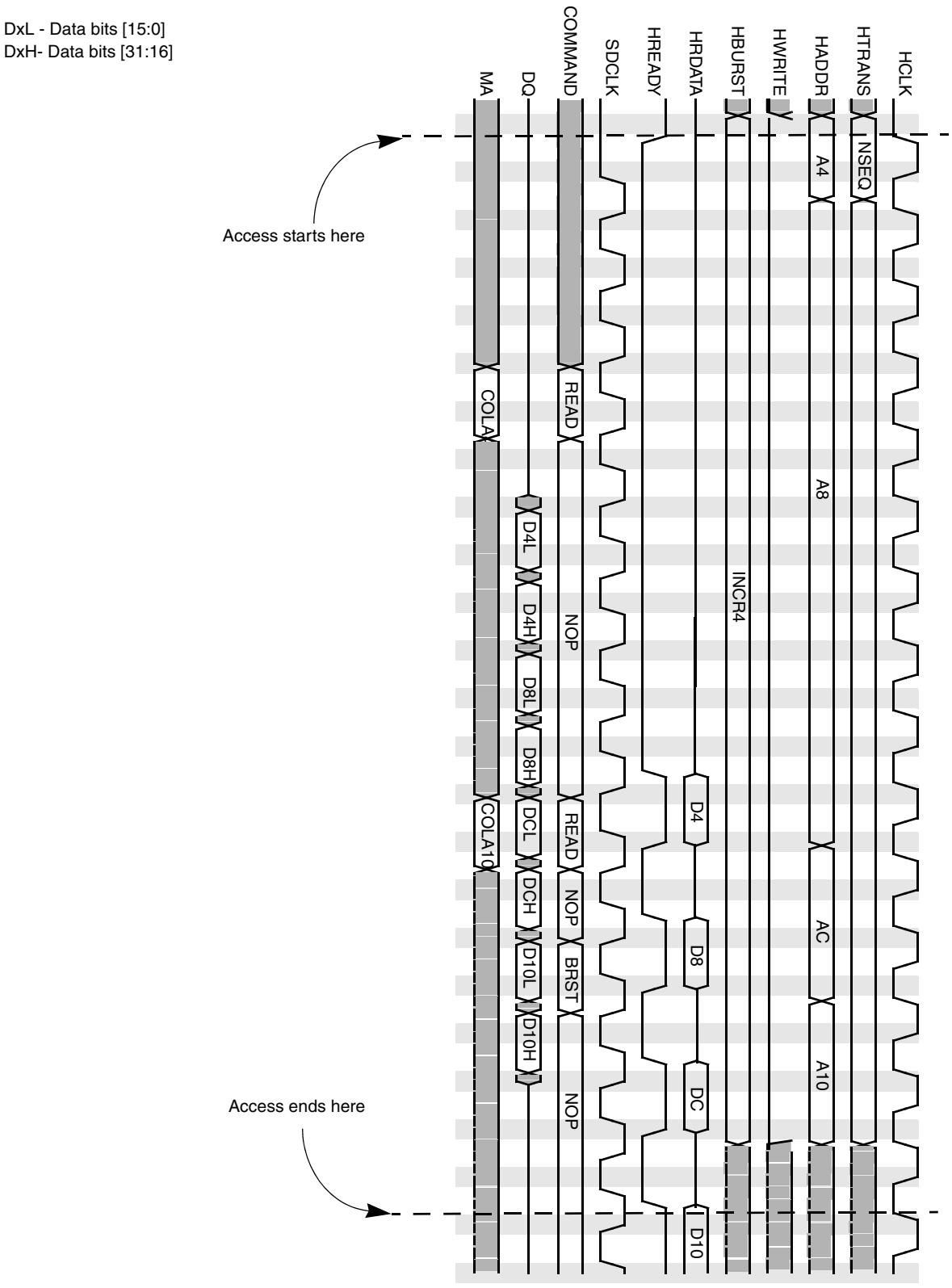


Figure 24-69. Misaligned on Page INCR4 Burst Read Access to 16-Bit Memory

24.4.8.1.3 Misaligned WRAP8 Burst Read Access to 32-bit Memory

The markers in Figure 24-70 marks the request access time, that is the time period between HREADY goes low (the ESDRAMC starts to execute the request) and HREADY goes high (the ESDRAMC request execution is completed).

Only one read command is issued to the SDRAM memory, since the misaligned read burst crosses the 32-bit SDRAM memory (8 words) boundary at the memory “natural” boundary. The read addresses are 0x04, 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C and 0x00. Without issuing the second read the last SDRAM data comes from address 0x00 since this is the “natural” 32-bit memory device boundary as well.

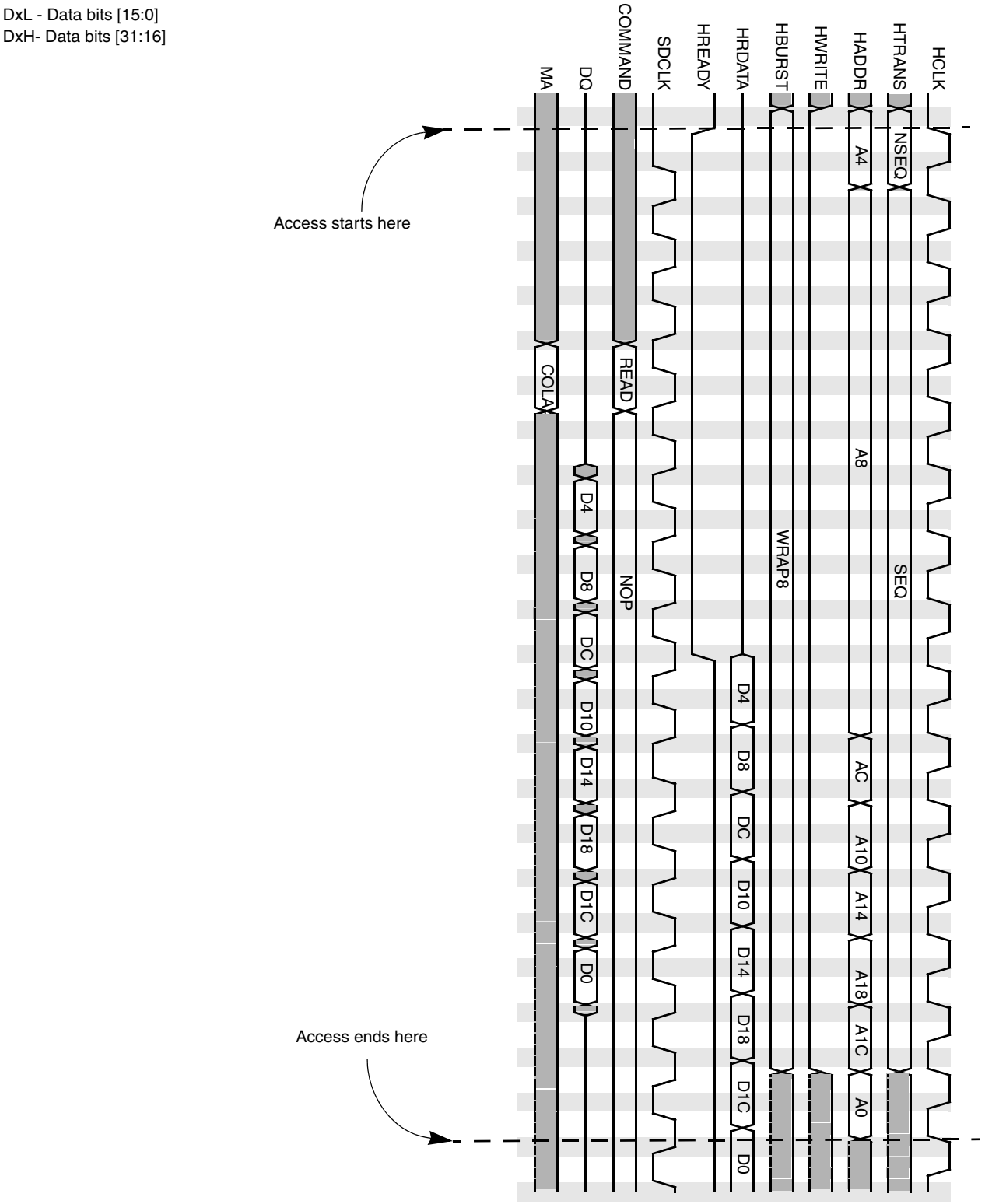


Figure 24-70. Misaligned WRAP8 Burst Read Access to 32-Bit Memory

24.4.8.2 SDRAM Command Sequence for Burst Accesses

Table 24-22 show the commands that the ESDRAMC performs for WRAP and INCR accesses from the AHB. The controller splits the transaction when needed in cases that the access cross WRAP boundaries of the SDRAM. The memory is configured to 8-beat bursts (BL=8).

Unspecified INCR burst accesses are translated to single accesses to allow the burst to terminate at any length with no additional delays.

The number in the brackets represent the address for read command and the last burst address for BTERM command.

Table 24-22. SDRAM Command Sequence for Burst Accesses

Type	Bus	Address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(0)	READ(0)	READ(0)	READ(0)
			READ(0)	READ(0)	READ(0)		READ(10)	READ(10)	READ(10)
			BTERM(0)	BTERM(4)	BTERM(8)		BTERM(10)	BTERM(14)	BTERM(18)
INCR8	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
		READ(10)	READ(10)	READ(10)	READ(10)	READ(20)	READ(20)	READ(20)	READ(20)
			READ(20)	READ(20)	READ(20)		READ(30)	READ(30)	READ(30)
			BTERM(20)	BTERM(24)	BTERM(28)		BTERM(30)	BTERM(34)	BTERM(38)
WRAP4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
INCR4	16-bit	READ(0)	READ(4)	READ(8)	READ(C)	READ(10)	READ(14)	READ(18)	READ(1C)
			READ(10)	READ(10)	READ(10)		READ(20)	READ(20)	READ(20)
			BTERM(10)	BTERM(14)	BTERM(18)		BTERM(20)	BTERM(24)	BTERM(28)

24.4.9 Precharge Command Mode

The precharge command mode (SMODE = 0b001) is used during SDRAM/LPDDR device initialization, and to manually deactivate an active bank(s). While in this mode, an access (either read or write) to the SDRAM/LPDDR address space generate a precharge command cycle. SDRAM/LPDDR address bit A10 determines whether a single bank, or all banks, are precharged by the command. Accessing an address with the SDRAM/LPDDR address A10 low precharges only the bank selected by the bank addresses, as illustrated in Figure 24-71. Conversely, accesses with A10 high precharges all banks regardless of the bank address, as illustrated in Figure 24-72. Note that A10 is the SDRAM pin, not the A10 bit ARM address bus. Translation of the SDRAM A10 to the corresponding ARM address is dependent on the memory configuration. The precharge command access is two clocks in length on the ARM, and one cycle to the SDRAM/LPDDR.

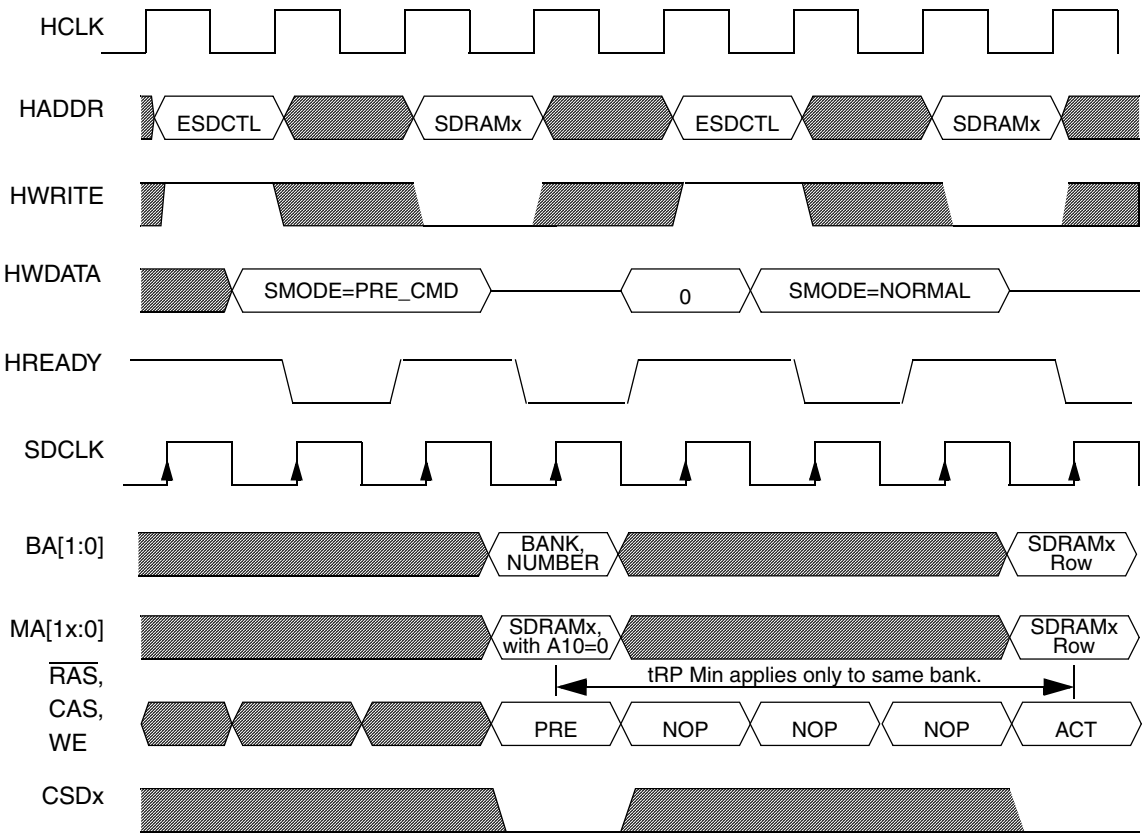


Figure 24-71. Precharge Specific Bank Timing Diagram

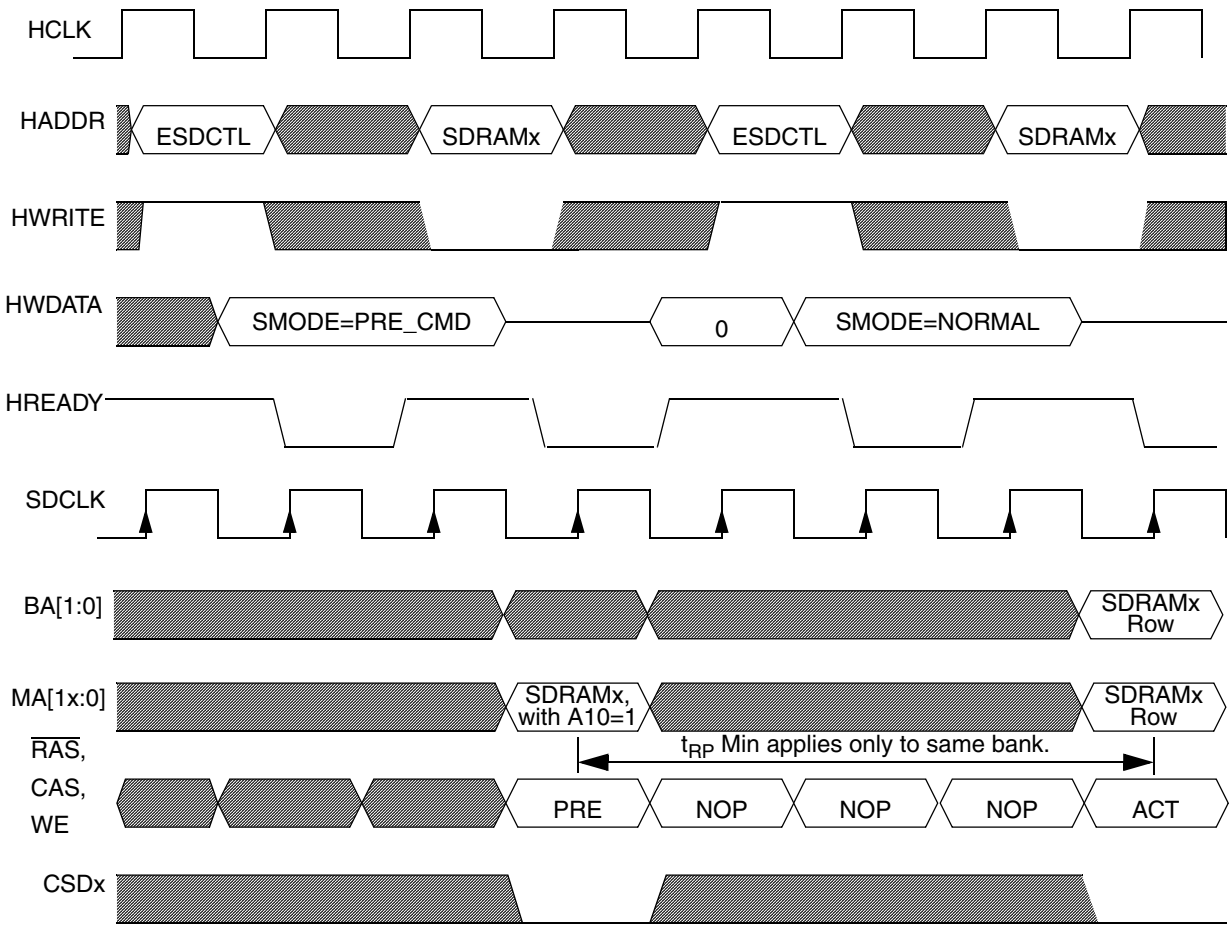


Figure 24-72. Precharge-All Banks Timing Diagram

24.4.10 Auto-Refresh Mode

The auto-refresh mode (SMODE=010) is used to manually request SDRAM/LPDDR refresh cycles and is normally used only during device initialization, since the ESDRAMC automatically generates refresh cycles when properly configured. The auto-refresh command (see Figure 24-73) refreshes all banks in the device, therefore the address supplied during the refresh command need only specify the correct SDRAM/LPDDR device. The lower address lines are a don't care. Either a read or write cycle may be used. If a write is used, the data is ignored and the external data bus is not driven. The cycle is 2 clocks on the ARM and a single clock to the SDRAM/LPDDR device.

The ESDRAMC does not guarantee that the SDRAM/LPDDR is in the idle state before the auto-refresh command is given. If one or more rows are active, a precharge-all command should be issued by the software prior to the auto-refresh command.

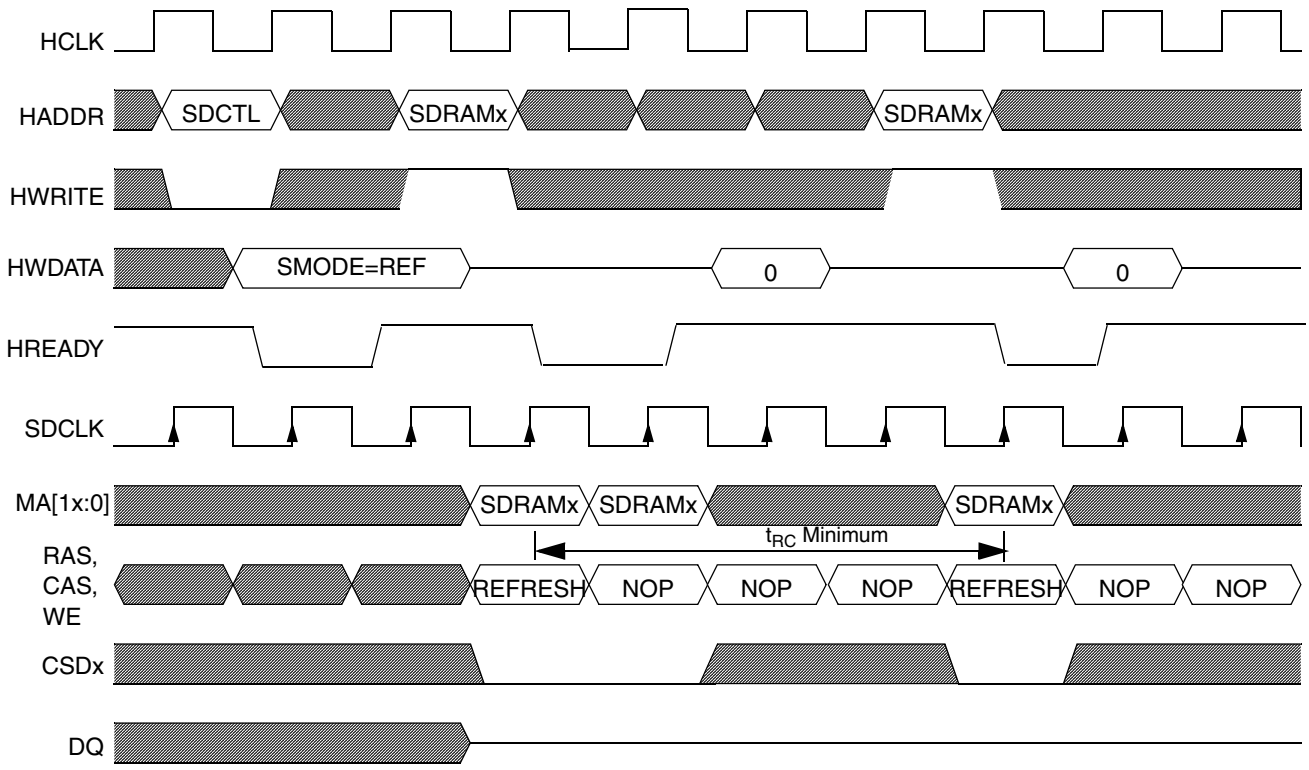


Figure 24-73. Software-Initiated Auto-Refresh Timing Diagram

24.4.11 Manual Self-Refresh Mode

The manual self-refresh mode (SMODE=100) is used to enter the SDRAM/LPDDR external device in self-refresh low power operating mode during the RUN system operating mode. Details regarding this operating mode are provided in Section 24.4.6.2, “Manual Self-Refresh Mode for SDRAM/LPDDR Devices.”

24.4.12 Load Mode Register Mode

Load mode register mode (SMODE=011) is used to program the SDRAM/LPDDR mode register (see Example 24-1 and Section 24.5.4.2, “SDR SDRAM Load Mode Register”). This mode differs from normal SDRAM write cycles because the data to be written is transferred across the address bus. Reads of the mode register are not allowed.

After SMODE bits are set to 011, either a read or write cycle may be used to write the external memory device mode register. In both cases (read or write), the ARM data is ignored and the external data bus is not driven. The row and bank address signals are used to transfer the data to the external memory device mode register (see Example 24-1 and Section 24.5.4.2, “SDR SDRAM Load Mode Register”). The cycle is 2 clocks on the ARM and a single clock to the SDRAM device.

Figure 24-74 illustrates the bus sequence for a load mode register operation. Load mode register commands must be issued while the SDRAM/LPDDR is idle. The enhanced SDRAM controller does not guarantee that the SDRAMs have been returned to the idle state before issuing the load mode register

command. Software is responsible to generate a precharge-all sequence before issuing the load mode register command if there is any possibility that one or more banks could be active. Also note, the row cycle time (t_{RC}) must be met before the load mode register command is issued.

Section 24.5.4.2, “SDR SDRAM Load Mode Register,” provides a detailed example of the mode register data value calculation and mapping to the ARM address.

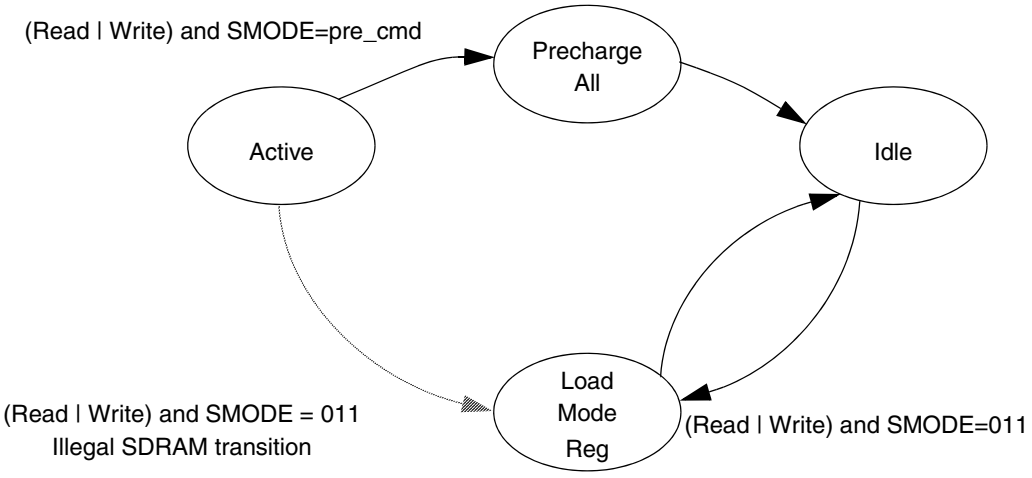


Figure 24-74. Load Mode Register State Diagram

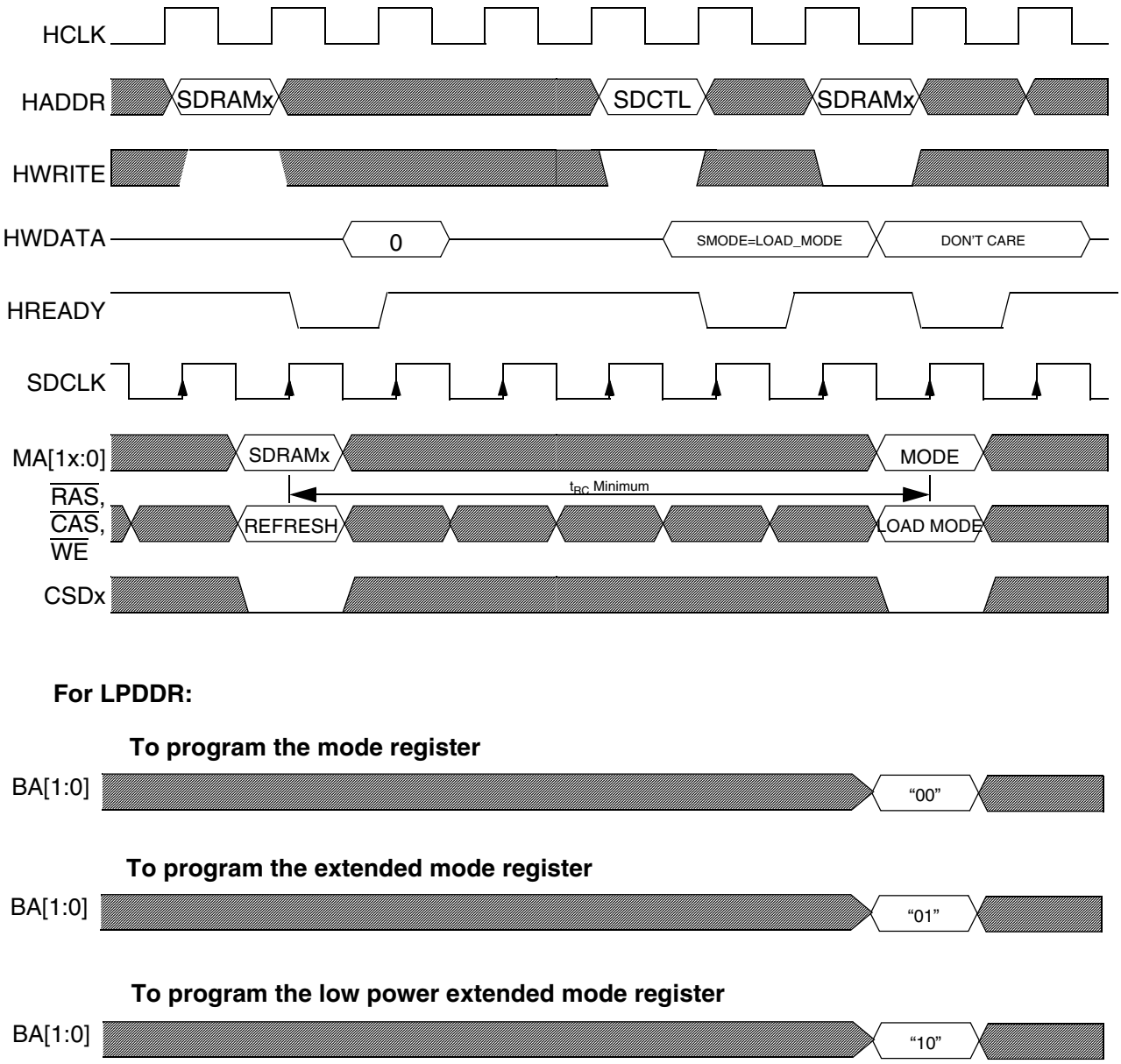


Figure 24-75. SDR and LPDDR Load Mode Register Timing Diagram

24.5 Initialization/Application Information

The following paragraphs provide details on selecting compatible SDRAM memories and configuring the controller to work with the memory system.

24.5.1 Memory Device Selection

Many SDRAM/LPDDR memory types are supported by the enhanced SDRAM controller. Important characteristics to consider when choosing a memory device are:

- The page comparators expect 4-bank memories. 2-bank devices are not supported, and their use results in the memory and controller losing synchronization when crossing page boundaries.
- Page (column) addressing must match one of the supported sizes.
- Memory density can be larger or smaller than those directly supported, although some memory may be inaccessible or redundantly mapped. The bank addresses are the most significant addresses and connecting a memory smaller than the selected density results in one or more banks being inaccessible.
- The controller is designed for memories meeting PC133 timing specifications up to 133 MHz system operation. Use of non-compliant memories require a thorough analysis of all timing parameters.

24.5.2 Configuring the Controller for SDRAM Memory Arrays

Configuration register programming options and controller-to-memory physical connections provide flexibility to accommodate different memory types and system configurations. Options are broadly grouped into 3 categories:

- Physical characteristics: row and column address bus widths and data bus width.
- Timing parameters: CAS latency, row precharge, cycle delays, refresh rate, etc.
- Functional features: clock suspend timer and supervisor/user protection.

Table 24-25 through Table 24-30 are provided to assist the designer with the selection of the correct physical parameters for a number of preferred memory configurations. Timing parameters are addressed in the following subsections.

24.5.3 CAS Latency

CAS latency is determined by the operating frequency and access time of the memories. For a 133 MHz system clock frequency and PC133-compatible memories, the CAS latency is generally programmed as 3 clocks, although it is recommended that memory specifications be consulted to confirm this value. CAS latency must be programmed in two places: the chip select Control Register and the device Mode Register. See Table 24-9 for a description of the control register encoding and Section 24.4.12, “Load Mode Register Mode,” for the details on programming the SDRAM mode register.

24.5.4 SDRAM/LPDDR Initialization Sequence

Prior to normal operation (read/write accesses), the external memory device must be initialized. The following paragraphs provide detailed information covering device initialization. Register definition, command descriptions and device operation information has been thoroughly described throughout the chapter.

24.5.4.1 SDRAM Initialization

SDR and LPDDR SDRAMs must be powered up and initialized in a predefined manner. Operational procedures other than those specified by the SDRAM manufacture specification may results in undefined operation.

24.5.4.1.1 SDR SDRAM Initialization

Once power is applied to the device and the clock is stable, the SDRAM requires a 200µs delay prior to issuing any command other than a COMMAND INHIBIT or a NOP. Starting at some point during this 200µs period and continuing at least through the end of this period, COMMAND INHIBIT or NOP commands should be applied.

Once the 200µs delay has been satisfied and at least one COMMAND INHIBIT or NOP command has been applied (the SDRAM_RDY status bit is asserted), a precharge command should be applied. All banks must then be precharged, thereby placing the device in the all banks idle state.

Once in idle state, several (manufacture-dependent) auto-refresh cycles must be performed. After the auto-refresh cycles are complete, the SDRAM is ready for mode register programming. Because the mode register powers up in an unknown state, it is recommended to be loaded prior to applying any operational command. Figure 24-76 illustrates a SDRAM initialization routine with 8 auto-refresh cycles. Table 24-1 shows an initialization SDRAM example code.

It is crucial that the dual parameters (those parameters that are defined both in the SDRAM device register and in ESDRAMC registers, like CAS latency, burst length, etc.) to be identical for proper operation of both SDRAM memory and enhanced SDRAM controller.

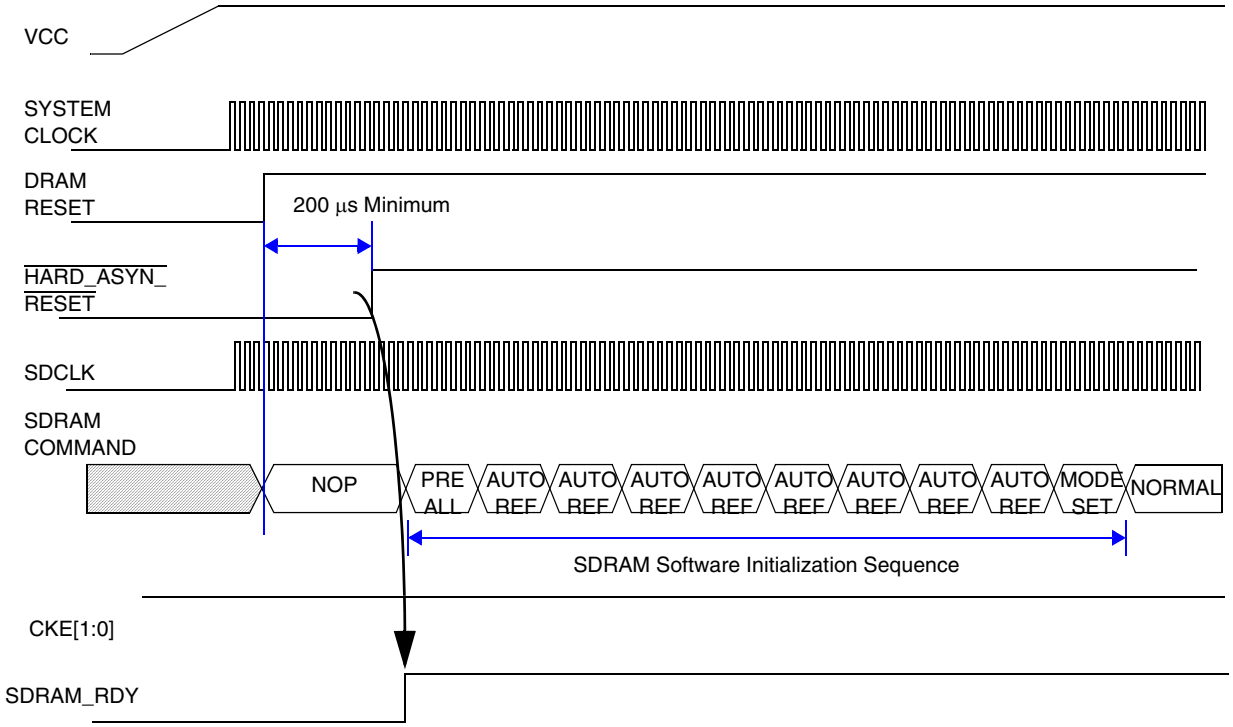


Figure 24-76. SDR SDRAM Initialization and Load Mode Register Sequence

Example 24-1. INIT_SDRAM Example Code (ARM Assembler)

```
init_sdram:
                ldr     r2, =ESD_ESDCTL0      // base address of registers
                ldr     r3, =PRE_ALL_CMD      // SMODE=001
                str     r3,(r2,#0x0)          // put CSD0 in precharge command mode
                ldr     r4, =SDRAM_CSD0       // CSD0 precharge address (A10=1)
                str     r1,(r4,#0x0)          // precharge CSD0 all banks
                ldr     r3, =AUTO_REF_CMD     // SMODE=010
                str     r3,(r2,#0x0)          // put array 0 in auto-refresh mode
                ldr     r4, =SDRAM_CSD0_BASE  // CSD0 base address
                ldr     r6,=0x7               // load loop counter
0:              ldr     r5,(r4,#0x0)          // run auto-refresh cycle to array 0
                subs    r6,r6,#1              // decrease counter value
                bne     0b
                ldr     r3, =SET_MODE_REG_CMD // SMODE=011
                str     r3,(r2,#0x0)          // setup CSD0 for mode register write
                ldr     r3, =MODE_REG_VAL0    // array 0 mode register value
                ldrb    r5,(r3,#0x0)          // New mode register value on address bus

                ldr     r3, =NORMAL_MODE      // SMODE=000
                str     r3,(r2,#0x0)          // setup CSD0 for normal operation




ESD_ESDCTL0             .long   0xXXXX_XXXX   // system/external device dependent data
SDRAM_CSD0:             .long   0xXXXX_XXXX   // system/external device dependent data
SDRAM_CSD0_BASE:        .long   0xXXXX_XXXX   // system/external device dependent data
PRE_ALL_CMD             .long   0xXXXX_XXXX   // system/external device dependent data (SMODE=001)
AUTO_REF_CMD            .long   0xXXXX_XXXX   // system/external device dependent data (SMODE=010)
SET_MODE_REG_CMD        .long   0xXXXX_XXXX   // system/external device dependent data (SMODE=011)
MODE_REG_VAL0           .long   0xXXXX_XXXX   // system/external device dependent data
NORMAL_MODE             .long   0xXXXX_XXXX   // system/external device dependent data (SMODE=000)
```

NOTE

To load the mode register the address starts from bit 0, so ldrb should be used.

24.5.4.1.2 LPDDR SDRAM Initialization

The DDR mobile SDRAM (LPDDR) must be powered up and initialized in a predefined manner. Operational procedures other than those specified may result in undefined operation. Power must first be applied to VDD and VDDQ according to the LPDDR SDRAM manufacture data sheet. Clock enable mast be driven through the SDRAM controller registers to CS0 and/or CS1.

After all power supply voltages are stable, and the clock is stable, the DDR Mobile-SDRAM requires a 200µs delay prior to applying a command other than DESELECT or NOP.

CKE is driven high by the SDRAM controller on the first edge of the clock.

Once the 200 µs delay has been satisfied, the following command sequence shall be applied using the SDRAM controller registers:

1. A DESELECT or NOP command.
2. A PRECHARGE ALL command should then be applied, placing the device in the all banks idle state.

3. Once in the idle state, two auto-refresh cycles must be performed (tRFC must be satisfied).
4. Two load mode register commands for the mode register and extended mode register.

Following these cycles, the LPDDR is ready for normal operation.

NOTE

A write access should be performed before the first read access to the LPDDR (in order to assure that a 0 value is driven on the DQS pins and held by the keeper of the DDR pads).

Figure 24-77 shows the LPDDR SDRAM initialization and load mode register sequence.

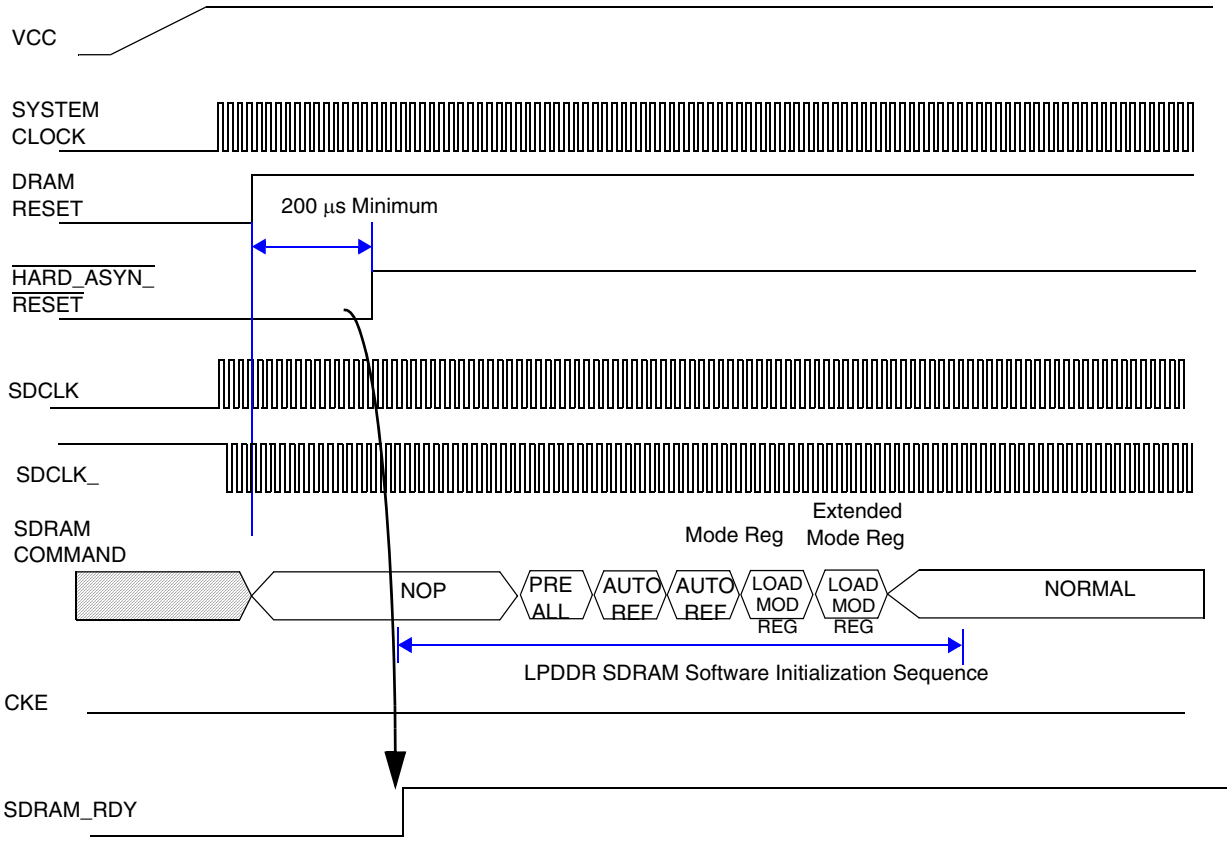


Figure 24-77. Simplified LPDDR SDRAM Initialization and Load Mode Register Sequence

24.5.4.2 SDR SDRAM Load Mode Register

The mode register is used to set the SDRAM operating characteristics including CAS latency, burst length, burst mode, and write data length. The settings depend on system characteristics including the operating frequency, memory device type, burst buffer/cache line length, and bus width. Operating characteristics vary by device type, so the data sheet must be consulted to determine the actual value to be written. In order to demonstrate the procedure, the following system characteristics is used:

- Micron MT48LC4M32B2 128Mbit (1M x 32 x 4 banks) SDR SDRAMs
- 133 MHz System Clock Frequency

- Sequential burst, burst length of 8
- Single word writes (for example, no bursting on writes)

Figure 24-78 shows the mode register bit assignments for the Micron 128 Mbit SDRAM. Table 24-23 lists the bit field descriptions.

	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
TYPE	Reserved*		WB	Op Mode		CAS Latency			BT	Burst Length		

Figure 24-78. 128 Mbit SDR SDRAM Mode Register

Table 24-23. SDRAM Mode Register Description

Name	Description
A11-A10	Reserved
A9 Write Burst	Write Burst Mode (WB). Selects between burst writes and single location writes. 0 Programmed Burst Length 1 Single Location Access[1]
A8–A7 Op Mode	Operating Mode. Defines operating modes. 00 Standard operation All other states reserved
A6–A4 CAS Latency	CAS Latency (CL). Sets latency between column address and data 000 Reserved 001 1 clock[1] 010 2 clocks 011 3 clocks 1*xx* Reserved
A3 Burst Type	Burst Type (BT). Selects burst type 0 Sequential 1 Interleave
A2–A0 Burst Length	Burst Length (BL). A 16-bit wide SDRAM requires a burst length of eight because the four 32-bit line fill cycles is decomposed into eight 16-bit accesses. 000 = 1[1] 001 = 2[1] 010 = 4[2] 011 = 8 111 = Full Page 10x = Reserved 1x0 = Reserved

1 Not supported in Micron SDR devices

2 Not supported by the ESDRAMC for 16-bit external memory devices.

For this example:

- Sequential burst (BT = 0)
- Burst length of 8 (BL = 011)
- Programmed burst length (during writes) (WB = 0)
- 3 Clock Latency (CAS Latency= 011)

Once the mode register value has been determined, it must be converted to an address. The mode register is written using the address bus, and the memory data sheet specifies the SDRAM address bits where the data is to be placed. The enhanced SDRAM controller drives the LSB address bits to the pins, so the memory density and bus width don't need to be taken into account during the conversion. Table 24-24 provides an example conversion using the same system characteristics used in the previous example.

Table 24-24. Example Address Calculation for Mode Register

Mode Register	0	0	WB	0	0	CAS LATENCY			BT	BL		
Program Value	0	0	0	0	0	0	1	1	0	0	1	1
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
SDRAM Pin	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
ARM Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

24.5.4.3 SDRAM Memory Configuration Examples

15 different SDRAM (SDR and LPDDR) configurations are demonstrated. These examples are 64 Mbyte, 128 Mbyte, and 256 Mbyte SDRAM memories in single x16 configurations.

All single-configuration 16-bit examples are shown connected to the lower half of the data bus. Alternatively, the memory can be connected to the upper half of the data bus by swapping the data connections to D [31:16] and the data qualifier mask connections to DQM3 and DQM2. In this case, it is necessary to program the DSIZ field in the Control Register to a value of 0 (configurations shown require a value of 1).

24.5.4.3.1 Single 64-Mbit (4 Mbits x 16) SDRAM Configuration

Table 24-25. Single 4-Mbit x16 Control Register Value

Control Field	Value
Density	64Mbit
Page Size	512 Byte
ROW	12 bit
COL	8 bit
DSIZ	16 (D [15:0]) bit
SREFR	2 refreshes

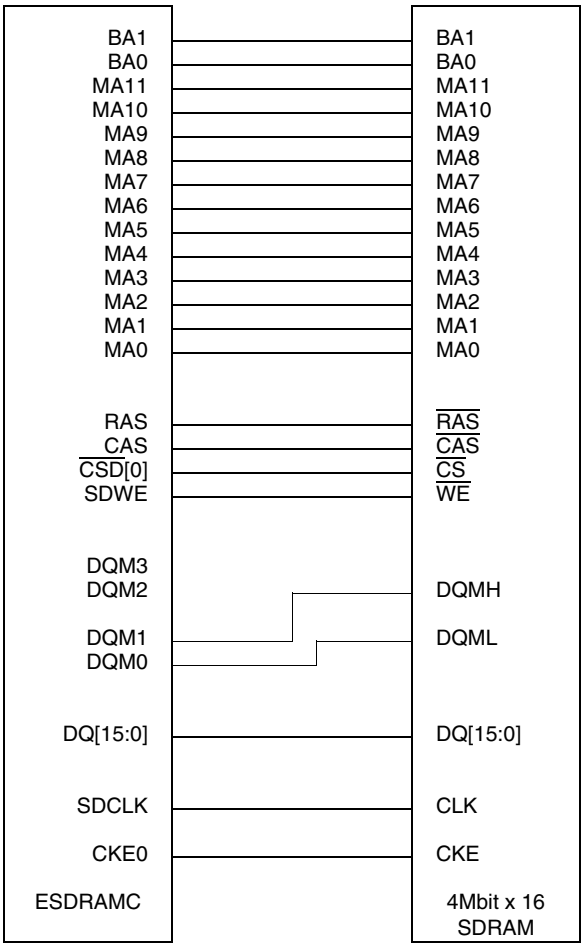


Figure 24-79. Single 64-Mbit (4 Mbits x 16) SDRAM Connection Diagram

24.5.4.3.2 Single 128-Mbit (8 Mbits x 16) SDRAM Configuration

Table 24-26. Single 8-Mbit x 16 Control Register Value

Control Field	Value
Density	128 Mbit
Page Size	1024Byte
ROW	12bit
COL	9bit
DSIZ	16 (D [15:0])bit
SREFR	4 refreshes

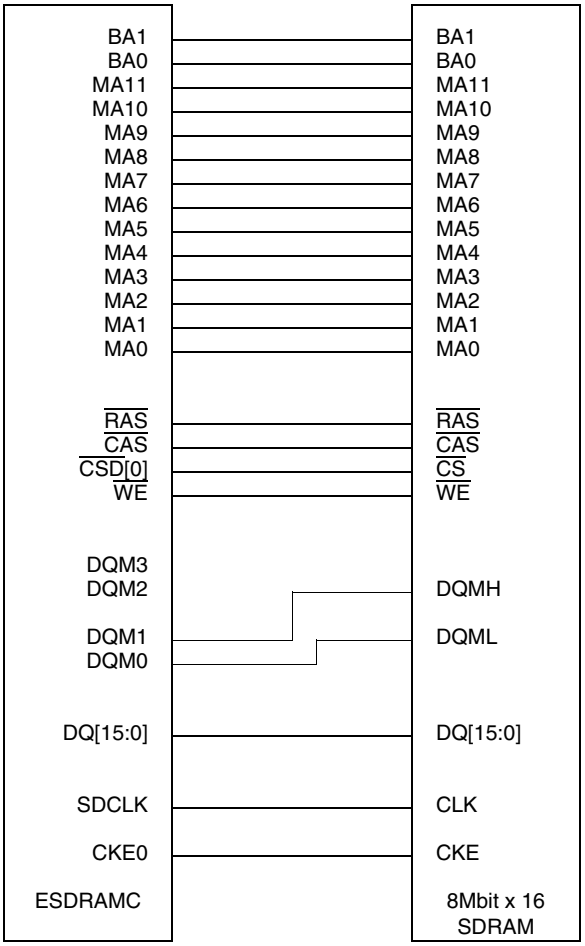


Figure 24-80. Single 128-Mbit (8-Mbits x 16) SDRAM Connection Diagram

24.5.4.3.3 Single 256-Mbit (16 Mbits x 16) SDRAM Configuration

Table 24-27. Single 16-Mbit x 16 Control Register Value

Control Field	Value
Density	256 Mbit
Page Size	1024Byte
ROW	13bit
COL	9bit
DSIZ	16 (D [15:0])bit
SREFR	4 refreshes

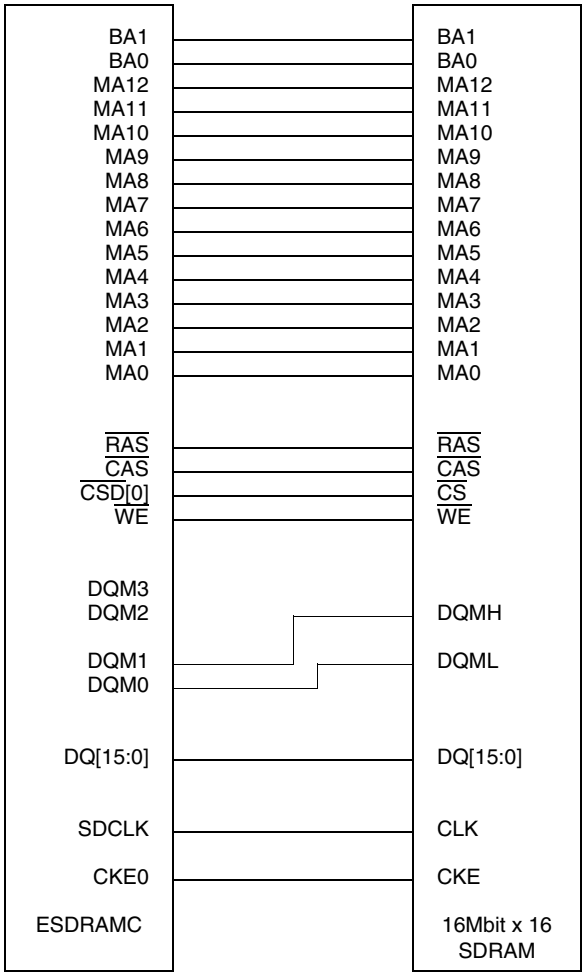


Figure 24-81. Single 256-Mbit (16 Mbits x 16) Connection Diagram

24.5.4.3.4 Single 512-Mbit (32 Mbits x 16) SDRAM Configuration

Table 24-28. Single 32-Mbit x 16 Control Register Value

Control Field	Value
Density	512 Mbits
Page Size	2048 bytes
ROW	13 bits
COL	10 bits
DSIZ	16 bits (D[15:0])
SREFR	4 refreshes

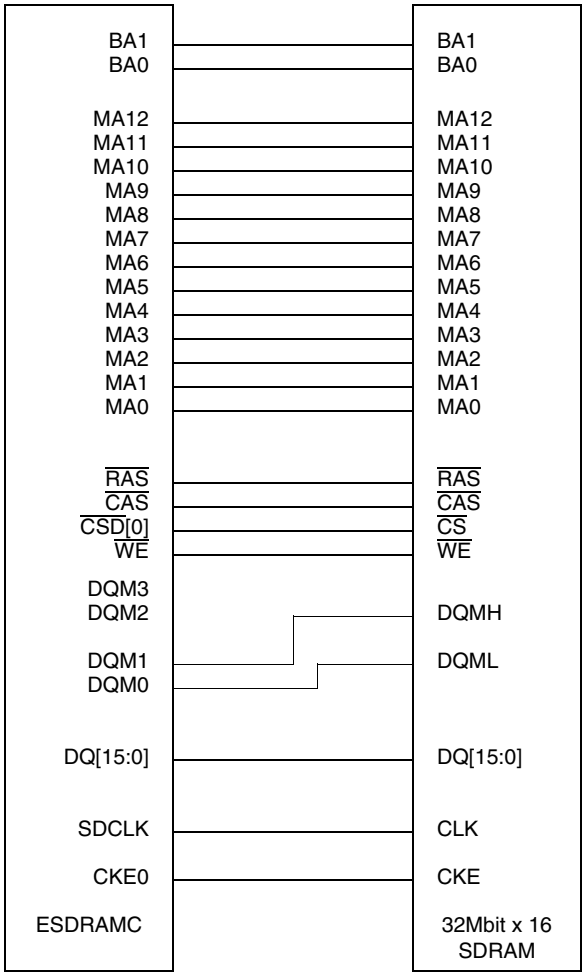


Figure 24-82. Single 512-Mbit (32 Mbits x 16) SDRAM Connection Diagram

24.5.4.3.5 Single 1-Gbit (64 Mbits x 16) SDRAM Configuration

Table 24-29. Single 64-Mbit x 16 Control Register Value

Control Field	Value
Density	1 Gbit
Page size	2048 byte
ROW	14 bits
COL	10 bits
DSIZ	16 (D [15:0]) bits
SREFR	8 refreshes

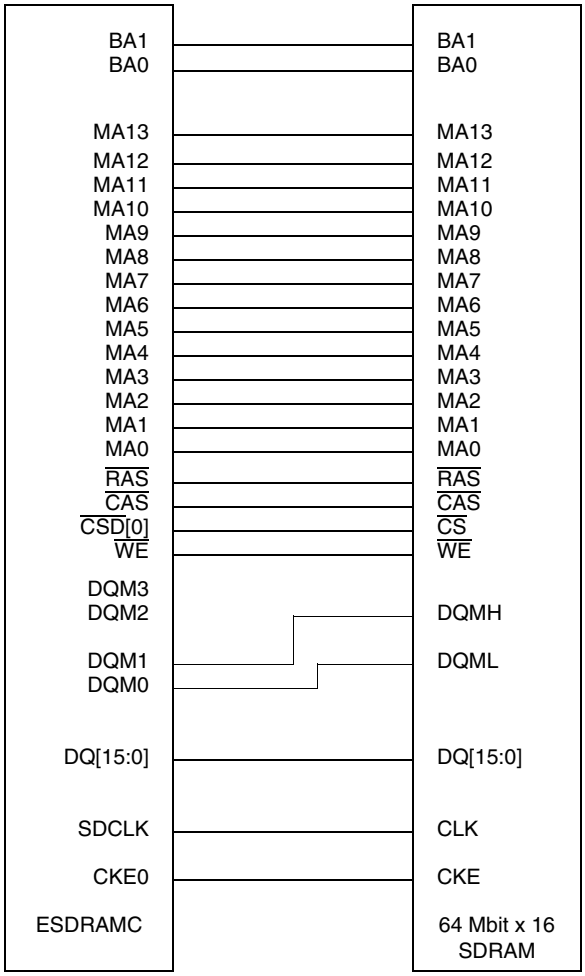


Figure 24-83. Single 1-Gbit (64 Mbits x 16) SDRAM Connection Diagram

24.5.4.3.6 Single 512-Mbit (32 Mbits x 16) Mobile DDR SDRAM Configuration

Table 24-30. Single 32-Mbit x 16 Control Register Value

Control Field	Value
Density	512 Mbit
Page Size	2048Byte
ROW	13 bit
COL	10 bit
DSIZ	16 (D [15:0]) bit
SREFR	4 refreshes

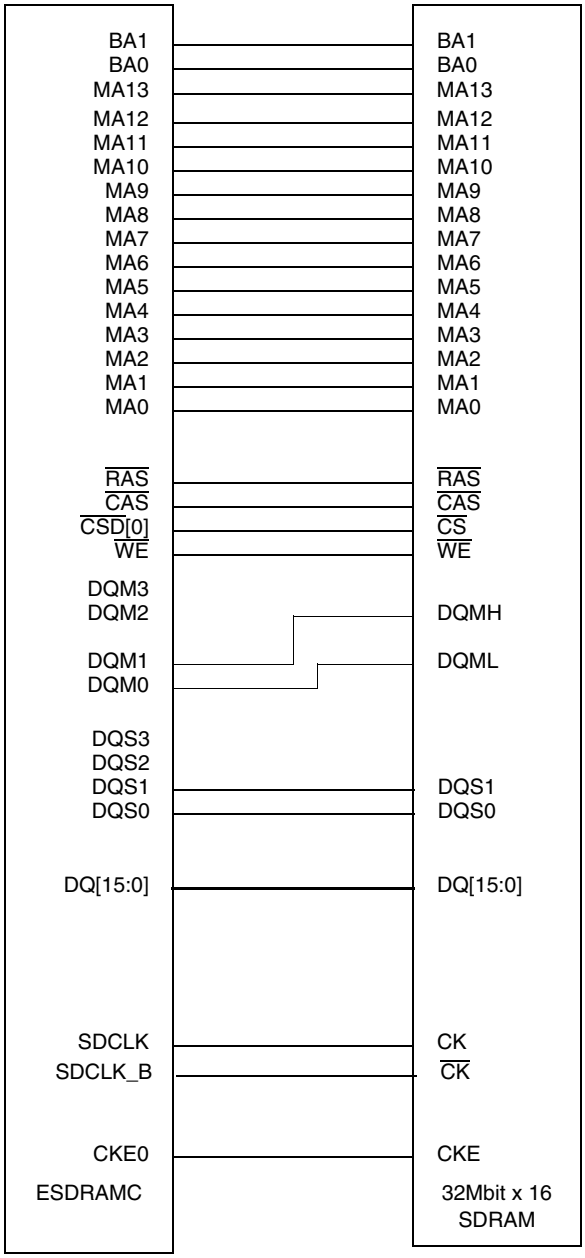


Figure 24-84. Single 512-Mbit (32 Mbits x 16) LPDDR SDRAM Connection Diagram

Chapter 25
Fast Ethernet Controller (FEC)

This chapter provides a feature-set overview, a functional block diagram, and transceiver connection information for the 10- and 100-Mbps media independent interfaces (MII), 10- and 100-Mbps reduced media independent interfaces (RMII), and the 7-wire serial interface. Detailed functional descriptions and application/initialization information are included.

25.1 Overview

The fast Ethernet controller (FEC) is designed to support both 10- and 100-Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports three different standard physical interfaces (MAC-PHY) for connection to an external Ethernet transceiver. The FEC is implemented with a combination of hardware and microcode. Figure 25-1 is a block diagram of the FEC.

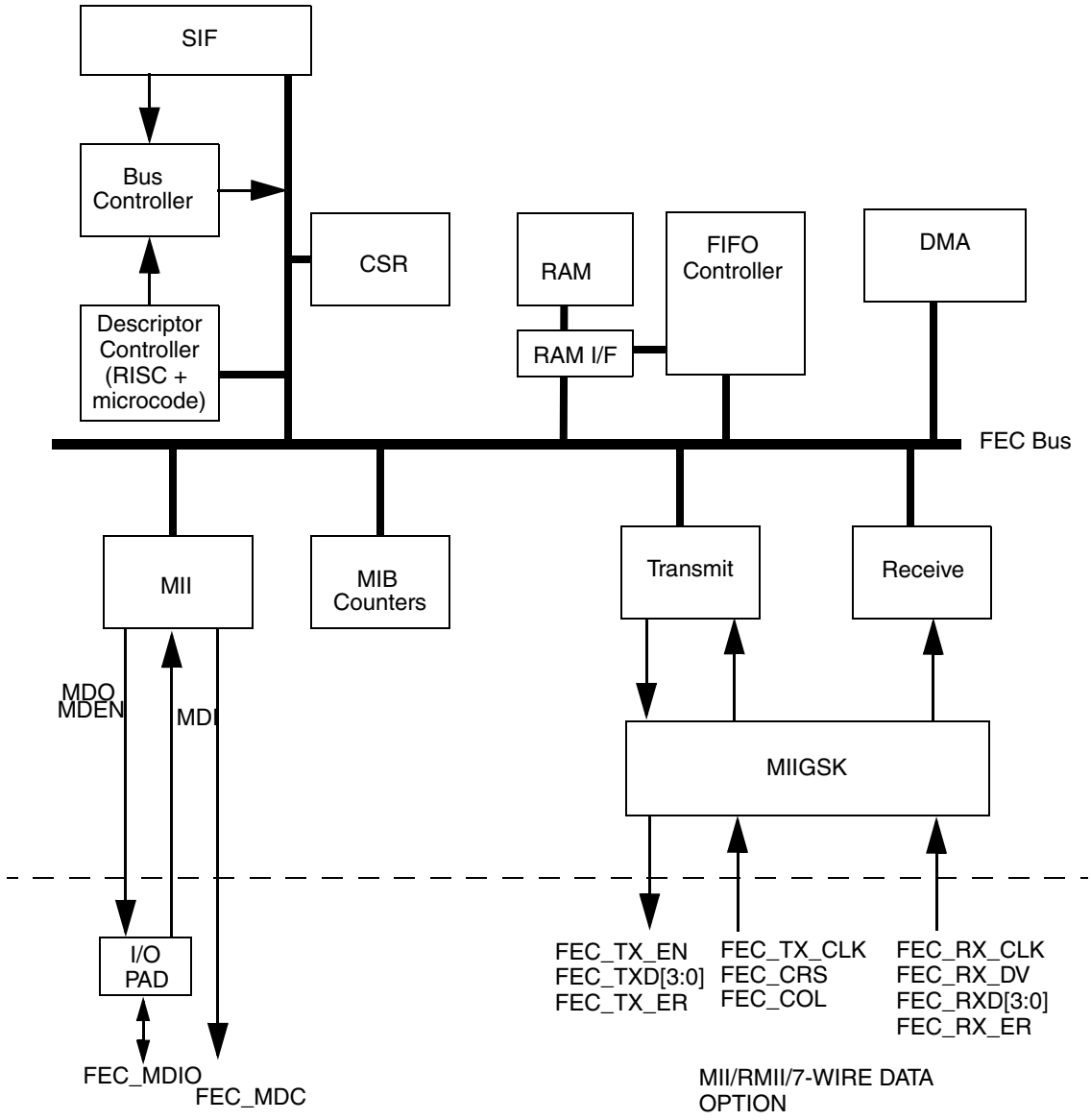


Figure 25-1. FEC Block Diagram

The FEC submodules shown in Figure 25-1 are described as follows:

- The descriptor controller is a RISC-based controller that provides the following functions in the FEC:
  - Initialization (those internal registers not initialized by the user or hardware)
  - High-level control of the DMA channels (initiating DMA transfers)
  - Interpreting buffer descriptors
  - Address recognition for receive frames
  - Random number generation for transmit collision backoff timer

NOTE

This DMA engine is for the transfer of FEC data only and is not related to the system DMA controller.

- The RAM is the focal point of all data flow in the FEC and is divided into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to and from the DMA block, from and to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO to the transmit block, and receive data flows from the receive block to the receive FIFO.
- The user controls the FEC by writing, through the slave interface (SIF) submodule, to control registers located in each block. The control and status register (CSR) block provides global control (for example, Ethernet reset and enable) and interrupt handling registers.
- The MII block provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of the management data clock and management data input/output lines of the MII interface (FEC_MDC and FEC_MDIO, respectively).
- The DMA block provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to run independently.
- The transmit and receive blocks provide the Ethernet MAC functionality (with some assistance from microcode).
- The message information block (MIB) maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC but provides valuable counters for network management. The counters supported are the RMON (RFC 1757) Ethernet statistics group and some of the IEEE 802.3 counters. See Section 25.3.3, “Message Information Block (MIB) Counters Memory Map,” for more information.
- The MII gasket block (MIIGSK) provides an interface between the MII (Media Independent Interface specified by IEEE 802.3) I/F and RMII (low pin count Reduced Media Independent Interface as specified by the RMII Consortium™ standard) I/F.

25.1.1 Features

The FEC incorporates the following features:

- Support for three different Ethernet physical interfaces:
  - 10-Mbps and 100-Mbps IEEE 802.3 MII
  - 10-Mbps and 100-Mbps RMII
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full-duplex flow control
- Programmable max frame length supports IEEE Std 802.1™ VLAN tags and priority
- Support for full-duplex operation (200Mbps throughput) with a minimum system clock rate of 50 MHz
- Support for half-duplex operation (100Mbps throughput) with a minimum system clock rate of 25 MHz

- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address can be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

25.2 Modes of Operation

The primary operational modes are described in this section.

25.2.1 Full- and Half-Duplex Operation

Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters. Selection of the duplex mode is controlled by the TCR[FDEN] bit. When configured for full-duplex mode, flow control can be enabled, which is effected by the TCR[RFC_PAUSE], TCR[TFC_PAUSE], and RCR[FCE] bits. See Section 25.4.4, “Full-Duplex Flow Control,” for more details.

25.2.2 Interface Options

The following interface options are supported. A detailed discussion of the interface configurations is provided in Section 25.4.1, “Network Interface Options.”

25.2.2.1 10-Mbps and 100-Mbps Media Independent Interface (MII)

The MII is defined by the IEEE 802.3 standard for 10/100-Mbps operation. The MAC-PHY interface can be configured to operate in MII mode by asserting RCR[MII_MODE].

The speed of operation is determined by the FEC_TX_CLK and FEC_RX_CLK signals which are driven by the external transceiver. The transceiver either autonegotiates the speed or control by software using the serial management interface (FEC_MDC/FEC_MDIO) signals to the transceiver. See the descriptions in Section 25.3.5.6, “MII Management Frame Register (MMFR),” and Section 25.3.5.7, “MII Speed Control Register (MSCR),” respectively, as well as MII documentation for a description of how to read and write registers in the transceiver using this interface.

25.2.2.2 10 Mbps and 100 Mbps RMII Interface

RMII is a reduced MII interface specified by the RMII Consortium™ standard. The purpose of this interface is to provide a low cost alternative to the MII, with 8 pin interface instead of 16 (Not including MDC,MDIO pins).

The FEC uses a gasket (MIIGSK) to support RMII. It will operate in RMII mode by setting MIIGSK_CFGR[I/F_MODE] = 01, while setting MIIGSK_CFGR[I/F_MODE] = 00 will disable RMII mode. The speed selection in RMII Mode is supported by configuring the MIIGSK_CFGR[FRCONT]. The serial management interface is still available in RMII mode.

25.2.2.3 10-Mbps 7-Wire Interface Operation

The FEC supports a 7-wire interface as used by many 10 Mbps ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, the MII mode is disabled and the 10 Mbps, 7-wire mode is enabled.

25.2.3 Address Recognition Options

The address options supported are promiscuous, broadcast reject, individual address (hash or exact match), and multicast hash match. Address recognition options are discussed in detail in Section 25.4.3.2, “Ethernet Address Recognition.”

25.2.4 Internal Loopback

Two levels of internal loopback modes are supported. The first level, which will loop data back before they enter MIIGSK, is selected using RCR[LOOP]. The second level, which will loop data back inside MIIGSK RMII domain, is selected by setting MIIGSK_CFGR[LBMODE] and MIIGSK_CFGR[I/F_MODE] = 01. Loopback mode is discussed in detail in Section 25.4.5, “Internal and External Loopback.”

25.3 Memory Map and Register Definition

This section includes the FEC memory map and detailed descriptions of all the registers, followed by a description of the buffers.

The FEC is programmed by a combination of control and status registers (CSRs) and buffer descriptors. The CSRs are used for mode control and to extract global status information. The descriptors are used to pass data buffers and related buffer information between the hardware and software.

25.3.1 Top Level Module Memory Map

The FEC implementation requires a 1 Kbyte memory space. This space is divided into 2 sections of 512 bytes each. The first is used for control/status registers, and the second contains event/statistics counters held in the MIB block.

Table 25-1 defines the top level memory map.For the base address of a particular module instantiation, see the system memory map.

Table 25-1. Module Memory Map

Base Address Offset	Function
0x0000–01FF	Control/Status Registers

Table 25-1. Module Memory Map (continued)

Base Address Offset	Function
0x0200–02FF	MIB Block Counters
0x0300–03FF	Extension Registers (MIIGSK)

25.3.2 Detailed Memory Map (Control/Status Registers)

Table 25-2 shows the FEC register memory map. For the base address of a particular module instantiation, see the system memory map.

25.3.3 Message Information Block (MIB) Counters Memory Map

Table 25-2. FEC Registers Memory Map

Base Address Offset (Register Abbreviation	Register	Access	Reset Value	Section/Page
General Registers				
0x0004 (EIR)	Ethernet interrupt event register	R/W	0x0000_0000	25.3.5.1/25-10
0x0008 (EIMR)	Ethernet interrupt mask register	R/W	0x0000_0000	25.3.5.2/25-12
0x0010 (RDAR)	Receive descriptor active register	R/W	0x0000_0000	25.3.5.3/25-12
0x0014 (TDAR)	Transmit descriptor active register	R/W	0x0000_0000	25.3.5.4/25-13
0x0024 (ECR)	Ethernet control register	R/W	0xF000_0000	25.3.5.5/25-14
0x0040 (MMFR)	MII management frame register	R/W	Unaffected by reset	25.3.5.6/25-15
0x0044 (MSCR)	MII speed control register	R/W	0x0000_0000	25.3.5.7/25-16
0x0064 (MIBC)	MIB control register	R/W	0xC000_0000	25.3.5.8/25-18
0x0084 (RCR)	Receive control register	R/W	0x05EE_0001	25.3.5.9/25-18
0x00C4 (TCR)	Transmit control register	R/W	0x0000_0000	25.3.5.10/25-20
0x00E4 (PALR)	Physical address low register	R/W	Unaffected by reset	25.3.5.11/25-21
0x00E8 (PAUR)	Physical address upper register	Mixed	0xuuuu_8808	25.3.5.12/25-22
0x00EC (OPDR)	Opcode and pause duration register	Mixed	0x0001_uuuu	25.3.5.13/25-22
0x0118 (IAUR)	Descriptor individual address upper register	R/W	Unaffected by reset	25.3.5.14/25-23
0x011C (IALR)	Descriptor individual address lower register	R/W	Unaffected by reset	25.3.5.15/25-24
0x0120 (GAUR)	Descriptor group address upper register	R/W	Unaffected by reset	25.3.5.16/25-24
0x0124 (GALR)	Descriptor group address lower register	R/W	Unaffected by reset	25.3.5.17/25-25
0x0144 (TFWR)	Transmit FIFO watermark register	R/W	0x0000_0000	25.3.5.18/25-26
0x014C (FRBR)	FIFO receive bound register	R/O	0x0000_0600	25.3.5.19/25-26
0x0150 (FRSR)	FIFO receive FIFO start registers	R/W	0x0000_0500	25.3.5.20/25-27
0x0180 (ERDSR)	Receive buffer descriptor ring start register	R/W	Unaffected by reset	25.3.5.21/25-28
0x0184 (ETDSR)	Transmit buffer descriptor ring start register	R/W	Unaffected by reset	25.3.5.22/25-29
0x0188 (EMRBR)	Maximum receive buffer size register	R/W	Unaffected by reset	25.3.5.23/25-29

Table 25-3 shows the MIB counters memory map, which defines the locations in the MIB RAM space where hardware maintained counters reside. It is the responsibility of software to poll the counters often enough to ensure that rollover is detected. For example, on a 100 Mbps channel an octets counter could roll over every 5.7 minutes.

These counters fall in the 0x0200-0x02FF address offset range, and are divided into RMON counters and IEEE counters, as follows:

- RMON counters are included which cover the Ethernet statistics counters defined in RFC 1757. In addition to the counters defined in the Ethernet statistics group, a counter is included to count truncated frames as the FEC only supports frame lengths up to 2047 bytes. The RMON counters are implemented independently for transmit and receive to insure accurate network statistics when operating in full-duplex mode.
- IEEE counters are included which support the mandatory and recommended counter packages defined in section 5 of ANSI/IEEE 802.3 (1998 edition). The IEEE basic package objects are supported by the FEC but do not require counters in the MIB block. In addition, some of the recommended package objects which are supported do not require MIB counters. Counters for transmit and receive full-duplex flow control frames are included as well.

Table 25-3. MIB Counters Memory Map

Base Address Offset	Mnemonic	Description
0x0200	RMON_T_DROP	Count of frames not counted correctly
0x0204	RMON_T_PACKETS	RMON Tx packet count
0x0208	RMON_T_BC_PKT	RMON Tx broadcast packets
0x020C	RMON_T_MC_PKT	RMON Tx multicast packets
0x0210	RMON_T_CRC_ALIGN	RMON Tx packets w CRC/Align error
0x0214	RMON_T_UNDERSIZE	RMON Tx packets < 64 bytes, good crc
0x0218	RMON_T_OVERSIZE	RMON Tx packets > MAX_FL bytes, good crc
0x021C	RMON_T_FRAG	RMON Tx packets < 64 bytes, bad crc
0x0220	RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad crc
0x0224	RMON_T_COL	RMON Tx collision count
0x0228	RMON_T_P64	RMON Tx 64 byte packets
0x022C	RMON_T_P65TO127	RMON Tx 65 to 127 byte packets
0x0230	RMON_T_P128TO255	RMON Tx 128 to 255 byte packets
0x0234	RMON_T_P256TO511	RMON Tx 256 to 511 byte packets
0x0238	RMON_T_P512TO1023	RMON Tx 512 to 1023 byte packets
0x023C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 byte packets
0x0240	RMON_T_P_GTE2048	RMON Tx packets w > 2048 bytes
0x0244	RMON_T_OCTETS	RMON Tx octets
0x0248	IEEE_T_DROP	Count of frames not counted correctly
0x024C	IEEE_T_FRAME_OK	Frames transmitted OK
0x0250	IEEE_T_1COL	Frames transmitted with single collision
0x0254	IEEE_T_MCOL	Frames transmitted with multiple collisions

Table 25-3. MIB Counters Memory Map (continued)

Base Address Offset	Mnemonic	Description
0x0258	IEEE_T_DEF	Frames transmitted after deferral delay
0x025C	IEEE_T_LCOL	Frames transmitted with late collision
0x0260	IEEE_T_EXCOL	Frames transmitted with excessive collisions
0x0264	IEEE_T_MACERR	Frames transmitted with Tx FIFO underrun
0x0268	IEEE_T_CSERR	Frames transmitted with carrier sense error
0x026C	IEEE_T_SQE	Frames transmitted with SQE error
0x0270	IEEE_T_FDXFC	Flow control pause frames transmitted
0x0274	IEEE_T_OCTETS_OK	Octet count for frames transmitted w/o error
0x0284	RMON_R_PACKETS	RMON Rx packet count
0x0288	RMON_R_BC_PKT	RMON Rx broadcast packets
0x028C	RMON_R_MC_PKT	RMON Rx multicast packets
0x0290	RMON_R_CRC_ALIGN	RMON Rx packets w CRC/Align error
0x0294	RMON_R_UNDERSIZE	RMON Rx packets < 64 bytes, good crc
0x0298	RMON_R_OVERSIZE	RMON Rx packets > MAX_FL bytes, good crc
0x029C	RMON_R_FRAG	RMON Rx packets < 64 bytes, bad crc
0x02A0	RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad crc
0x02A4	RMON_R_RESVD_0	—
0x02A8	RMON_R_P64	RMON Rx 64 byte packets
0x02AC	RMON_R_P65TO127	RMON Rx 65 to 127 byte packets
0x02B0	RMON_R_P128TO255	RMON Rx 128 to 255 byte packets
0x02B4	RMON_R_P256TO511	RMON Rx 256 to 511 byte packets
0x02B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 byte packets
0x02BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 byte packets
0x02C0	RMON_R_P_GTE2048	RMON Rx packets w > 2048 bytes
0x02C4	RMON_R_OCTETS	RMON Rx octets
0x02C8	IEEE_R_DROP	Count of frames not counted correctly
0x02CC	IEEE_R_FRAME_OK	Frames received OK
0x02D0	IEEE_R_CRC	Frames received with CRC error
0x02D4	IEEE_R_ALIGN	Frames received with alignment error
0x02D8	IEEE_R_MACERR	Receive FIFO overflow count
0x02DC	IEEE_R_FDXFC	Flow control pause frames received
0x02E0	IEEE_R_OCTETS_OK	Octet count for frames received w/o error

25.3.4 MIIGSK Registers Memory Map

Table 25-4 lists the MIIGSK registers.

Table 25-4. MIIGSK Registers Memory Map

Offset	Mnemonic	Description
0x0300	MIIGSK_CFGR	MIIGSK Configuration Register
0x0308	MIIGSK_ENR	MIIGSK Enable Register

25.3.5 Register Descriptions

This section provides detailed descriptions of the FEC's registers.

25.3.5.1 Ethernet Interrupt Event Register (EIR)

The EIR bit assignments are shown in Figure 25-2 and described in Table 25-5. When an event occurs that sets a bit in the EIR, an interrupt is generated if the corresponding bit in the interrupt mask register (EIMR) is also set. The bit in the EIR is cleared if a one is written to that bit position; writing zero has no effect. This register is cleared upon hardware reset.

Interrupts can be divided into three classes as follows:

- Interrupts that occur in normal operation: these include GRA, TXF, TXB, RXF, RXB, and MII.
- Interrupts that result from errors or problems detected in the network or transceiver: these include HBERR, BABR, BABT, LC, and RL.
- Interrupts that result from internal errors are EBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters. The correspondence between interrupts and counters is shown in Table 25-6. Software can choose to mask off the interrupts since these errors is visible to network management using the MIB counters.

Offset 0x0004 (EIR)
0x0008 (EIMR)

Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-2. Ethernet Interrupt Event Register (EIR)

Table 25-5. EIR Field Descriptions

Bits	Name	Description
31	HBERR	Heartbeat error. This interrupt indicates that HBC is set in the TCR register and that the COL input was not asserted within the Heartbeat window following a transmission.
30	BABR	Babbling receive error. This bit indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
29	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded RCR[MAX_FL] bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation does not occur.
28	GRA	Graceful stop complete. This interrupt is asserted for one of three reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. 1) A graceful stop, which was initiated by the setting of the TCR[GTS] bit is now complete. 2) A graceful stop, which was initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. 3) A graceful stop, which was initiated by the reception of a valid full-duplex flow control “pause” frame is now complete. See the “Full-Duplex Flow Control” section of the Functional Description chapter.
27	TXF	Transmit frame interrupt. This bit indicates that a frame has been transmitted and that the last corresponding buffer descriptor has been updated.
26	TXB	Transmit buffer interrupt. This bit indicates that a transmit buffer descriptor has been updated.
25	RXF	Receive frame interrupt. This bit indicates that a frame has been received and that the last corresponding buffer descriptor has been updated.
24	RXB	Receive buffer interrupt. This bit indicates that a receive buffer descriptor has been updated that was not the last in the frame.
23	MII	MII interrupt. This bit indicates that the MII has completed the data transfer requested.
22	EBERR	Ethernet bus error. This bit indicates that a system bus error occurred when a DMA transaction was underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs software needs to insure that the FIFO controller and DMA are also soft reset.
21	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded.
20	RL	Collision retry limit. This bit indicates that a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. Can only occur in half-duplex mode.
19	UN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18–0	—	Reserved

Table 25-6. Error Interrupts and Block Counters

Interrupt	Counter(s)
HBERR	IEEE_T_SQE
BABR	RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
BABT	RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
LATE_COL	IEEE_T_LCOL

Table 25-6. Error Interrupts and Block Counters (continued)

Interrupt	Counter(s)
COL_RETRY_LIM	IEEE_T_EXCOL
XFIFO_UN	IEEE_T_MACERR

25.3.5.2 Ethernet Interrupt Mask Register (EIMR)

The EIMR controls which of the interrupt events flagged in the EIR are allowed to generate actual interrupts. If the corresponding bits in both the EIR and EIMR are set, the interrupt is signaled to the CPU. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit. This register is cleared upon a hardware reset.

The EIMR bit assignments are the same as for the EIR: they are shown in Figure 25-2 and described in Table 25-7.

Table 25-7. EIMR Field Descriptions

Bits	Name	Description
31–19	See Table 25-5.	Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every system clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
18–0	—	Reserved.

25.3.5.3 Receive Descriptor Active Register (RDAR)

RDAR is a user-writeable command register which indicates that the receive descriptor ring has been updated, and that empty receive buffers have been produced by the driver with the empty bit set.

The RDAR[R_DES_ACTIVE] bit is set whenever the register is written, independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (provided ECR[ETHER_EN] is also set to 1). After the FEC polls a receive descriptor whose empty bit is not set, then the FEC clears the RDAR[R_DES_ACTIVE] bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors have been placed into the receive descriptor ring.

The RDAR is cleared at reset, and when ECR[ETHER_EN] is cleared.

The RDAR bit assignments are shown in Figure 25-3 and described in Table 25-8.

Offset 0x0010 (RDAR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	R_DES_ACTIVE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-3. Receive Descriptor Active Register (RDAR)

Table 25-8. RDAR Field Descriptions

Bits	Name	Description
31–25	—	Reserved.
24	R_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a receive descriptor whose empty bit is not set. Also cleared when ECR[ETHER_EN] is cleared.
23–0	—	Reserved.

25.3.5.4 Transmit Descriptor Active Register (TDAR)

The TDAR is a command register, written to by the user, to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

Whenever the register is written the TDAR[X_DES_ACTIVE] bit is set to 1, independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and process transmit frames (provided ECR[ETHER_EN] is also set to 1). After the FEC polls a transmit descriptor whose ready bit is not set, then the FEC clears the TDAR[X_DES_ACTIVE] bit and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The TDAR is cleared at reset, when ECR[ETHER_EN] is cleared, or when ECR[RESET] is set to 1.

The TDAR bit assignments are shown in Figure 25-4 and described in Table 25-9.

Offset 0x0014 (TDAR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	X_DES_ACTIVE	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-4. Transmit Descriptor Active Register (TDAR)

Table 25-9. TDAR Field Descriptions

Bits	Name	Description
31–25	—	Reserved.
24	X_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device when the FEC polls a transmit descriptor whose ready bit is not set. Also cleared when ECR[ETHER_EN] is cleared.
23–0	—	Reserved.

25.3.5.5 Ethernet Control Register (ECR)

The ECR is used to enable/disable the FEC. ECR is a read/write user register, though both fields in this register can also be altered by hardware.

ECR bit assignments are shown in Figure 25-5 and described in Table 25-10.

Offset 0x0024 (ECR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-5. Ethernet Control Register (ECR)

Table 25-10. ECR Field Descriptions

Bits	Name	Description
31-2	—	Reserved.
1	ETHER_EN	When this bit is set, the FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception is immediately stopped and transmission is stopped after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. The ETHER_EN bit is altered by hardware under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared • an error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN isis cleared
0	RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ETHER_EN is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 system clock cycles after RESET is written with a 1.

25.3.5.6 MII Management Frame Register (MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to MMFR causes a management frame to be generated unless the MII-SPEED field of the MSCR has been set to 0, in which case MSCR is set to a non-zero value and an MII frame is generated with the data previously written to MMFR. This allows MMFR and MSCR to be programmed in either order if the MII-SPEED field of MSCR is zero (for further details see Section 25.3.5.7, “MII Speed Control Register (MSCR)”).

The MMFR does not reset to a definite value. MMFR bit assignments are shown in Figure 25-6 and described in Table 25-11.

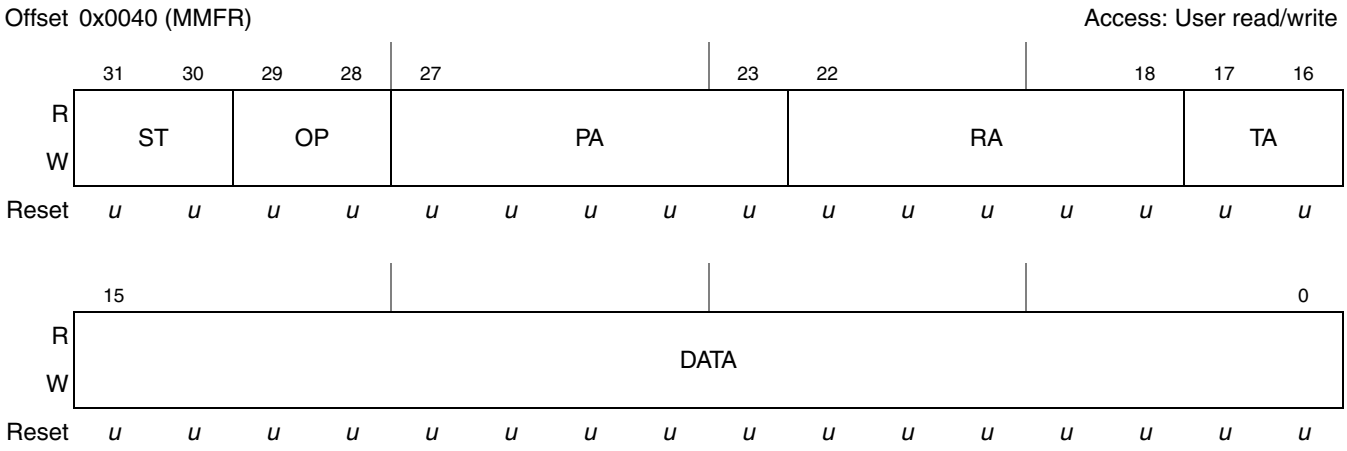


Figure 25-6. MII Management Frame Register (MMFR)

Table 25-11. MMFR Field Descriptions

Bits	Name	Description
31–30	ST	Start of frame delimiter. These bits must be programmed to 01 for a valid MII management frame.
29–28	OP	Operation code. This field must be programmed to 10 (read) or 01 (write) to generate a valid MII management frame. A value of 11 produces “read” frame operation while a value of 00 produces “write” frame operation, but these frames are not MII-compatible.
27–23	PA	PHY address. This field specifies one of up to 32 attached PHY devices.
22–18	RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
17–16	TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
15–0	DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

To perform a read or write operation on the MII Management Interface, the MMFR must be written to by the user. To generate a valid read or write management frame, the ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame isis generated but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compatible MII management interface write frame (write to a PHY register), the user must write {01 01 PHYAD REGAD 10 DATA} to the bit fields of the MMFR, as shown in Table 25-11. Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR is altered as the contents are serially shifted and is unpredictable if read by the user. After the write management frame operation has completed, the MII interrupt is generated. At this time the contents of the MMFR matches the original value written.

To generate an MII management interface read frame (read a PHY register) the user must write {01 10 PHYAD REGAD 10 XXXX} to the bit fields of the MMFR shown in Table 25-11 (the contents of the 4-bit DATA field are arbitrary). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time the contents of the MMFR is altered as the contents are serially shifted, and is unpredictable if read by the user. After the read management frame operation has completed, the MII interrupt is generated. At this time the contents of the MMFR matches the original value written except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the MMFR is written to while frame generation is in progress, the frame contents is altered. Software uses the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

25.3.5.7 MII Speed Control Register (MSCR)

MSCR bit assignments are shown in Figure 25-7 and described in Table 25-12. The MSCR provides control of the frequency of the MII clock (FEC_MDC signal), and allows a preamble drop on the MII management frame.

Offset 0x0044 (MSCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DIS_PREAMBLE	MII_SPEED						0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-7. MII Speed Control Register (MSCR)

Table 25-12. MSCR Field Descriptions

Bits	Name	Description
31–8	—	Reserved.
7	DIS_PREAMBLE	Asserting this bit causes preamble (0xFFFF_FFFF) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
6–1	MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (FEC_MDC) relative to the system clock. A value of 0 in this field “turns off” the FEC_MDC and leave it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED*2) of the system clock frequency.
0	—	Reserved.

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compatible with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value in order to generate a read or write management frame. After the management frame is complete the MSCR can optionally be set to zero to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED is changed during operation (change takes effect following either a rising or falling edge of FEC_MDC).

The FEC_MDC frequency depends on both the system clock frequency and the MII_SPEED register. If the system clock is 25 MHz, programming the MII_SPEED register to 0x0000_0005 results in an FEC_MDC frequency of 25 MHz * 1/10 = 2.5 MHz. A table showing optimum values for MII_SPEED for different system clock frequencies is provided below.

Table 25-13. Programming Examples for MSCR

System Clock Frequency	MII_SPEED (Field in Reg)	FEC_MDC Frequency
25 MHz	0x5	2.5 MHz
33 MHz	0x7	2.36 MHz
40 MHz	0x8	2.5 MHz

Table 25-13. Programming Examples for MSCR (continued)

System Clock Frequency	MII_SPEED (Field in Reg)	FEC_MDC Frequency
50 MHz	0xA	2.5 MHz
66 MHz	0xD	2.54 MHz

25.3.5.8 MIB Control Register (MIBC)

The MIB control register is a read/write register used to provide control of and to observe the state of the Message Information Block (MIB). This register is accessed by user software if there is a need to disable the MIB operation. For example, in order to clear all MIB counters in RAM the user disables the MIB, then clears all the MIB RAM locations, then enables the MIB. The MIB_DISABLE bit is reset to 1. See Table 25-3 for the locations of the MIB counters.

MIBC bit assignments are shown in Figure 25-8 and described in Table 25-14.

Offset 0x0064 (MIBC) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MIB_DISABLE	MIB_IDLE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-8. MIB Control Register (MIBC)

Table 25-14. MIBC Field Descriptions

Bits	Name	Description
31	MIB_DISABLE	A read/write control bit. If set, the MIB logic halts and not update any MIB counters.
30	MB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
29–0	—	Reserved.

25.3.5.9 Receive Control Register (RCR)

RCR bit fields are shown in Figure 25-9 and described in Table 25-15. The RCR is programmed by the user, and controls the operational mode of the receive block. It can only be written to when ECR[ETHER_EN] = 0 (that is, during initialization).

Offset 0x0084 (RCR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	MAX_FL										
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 25-9. Receive Control Register (RCR)

Table 25-15. RCR Field Descriptions

Bits	Name	Description
31–27	—	Reserved.
26–16	MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive Frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. The recommended default value to be programmed by the user is 1518 or 1522 (if VLAN Tags are supported).
15–6	—	Reserved.
5	FCE	Flow control enable. When FCE is set to 1, the receiver detects pause frames. Upon pause frame detection, the transmitter stops transmitting data frames for a given duration.
4	BC_REJ	Broadcast frame reject. When BC_REJ is set to 1, frames with DA (destination address) = 0xFF_FF_FF_FF_FF_FF are rejected unless the PROM bit is set to 1. If both BC_REJ and PROM are set to 1, then frames with broadcast DA is accepted and the M (MISS) bit is set in the receive buffer descriptor.
3	PROM	Promiscuous mode. All frames are accepted regardless of address matching.
2	MII_MODE	Media independent interface mode. Selects external interface mode. Setting this bit to one selects MII mode, setting this bit equal to zero selects 7-wire mode (used only for serial 10 Mbps). This bit controls the interface mode for both transmit and receive blocks.
1	DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full-duplex or to monitor transmit activity in half-duplex mode). 1 Disable reception of frames while transmitting (normally used for half-duplex mode).
0	LOOP	Internal loopback.When LOOP is set to 1, transmitted frames are looped back internal to the device and the transmit output signals are not asserted. The system clock is substituted for the FEC_TX_CLK when LOOP is set to 1. DRT must be set to zero when setting LOOP to 1.

25.3.5.10 Transmit Control Register (TCR)

TCR bit fields are shown in Figure 25-9 and described in Table 25-15. This register is read/write, and is written by the user to configure the transmit block. Bits [2:1] must only be modified when ECR[ETHER_EN] = 0 (that is, during initialization). This register is cleared at system reset.

Offset 0x00C4 (TCR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-10. Transmit Control Register (TCR)

Table 25-16. TCR Field Descriptions

Bits	Name	Description
31–5	—	Reserved.
4	RFC_PAUSE	Receive frame control pause. This read-only status bit is set to 1 when a full-duplex flow control pause frame has been received and the transmitter is paused for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete. 0 Transmitter is not paused 1 Transmitter is paused after reception of full-duplex flow control pause frame
3	TFC_PAUSE	Transmit frame control pause. When this bit is set to 1, a pause frame is transmitted according to the following steps: 1)FEC stops transmission of data frames after the current transmission is complete. 2)The GRA interrupt in the EIR register is asserted. 3)With transmission of data frames stopped, the FEC transmits a MAC control pause frame. 4)The FEC clears the TFC_PAUSE bit and resume transmitting data frames. Note that the FEC can still transmit a MAC control pause frame when the transmitter is paused due to user assertion of GTS or reception of a pause frame. 0 No pause frame is transmitted 1 Pause frame is transmitted
2	FDEN	Full duplex enable. When FDEN is set to 1, frames are transmitted independent of carrier sense and collision inputs. This bit must only be modified when ETHER_EN is cleared. 0 Full duplex is not enabled 1 Full duplex is enabled

Table 25-16. TCR Field Descriptions (continued)

Bits	Name	Description
1	HBC	Heartbeat control. When HBC is set to 1, the heartbeat check is performed after end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This bit must only be modified when ETHER_EN is cleared. 0 Heartbeat check is not performed after end of transmission 1 Heartbeat check is performed after end of transmission
0	GTS	Graceful transmit stop. When GTS is set to 1, the FEC stops transmission after any frame that is currently being transmitted is complete and the GRA interrupt in the EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission has completed, a “restart” can be accomplished by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS = 1, transmission stops after the collision. The frame is transmitted again after GTS is cleared. **Note:** There can be old frames in the transmit FIFO that is transmitted when GTS is reasserted. To avoid this, clear ECR[ETHER_EN] following the GRA interrupt. 0 Graceful transmit stop is not enabled 1 Graceful transmit stop is enabled.

25.3.5.11 Physical Address Low Register (PALR)

The PALR is written by the user, and contains the lower 32 bits (bytes 0,1,2,3) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is also used for bytes 0 through 3 of the 6-byte source address field when transmitting pause frames. This register is unaffected by reset and must be initialized by the user.

PALR bit fields are shown in Figure 25-11 and described in Table 25-17.

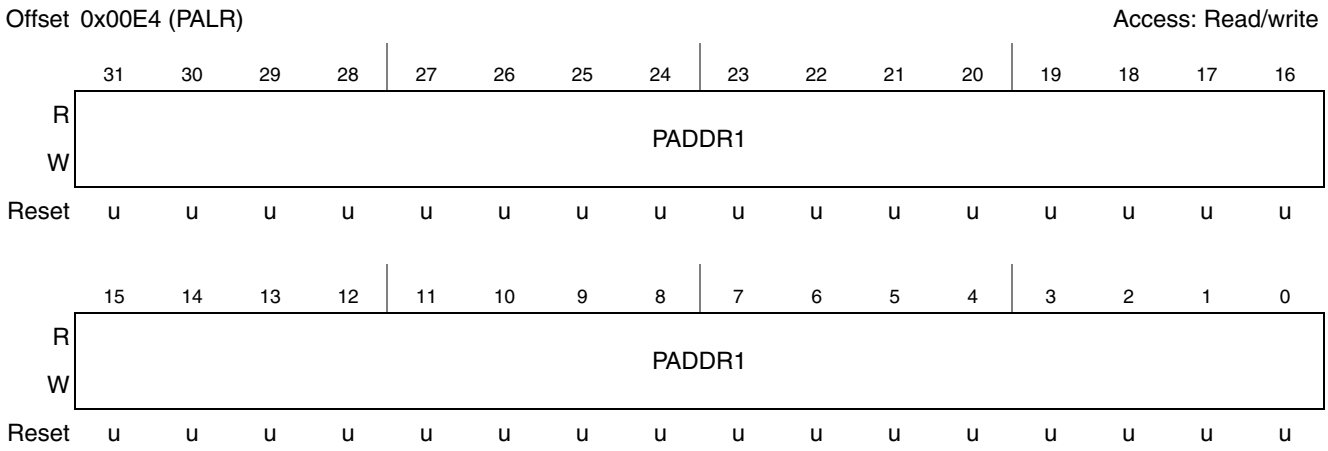


Figure 25-11. Physical Address Low Register (PALR)

Table 25-17. PALR Field Descriptions

Bits	Name	Description
31–0	PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8) and 3 (bits 7:0) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.

25.3.5.12 Physical Address Upper Register (PAUR)

The PAUR is written by the user, and contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte source address field when transmitting pause frames. Bits 15:0 of PAUR contain a constant type field (0x8808) used for transmission of pause frames. This register is unaffected by reset, and bits 31:16 must be initialized by the user.

PAUR bit fields are shown in Figure 25-12 and described in Table 25-18.

Offset 0x00E8 (PAUR) Access: Mixed

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PADDR2															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TYPE															
W																
Reset	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 25-12. Physical Address Upper Register (PAUR)

Table 25-18. PAUR Field Descriptions

BIts	Name	Description
31–16	PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address to be used for exact match, and the source address field in pause frames.
15–0	TYPE	Type field in pause frames. This field has a constant value of 0x8808.

25.3.5.13 Opcode/Pause Duration Register (OPDR)

OPDR contains the 16-bit Opcode, and 16-bit pause duration fields used in transmission of a pause frame. The Opcode field is a constant value, 0x0001. When another node detects a pause frame, that node pauses transmission for the duration specified in the pause duration field. This register is not reset and must be initialized by the user.

OPDR bit fields are shown in Figure 25-13 and described in Table 25-19.

Offset 0x00EC (OPDR) Access: Mixed

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OPCODE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PAUSE_DUR															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-13. Opcode/Pause Duration Register (OPDR)

Table 25-19. OPD Field Descriptions

Bits	Name	Description
31–16	OPCODE	Opcode field used in pause frames. These bits are a constant, 0x0001.
15–0	PAUSE_DUR	Pause duration field used in pause frames.

25.3.5.14 Descriptor Individual Address Upper Register (IAUR)

IAUR bit fields are shown in Figure 25-14 and described in Table 25-20. The IAUR is written by the user, and contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match between the DA field of receive frames and an individual DA. This register is unaffected by reset, and must be initialized by the user.

Offset 0x0118 (IAUR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IADDR1															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IADDR1															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-14. Descriptor Individual Upper Address Register (IAUR)

Table 25-20. IAUR Field Descriptions

Bits	Name	Description
31–0	IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

25.3.5.15 Descriptor Individual Address Lower Register (IALR)

The IALR is written by the user, and contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is unaffected by reset, and must be initialized by the user.

IAUR bit fields are shown in Figure 25-15 and described in Table 25-21.

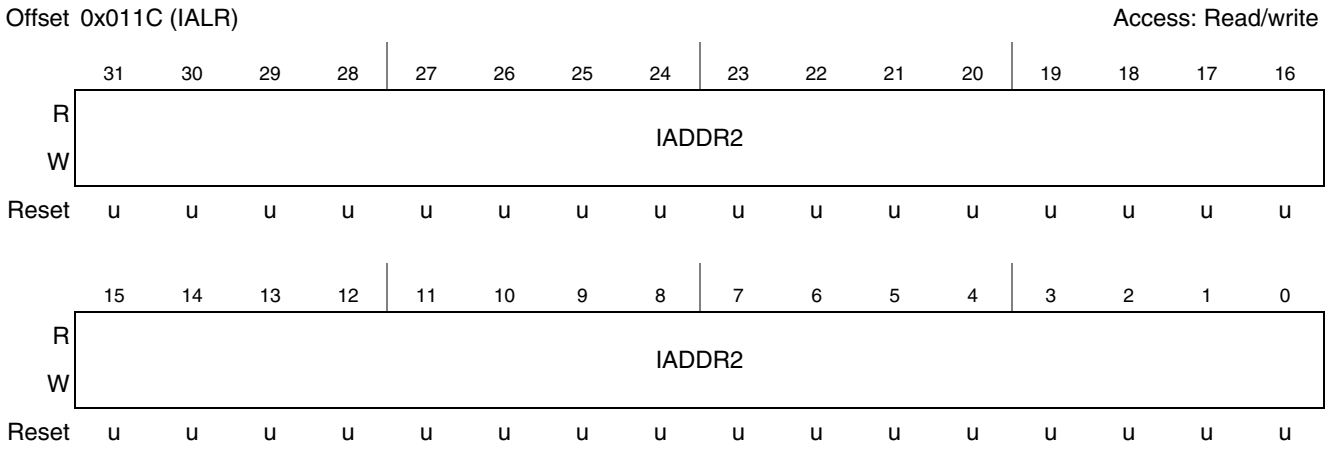


Figure 25-15. Descriptor Individual Address Lower Register (IALR)

Table 25-21. IALR Field Descriptions

Bits	Name	Description
31–0	IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

25.3.5.16 Descriptor Group Address Upper Register (GAUR)

The GAUR is written by the user, and contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

GAUR bit fields are shown in Figure 25-16 and described in Table 25-22.

Offset 0x0120 (GAUR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GADDR1															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR1															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-16. Descriptor Group Upper Address Register (GAUR)

Table 25-22. GAUR Field Descriptions

Bits	Name	Description
31–0	GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

25.3.5.17 Descriptor Group Address Lower Register (GALR)

The GALR is written by the user, and contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized by the user.

GALR bit fields are shown in Figure 25-17 and described in Table 25-23.

Offset 0x0124 (GALR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GADDR2															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR2															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-17. Descriptor Group Lower Address Register (GALR)

Table 25-23. GALR Field Descriptions

Bits	Name	Description
31–0	GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

25.3.5.18 Transmit FIFO Watermark Register (TFWR)

The TFWR is programmed by the user to control the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows the user to minimize transmit latency (TFWR[1:0] = 0*n*) or allow for larger bus access latency (TFWR[1:0] = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. In some use cases the byte counts associated with the TFWR field need to be modified to match system requirements, such as the worst-case bus access latency by the transmit data DMA channel.

TFWR bit fields are shown in Figure 25-18 and described in Table 25-24.

Offset 0x0144 (TFWR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X_WMRK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-18. Transmit FIFO Watermark Register (TFWR)

Table 25-24. TFWR Field Descriptions

Bits	Name	Description
31–2	—	Reserved.
1–0	X_WMRK	Number of bytes written to transmit FIFO before transmission of a frame begins 0x 64 bytes written 10 128 bytes written 11 192 bytes written

25.3.5.19 FIFO Receive Bound Register (FRBR)

The FRBR register can be read to determine the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR to appropriately divide the available FIFO RAM between the transmit and receive data paths.

FRBR bit fields are shown in Figure 25-19 and described in Table 25-25.

Offset 0x014C (FRBR) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	1	R_BOUND								0	0
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 25-19. FIFO Receive Bound Register (FRBR)

Table 25-25. FRBR Field Descriptions

Bits	Name	Description
31–10	—	Reserved, read as 0 (except bit 10, which is read as 1).
9–2	R_BOUND	Read-only. Highest valid FIFO RAM address.
1–0	—	Reserved.

25.3.5.20 FIFO Receive Start Register (FRSR)

FRSR is an 8-bit register programmed by the user to indicate the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

The default value of the receive FIFO starting address is 0x40. This is the value assigned by hardware at reset.

FRSR bit assignments are shown in Figure 25-20 and described in Table 25-26.

Offset 0x0150 (FRSR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	1	R_FSTART								0	0
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

Figure 25-20. FIFO Receive Start Register (FRSR)

Table 25-26. FRSR Field Descriptions

Bits	Name	Description
31–10	—	Reserved, read as 0 (except bit 10, which is read as 1).
9–2	R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs.
1–0	—	Reserved.

25.3.5.21 Receive Buffer Descriptor Ring Start Register (ERDSR)

The register is written by the user, and provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 128-bit aligned (that is, evenly divisible by 16).

ERDSR bit assignments are shown in Figure 25-21 and described in Table 25-27.

This register is unaffected by reset and must be initialized by the user prior to operation.

Offset 0x0180 (ERDSR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	R_DES_START															
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R_DES_START														0	0
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-21. Receive Descriptor Ring Start Register (ERDSR)

Table 25-27. ERDSR Field Descriptions

Bits	Name	Description
31–2	R_DES_START	Pointer to start of receive buffer descriptor queue.
1–0	—	Reserved.

25.3.5.22 Transmit Buffer Descriptor Ring Start Register (ETDSR)

ETDSR bit assignments are shown in Figure 25-21 and described in Table 25-27. The register is written by the user, and provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 128-bit aligned (that is, evenly divisible by 16).

This register is unaffected by reset and must be initialized by the user prior to operation.

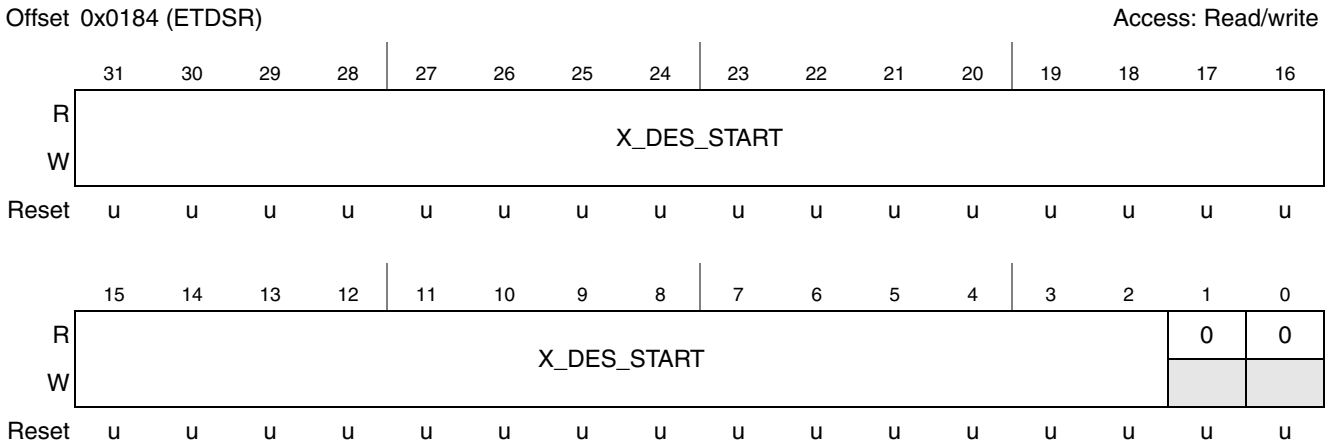


Figure 25-22. Transmit Buffer Descriptor Ring Start Register (ETDSR)

Table 25-28. ETDSR Field Descriptions

Bits	Name	Description
31–2	X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0	—	Reserved.

25.3.5.23 Maximum Receive Buffer Size Register (EMRBR)

The EMRBR, shown in Figure 25-23 and Table 25-29, is a user-programmable register which dictates the maximum size of all receive buffers. Note that because receive frames is truncated at 2k-1(2047) bytes, bits 31-11 are not used. The programmed value accounts for the fact that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. The EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low, and hence only bits 10-4 are actually used. To minimize bus utilization (descriptor fetches) it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is unaffected by reset, and must be initialized by the user.

Offset 0x0188 (EMRBR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	R_BUF_SIZE							0	0	0	0
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 25-23. Receive Buffer Size Register (EMRBR)

Table 25-29. EMRBR Field Descriptions

Bits	Name	Description
31–11	—	Reserved.
10–4	R_BUF_SIZE	Receive buffer size.
3–0	—	Reserved.

25.3.5.24 MIIGSK Configuration Register (MIIGSK_CFGR)

The MIIGSK Configuration Register contains configuration bits for various features and modes of the MIIGSK block.

NOTE

The register should be written only when MIIGSK_ENR[READY] bit is negated.

Offset 0x0300 (MIIGSK_CFGR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	FRCONT	0	LBMODE	0	0	I/F_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-24. MIIGSK Configuration Register (MIIGSK_CFGR)

Table 25-30. MIIGSK_CFGR Field Descriptions

Bits	Description
31-7	Reserved.
6 FRCONT	Frequency Control. This field determines the clock frequency of the clock source to the MIIGSK RMII logic in order to support 10/100 Mbps operations. 0 In RMII Mode the Clock source (REF_CLOCK) for the MIIGSK RMII Logic is 50 MHz in order to support 100 Mbps operation. 1 In RMII mode the clock source (REF_CLOCK) for the MIIGSK RMII Logic is divided by 10 (5 MHz) in order to support 10 Mbps operation. **Note:** This bit has no effect in MII mode.
5	Reserved.
4 LBMODE	RMII domain internal loopback mode. 0 Normal operation of the MIIGSK block (default). 1 The MIIGSK MII transmit inputs (from ethernet controller) are looped back to the MIIGSK MII receive outputs (to ethernet controller) through the MIIGSK RMII Tx/Rx Logic **Note:** Proper operation is guaranteed only when I/F Mode fields = '01'.
3-2	Reserved.
1-0 I/F_MODE	This field determines the type of interface that the MIIGSK is connected to. 00 - MII Mode 01- RMII Mode

25.3.5.25 MIIGSK Enable Register (MIIGSK_ENR)

The MIIGSK Enable Register contains two bits which allow the user to enable/disable the MIIGSK operation and also gives an indication about the MIIGSK operation status (enabled or disabled).The status bit is used to ensure proper configuration of the MIIGSK block.

Offset 0x0308 (MIIGSK_ENR) Access: Read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	READY	EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 25-25. MIIGSK Configuration Register (MIIGSK_ENR)

Table 25-31. MIIGSK_ENR Field Descriptions

Bits	Description
31-3	Reserved.
2 READY	Enable bit status This bit reflects the value of the MIIGSK Enable bit. This bit is Read Only.
1 EN	MIIGSK Enable. Setting this bit allows the MIIGSK to transmit/receive frames to/from the Ethernet Controller. Clearing this bit prevents the transmission/reception of frames. This bit provides a mean to change RMII speed option (MIIGSK_CFGR[FRCONT]) without disabling the whole FEC. This bit is set by default.
0	Reserved.

25.4 Functional Description

This section provides a detailed description of the functions of the FEC including:

- Network interface options
- Frame transmission and reception
- Full-duplex flow control
- Internal and external loopback

25.4.1 Network Interface Options

The FEC supports an MII interface, an RMII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet. The interface mode is selected by MIIGSK_CFGR[I/F_MODE] together with the RCR[MII_MODE] bit as shown in Table 25-32.

Table 25-32. Interface Mode Selection

MIIGSK_CFGR[I/F_MODE]	RCR[MII_MODE]	Interface Mode Selected
00	0	7-Wire
00	1	MII
01	1	RMII

In MII mode, there are 18 signals defined by IEEE 802.3 and supported by the FEC. These signals are shown in Table 25-33 below.

Table 25-33. MII Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Transmit clock	FEC_TX_CLK	In
Transmit enable	FEC_TX_EN	Out
Transmit data	FEC_TXD[3:0]	Out
Transmit error	FEC_TX_ER	Out

Table 25-33. MII Mode Signal Configuration (continued)

Signal Description	FEC Signal Name	Direction
Collision	FEC_COL	In
Carrier sense	FEC_CRS	In
Receive clock	FEC_RX_CLK	In
Receive data valid	FEC_RX_DV	In
Receive data	FEC_RXD[3:0]	In
Receive error	FEC_RX_ER	In
Management data clock	FEC_MDC	Out
Management data input/output	FEC_MDIO	I/O

In RMII mode a reduced set of 10 signals are used. These signals are shown in Table 25-34.

Table 25-34. RMII Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Synchronous clock reference (REF_CLK)	FEC_TX_CLK	In
Transmit Enable	FEC_TX_EN	Out
Transmit Data	FEC_TXD[1:0]	Out
Carrier Sense/Receive Data Valid (CRS_DV)	FEC_RX_DV	In
Receive Data	FEC_RXD[1:0]	In
Receive Error	FEC_RX_ER	In
Management Data Clock	FEC_MDC	Out
Management Data Input/Output	FEC_MDIO	I/O

The 7-wire serial interface operates in what is generally referred to as AMD mode. 7-wire mode connections to the external transceiver are shown in Table 25-35.

Table 25-35. 7-Wire Mode Signal Configuration

Signal Description	FEC Signal Name	Direction
Transmit clock	FEC_TX_CLK	In
Transmit enable	FEC_TX_EN	Out
Transmit data	FEC_TXD[0]	Out
Collision	FEC_COL	In
Receive clock	FEC_RX_CLK	In
Receive data valid	FEC_RX_DV	In
Receive data	FEC_RXD[0]	In

25.4.2 FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set to 1 and data appears in the transmit FIFO, the FEC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the TFWR register), the FEC transmit logic asserts FEC_TX_EN and start transmitting the preamble (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller postpones transmission if the network is busy (that is, the carrier sense signal FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense remains inactive for 60 bit periods. If so, the transmission begins after waiting an additional 36 bit periods (96 bit periods after carrier sense originally became inactive). See Section 25.4.2.3, “Transmission Error Handling,” for more details.

If a collision occurs during transmission of a frame in half-duplex mode, the Ethernet controller follows the specified backoff procedures (see Section 25.4.2.2, “Collision Handling”) and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, if the TC bit is set in the transmit frame control word then a 32-bit cyclic redundancy check (CRC) known as the frame check sequence (FCS) is appended. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set to 1).

Both buffer (TXB) and frame (TXF) interrupts can be generated as determined by the settings in the EIMR.

The transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, and XFIFO_UN. If the transmit frame length exceeds MAX_FL bytes the BABT interrupt is asserted: however, the entire frame is transmitted (no truncation).

Transmission is paused by setting the graceful transmit stop (GTS) bit in the TCR register. When the TCR[GTS] is set to 1, the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

The Ethernet controller transmits bytes LSB first.

25.4.2.1 Transmit Inter-Packet Gap (IPG) Time

The minimum inter-packet gap (IPG) time for back-to-back transmission is 96 bit periods. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96-bit–period IPG counter. Frame transmission begins 96 bit periods after carrier sense is negated if it stays negated for at least 60 bit periods. If carrier sense is asserted during the last 36 bit periods, it is ignored and a collision occurs.

25.4.2.2 Collision Handling

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit periods, transmitting a jam pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit periods, the retry process is initiated. The transmitter waits a random number of slot periods, where one slot period is 512 bit periods. If a collision occurs after 512 bit periods, then no retransmission is performed and the end of frame buffer is closed with a late collision (LC) error indication.

25.4.2.3 Transmission Error Handling

The Ethernet controller reports frame transmission error conditions using the FEC RxBDs, the EIR register, and the MIB block counters

There are four types of transmission errors:

- Transmitter underrun
- Retransmission attempts limit expired
- Late collision
- Heartbeat

The FEC's procedures for handling these errors are described in the following subsections.

25.4.2.3.1 Transmitter Underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error, then stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the EIR and the UN interrupt is asserted (if enabled in the EIMR register). The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

25.4.2.3.2 Retransmission Attempts Limit Expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the RL bit is set in the EIR. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The "RL" interrupt is asserted if enabled in the EIMR register.

25.4.2.3.3 Late Collision

When a collision occurs after the slot time (512 bits starting at the preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and the LC bit is set in the EIR register. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame.

The LC interrupt is asserted if enabled in the EIMR register.

25.4.2.3.4 Heartbeat

Some transceivers have a self-test feature called “heartbeat” or “signal quality error.” To signify a good self-test, the transceiver indicates a collision to the FEC within 4 microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the TCR register and the heartbeat condition is not detected by the FEC after a frame transmission, then a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets the HB bit in the EIR register, and generates the HBERR interrupt if it is enabled.

25.4.3 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking.

When the driver enables the FEC receiver by setting ECR[ETHER_EN] to 1, it immediately starts processing receive frames. When FEC_RX_DV asserts, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the frame is processed by the receiver. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit periods of RX_D0 following assertion of FEC_RX_DV are ignored. Following the first 16 bit periods the data sequence is checked for alternating 1/0s. If a 0b11 or 0b00 data sequence is detected during bit periods 17 to 21, the remainder of the frame is ignored. After bit period 21, the data sequence is monitored for a valid SFD (11). If a 0b00 is detected, the frame is rejected. When a 0b11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes can occur, but if a 0b00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, the receive FIFO is signaled that the frame is accepted and can be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Thus no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV and TR status bits, and the frame length. See Section 25.4.3.3, “Reception Error Handling” for more details.

Receive Buffer (RXB) and Frame Interrupts (RXF) can be generated if enabled by the EIMR register. The only receive error interrupt is babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See Section 25.5.2.3, “Ethernet Receive Buffer Descriptor (RxBD)” for more details.

When the receive frame is complete, the FEC sets the L bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E bit. The Ethernet controller next generates a maskable interrupt (RXF bit in EIR, maskable by RXF bit in EIMR), indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

25.4.3.1 Receive Inter-Packet Gap (IPG) Time

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit periods. If an inter-packet gap between receive frames is less than 28 bit periods, the second frame may be discarded by the receiver.

25.4.3.2 Ethernet Address Recognition

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 25-26 illustrates the address recognition decisions made by the receive block, while Figure 25-27 illustrates the decisions made by the microcontroller.

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the DA field.

If the DA is a broadcast address, and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in Figure 25-26. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 25-27.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame. The hash algorithm is described in Section 25.4.3.2.1, “Hash Algorithm.”

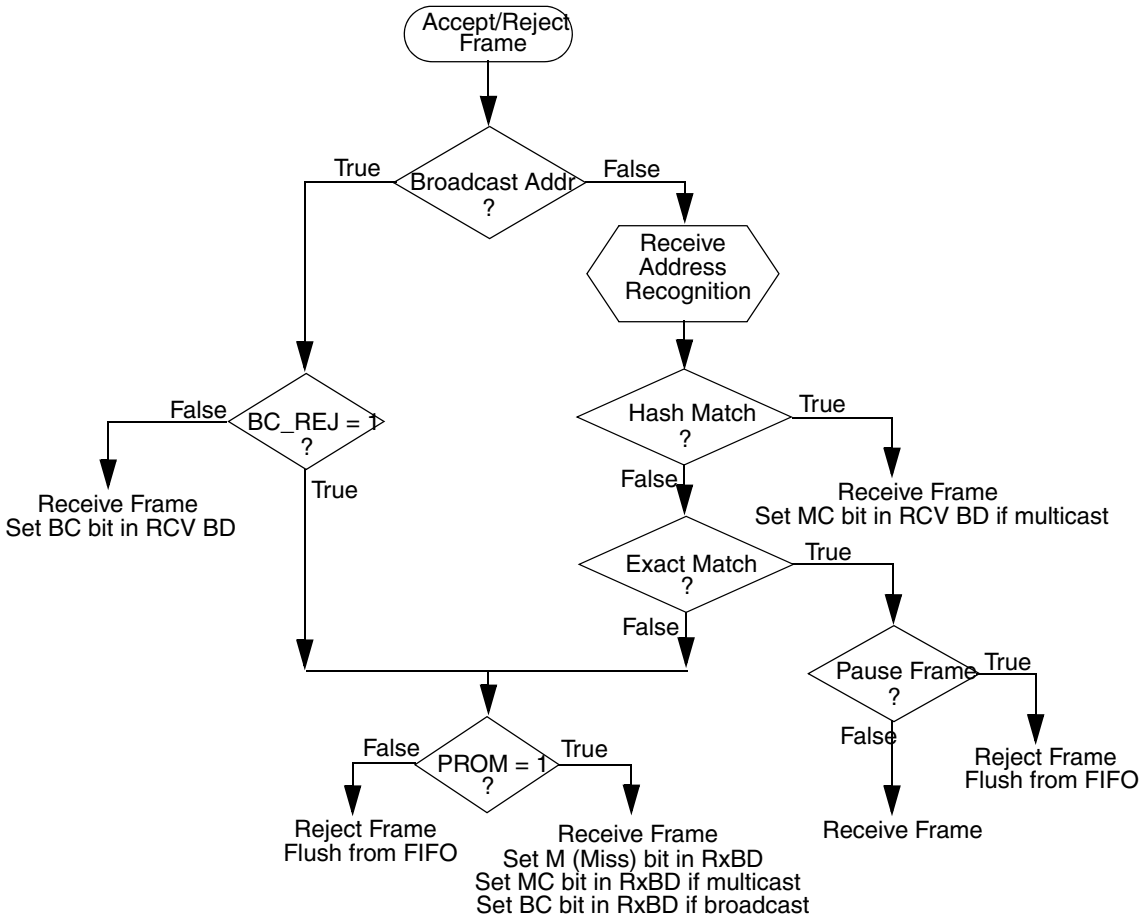
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines that the received frame is a valid pause frame, then the frame is rejected. Note the receiver detects a pause frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR (the hash algorithm is described in Section 25.4.3.2.1, “Hash Algorithm”). In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on pause frame detection, shown in Figure 25-26.

If neither a hash match (group or individual), nor an exact match occur, then if promiscuous mode is enabled (RCR[PROM] = 1), then the frame is accepted and the MISS bit in the receive buffer descriptor is set: otherwise, the frame is rejected.

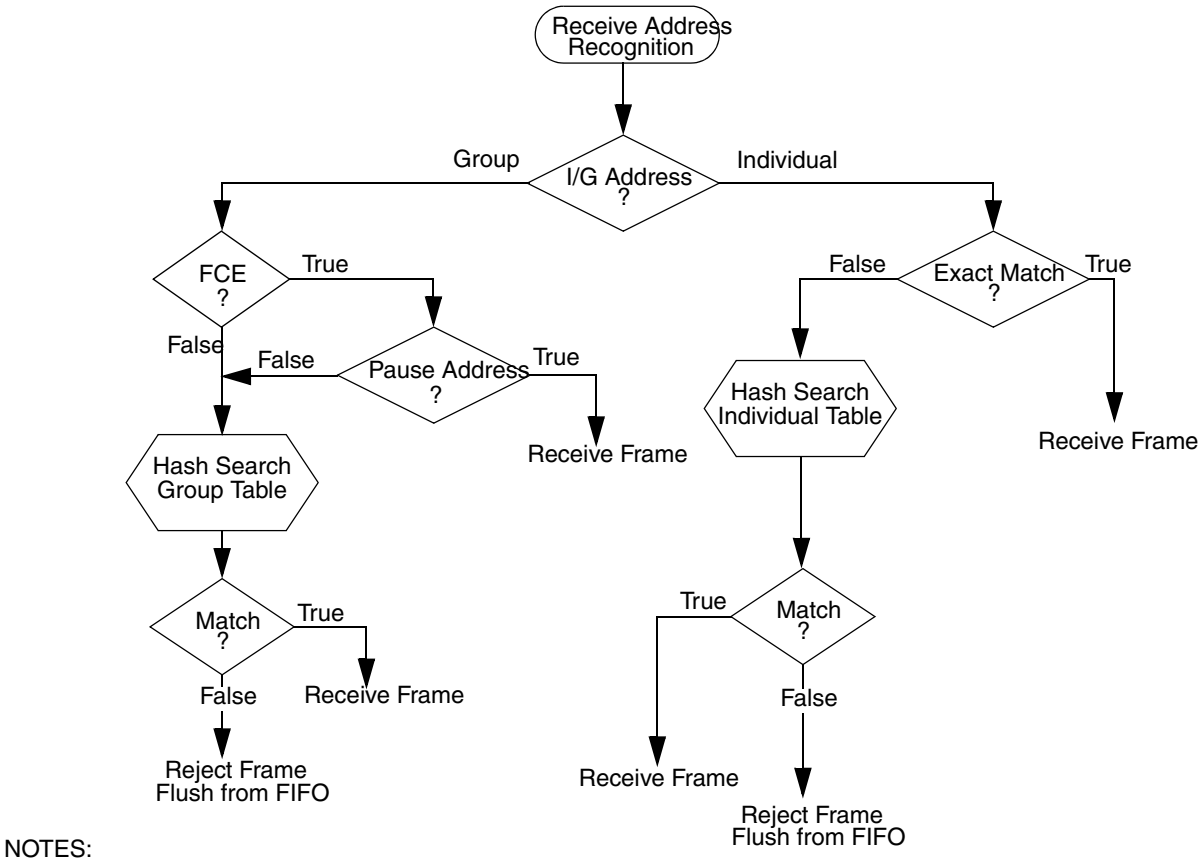
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, then the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



NOTES:
BC_REJ—Field in RCR register (BroadCast REJect)
PROM—Field in RCR register (PROMiscuous mode)
Pause Frame—Valid PAUSE frame received

Figure 25-26. Ethernet Address Recognition—Receive Block Decisions



NOTES:
FCE - field in RCR register (Flow Control Enable)
I/G - Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 25-27. Ethernet Address Recognition—Microcode Decisions

25.4.3.2.1 Hash Algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, which are represented by 64 bits stored in GAUR, GALR (group address hash match) or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects GAUR (MSB = 1) or GALR (MSB = 0). The least significant five bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected. For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The CRC32 polynomial to use in computing the hash is shown in Equation :

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

Table 25-36 shows example destination addresses and corresponding hash values is included below for reference.

Table 25-36. Destination Address to 6-Bit Hash

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
65:FF:FF:FF:FF:FF	0x0	0
55:FF:FF:FF:FF:FF	0x1	1
15:FF:FF:FF:FF:FF	0x2	2
35:FF:FF:FF:FF:FF	0x3	3
B5:FF:FF:FF:FF:FF	0x4	4
95:FF:FF:FF:FF:FF	0x5	5
D5:FF:FF:FF:FF:FF	0x6	6
F5:FF:FF:FF:FF:FF	0x7	7
DB:FF:FF:FF:FF:FF	0x8	8
FB:FF:FF:FF:FF:FF	0x9	9
BB:FF:FF:FF:FF:FF	0xA	10
8B:FF:FF:FF:FF:FF	0xB	11
0B:FF:FF:FF:FF:FF	0xC	12
3B:FF:FF:FF:FF:FF	0xD	13
7B:FF:FF:FF:FF:FF	0xE	14
5B:FF:FF:FF:FF:FF	0xF	15
27:FF:FF:FF:FF:FF	0x10	16
07:FF:FF:FF:FF:FF	0x11	17
57:FF:FF:FF:FF:FF	0x12	18
77:FF:FF:FF:FF:FF	0x13	19
F7:FF:FF:FF:FF:FF	0x14	20
C7:FF:FF:FF:FF:FF	0x15	21
97:FF:FF:FF:FF:FF	0x16	22
A7:FF:FF:FF:FF:FF	0x17	23
99:FF:FF:FF:FF:FF	0x18	24
B9:FF:FF:FF:FF:FF	0x19	25
F9:FF:FF:FF:FF:FF	0x1A	26

Table 25-36. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
C9:FF:FF:FF:FF:FF	0x1B	27
59:FF:FF:FF:FF:FF	0x1C	28
79:FF:FF:FF:FF:FF	0x1D	29
29:FF:FF:FF:FF:FF	0x1E	30
19:FF:FF:FF:FF:FF	0x1F	31
D1:FF:FF:FF:FF:FF	0x20	32
F1:FF:FF:FF:FF:FF	0x21	33
B1:FF:FF:FF:FF:FF	0x22	34
91:FF:FF:FF:FF:FF	0x23	35
11:FF:FF:FF:FF:FF	0x24	36
31:FF:FF:FF:FF:FF	0x25	37
71:FF:FF:FF:FF:FF	0x26	38
51:FF:FF:FF:FF:FF	0x27	39
7F:FF:FF:FF:FF:FF	0x28	40
4F:FF:FF:FF:FF:FF	0x29	41
1F:FF:FF:FF:FF:FF	0x2A	42
3F:FF:FF:FF:FF:FF	0x2B	43
BF:FF:FF:FF:FF:FF	0x2C	44
9F:FF:FF:FF:FF:FF	0x2D	45
DF:FF:FF:FF:FF:FF	0x2E	46
EF:FF:FF:FF:FF:FF	0x2F	47
93:FF:FF:FF:FF:FF	0x30	48
B3:FF:FF:FF:FF:FF	0x31	49
F3:FF:FF:FF:FF:FF	0x32	50
D3:FF:FF:FF:FF:FF	0x33	51
53:FF:FF:FF:FF:FF	0x34	52
73:FF:FF:FF:FF:FF	0x35	53
23:FF:FF:FF:FF:FF	0x36	54
13:FF:FF:FF:FF:FF	0x37	55
3D:FF:FF:FF:FF:FF	0x38	56
0D:FF:FF:FF:FF:FF	0x39	57
5D:FF:FF:FF:FF:FF	0x3A	58
7D:FF:FF:FF:FF:FF	0x3B	59

Table 25-36. Destination Address to 6-Bit Hash (continued)

48-bit DA (in hex)	6-bit Hash (in hex)	Hash Decimal Value
FD:FF:FF:FF:FF:FF	0x3C	60
DD:FF:FF:FF:FF:FF	0x3D	61
9D:FF:FF:FF:FF:FF	0x3E	62
BD:FF:FF:FF:FF:FF	0x3F	63

25.4.3.3 Reception Error Handling

The Ethernet controller reports frame reception error conditions using the FEC RxBDs, the EIR register, and the MIB block counters

There are five types of reception errors:

- Overrun
- Non-octet (dribbling bits)
- CRC
- Frame-length violation
- Truncation

These are described in the following subsections.

25.4.3.3.1 Overrun

If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the RxBD. All subsequent data in the frame is discarded, and subsequent frames can also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. This frame must be discarded by the driver.

25.4.3.3.2 Non-Octet (Dribbling Bits)

The Ethernet controller handles up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, then the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, then no error is reported.

25.4.3.3.3 CRC

When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the RxBD. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

25.4.3.3.4 Frame Length Violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and the LG bit in the end of frame RxBD is set. The frame is not truncated unless the frame length exceeds 2047 bytes).

25.4.3.3.5 Truncation

When the receive frame length exceeds 2047 bytes, the frame is truncated and the TR bit is set in the RxBD.

25.4.4 Full-Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, FEC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] asserted) and flow control enable (RCR[FCE]) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in Table 25-37. In addition, the receive status associated with the frame indicates that the frame is valid.

Table 25-37. Pause Frame Field Specification

48-bit destination address	0x0180_C200_0001 or physical address
48-bit source address	Any
16-bit type	0x8808
16-bit opcode	0x0001
16-bit pause duration	0x0000–0xFFFF

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is asserted by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. Note that the pause timer makes use of the transmit backoff timer hardware, which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments after every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing FEC data frame transmission to resume. Note that the receive flow control pause (TCR[RFC_PAUSE]) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and the user must assert flow control pause (TCR[TFC_PAUSE]). On assertion of transmit flow control pause (TCR[TFC_PAUSE]), the transmitter asserts TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts. Following EIR[GRA] assertion, the pause frame is transmitted. On completion of pause frame transmission, flow control pause (TCR[TFC_PAUSE]) and TCR[GTS] are cleared internally.

The user must specify the desired pause duration in the OPD register.

Note that when the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (TCR[TFC_PAUSE]) still can be asserted and causes the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

25.4.5 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the RCR register, the FDEN bit in the TCR register and the LBMODE bit in MIIGSK_CFGR register. Furthermore internal loopback has two levels which loops data back in MII domain and RMII domain respectively.

For both internal and external loopback set FDEN = 1 and RCR[DRT] = 0.

For the MII domain internal loopback set RCR[LOOP] = 1 and MIIGSK_CFGR[LBMODE] = 0. FEC_TX_EN and FEC_TX_ER will not assert during internal loopback. During the MII domain internal loopback, the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This will cause an increase in the required system bus bandwidth for transmit and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For the RMII domain internal loopback, set RCR[LOOP] = 0 and MIIGSK_CFGR[LBMODE] = 1. The data is sent from the MII domain into MIIGSK, converted to RMII format by a RMII transmitter inside the MIIGSK, then looped back through a RMII receiver of the MIIGSK, finally gets back to the MII domain.

For external loopback set RCR[LOOP] = 0, MIIGSK_CFGR[LBMODE] = 0, and configure the external transceiver for loopback.

25.5 Initialization/Application Information

25.5.1 Initialization Sequence

This section describes which registers are reset due to hardware reset, which are reset by the FEC RISC, and what locations the user must initialize prior to enabling the FEC.

25.5.1.1 Hardware Controlled Initialization

In the FEC, registers and control logic that generate interrupts are reset by hardware. A hardware reset negates output signals and resets general configuration bits.

Other registers are reset when the ECR[ETHER_EN] bit is cleared, as indicated in Table 25-38. ECR[ETHER_EN] is cleared by a hard reset or can be cleared by software to halt operation. By clearing ECR[ETHER_EN], the configuration control registers such as the TCR and RCR do not reset, but the entire data path resets.

Table 25-38. Effect on FEC of Clearing ECR[ETHER_EN]

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated

Table 25-38. Effect on FEC of Clearing ECR[ETHER_EN] (continued)

Register/Machine	Reset Value
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation
MIIGSK	Interface to transceiver is disabled

25.5.1.2 User Initialization (Prior to Asserting ECR[ETHER_EN])

The user must initialize portions of the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The order of initializations is not important.

The FEC and FEC FIFO/DMA registers requiring initialization are defined in Table 25-39.

Table 25-39. Registers Requiring Initialization before Setting ECR[ETHER_EN]

Register	Location (FEC or FIFO/DMA)	Comments
EIMR	FEC	Requires initialization
EIR	FEC	Must be cleared by writing 0xFFFF_FFFF
TFWR	FEC	Optional
IALR / IAUR	FEC	—
GAUR / GALR	FEC	—
PALR / PAUR	FEC	—
OPD	FEC	Only needed for full-duplex flow control
RCR	FEC	—
TCR	FEC	—
MSCR	FEC	Optional
MIIGSK_CFGR	FEC	—
MIB counters	FEC	Must be cleared
FRSR	FIFO/DMA	Initialization is optional
EMRBR	FIFO/DMA	—
ERDSR	FIFO/DMA	—
ETDSR	FIFO/DMA	—
Transmit Descriptor ring	FIFO/DMA	Must be emptied
Receive Descriptor ring	FIFO/DMA	Must be emptied

25.5.1.3 Microcontroller Initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, the hardware is ready for operation.

The microcontroller initialization sequence follows:

1. Initialize BackOff Random Number Seed

2. Activate Receiver
3. Activate Transmitter
4. Clear Transmit FIFO
5. Clear Receive FIFO
6. Initialize Transmit Ring Pointer
7. Initialize Receive Ring Pointer
8. Initialize FIFO Count Register

25.5.1.4 User Initialization (after asserting ECR[ETHER_EN])

After asserting ECR[ETHER_EN], the user can set up the buffer/frame descriptors and write to the TDAR and RDAR. See Section 25.5.2, “Buffer Descriptors,” for more details.

25.5.2 Buffer Descriptors

This section provides a description of the operation of the driver/DMA using the buffer descriptors (BD). It is followed by a detailed description of the receive and transmit descriptor fields.

25.5.2.1 Driver/DMA Operation with Buffer Descriptors

The data for the FEC frames must reside in memory external to the FEC. The data for a frame is placed in one or more buffers. Associated with each buffer is a buffer descriptor (BD) which contains a starting address (pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read in by the FEC DMA engine.

Software “produces” buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit “produces” the buffer. Software writing to either the TDAR or RDAR tells the FEC that a buffer has been placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and “consumes” the buffers after they have been produced. After the data DMA is complete and the buffer descriptor status bits have been written by the DMA engine, the RxBD[E] or TxBD[R] bit is cleared by hardware to signal the buffer has been “consumed.” Software can poll the BDs to detect when the buffers have been consumed or can rely on the buffer/frame interrupts. These buffers can then be processed by the driver and returned to the free list.

The ECR[ETHER_EN] signal operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before the ECR[ETHER_EN] bit is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The last buffer descriptor in each ring is defined by the Wrap (W) bit. When set, W indicates that the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively.

NOTE

Buffer descriptor rings must start on a 128-bit boundary.

25.5.2.2 Ethernet Transmit Buffer Descriptor (TxBD)

Figure 25-28 shows the transmit buffer descriptor format. Table 25-40 describes the TxBD fields.

Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated using individual interrupt bits (error conditions) and in statistic counters in the MIB block. See Section 25.3.3, “Message Information Block (MIB) Counters Memory Map,” for more details.

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	R	TO1	W	TO2	L	TC	ABC									
	Data Length															
0x0004	Tx Data Buffer Pointer A[31:16]															
	Tx Data Buffer Pointer A[15:0]															

Figure 25-28. Transmit Buffer Descriptor (TxBD)

Table 25-40. Transmit Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 R	Ready. Written by the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD can be written by the user after this bit is set.
	30 TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware, nor does its value affect hardware.
	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
	28 TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
	27 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
	26 TC	Tx CRC. Written by user (only valid if L = 1). 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
	25 ABC	Append bad CRC. Written by user (only valid if L = 1). 0 No effect 1 Transmit the CRC sequence inverted after the last data byte (regardless of TC value).
	24–16	Reserved.
	15–0 Data Length	Data Length, written by user. Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. Bits [10:0] are used by the DMA engine, bits[15:11] are ignored.
0x0004	31–0 A	Tx data buffer pointer[1]

[1] The transmit buffer pointer, which contains the address of the associated data buffer, must always be divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

25.5.2.2.1 Driver/DMA Operation with Transmit Buffer Descriptors

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. After the software driver has set up the buffers for a frame, it sets up the corresponding BDs. In the TxBD the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword, and the buffer pointer in the second longword.The last step in setting up the BDs for a transmit frame is to set the R bit in the first BD for the frame. The driver follows that with a write to TDAR which triggers the FEC to poll the next BD in the ring.

The Ethernet controller confirms transmission by clearing the ready bit (R bit) when DMA of the buffer is complete.

Transmit Frame in Multiple Buffers

Typically a transmit frame is divided between multiple buffers. For example, it is possible to have an application payload in one buffer, TCP header in a 2nd buffer, IP header in a 3rd buffer, and Ethernet/IEEE 802.3 header in a 4th buffer. The FEC does not prepend the Ethernet header (Destination Address, Source Address, Length/Type field(s)), so this must be provided by the driver in one of the transmit buffers. The FEC can append the Ethernet CRC to the frame. Whether the CRC is appended by the FEC or by the driver is determined by the TC bit in the transmit BD, which must be set by the driver.

The driver (TxBD software producer) sets up Tx BDs in such a way that a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, first the BD's are initialized with pointer, length and control (W, L, TC, ABC) and then the TxBD[R] bits are set to 1 in *reverse order* (3rd, 2nd, 1st BD) to insure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the 2nd was made available, potentially causing a transmit FIFO underrun.

In the FEC, the DMA is notified by the driver that new transmit frame(s) are available by writing to the TDAR register. When this register is written to (data value is not significant) the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers, until a transmit BD is encountered with the R bit = 0. At this point the FEC polls this BD one more time. If the R bit = 0 the second time, then the RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

25.5.2.3 Ethernet Receive Buffer Descriptor (RxBD)

Figure 25-29 shows the receive buffer descriptor format. Table 25-41 describes the RxBD fields.

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	E	RO1	W	RO2	L			M	B	MC	LG	NO		CR	OV	TR
	Data Length															
0x0004	Rx Data Buffer Pointer A[31:16]															
	Rx Data Buffer Pointer A[15:0]															

Figure 25-29. Receive Buffer Descriptor (RxBD)

Table 25-41. Receive Buffer Descriptor Field Definitions

Offset	Field	Description
0x0000	31 E	Empty. Written by the FEC (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
0x0000	30 RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	29 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ERDSR.
0x0000	28 RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
0x0000	27 L	Last in frame. Written by the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
0x0000	26–25	Reserved.
0x0000	24 M	Miss. Written by the FEC. This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition. Thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
0x0000	23 BC	is set if the DA is broadcast (FF:FF:FF:FF:FF:FF).
0x0000	22 MC	is set if the DA is multicast and not BC.
0x0000	21 LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
0x0000	20 NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit is not set.
0x0000	19	Reserved
0x0000	18 CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
0x0000	17 OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and is zero. This bit is valid only if the L-bit is set.
0x0000	16 TR	is set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set the frame is discarded and the other error bits ignored as they can be incorrect.

Table 25-41. Receive Buffer Descriptor Field Definitions (continued)

Offset	Field	Description
0x0000	15–0 Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L = 0 (the value is equal to EMRBR), or the length of the frame including CRC if L = 1. It is written by the FEC once as the BD is closed.
0x0004	31–0	RX data buffer pointer[1]

1 The receive buffer pointer, which contains the address of the associated data buffer, must always be divisible by 16. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

25.5.2.3.1 Driver/DMA Operation with Receive Buffer Descriptors

Unlike the transmit case, the length of the receive frame is unknown by the driver ahead of time. Therefore the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) sets up some number of "empty" buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L (1 indicates last buffer in frame) bit, the frame status bits (if L = 1) and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end of frame buffers, the L=1 bit is set in the receive BD, and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame is discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not just the length of the last buffer.

For simplicity the driver can assign the default receive buffer length to be large enough to contain an entire frame, keeping in mind that a malfunction on the network or out-of-specification implementation could result in giant frames. Frames of 2 Kbytes (2048 bytes) or larger are truncated by the FEC at 2047 bytes, so software never sees a receive frame larger than 2047 bytes.

As in the transmit case, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received the FEC fills receive buffers and update the associated BDs, then reads the next BD in the receive descriptor ring. If the FEC reads a receive BD and finds the E bit is cleared, it polls this BD once more. If E is still cleared, then the FEC stops reading receive BDs until the driver writes to RDAR.

NOTE

Whenever the software driver sets an E bit in one or more receive descriptors, the driver follows that with a write to RDAR.

Chapter 26
Controller Area Network (FlexCAN)

The FlexCAN module is a communication controller that implements the CAN protocol according to the CAN 2.0B protocol specification. The CAN protocol was designed primarily (but not solely) to meet requirements suitable for a serial data bus in vehicle applications, including: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and sufficient bandwidth.

References: This document assumes an understanding of the following references:

- Controller Area Network – CAN Specification Version 2.0 Part A, Part B, Robert Bosch GmbH, 1991.
- ISO International Standard – ISO 11898 First Edition 1993 Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for high-speed Communication.

26.1 Introduction

The FlexCAN module is a full implementation of the CAN protocol specification (version 2.0B), which supports both standard and extended message frames.

Figure 26-1 is a block diagram showing the main submodules of the FlexCAN.

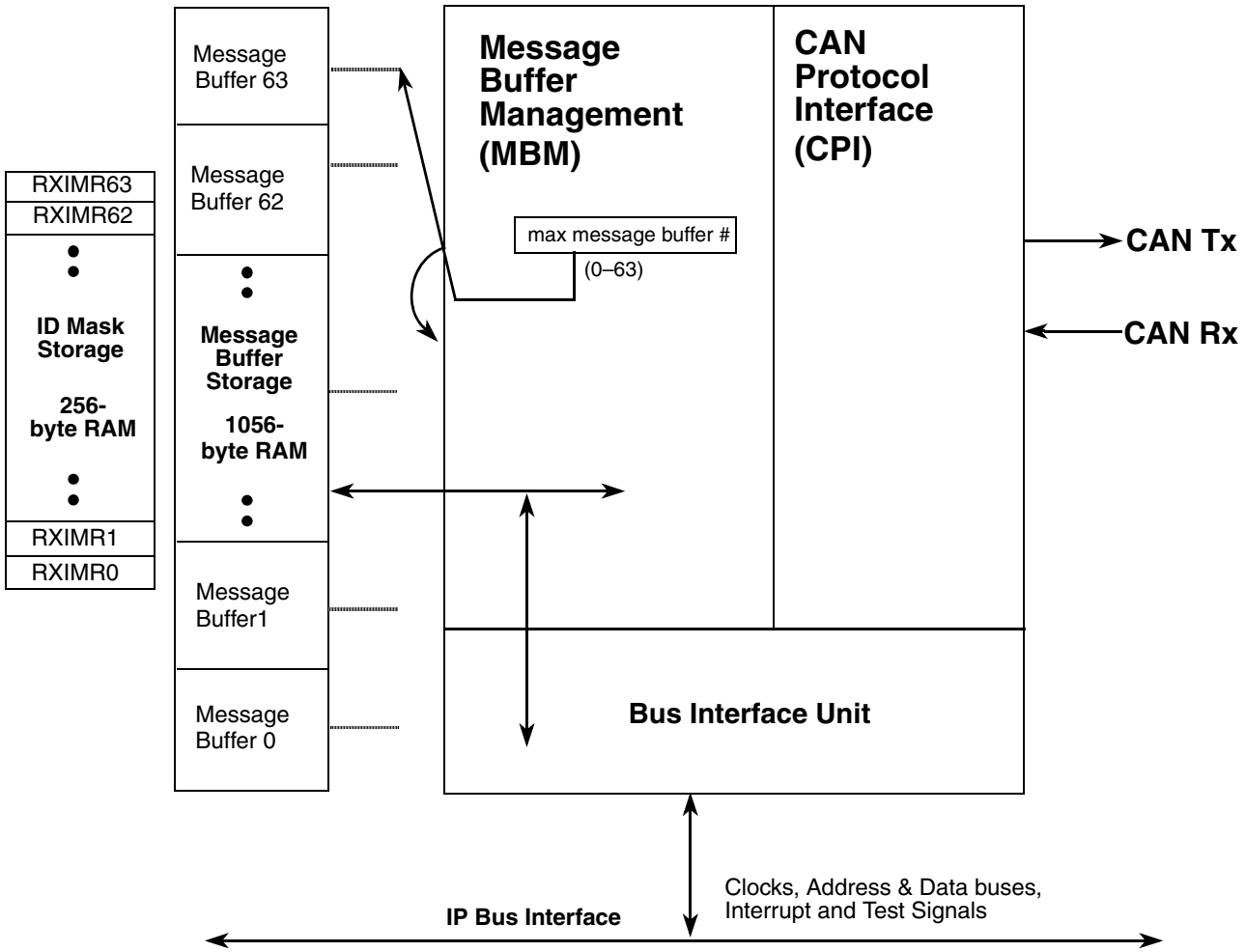


Figure 26-1. FlexCAN Block Diagram

The FlexCAN includes the following submodules:

- An embedded RAM for storing message buffers (64 message buffers are supported)
- An embedded RAM for storing individual Rx mask registers
- A CAN protocol interface (CPI) submodule that manages the serial communication on the CAN bus: requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling.
- A message buffer management (MBM) submodule which handles message buffer selection for reception and transmission, taking care of arbitration and ID-matching algorithms.
- A bus interface unit (BIU) submodule which controls the access to and from the internal interface bus, in order to establish connection to the ARM and to other blocks. Clocks, address and data buses, interrupt outputs, and test signals are accessed through the bus interface unit.

26.1.1 FlexCAN Module Features

The FlexCAN module includes the following features:

- Full implementation of the CAN protocol specification, version 2.0B
  - — Standard data and remote frames
  - — Extended data and remote frames
  - — Zero to eight bytes data length
  - — Programmable bit rate up to 1 Mbyte/sec
  - — Content-related addressing
- Flexible message buffers of zero to eight bytes data length
- Each message buffer configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx mask registers per message buffer
- Includes 1056 bytes of RAM used for message buffer storage (64 message buffers)
- Includes 256 bytes of RAM used for individual Rx mask registers (64 mask registers, one for each message buffer)
- Full-featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard, or 32 partial (8-bit) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Selectable clock source to the CAN protocol interface (bus clock or crystal oscillator)
- Unused message buffer and Rx mask register space can be used as general-purpose RAM space
- Listen-only mode capability
- Programmable loopback mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency due to an arbitration scheme for high-priority messages
- Low-power modes, with programmable wake-up on bus activity

26.1.2 Modes of Operation

The FlexCAN module has four functional modes: normal mode (user and supervisor), freeze mode, listen-only mode and loopback mode. There are also three low-power modes: disable mode, doze mode and stop mode. These modes are described as follows:

- Normal mode (user or supervisor)—The module receives and/or transmits message frames, errors are handled normally, and all the CAN protocol functions are enabled. User and supervisor modes differ in the access to some restricted control registers.

- Freeze mode—Enabled when the FRZ bit in the module configuration register (MCR) is set to 1. If enabled, freeze mode is entered when the HALT bit in the MCR is set, or when debug mode is requested at the core level. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Section 26.4.9.1, “Freeze Mode,” for more information.
- Listen-only mode—The module enters this mode when the LOM bit in the control register is set to 1. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN error passive model (see CAN protocol specification). Only messages acknowledged by another CAN station are received. If FlexCAN detects a message that has not been acknowledged, it flags a BIT0 error (without changing the REC), as if it were trying to acknowledge the message.
- Loopback mode—The module enters this mode when the LPB bit in the control register is set to 1. In this mode, FlexCAN performs an internal loopback that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input signal is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.
- Module disable mode—This low-power mode is entered when the MDIS bit in the MCR is set to 1. When disabled, the module shuts down the clocks to the CAN protocol interface and message buffer management submodules. This mode is exited by clearing the MDIS bit in the MCR. See Section 26.4.9.2, “Module Disable Mode,” for more information.
- Doze mode—This low-power mode is entered when the DOZE bit in the MCR is set to 1 and doze mode is requested at the core level. When in doze mode, the module shuts down the clocks to the CAN protocol interface and the message buffer management submodules. This mode is exited when the DOZE bit in the MCR is cleared, when the core is removed from doze mode, or when activity is detected on the CAN bus and the self wake-up mechanism is enabled. See Section 26.4.9.3, “Doze Mode” for more information.
- Stop mode—This low-power mode is entered when stop mode is requested at the core level. When in stop mode, the module puts itself in an inactive state and then informs the ARM that the clocks can be shut down globally. This mode is exited when the stop-mode request is removed or when activity is detected on the CAN bus and the self wake-up mechanism is enabled. See Section 26.4.9.4, “Stop Mode,” for more information.

26.2 External Signal Description

26.2.1 Overview

The FlexCAN module has two I/O signals, which are summarized in Table 26-1 and described in more detail in the following subsections.

Table 26-1. FlexCAN External Signals

Signal Name	Direction	Description
CAN Rx	Input	CAN receive signal
CAN Tx	Output	CAN transmit signal

26.2.2 CAN Rx Signal

This is the receive signal from the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

26.2.3 CAN Tx Signal

This is the transmit signal to the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

26.3 Memory Map and Register Definition

This section includes the module memory map, and detailed descriptions of all registers and buffers. For the base address of a particular module instantiation, see the system memory map.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address: followed by message buffer storage space in embedded RAM starting at offset 0x0060; and an extra ID mask storage space in a separate embedded RAM starting at offset 0x0880.

26.3.1 Memory Map

Table 26-2 shows the module memory map.

Access type can be supervisor (S) or unrestricted (U). Most registers can be configured to have either supervisor or unrestricted access by programming the SUPV bit in the MCR: these registers are identified as S/U in the Access column of Table 26-2.

The Rx global mask (RXGMASK), Rx buffers 14 mask and 15 mask (RX14MASK and RX15MASK) registers are provided for backwards compatibility, and are not used when the BCC bit in the MCR is set to 1.

The address offsets 0x0060–0x047F (1056 bytes) and 0x0880–0x097F (256 bytes) are occupied by two separate embedded memories. These two ranges are completely occupied by RAM.

Table 26-2. FlexCAN Module Memory Map

Address Offset (Register/Buffer Abbreviation)	Register/Buffer	Access	Affected by Hard Reset	Affected by Soft Reset	Section/Page
0x0000 (MCR)	Module configuration	S R/W	Yes	Yes	26.3.2.1/26-7
0x0004 (CTRL)	Control register	S/U R/W	Yes	No	26.3.2.2/26-11
0x0008 (TIMER)	Free-running timer	S/U R/W	Yes	Yes	26.3.2.3/26-14
0x0010 (RXGMASK)	Rx global mask	S/U R/W	Yes	No	26.3.2.4/26-14
0x0014 (RX14MASK)	Rx buffer 14 mask	S/U R/W	Yes	No	26.3.2.5/26-15
0x0018 (RX15MASK)	Rx buffer 15 mask	S/U R/W	Yes	No	26.3.2.6/26-15
0x001C (ECR)	Error counter register	S/U R/W	Yes	Yes	26.3.2.7/26-16
0x0020 (ESR)	Error and status register	S/U R/W	Yes	Yes	26.3.2.8/26-17
0x0024 (IMASK2)	Interrupt masks 2	S/U R/W	Yes	Yes	26.3.2.9/26-19
0x0028 (IMASK1)	Interrupt masks 1	S/U R/W	Yes	Yes	26.3.2.10/26-21
0x002C (IFLAG2)	Interrupt flags 2	S/U R/W	Yes	Yes	26.3.2.11/26-22
0x0030 (IFLAG1)	Interrupt flags 1	S/U R/W	Yes	Yes	26.3.2.12/26-23
0x0080–0x017F (MB0–15)	Message buffers 0–15	S/U R/W	No	No	26.3.3.1/26-25
0x0180–0x027F (MB16–31)	Message buffers 16–31	S/U R/W	No	No	26.3.3.1/26-25
0x0280–0x047F (MB32–63)	Message buffers 32–63	S/U R/W	No	No	26.3.3.1/26-25
0x0880–0x08BF (RXIMR0–15)	Rx individual mask registers	S/U R/W	No	No	26.3.2.13/26-24
0x08C0–0x08FF (RXIMR16–31)	Rx individual mask registers RXIMR16-RXIMR31	S/U R/W	No	No	26.3.2.13/26-24
0x0900–0x097F (RXIMR32–63)	Rx individual mask registers RXIMR32-RXIMR63	S/U R/W	No	No	26.3.2.13/26-24

26.3.2 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 26-2 and Table 26-3 explain conventions used in register diagrams and tables.

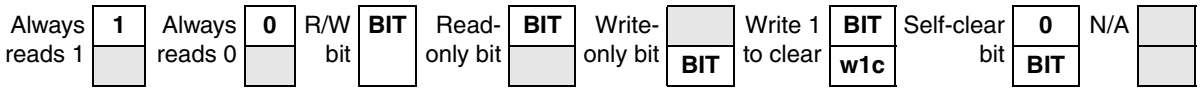


Figure 26-2. Key to Register Fields

Table 26-3. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that can be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

26.3.2.1 Module Configuration Register (MCR)

The MCR, shown in Figure 26-3, defines global system configurations such as the module operation mode (for example, low-power mode) and maximum message buffer configuration. The MAXMB field must only be changed while the module is in freeze mode: all other fields in this register can be accessed at any time.

Offset 0x0000 (MCR) Access: Supervisor read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDIS	FRZ	FEN	HALT	NOT_RDY	WAK_MSK	SOFT_RST	FRZ_ACK	SUPV	SLF_WAK	WRN_EN	LPM_ACK	WAK_SRC	DOZE	SRX_DIS	BCC
W																
Reset	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	LPRO_EN	AEN	0	0	IDAM		0	0	MAXMB					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Figure 26-3. Module Configuration Register (MCR)

Table 26-4. Module Configuration Register Description

Field	Description
31 MDIS	This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the clocks to the CAN protocol interface and message buffer management submodules. This is the only bit in the MCR not affected by soft reset. See Section 26.4.9.2, “Module Disable Mode,” for more information. 0 Enable the FlexCAN module 1 Disable the FlexCAN module
30 FRZ	The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR is set or when debug mode is requested at core level. When FRZ is set to 1, FlexCAN is enabled to enter freeze mode. Clearing this bit field causes FlexCAN to exit from freeze mode. 0 Not enabled to enter freeze mode 1 Enabled to enter freeze mode
29 FEN	This bit controls whether the FIFO feature is enabled or not. When FEN is set to 1, message buffers 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x0080-0x00FF) is used by the FIFO engine. See Section 26.3.3.2, “Rx FIFO Description,” and Section 26.4.7, “Rx FIFO,” for more information. 0 FIFO not enabled 1 FIFO enabled
28 HALT	Setting this bit to 1 puts the FlexCAN module into freeze mode. The ARM clears it after initializing the message buffers and control register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in freeze mode, the ARM has write access to the error counter register, that is otherwise read-only. freeze mode cannot be entered while FlexCAN is in any of the low-power modes. See Section 26.4.9.1, “Freeze Mode,” for more information. 0 No freeze-mode request. 1 Enters freeze mode if the FRZ bit is set to 1.
27 NOT_RDY	This read-only bit indicates that FlexCAN is either in disable mode, doze mode, stop mode or freeze mode. It is cleared after FlexCAN has exited these modes. 0 FlexCAN module is either in normal mode, listen-only mode or loopback mode 1 FlexCAN module is either in disable mode, doze mode, stop mode or freeze mode
26 WAK_MSK	This bit enables the wake up interrupt generation. 0 Wake-up interrupt is disabled 1 Wake-up interrupt is enabled
25 SOFT_RST	When this bit is set to 1, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMASK1, IMASK2, IFLAG1, IFLAG2. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: • CTRL • RXIMR0–RXIMR63 • RXGMASK, RX14MASK, RX15MASK • All message buffers The SOFT_RST bit can be set to1 directly by the ARM writing to the MCR, but it is also set to 1 when a global soft reset is requested at the core level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it can take some time to fully propagate its effect. The SOFT_RST bit remains set while reset is pending, and is automatically cleared when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low-power modes. The module must be first removed from low-power mode, and then soft reset can be applied. 0 No reset request 1 Resets the registers marked as “affected by soft reset” in Table 26-2

Table 26-4. Module Configuration Register Description (continued)

Field	Description
24 FRZ_ACK	This read-only bit indicates that FlexCAN is in freeze mode and its prescaler is stopped. The freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered freeze mode. If freeze mode request is negated, then this bit is cleared after the FlexCAN prescaler is running again. If freeze mode is requested while FlexCAN is in any of the low-power modes, then the FRZ_ACK bit is only set when the low-power mode is exited. See Section 26.4.9.1, “Freeze Mode,” for more information. 0 FlexCAN not in freeze mode, prescaler running 1 FlexCAN in freeze mode, prescaler stopped
23 SUPV	This bit configures some of the FlexCAN registers to be either in supervisor or unrestricted memory space. The registers affected by this bit are marked as S/U in the access type column of Table 26-2. Reset value of this bit is ‘1’, so the affected registers start with supervisor access restrictions. 0 Affected registers are in unrestricted memory space 1 Affected registers are in supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location
22 SLF_WAK	This bit enables the self wake-up feature when FlexCAN is in doze mode or stop mode. If this bit is set to 1 before FlexCAN enters doze mode or stop mode, then FlexCAN looks for a recessive-to-dominant transition on the bus during these modes. If a recessive-to-dominant transition is detected during doze mode, FlexCAN resumes its clocks and, if enabled to do so, generates a wake-up interrupt to the ARM. If a recessive-to-dominant transition is detected during stop mode, then FlexCAN generates, if enabled to do so, a wake-up interrupt to the ARM so that it can resume the clocks globally. This bit cannot be written while the module is in doze mode or stop mode. 0 FlexCAN self wake-up feature is disabled 1 FlexCAN self wake-up feature is enabled
21 WRN_EN	When set to 1, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the error and status register. If WRN_EN is cleared, the TWRN_INT and RWRN_INT flags are always zero, independent of the values of the error counters, and no warning interrupt is ever generated. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters. 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96.
20 LPM_ACK	This read-only bit indicates that FlexCAN is either in disable mode, doze mode or stop mode. Either of these low-power modes cannot be entered until all current transmission or reception processes have finished, so the ARM can poll the LPM_ACK bit to know when FlexCAN has actually entered low-power mode. See Section 26.4.9.2, “Module Disable Mode,” Section 26.4.9.3, “Doze Mode,” and Section 26.4.9.4, “Stop Mode,” for more information. 0 FlexCAN not in any of the low-power modes 1 FlexCAN is either in disable mode, doze mode or stop mode
19 WAK_SRC	This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake-up. See Section 26.4.9.3, “Doze Mode,” and Section 26.4.9.4, “Stop Mode,” for more information. 0 FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1 FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus
18 DOZE	This bit defines whether FlexCAN is allowed to enter low-power mode when doze mode is requested at the core level. This bit is automatically reset when FlexCAN wakes up from doze mode upon detecting activity on the CAN bus (self wake-up enabled). 0 FlexCAN is not enabled to enter low-power mode when doze mode is requested 1 FlexCAN is enabled to enter low-power mode when doze mode is requested
17 SRX_DIS	This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is set to 1, frames transmitted by the module are not stored in any message buffer, regardless if the message buffer is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal is generated due to the frame reception. 0 Self reception enabled 1 Self reception disabled

Table 26-4. Module Configuration Register Description (continued)

Field	Description
16 BCC	This bit is provided to support backwards compatibility with previous FlexCAN versions. When this bit is cleared, the following configuration is applied: • Individual Rx ID masking is disabled (for cores supporting this feature). Instead of individual ID masking per message buffer, FlexCAN uses the legacy masking scheme with RXGMASK, RX14MASK and RX15MASK. • The reception queue feature is disabled. Upon receiving a message, if the first message buffer with a matching ID that is found is still occupied by a previous unread message, FlexCAN does not look for another matching message buffer. It overrides this message buffer with the new message and set the CODE field to '0110' (OVERRUN). Upon reset this bit is cleared, allowing legacy software to work without modification. 0 Individual Rx masking and queue feature are disabled. 1 Individual Rx masking and queue feature are enabled.
15–14	Reserved
13 LPRIO_EN	This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11 bits for standard frames and 29 bits for extended frames. 0 Local priority disabled 1 Local priority enabled
12 AEN	This bit is supplied for backwards compatibility reasons. When set to 1, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. 0 Abort disabled 1 Abort enabled
11–10	Reserved
9–8 IDAM	This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown below. All elements of the table are configured at the same time by this field (they are all the same format). See Section 26.3.3.2, "Rx FIFO Description." 00 Format A One full ID (standard or extended) per filter element. 01 Format B Two full standard IDs or two partial 14-bit extended IDs per filter element 10 Format C Four partial 8-bit IDs (standard or extended) per filter element. 11 Format D All frames rejected.
7–6	Reserved
5–0 MAXMB	This 6-bit field defines the maximum number of message buffers that take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 message buffer configuration. Change this field only while the module is in freeze mode. The maximum number of message buffers in use is equal to MAXMB + 1. MAXMB must be programmed with a value smaller or equal to the number of available message buffers, otherwise, FlexCAN does not transmit or receive frames.

26.3.2.2 Control Register (CTRL)

The CTRL register, shown in Figure 26-4, is defined for specific FlexCAN control features related to the CAN bus, such as bit rate, programmable sampling point within an Rx bit, loopback mode, listen-only mode, bus-off recovery behavior and interrupt enabling (bus-off, error, warning). It also determines the division factor for the clock prescaler. Most of the fields in this register can only be changed while the module is in disable mode or in freeze mode. Exceptions are the BOFF_MSK, ERR_MSK, TWRN_MSK, RWRN_MSK and BOFF_REC bits, that can be accessed at any time.

Offset 0x0004 (CTRL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BOFF_ MSK	ERR_ MSK	CLK_SRC	LPB	TWRN_ MSK	RWRN_ MSK	0	0	SMP	BOFF_ REC	TSYN	LBUF	LOM	PROPSEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-4. Control Register (CTRL)

Table 26-5. Control Register Description

Field	Description
31–24 PRESDIV	This 8-bit field defines the ratio between the CPI clock frequency and the serial clock (SCLK) frequency. The SCLK period defines the time quantum of the CAN protocol. For the reset value, the SCLK frequency is equal to the CPI clock frequency. The maximum value of this register is 0xFF, that gives a minimum SCLK frequency equal to the CPI clock frequency divided by 256. For more information see Section 26.4.8.4, “Protocol Timing.” SCLK frequency = CPI clock frequency / (PRESDIV + 1)
23–22 RJW	This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one re-synchronization. The valid programmable values are 0–3. Resync Jump Width = RJW + 1.
21–19 PSEG1	This 3-bit field defines the length of phase buffer segment 1 in the bit time. The valid programmable values are 0–7. Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.
18–16 PSEG2	This 3-bit field defines the length of phase buffer segment 2 in the bit time. The valid programmable values are 1–7. Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
15 BOFF_MSK	This bit provides a mask for the bus-off interrupt. 0 Bus-off interrupt disabled 1 Bus-off interrupt enabled
14 ERR_MSK	This bit provides a mask for the error interrupt. 0 Error interrupt disabled 1 Error interrupt enabled

Table 26-5. Control Register Description (continued)

Field	Description
13 CLK_SRC	This bit selects the clock source to the CAN protocol interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the SCLK (SCLK). In order to guarantee reliable operation, this bit must only be changed while the module is in disable mode. See Section 26.4.8.4, “Protocol Timing,” for more information. 0 The CAN engine clock source is the oscillator clock (24.576 MHz) 1 The CAN engine clock source is the bus clock (66.5 MHz)
12 LPB	This bit configures FlexCAN to operate in loopback mode. In this mode, FlexCAN performs an internal loopback that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input signal is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loopback disabled 1 Loopback enabled
11 TWRN_MSK	This bit provides a mask for the Tx warning interrupt associated with the TWRN_INT flag in the error and status register. This bit has no effect if the WRN_EN bit in the MCR is cleared and it is read as zero when WRN_EN is cleared. 0 Tx warning interrupt disabled 1 Tx warning interrupt enabled
10 RWRN_MSK	This bit provides a mask for the Rx warning interrupt associated with the RWRN_INT flag in the error and status register. This bit has no effect if the WRN_EN bit in the MCR is cleared and it is read as zero when WRN_EN is cleared. 0 Rx warning interrupt disabled 1 Rx warning interrupt enabled
9–8	Reserved
7 SMP	This bit defines the sampling mode of CAN bits at the Rx input. 0 Just one sample is used to determine the bit value 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used
6 BOFF_REC	This bit defines how FlexCAN recovers from the bus-off state. If this bit is cleared, automatic recovering from the bus-off state occurs according to the CAN Specification 2.0B. If the bit is set to 1, automatic recovering from bus-off is disabled and the module remains in the bus-off state until the bit is cleared by the user. If the bit is cleared before 128 sequences of 11 recessive bits are detected on the CAN bus, then bus-off recovery happens as if the BOFF_REC bit had never been set to 1. If the bit is cleared after 128 sequences of 11 recessive bits have occurred, then FlexCAN re-synchronizes to the bus by waiting for 11 recessive bits before joining the bus. After it is cleared, the BOFF_REC bit can be set to 1 during bus-off, but it is only effective the next time the module enters bus-off. If BOFF_REC was cleared when the module entered bus-off, setting it to 1 during bus-off is not effective for the current bus-off recovery. 0 Automatic recovering from the bus-off state enabled, according to CAN Specification 2.0 part B 1 Automatic recovering from the bus-off state disabled
5 TSYN	This bit enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (for example, global network time). If the FEN bit in the MCR is set (FIFO enabled), message buffer 8 is used for timer synchronization instead of message buffer 0. 0 Timer Sync feature disabled 1 Timer Sync feature enabled

Table 26-5. Control Register Description (continued)

Field	Description
4 LBUF	This bit defines the ordering mechanism for message buffer transmission. When set to 1, the LPRIO_EN bit does not affect the priority arbitration. 0 Buffer with highest priority is transmitted first 1 Lowest number buffer is transmitted first
3 LOM	This bit configures FlexCAN to operate in listen-only mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN error passive mode [Ref. 1]. Only messages acknowledged by another CAN station are received. If FlexCAN detects a message that has not been acknowledged, it flags a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. 0 Listen-only mode is deactivated 1 FlexCAN module operates in listen-only mode
2–0 PROP_SEG	This 3-bit field defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. Propagation segment time = (PROPSEG + 1) * Time-Quantum, where Time-Quantum = one SCLK period.

[1] One time quantum is equal to the SCLK period.

26.3.2.3 Free-Running Timer (TIMER)

The TIMER register, shown in Figure 26-5, represents a 16-bit free-running counter that can be read and written by the ARM. The timer starts at 0x0000 upon reset, counts linearly to 0xFFFF, then wraps back to 0x0000.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments each time a bit is received or transmitted. When there is no message on the bus, it counts using the previously-programmed baud rate. During freeze mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the time stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data takes some time to actually be written to the register. If desired, software can poll the register to discover when the data was actually written.

Offset 0x0008 (TIMER) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-5. Free-Running Timer (TIMER)

26.3.2.4 Rx Global Mask (RXGMASK)

The RXGMASK register, shown in Figure 26-6, is provided for legacy support and for low cost MCUs that do not have the individual masking per message buffer feature. For MCUs supporting individual masks per message buffer, setting the BCC bit in the MCR causes the RXGMASK register to have no effect on the module operation. For MCUs not supporting individual masks per message buffer, this register is always effective.RXGMASK is used as acceptance mask for all Rx message buffers, excluding message buffers 14–15, which have individual mask registers. When the FEN bit in the MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 6–7, which have individual masks. Setting the BCC bit in MCR causes the RXGMASK register to have no effect on the module's operation.

The contents of this register must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

Offset 0x0010 (RXGMASK)
0x0014 (RX14MASK)
0x0018 (RX15MASK)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 26-6. Rx Global Mask Register (RXGMASK)

Table 26-6. Rx Global Mask Register Description

Field	Description
31–0 MI31–MI0	For normal Rx message buffers, the mask bits affect the ID filter programmed on the message buffer. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

26.3.2.5 Rx 14 Mask (RX14MASK)

RX14MASK is used as acceptance mask for the identifier in message buffer 14. When the FEN bit in the MCR is set (FIFO enabled), the RXG14MASK also applies to element 6 of the ID filter table. Setting the BCC bit in the MCR causes the RX14MASK register to have no effect on the module operation.

This register has the same structure as RXGMASK. It must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

- Address offset: 0x0014
- Reset value: 0xFFFF_FFFF

26.3.2.6 Rx 15 Mask (RX15MASK)

When the BCC bit is cleared, RX15MASK is used as acceptance mask for the identifier in message buffer 15. When the FEN bit in the MCR is set (FIFO enabled), the RXG14MASK also applies to element 7 of the ID filter table. Setting the BCC bit in the MCR causes the RX15MASK register to have no effect on the module operation

.This register has the same structure as the RXGMASKr. It must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

- Address offset: 0x0018
- Reset value: 0xFFFF_FFFF

26.3.2.7 Error Counter Register (ECR)

The ECR register, shown in Figure 26-7, has two 8-bit fields which reflect the value of the transmit error counter (tx_err_counter field) and receive error counter (rx_err_counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read only except in freeze mode, where they can be written by the ARM.

Writing to the error counter register while in freeze mode is an indirect operation. The data is first written to an auxiliary register, and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data takes some time to actually be written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, for example, by transmitting an “error active” or “error passive” flag, delaying its transmission start time (“error passive”), and avoiding any influence on the bus when in the bus-off state. The following are the basic rules for FlexCAN bus state transitions:

- If the value of tx_err_counter or rx_err_counter increases to exceed 127, the FLT_CONF field in the error and status register is updated to reflect “error passive” state.
- If the FlexCAN state is “error passive,” and either tx_err_counter or rx_err_counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the error and status register is updated to reflect “error active” state.
- If the value of tx_err_counter increases to be greater than 255, the FLT_CONF field in the error and status register is updated to reflect the bus-off state, and an interrupt is issued. The value of tx_err_counter is then reset to zero.
- If FlexCAN is in the bus-off state, then tx_err_counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, tx_err_counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the tx_err_counter. when tx_err_counter reaches the value of 128, the FLT_CONF field in the error and status register is updated to be “error active” and both error counters are reset to zero. at any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the tx_err_counter value.
- If during system start-up, only one node is operating, then its tx_err_counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the error and status register). After the transition to “error passive” state, the tx_err_counter does not increment anymore by acknowledge errors. Therefore the device never goes to the bus-off state.
- If the rx_err_counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to “error active” state.

Offset 0x001C (ECR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	rx_err_counter								tx_err_counter							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-7. Error Counter Register (ECR)

26.3.2.8 Error and Status Register (ESR)

The ESR register, shown in Figure 26-8, reflects various error conditions, some general status of the device and it is the source of four interrupts to the ARM. The reported error conditions are those that occurred since the last time the ARM read this register. The ARM read action clears bits. Bits are status bits.

Most bits in this register are read-only, except TWRN_INT, RWRN_INT, BOFF_INT, WAK_INT and ERR_INT, which are interrupt flags that can be cleared by writing 1 to them (writing ‘0’ has no effect). See Section 26.4.10, “Interrupts,” for more details.

Offset 0x0020 (ESR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TWRN_INT	RWRN_INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FRM_ERR	STF_ERR	TX_WRN	RX_WRN	IDLE	TXRX	FLT_CONF		0	BOFF_INT	ERR_INT	WAK_INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-8. Error and Status Register (ESR)

Table 26-7. Error and Status Register Description

Field	Description
31–18	Reserved
17 TWRN_INT	If the WRN_EN bit in the MCR is set to 1, the TWRN_INT bit is set when the TX_WRN flag transition from '0' to '1', meaning that the Tx error counter reached 96. If the corresponding mask bit in the control register (TWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. Writing '0' has no effect. 0 No Tx error counter transition detected 1 The Tx error counter has transitioned from < 96 to ≥ 96
16 RWRN_INT	If the WRN_EN bit in the MCR is set to 1, the RWRN_INT bit is set when the RX_WRN flag transition from '0' to '1', meaning that the Rx error counters reached 96. If the corresponding mask bit in the control register (RWRN_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to '1'. Writing '0' has no effect. 0 No Rx error counter transition detected 1 The Rx error counter has transitioned from < 96 to ≥ 96
15 BIT1_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No bit inconsistency detected 1 At least one bit sent as recessive is received as dominant This bit is not set by a transmitter in the case of an arbitration field or ACK slot, or in the case of a node sending a passive error flag that detects dominant bits.
14 BIT0_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No bit inconsistency detected 1 At least one bit sent as dominant is received as recessive
13 ACK_ERR	This bit indicates that an acknowledge (ACK) error has been detected by the transmitter node; that is, a dominant bit has not been detected during the ACK SLOT. 0 No ACK error detected 1 An ACK error occurred since last read of this register
12 CRC_ERR	This bit indicates that a CRC error has been detected by the receiver node, in other words the calculated CRC is different from the received. 0 No CRC error detected 1 A CRC error occurred since last read of this register.
11 FRM_ERR	This bit indicates that a form error has been detected by the receiver node, in other words a fixed-form bit field contains at least one illegal bit. 0 No form error detected 1 A form error occurred since last read of this register
10 STF_ERR	This bit indicates that a stuffing error has been detected. 0 No stuffing error detected 1 A stuffing error occurred since last read of this register
9 TX_WRN	This bit indicates when repetitive errors are occurring during message transmission. 0 Repetitive errors not detected 1 tx_err_counter ≥ 96
8 RX_WRN	This bit indicates when repetitive errors are occurring during message reception. 0 Repetitive errors not detected 1 rx_err_counter ≥ 96
7 IDLE	This bit indicates when CAN bus is in IDLE state. 0 CAN bus is not in IDLE state 1 CAN bus is now in IDLE state

Table 26-7. Error and Status Register Description (continued)

Field	Description
6 TXRX	This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is set to 1. 0 FlexCAN is receiving a message (IDLE=0) 1 FlexCAN is transmitting a message (IDLE=0)
5–4 FLT_CONF	This 2-bit field indicates the confinement state of the FlexCAN module, as follows:. 00 Error active 01 Error passive 1x Bus off If the LOM bit in the control register is set to 1, the FLT_CONF field indicates “Error Passive”. Since the control register is not affected by soft reset, the FLT_CONF field is not affected by soft reset if the LOM bit is set to 1.
3	Reserved
2 BOFF_INT	This bit is set when FlexCAN enters the bus-off state. If the corresponding mask bit in the control register (BOFF_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing a 1 to it. Writing 0 has no effect. 0 FlexCAN has not entered the bus-off state 1 FlexCAN module entered the bus-off state
1 ERR_INT	This bit indicates that at least one of the error bits is set. If the corresponding mask bit in the control register (ERR_MSK) is set, an interrupt is generated to the ARM. This bit is cleared by writing it to ‘1’.Writing ‘0’ has no effect. 0 No error bits set in the error and status register 1 Indicates setting of any error bit in the error and status register
0 WAK_INT	When FlexCAN is in doze mode or stop mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR is set, an interrupt is generated to the ARM. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect. 0 No recessive to dominant transition detected 1 Indicates a recessive to dominant transition received on the CAN bus when the FlexCAN module is in doze mode or stop mode

26.3.2.9 Interrupt Masks 2 Register (IMASK2)

The IMASK2 register, shown in Figure 26-9, allows any number of a range of 32 message buffer interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFLAG2 bit is set).

Offset 0x0024 (IMASK2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF 63M	BUF 62M	BUF 61M	BUF 60M	BUF 59M	BUF 58M	BUF 57M	BUF 56M	BUF 55M	BUF 54M	BUF 53M	BUF 52M	BUF 51M	BUF 50M	BUF 49M	BUF 48M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF 47M	BUF 46M	BUF 45M	BUF 44M	BUF 43M	BUF 42M	BUF 41M	BUF 40M	BUF 39M	BUF 38M	BUF 37M	BUF 36M	BUF 35M	BUF 34M	BUF 33M	BUF 32M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-9. Interrupt Masks 2 Register (IMASK2)

Table 26-8. Interrupt Masks 2 Register Description

Field	Description
31–0 BUF63M–BUF32M	Each bit enables or disables the respective FlexCAN message buffer interrupt (message buffers 32–63). 0 The corresponding buffer interrupt is disabled 1 The corresponding buffer interrupt is enabled

NOTE

Setting or clearing a bit in the IMASK2 register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set.

26.3.2.10 Interrupt Masks 1 Register (IMASK1)

The IMASK1 register, shown in Figure 26-10, enables or disables any number of a range of 32 message buffer interrupts. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFLAG1 bit is set).

Offset 0x0028 (IMASK1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF 31M	BUF 30M	BUF 29M	BUF 28M	BUF 27M	BUF 26M	BUF 25M	BUF 24M	BUF 23M	BUF 22M	BUF 21M	BUF 20M	BUF 19M	BUF 18M	BUF 17M	BUF 16M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF 15M	BUF 14M	BUF 13M	BUF 12M	BUF 11M	BUF 10M	BUF 9M	BUF 8M	BUF 7M	BUF 6M	BUF 5M	BUF 4M	BUF 3M	BUF 2M	BUF 1M	BUF 0M
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-10. Interrupt Masks 1 Register (IMASK1)

Table 26-9. Interrupt Masks 1 Register Description

Field	Description
31–0 BUF31M–BUF0M	Each bit enables or disables the respective FlexCAN message buffer interrupt (message buffers 0–31). 0 The corresponding buffer interrupt is disabled 1 The corresponding buffer interrupt is enabled

NOTE

Setting or clearing a bit in the IMASK1 register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set.

26.3.2.11 Interrupt Flags 2 Register (IFLAG2)

The IFLAG2 register, shown in Figure 26-11, defines the flags for 32 message buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1; writing 0 has no effect.

When the AEN bit in the MCR is set (abort enabled), while the IFLAG2 bit is set for a message buffer configured as Tx, ARM write access to the corresponding message buffer is blocked.

Offset 0x002C (IFLAG2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF 63I	BUF 62I	BUF 61I	BUF 60I	BUF 59I	BUF 58I	BUF 57I	BUF 56I	BUF 55I	BUF 54I	BUF 53I	BUF 52I	BUF 51I	BUF 50I	BUF 49I	BUF 48I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF 47I	BUF 46I	BUF 45I	BUF 44I	BUF 43I	BUF 42I	BUF 41I	BUF 40I	BUF 39I	BUF 38I	BUF 37I	BUF 36I	BUF 35I	BUF 34I	BUF 33I	BUF 32I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-11. Interrupt Flags 2 Register (IFLAG2)

Table 26-10. Interrupt Flags 2 Register Description

Field	Description
31–0 BUF63I–BUF32I	Each bit flags the respective FlexCAN message buffer interrupt (message buffer 32–63). 0 No such occurrence 1 The corresponding buffer has successfully completed transmission or reception

26.3.2.12 Interrupt Flags 1 Register (IFLAG1)

The IFLAG1 register, shown in Figure 26-12, defines the flags for 32 message buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1 to it. Writing 0 has no effect.

Setting the abort enabled (AEN) bit in the MCR while the IFLAG1 bit is set for a message buffer configured as Tx blocks the ARM's write access to the corresponding message buffer.

Setting the FIFO enable (FEN) bit in the MCR changes the function of the 8 least significant interrupt flags (BUF7I–BUF0I) to support the FIFO's operation. BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO; BUF4I–BUF0I are not used.

Offset 0x0030 (IFLAG1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF 31I	BUF 30I	BUF 29I	BUF 28I	BUF 27I	BUF 26I	BUF 25I	BUF 24I	BUF 23I	BUF 22I	BUF 21I	BUF 20I	BUF 19I	BUF 18I	BUF 17I	BUF 16I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF 15I	BUF 14I	BUF 13I	BUF 12I	BUF 11I	BUF 10I	BUF 9I	BUF 8I	BUF 7I	BUF 6I	BUF 5I	BUF 4I	BUF 3I	BUF 2I	BUF 1I	BUF 0I
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-12. Interrupt Flags 1 Register (IFLAG1)

Table 26-11. Interrupt Flags 1 Register Description

Field	Description
31–8 BUF31I–BUF8I	Each bit flags the respective FlexCAN message buffer interrupt (message buffers 8–31). 0 No such occurrence 1 The corresponding message buffer has successfully completed transmission or reception
7 BUF7I	If the FIFO is not enabled, this bit flags the interrupt for message buffer 7. If the FIFO is enabled, this flag indicates an overflow condition in the FIFO (frame lost because FIFO is full). 0 No such occurrence 1 Message buffer 7 completed transmission/reception or FIFO overflow
6 BUF6I	If the FIFO is not enabled, this bit flags the interrupt for message buffer 6. If the FIFO is enabled, this flag indicates that 4 out of 6 buffers of the FIFO are already occupied (FIFO almost full). 0 No such occurrence 1 Message buffer 6 completed transmission/reception or FIFO almost full

Table 26-11. Interrupt Flags 1 Register Description (continued)

Field	Description
5 BUF5I	If the FIFO is not enabled, this bit flags the interrupt for message buffer 5. If the FIFO is enabled, this flag indicates that at least one frame is available to be read from the FIFO. 0 No such occurrence 1 Message buffer 5 completed transmission/reception or frames available in the FIFO
4–0 BUF4I–BUF0I	If the FIFO is not enabled, these bits flag the interrupts for message buffers 0–4. If the FIFO is enabled, these flags are not used and must be considered as reserved locations. 0 No such occurrence 1 Corresponding message buffer completed transmission/reception

26.3.2.13 Rx Individual Mask Registers (RXIMR0–RXIMR63)

The RXIMR0–RXIMR63 registers, shown in Figure 26-13, are used as acceptance masks for ID filtering in Rx message buffers and the FIFO. If the FIFO is not enabled, one mask register is provided for each available message buffer, providing ID masking capability on a per message buffer basis. When the FIFO is enabled (FEN bit in the MCR is set), the first 8 mask registers apply to the 8 elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular message buffers, starting from message buffer 8.

The individual Rx mask registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception. Furthermore, they can only be accessed by the ARM while the module is in freeze mode. Outside of freeze mode, write accesses are blocked and read accesses return “all zeros”. Furthermore, if the BCC bit in the MCR is cleared, any read or write operation to these registers results in an access error.

Offset 0x0880–0x08BF (RXIMR0–15)
0x08C0–0x08FF (RXIMR16–31)
0x0900–0x097F (RXIMR32–63)

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Figure 26-13. Rx Individual Mask Registers (RXIMR0 - RXIMR63)

Table 26-12. Rx Individual Mask Registers Description

Field	Description
31–0 MI31–MI0	For normal Rx message buffers, the mask bits affect the ID filter programmed on the message buffer. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 the corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

26.3.3 Buffer Descriptions

The address offset range 0x0080–0x047F is used for message buffers, which store CAN messages for transmission and reception. Table 26-13 shows a standard/extended message buffer (message buffer 0) memory map, using 16 bytes total (0x0080–0x008F space). Each message buffer comprises control and status (C/S), identifier, and data fields, as shown in Table 26-13.

Table 26-13. Message Buffer 0 (MB0) Memory Map

Address Offset	Message Buffer Field
0x0080	Control and Status (C/S)
0x0084	Identifier field
0x0088–0x008F	Data fields 0–7 (1 byte each)

Alternatively, when the MCR’s FEN bit is set, the memory area from 0x0080 to 0x00FF (which is normally occupied by message buffers 0 to 7) is used by the reception FIFO engine.

The following subsections describe the structure of the message buffers and the receive FIFO.

26.3.3.1 Message Buffer Description

Figure 26-14 shows the message buffer structure used in FlexCAN. Both extended and standard frames (29-bit and 11-bit identifiers, respectively) used in the CAN specification (Version 2.0, Part B) are represented.

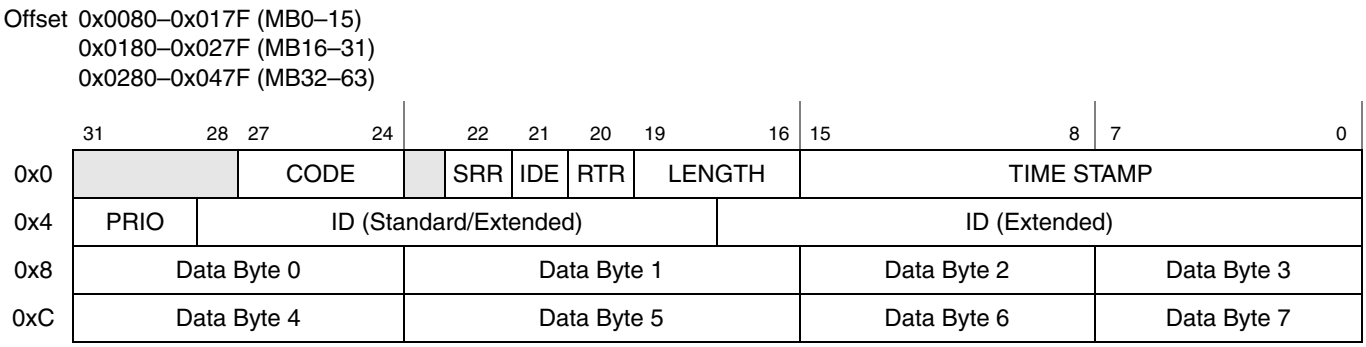


Figure 26-14. Message Buffer Structure

The fields shown in Figure 26-14 are described in Table 26-14.

Table 26-14. Message Buffer Field Descriptions

Field	Description
CODE	Message buffer code. Can be accessed (read or written to) by the ARM core or the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 26-15 and Table 26-16. See Section 26.4, "Functional Description," for additional information.
SRR	Substitute remote request. In the Tx buffers, for transmission of frames in extended format, the bit must be set to 1—a cleared bit is invalid. In the Rx buffers, the bit takes the value received on the CAN bus. A received value of 0 is interpreted as an arbitration loss. 0 Dominant is not a valid value for transmission in extended format frames 1 Recessive value is compulsory for transmission in extended format frames
IDE	ID extended bit. This bit identifies whether the frame format is standard or extended. 0 Extended frame format 1 Standard frame format
RTR	Remote transmission request. Used for requesting transmissions of a data frame. If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered a successful bit transmission. 0 Indicates the current message buffer has a data frame to be transmitted 1 Indicates the current message buffer has a remote frame to be transmitted
LENGTH	Length of data in bytes of the Rx or Tx data, which is located in offsets 0x0008 through 0x000F of the message buffer space (see Figure 29-2). In reception, this field is copied by the FlexCAN module from the data length code (DLC) field of the received frame. In transmission this field is written by the ARM, and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the value of LENGTH.
TIME STAMP	Free-running counter time stamp. This field is a copy of the free-running timer, captured for Tx and Rx frames when the identifier field first appears on the CAN bus.
PRIO	Local priority. Only used when the LPRIO_EN bit is set in the MCR. Only applies to Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Section 26.4.3, "Arbitration Process."
ID	Frame identifier. In standard frame format (Tx or Rx), only the 11 most significant bits are used for frame identification—the 18 least significant bits are ignored. In extended frame format (Tx or Rx), all bits are used for frame identification.
DATA	Data field. Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the ARM prepares the data field to be transmitted within the frame.

Table 26-15. Message Buffer Codes for Rx Buffers

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0000	INACTIVE: message buffer is not active.	—	Message buffer does not participate in the matching process.
0100	EMPTY: message buffer is active and empty.	0010	Message buffer participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.

Table 26-15. Message Buffer Codes for Rx Buffers (continued)

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0010	FULL: message buffer is full.	0010	The act of reading the C/S word followed by unlocking the message buffer does not make the code return to EMPTY. It remains FULL. If a new frame is written to the message buffer after the C/S word is read and the message buffer is unlocked, the code still remains FULL.
		0110	If the message buffer is FULL and a new frame is overwritten to this message buffer before the ARM has time to read it, the code is automatically updated to OVERRUN. See Section 26.4.5, “Matching Process,” for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the ARM reads the C/S word and then unlocks the message buffer, when a new frame is written to the message buffer the code returns to FULL.
		0110	If the code already indicates OVERRUN, yet another new frame must be written, the message buffer is overwritten again, and the code remains OVERRUN. See Section 26.4.5, “Matching Process,” for details about OVERRUN behavior.
0*nm*1[1]	BUSY: FlexCAN is updating the contents of the message buffer. The ARM must not access the message buffer.	0010	An EMPTY buffer was written with a new frame (*nm* was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

Note:

[1] For Tx message buffers (see Table 26-16), the BUSY bit should be ignored upon read, except when the AEN bit is set in the MCR.

Table 26-16. Message Buffer Code for Tx Buffers

RTR	Initial Tx Code	Code After Successful Transmission	Description
X	1000	—	INACTIVE: message buffer does not participate in the arbitration process.
X	1001	—	ABORT: message buffer was configured as Tx and ARM aborted the transmission. This code is only valid when the AEN bit in the MCR is set to 1. Message buffer does not participate in the arbitration process.
0	1100	1000	Transmit data frame once unconditionally. After transmission, the message buffer automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame once unconditionally. After transmission, the message buffer automatically becomes an Rx message buffer with the same ID.

Table 26-16. Message Buffer Code for Tx Buffers (continued)

RTR	Initial Tx Code	Code After Successful Transmission	Description
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This message buffer participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the message buffer. If a match occurs, this message buffer is allowed to participate in the current arbitration process and the CODE field is automatically updated to 0b1110 to allow the message buffer to participate in future arbitration runs. When the frame is eventually transmitted successfully, the code automatically returns to 0b1010 to restart the process.
0	1110	1010	This is an intermediate code automatically written to the message buffer by the MBM as a result of a match to a remote request frame. The data frame is transmitted once unconditionally and then the code automatically returns to 0b1010. The ARM can also write this code with the same effect.

26.3.3.2 Rx FIFO Description

When the MCR's FEN bit is set, the memory area from 0x0080 to 0x00FF (which is normally occupied by message buffers 0–7) is used by the reception FIFO engine. Figure 26-15 shows the Rx FIFO data structure. The region 0x0000-0x000C contains an message buffer structure which is the port through which the ARM reads data from the FIFO (the oldest frame received and not read yet). The region 0x0010-0x00DF is reserved for internal use of the FIFO engine. The region 0x00E0-0x00FF contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. Figure 26-16 shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the

MCR. All elements of the table must have the same format. See Section 26.4.7, “Rx FIFO,” for more information.

	31 28	27 24	22	21	20	19 16	15 8	7 0
0x0			SRR	IDE	RTR	LENGTH	TIME STAMP	
0x4		ID (Standard/Extended)					ID (Extended)	
0x8	Data Byte 0		Data Byte 1				Data Byte 2	Data Byte 3
0xC	Data Byte 4		Data Byte 5				Data Byte 6	Data Byte 7
0x10 to 0xDF	Reserved							
0xE0	ID Table 0							
0xE4	ID Table 1							
0xE8	ID Table 2							
0xEC	ID Table 3							
0xF0	ID Table 4							
0xF4	ID Table 5							
0xF8	ID Table 6							
0xFC	ID Table 7							

Figure 26-15. Rx FIFO Structure

A	REM	EXT	RXIDA (Standard = 29-19, Extended = 29-1)				
B	REM	EXT	RXIDB_0 (Standard = 29-19, Extended = 29-16)	REM	EXT	RXIDB_1 (Standard = 13-3, Extended = 13-0)	
C	RXIDC_0 (Std/Ext = 31-24)		RXIDC_1 (Std/Ext = 23-16)		RXIDC_2 (Std/Ext = 15-8)		RXIDC_3 (Std/Ext = 7-0)

Figure 26-16. ID Table 0–7

Table 26-17. Rx FIFO Field Descriptions

Field	Description
REM	Remote frame bit. Specifies if remote frames are accepted into the FIFO if they match the target ID. 0 Remote frames can be accepted and data frames are rejected 1 Remote frames are rejected and data frames can be accepted
EXT	Extended frame bit.Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID. 0 Extended frames can be accepted and standard frames are rejected 1 Extended frames are rejected and standard frames can be accepted
RXIDA	Rx Frame Identifier (Format A). Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (3 to 13)are used for frame identification. In the extended frame format, all bits are used.
RXIDB_0, RXIDB_1	Rx Frame Identifier (Format B). Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (3 to 13)are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.
RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3	Rx Frame Identifier (Format C). Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

26.4 Functional Description

26.4.1 Overview

The FlexCAN module is a CAN protocol engine with a flexible mailbox system for transmitting and receiving CAN frames. The mailbox system includes a set of up to 64 message buffers that store configuration and control data, time stamp, message ID and data (see Section 26.3.3.1, “Message Buffer Description”). The memory corresponding to the first 8 message buffers can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 8 extended IDs, or 16 standard IDs, or 32 8-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into message buffers that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the message buffer with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of message buffers to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the message buffer ordering.

A message buffer is said to be active at a given time if it can participate in the currently-active matching and arbitration algorithms. An Rx message buffer with a code of 0b0000 is inactive, as is a Tx message buffer with a 0b1000 or 0b1001 (see Table 26-15 and Table 26-16, respectively). In other cases, a message buffer is temporarily deactivated (that is, it does not participate in the current arbitration or matching run) when the ARM writes to the C/S field of that message buffer (see Section 26.4.6.2, “Message Buffer Deactivation”).

26.4.2 Transmit Process

In order to transmit a CAN frame, the ARM must prepare a message buffer for transmission by executing the following procedure:

1. The first step depends on the state of the message buffer:
   — If the message buffer is active (transmission pending), write an ABORT code (0b1001) to the CODE field of the control and status word to request an abortion of the transmission, then read back the CODE field and the interrupt flag register to check if the transmission was aborted (see Section 26.4.6.1, “Transmission Abort Mechanism”).
   — Alternatively if backwards compatibility is desired (AEN in MCR is cleared), write 0b1000 to the CODE field to inactivate the message buffer. However, the pending frame may be transmitted without notification (see Section 26.4.6.2, “Message Buffer Deactivation”).
2. Write the ID word.
3. Write the data bytes.
4. Write the LENGTH, CODE, and other control fields of the control and status word to activate the message buffer.

After the message buffer is activated by following this procedure, it participates in the arbitration process and eventually is transmitted according to its priority. At the end of the successful transmission, the value of the free running timer is written into the time stamp field, the CODE field in the control and status word is updated, a status flag is set in the interrupt flag register and an interrupt is generated if allowed by the corresponding interrupt mask register bit. The new CODE field after transmission depends on the code that was used to activate the message buffer (see Table 26-15 and Table 26-16 in Section 26.3.3.1, “Message Buffer Description”).

When the abort feature is enabled (AEN in the MCR is set to 1), after the interrupt flag is asserted for a message buffer configured as transmit buffer, the message buffer is blocked: therefore the ARM is not able to update it until the interrupt flag be cleared by ARM. This means that the ARM must clear the corresponding interrupt flag before starting to prepare this message buffer for a new transmission or reception.

26.4.3 Arbitration Process

The arbitration process is an algorithm executed by the message buffer management submodule (MBM) that scans the whole message buffer memory looking for the highest priority message to be transmitted. All message buffers programmed as transmit buffers are scanned to find the lowest ID or the lowest message buffer number or the highest priority, depending on the LBUF and LPRIO_EN bits on the control register. (If LBUF is cleared, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.) The arbitration process is triggered by any of the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame

- During intermission, if the winner message buffer defined in a previous arbitration was deactivated, or if there was no message buffer to transmit, but the ARM wrote to the C/S word of any message buffer after the previous arbitration finished
- When the MBM is in the idle or bus-off state and the ARM writes to the C/S word of any message buffer
- Upon leaving freeze mode

When LBUF is set to 1, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both cleared, the message buffer with the lowest ID is transmitted first but. If LBUF is cleared and LPRIO_EN is set to 1, the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is done based on the full 32-bit ID and the PRIO bits define which message buffer is transmitted first: therefore message buffers with PRIO = 0b000 have higher priority. If two or more message buffers have the same priority, the regular ID determines the priority of transmission. If two or more message buffers have the same priority (3 extra bits) and the same regular ID, the lowest message buffer is transmitted first.

After the highest priority message buffer is selected, it is transferred to a temporary storage space called the serial message buffer (SMB), which has the same structure as a normal message buffer but is not user accessible. After this transfer, write access to the corresponding message buffer is blocked (if the AEN bit in the MCR is set to 1). Write access is restored by any of the following events:

- After the message buffer is transmitted
- FlexCAN enters HALT or BUS OFF
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (data length code) value is bigger.

26.4.4 Receive Process

To be able to receive CAN frames into the mailbox message buffers, the ARM must prepare one or more message buffers for reception by executing the following steps:

1. The first step depends on the state of the message buffer:
   - If the message buffer has a pending transmission, write an ABORT code (0b1001) to the CODE field of the control and status word to request an abortion of the transmission, then read back the CODE field and the interrupt flag register to check if the transmission was aborted (see Section 26.4.6.1, “Transmission Abort Mechanism”).
   - Alternatively if backwards compatibility is desired (AEN in the MCR is cleared), write 0b1000 to the CODE field to inactivate the message buffer. However, the pending frame may be transmitted without notification (see Section 26.4.6.2, “Message Buffer Deactivation”).
   - If the message buffer already programmed as a receiver, write 0b0000 to the CODE field of the control and status word to keep the message buffer inactive.
2. Write the ID word
3. Write 0b0100 to the CODE field of the control and status word to activate the message buffer

After the message buffer is activated, it is able to receive frames that match the programmed ID. At the end of a successful reception, the message buffer is updated by the MBM as follows:

1. The value of the free running timer is written into the time stamp field
2. The received ID, data (8 bytes at most), and length fields are stored
3. The CODE field in the control and status word is updated (see Table 26-15 and Table 26-16 in Section 26.3.3.1, “Message Buffer Description”)
4. A status flag is set in the interrupt flag register and an interrupt is generated if allowed by the corresponding interrupt mask register bit

Upon receiving the message buffer interrupt, the ARM services the received frame using the following procedure:

1. Read the control and status word (mandatory—activates an internal lock for this buffer)
2. Read the ID field (optional—needed only if a mask was used)
3. Read the data field
4. Read the free-running timer (optional—releases the internal lock)

The ARM’s read of the control and status word is necessary to assure data coherence (see Section 26.4.6, “Data Coherence”). After reading the control and status word, if the BUSY bit is set in the CODE field then the ARM postpones the access to the message buffer until this bit is cleared.

Reading the free-running timer is not mandatory. If not executed, the message buffer remains locked, unless the ARM reads the C/S word of another message buffer. Only one message buffer is locked at a time.

The ARM synchronizes to frame reception by the status flag bit for the specific message buffer in one of the interrupt flag registers and not by the CODE field of that message buffer. Polling the CODE field does not work because after a frame is received and the ARM services the message buffer (by reading the C/S word followed by unlocking the message buffer), the CODE field does not return to EMPTY. It remains FULL, as explained in Table 26-15. If the ARM tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the message buffer, then the message buffer is deactivated from any currently ongoing matching process. As a result, a newly-received frame matching the ID of that message buffer can possibly be lost.

NOTE

When polling, do not read directly the C/S word of the message buffers. Instead, read the interrupt flag registers.

The received ID field is always stored in the matching message buffer. The contents of the ID field in a message buffer can possibly change if the match was due to masking.
FlexCAN does receive frames transmitted by itself if there exists an Rx matching message buffer, provided the SRX_DIS bit in the MCR is cleared. If SRX_DIS is set to 1, FlexCAN does not store frames transmitted by itself in any message buffer, even if it contains a matching message buffer, and no interrupt flag or interrupt signal is generated due to the frame reception.

To receive CAN frames through the FIFO, the ARM must first enable and configure the FIFO during freeze mode (see Section 26.4.7, “Rx FIFO”). After receiving the frames available interrupt from FIFO, the ARM services the received frame using the following procedure:

1. Read the control and status word (optional—needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional—needed only if a mask was used)
3. Read the data field
4. Clear the frames available interrupt (mandatory—releases the buffer and allow the ARM to read the next FIFO entry)

26.4.5 Matching Process

The matching process is an algorithm executed by the MBM that scans the message buffer memory looking for Rx message buffers programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the 8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other message buffers are scanned. In the event that the FIFO is full, the matching algorithm always looks for a matching message buffer outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary message buffer called serial message buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular message buffers, the contents of the SMB are transferred to the FIFO or to the matched message buffer during the sixth bit of the end-of-frame field of the CAN protocol. This operation is called “move-in.” If any protocol error (such as CRC or ACK) is detected, than the move-in operation is not performed.

For the regular mailbox message buffers, a message buffer is said to be free to receive a new frame if all of the following conditions are satisfied:

- The message buffer is not locked (see Section 26.4.6.3, “Message Buffer Lock Mechanism”)
- The CODE field is EMPTY; alternatively, the CODE field is FULL or OVERRUN but the ARM has already serviced the message buffer (read the C/S word and then unlocked the message buffer)

If the first message buffer with a matching ID is not free to receive the new frame, then the matching algorithm keeps looking for another free message buffer until it finds one. If no free message buffer can be found, then the last matching message buffer is overwritten (unless it is locked) and the CODE field is set to OVERRUN (see Table 26-15 and Table 26-16). If the last matching message buffer is locked, then the new message remains in the SMB, waiting for the message buffer to be unlocked (see Section 26.4.6.3, “Message Buffer Lock Mechanism”).

Suppose for example that the FIFO is disabled and there are two message buffers with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these message buffers are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in the second message buffer. The code of this message buffer is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds the second message buffer again, but it is not free to receive, so it keeps looking, finds the fifth message buffer, and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching message buffers that are free to receive, so it overwrites the last matched message buffer (fifth message buffer) and sets the CODE field of that message buffer to OVERRUN.

The ability to match the same ID in more than one message buffer can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the ARM to service the message buffers. By programming more than one message buffer with the same ID, received messages are queued into the message buffers. The ARM can examine the time stamp field of the message buffers to determine the order in which the messages arrived.

The matching algorithm described above can be changed to that used in previous versions of the FlexCAN module by clearing the BCC bit in the MCR. In this case, the matching algorithm stops at the first message buffer with a matching ID that it founds, whether this message buffer is free or not. As a result, the message queueing feature does not work if the BCC bit is cleared.

Matching to a range of IDs is possible by using ID acceptance masks. FlexCAN supports individual masking per message buffer. See Section 26.3.2.13, “Rx Individual Mask Registers (RXIMR0–RXIMR63).” During the matching algorithm, if a mask bit is set to 1 then the corresponding ID bit is compared. If the mask bit is cleared, the corresponding ID bit is “don’t care”. The individual mask registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is set to 1 and while the module is in freeze mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR is cleared.

26.4.6 Data Coherence

In order to maintain data coherence and ensure the proper operation of FlexCAN, the ARM must obey the rules described in Section 26.4.2, “Transmit Process,” and Section 26.4.4, “Receive Process.” Any ARM access to a message buffer structure within FlexCAN which does not comply with these rules causes FlexCAN to behave in an unpredictable way.

26.4.6.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the ARM if the transmission was aborted or if the frame could not be aborted and was transmitted instead. In order to maintain backwards compatibility, the abort mechanism must be explicitly enabled by setting the AEN bit in the MCR.

In order to abort a transmission, the ARM must write a specific abort code (0b1001) to the CODE field of the control and status word. When the abort mechanism is enabled, the active message buffers configured as transmission must be aborted first before they can be updated. If the abort code is written to an message buffer that is currently being transmitted, or to an message buffer that was already loaded into the SMB for transmission, the write operation is blocked and the message buffer is not deactivated, but the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into freeze mode

If none of conditions above are reached, the message buffer is transmitted correctly, the interrupt flag is set in the interrupt flag register and an interrupt to the ARM is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the CODE field, the interrupt flag is set in the interrupt flag register and an interrupt is (optionally) generated to the ARM.

If the ARM writes the abort code before the transmission begins internally, then the write operation is not blocked, therefore the message buffer is updated and no interrupt flag is set. In this way the ARM just needs to read the abort code to make sure the active message buffer was deactivated. Although the AEN bit is set and the ARM wrote the abort code, in this case the message buffer is deactivated and not aborted, because the transmission did not start yet. One message buffer is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

1. ARM writes 0b1001 into the CODE field of the C/S word
2. ARM reads the CODE field and compares it to the value that was written
3. If the CODE field that is read is different from the value that was written, the ARM must read the corresponding interrupt flag (IFLAG) to check if the frame was transmitted or it is being currently transmitted. If the corresponding IFLAG is set, the frame was transmitted. If the corresponding IFLAG is reset, the ARM must wait for it to be set, and then the ARM must read the CODE field to check if the message buffer was aborted (CODE=0b1001) or it was transmitted (CODE=0b1000).

26.4.6.2 Message Buffer Deactivation

Deactivation is mechanism provided to maintain data coherence when the ARM writes to the control and status word of active message buffers out of freeze mode. Any ARM write access to the control and status word of a message buffer causes that message buffer to be excluded from the transmit or receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is to maintain data coherence. The match/arbitration process scans the message buffers to decide which message buffer to transmit or receive. If the ARM updates the message buffer in the middle of a match or arbitration process, the data of that message buffer may no longer be coherent. Deactivation of the message buffer prevents this from happening.

Even with the coherency mechanism described above, writing to the control and status word of active message buffers when not in freeze mode can produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If message buffers are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx message buffer with a matching ID is deactivated during the matching process after it was scanned, then this message buffer is marked as invalid to receive the frame, and FlexCAN keeps looking for another matching message buffer within those not yet scanned. If no matching buffer is found, then the message is lost. Suppose, for example, that two message buffers have a matching ID to a received frame, and the user deactivated the first matching message buffer after FlexCAN has scanned the second. The received frame is lost even if the second matching message buffer was “free to receive”.

- If a Tx message buffer containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN looks for another winner within the message buffers that it has not scanned yet. Therefore, it can possibly transmit a message buffer with an ID that is not the lowest at the time, if a lower ID is present in one of the message buffers that was already scanned before the deactivation.
- There is a point in time before which the deactivation of a Tx message buffer causes it not to be transmitted (end of move-out). After this time, the message buffer is transmitted but no interrupt is issued and the CODE field is not updated. To avoid this situation, use the abort procedures described in Section 26.4.6.1, “Transmission Abort Mechanism.”

26.4.6.3 Message Buffer Lock Mechanism

Besides message buffer deactivation, FlexCAN has another data coherence mechanism for the receive process. When the ARM reads the control and status word of an “active not empty” Rx message buffer, FlexCAN assumes that the ARM wants to read the whole message buffer in an atomic operation, and thus it sets an internal lock flag for that message buffer. The lock is released when the ARM reads the free running timer (global unlock operation), or when it reads the control and status word of another message buffer. The message buffer locking is done to prevent a new frame to be written into the message buffer while the ARM is reading it.

NOTE

The locking mechanism only applies to Rx message buffers which have a code different than INACTIVE (0b0000) or EMPTY (0b0100). Tx message buffers cannot be locked. In previous FlexCAN versions, reading the C/S word locked the message buffer even if it was empty. This behavior is followed when the BCC bit is cleared.

Suppose, for example, that the FIFO is disabled and the second and the fifth message buffers of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two message buffers. Suppose now that the ARM decides to read message buffer number 5 and at the same time another message with the same ID is arriving. When the ARM reads the control and status word of message buffer number 5, this message buffer is locked. The new message arrives and the matching algorithm finds out that there are no “free to receive” message buffers, so it decides to override message buffer number 5. However, this message buffer is locked, so the new message cannot be written there. It remains in the SMB waiting for the message buffer to be unlocked, and only then is it written to the message buffer. If the message buffer is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there is no indication of lost messages either in the CODE field of the message buffer or in the error and status register.

While the message is being moved-in from the SMB to the message buffer, the BUSY bit on the CODE field is set to 1. If the ARM reads the control and status word and finds out that the BUSY bit is set, the ARM postpones accessing the message buffer until the BUSY bit is cleared.

NOTE

If the BUSY bit is set to 1 or if the message buffer is empty, then reading the control and status word does not lock the message buffer.

Deactivation takes precedence over locking. If the ARM deactivates a locked Rx message buffer, then its lock status is cleared and the message buffer is marked as invalid for the current matching round. Any pending message on the SMB is not transferred to the message buffer.

26.4.7 Rx FIFO

The receive-only FIFO is enabled by setting the FEN bit to 1 in the MCR. The reset value of this bit is zero to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first 8 message buffers (0x0080-0x00FF) is now reserved for use of the FIFO engine (see Section 26.3.3.2, “Rx FIFO Description”). Management of read and write pointers is done internally by the FIFO engine. The ARM can read the received frames sequentially, in the order they were received, by repeatedly accessing a message buffer structure at the beginning of the memory.

The FIFO can store up to 6 frames pending service by the ARM. An interrupt is sent to the ARM when new frames are available in the FIFO. Upon receiving the interrupt, the ARM must read the frame (accessing an message buffer in the 0x0080 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the message buffer in 0x0080 with the next frame in the queue, and then issue another interrupt to the ARM. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the ARM and subsequent frames are not accepted until the ARM creates space in the FIFO by reading one or more frames. A warning interrupt is also generated when 4frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of eight 32-bit registers that can be configured to one of the following formats (see also Section 26.3.3.2, “Rx FIFO Description”):

- Format A: 8 extended or standard IDs (including IDE and RTR)
- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

NOTE

A chosen format is applied to all 8 registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight individual mask registers (RXIMR0–RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular message buffers, starting from message buffer 8. If the BCC bit is cleared, then the FIFO filter table is affected by the legacy mask registers as follows: element 6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0–5) are affected by RXGMASK.

26.4.8 CAN Protocol Related Features

26.4.8.1 Remote Frames

The user can program a message buffer to be a remote request frame by writing the message buffer as transmit with the RTR bit set to 1. After the remote request frame is transmitted successfully, the message buffer becomes a receive message buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with CODE field 0b1010. If there is a matching ID, then this message buffer frame is transmitted. If the matching message buffer has the RTR bit set, then FlexCAN transmits a remote frame as a response.

A received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame must match.

In the case that a remote request frame is received and matches a message buffer, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx message buffer, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (the FEN bit in the MCR is set to 1), FlexCAN does not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the FIFO and presented to the ARM. For filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

26.4.8.2 Overload Frames

FlexCAN transmits overload frames when any of the following conditions on CAN bus are detected:

- Dominant bit in the first or second bit of intermission
- Dominant bit at the 7th (last) bit of end-of-frame field (Rx frames)
- Dominant bit at the 8th (last) bit of the error frame delimiter or the overload frame delimiter

26.4.8.3 Time Stamp

The value of the free-running timer is sampled at the beginning of the identifier field on the CAN bus, and is stored at the end of “move-in” in the time stamp field, providing network behavior with respect to time.

The free-running timer can be reset upon a specific frame reception, enabling network time synchronization. See TSYN description in Section 26.3.2.2, “Control Register (CTRL).”

26.4.8.4 Protocol Timing

Figure 26-17 shows the structure of the clock generation circuitry that feeds the CAN protocol interface (CPI) submodule. The clock source bit (CLK_SRC) in the CTRL register defines whether the internal clock is connected to the output of a crystal oscillator (oscillator clock) or to the peripheral clock (generally

from a PLL). In order to guarantee reliable operation, the clock source must be selected while the module is in disable mode (bit MDIS set in the MCR).

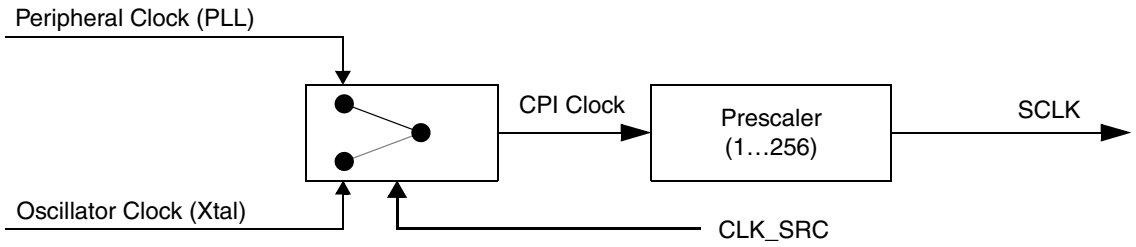


Figure 26-17. CAN Engine Clocking Scheme

The crystal oscillator clock must be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL-generated clocks.

The FlexCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The control register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. For more details see Section 26.3.2.2, “Control Register (CTRL).”

The PRESDIV field controls a prescaler that generates the SCLK, whose period defines the ‘time quantum’ used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(Prescaler\ \ value)}$$

A bit time is subdivided into three segments[1] (see Figure 26-18 and Figure 26-18):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time segment 1: This segment includes the propagation segment and the phase segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time segment 2: This segment represents phase segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL register (plus 1) to be 2–8 time quanta long

$$Bit\ \ Rate = \frac{f_{Tq}}{(number\ \ of\ \ Time\ \ Quanta)}$$

1. For further explanation of the underlying concepts see ISO/DIS 11519–1, Section 10.3. See also the Bosch CAN 2.0A/B protocol specification (September 1991) for bit timing.

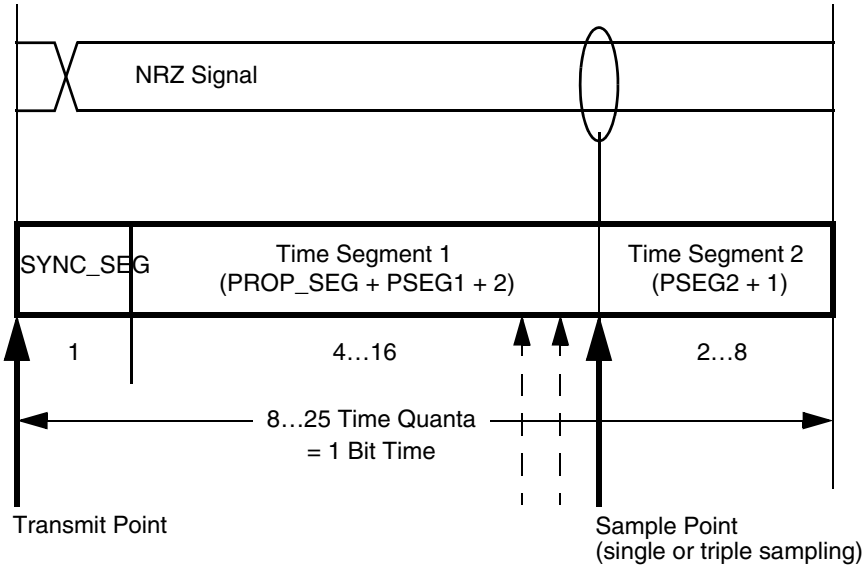


Figure 26-18. Segments Within the Bit Time

Table 26-18. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 26-19 gives an overview of the CAN-compatible segment settings and the related parameter values.

Table 26-19. CAN Standard-Compatible Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5…10	2	1…2
4…11	3	1…3
5…12	4	1…4
6…13	5	1…4
7…14	6	1…4
8…15	7	1…4
9…16	8	1…4

NOTE

It is the user's responsibility to ensure the bit time settings are compatible with the CAN standard. For bit time calculations, use an IPT (information processing time) of 2, which is the value implemented in the FlexCAN module.

26.4.8.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in Figure 26-19.

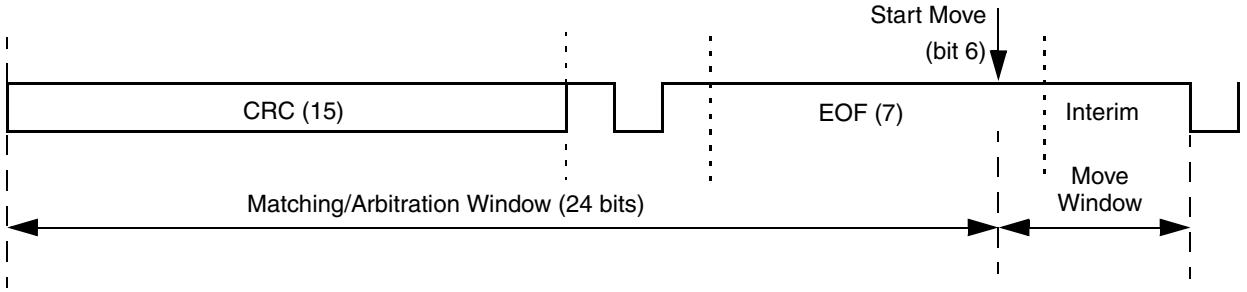


Figure 26-19. Arbitration, Match and Move Time Windows

When performing matching and arbitration, FlexCAN needs to scan the whole message buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 26-19
- The peripheral clock frequency cannot be smaller than the oscillator clock frequency, in other words the PLL cannot be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in Table 26-20

Table 26-20. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	Minimum Ratio
16	8
32	8
64	16

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency must be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 26-20 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 message buffers, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) must be at least 2. For prescaler factor equal

to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies must be at least 2.

26.4.9 Modes of Operation Detailed Descriptions

26.4.9.1 Freeze Mode

This mode is entered by setting the HALT bit to 1 in the MCR or when the core is put into debug mode. In both cases it is also necessary that the FRZ bit is set in the MCR and the module is not in any of the low-power modes (disable, doze, stop). When freeze mode is requested during transmission or reception, FlexCAN performs the following actions:

- Waits to be in either intermission, passive error, bus-off or idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores the Rx input signal and drives the Tx signal as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the error counter register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in the MCR

After requesting freeze mode, the user must wait for the FRZ_ACK bit to be set in the MCR before executing any other action—otherwise FlexCAN operation is unpredictable. In freeze mode, all memory mapped registers are accessible.

Freeze mode is exited in one of the following ways:

- ARM clears the FRZ bit in the MCR
- The core is removed from debug mode and/or the HALT bit is cleared

Once out of freeze mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

26.4.9.2 Module Disable Mode

This low-power mode is entered when the MDIS bit in the MCR is set to 1. If the module is disabled during freeze mode, it shuts down the clocks to the CPI and MBM submodules, sets the LPM_ACK bit and clears the FRZ_ACK bit. If the module is disabled during transmission or reception, FlexCAN performs the following actions:

- Waits to be in either the idle or bus-off state, or else waits for the third bit of intermission and then checks if it is recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input signal and drives its Tx signal as recessive
- Shuts down the clocks to the CPI and MBM submodules
- Sets the NOT_RDY and LPM_ACK bits in the MCR

The bus interface unit continues to operate, enabling the ARM to access memory mapped registers, except the free running timer, the error counter register and the message buffers, which cannot be accessed when

the module is in disable mode. Exiting from this mode is done by clearing the MDIS bit, which resumes the clocks and clears the LPM_ACK bit.

26.4.9.3 Doze Mode

This is a system low-power mode in which the ARM bus is kept alive and a global doze mode request is sent to all peripherals asking them to enter low-power mode. The DOZE bit in the MCR must be set to 1 in order for a global doze mode to trigger doze mode. If doze mode is triggered during freeze mode, FlexCAN shuts down the clocks to the CPI and MBM submodules, sets the LPM_ACK bit and clears the FRZ_ACK bit. If doze mode is triggered during transmission or reception, FlexCAN performs the following actions:

- Waits to be in either the idle or the bus-off state, or else waits for the third bit of intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input signal and drives its Tx signal as recessive
- Shuts down the clocks to the CPI and MBM submodules
- Sets the NOT_RDY and LPM_ACK bits in the MCR

The bus interface unit continues to operate, enabling the ARM to access memory mapped registers, except the free-running timer, the error counter register and the message buffers, which cannot be accessed in doze mode.

Doze mode is exited in one of the following ways:

- ARM removes the doze mode request
- ARM clears the DOZE bit of the MCR
- Self wake-up mechanism

In the self wake-up mechanism, if the SLF_WAK bit in the MCR was set at the time FlexCAN entered doze mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN clears the DOZE bit and resumes its clocks. It also sets the WAK_INT bit in the ESR and, if enabled by the WAK_MSK bit in the MCR, generates a wake-up interrupt to the ARM. FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it does not receive the frame that woke it up. Table 26-21 details the effect of SLF_WAK and WAK_MSK upon wake-up from doze mode.

Table 26-21. Wake-up from Doze Mode

SLF_WAK	WAK_MSK	FlexCAN Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	Yes	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in doze mode. See the WAK_SRC bit in Section 26.3.2.1, “Module Configuration Register

(MCR).” This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

26.4.9.4 Stop Mode

This is a system low-power mode in which all core clocks are stopped for maximum power savings. If FlexCAN receives the global stop mode request during freeze mode, it sets the LPM_ACK bit, clears the FRZ_ACK bit and then sends a stop acknowledge signal to the ARM, in order to shut down the clocks globally. If stop mode is requested during transmission or reception, FlexCAN performs the following actions:

- Waits to be in either the idle or bus-off state, or else waits for the third bit of intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input signal and drives its Tx signal as recessive
- Sets the NOT_RDY and LPM_ACK bits in the MCR
- Sends a Stop Acknowledge signal to the ARM, so that it can shut down the clocks globally

Stop mode is exited in one of the following ways:

- ARM resumes the clocks and removes the stop mode request
- ARM resumes the clocks and stop mode request as a result of the self wake-up mechanism

In the self wake-up mechanism, if the SLF_WAK bit in the MCR was set at the time FlexCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR and, if enabled by the WAK_MSK bit in the MCR, generates a wake-up interrupt to the ARM. Upon receiving the interrupt, the ARM resumes the clocks and remove the stop mode request. FlexCAN then waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it does not receive the frame that woke it up. Table 26-22 details the effect of SLF_WAK and WAK_MSK upon wake-up from stop mode. Wake-up from stop mode only works when both bits are set to 1.

Table 26-22. Wake-up from Stop Mode

SLF_WAK	WAK_MSK	Core Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	No	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in stop mode. See the WAK_SRC bit in Section 26.3.2.1, “Module Configuration Register (MCR). This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

26.4.10 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to ORed interrupts from message buffers, bus-off, error, Tx warning, Rx warning, and wake-up). Each of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the interrupt flag registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the ARM writes a 1 to it (unless another interrupt is generated at the same time).

NOTE

It must be guaranteed that the ARM only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions can cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (bit FEN of MCR is set to 1), the interrupts corresponding to message buffers 0–7 have a different behavior. Bit 7 of the IFLAG1 becomes the “FIFO Overflow” flag; bit 6 becomes the FIFO warning flag, bit 5 becomes the “Frames Available in FIFO flag” and bits 4–0 are unused. See Section 26.3.2.12, “Interrupt Flags 1 Register (IFLAG1),” for more information.

A combined interrupt for all message buffers is generated by an OR of all the interrupt sources from message buffer 0–7. This interrupt gets generated when any of the message buffers generates an interrupt. In this case the ARM must read the interrupt flag registers to determine which message buffer caused the interrupt.

The other five interrupt sources (bus-off, error, Tx warning, Rx warning, and wake up) generate interrupts like the message buffer ones, and can be read from the error and status register. The bus-off, error, Tx warning, and Rx warning interrupt mask bits are located in the control register, and the wake-up interrupt mask bit is located in the MCR.

26.5 Initialization/Application Information

This section provide instructions for initializing the FlexCAN module.

26.5.1 FlexCAN Initialization Sequence

The FlexCAN module can be reset in two ways:

- Core-level hard reset, which resets all memory mapped registers asynchronously
- SOFT_RST bit in the MCR, which resets some of the memory mapped registers synchronously (see Table 26-2 to see what registers are affected by soft reset)

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it can take some time to fully propagate its effects. The SOFT_RST bit remains set while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft

reset cannot be applied while clocks are shut down in any of the low-power modes. The low-power mode must be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) must be selected while the module is in disable mode. After the clock source is selected and the module is enabled (MDIS bit cleared), FlexCAN automatically goes to freeze mode. In freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in the MCR are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR are set. The Tx signal is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. The message buffers and the Rx individual mask registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into freeze mode (see Section 26.4.9.1, “Freeze Mode”). The following is a generic initialization sequence applicable to the FlexCAN module:

1. Initialize the module configuration register
   - Enable the individual filtering per message buffer and reception queue features by setting the BCC bit
   - Enable the warning interrupts by setting the WRN_EN bit
   - If required, disable frame self reception by setting the SRX_DIS bit
   - Enable the FIFO by setting the FEN bit
   - Enable the abort mechanism by setting the AEN bit
   - Enable the local priority feature by setting the LPRIO_EN bit
2. Initialize the control register
   - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
   - Determine the bit rate by programming the PRESDIV field
   - Determine the internal arbitration mode (LBUF bit)
3. Initialize the message buffers
   - The control and status word of all message buffers must be initialized
   - If FIFO was enabled, the 8-entry ID table must be initialized
   - Other entries in each message buffer must be initialized as required
4. Initialize the Rx individual mask registers
5. Set required interrupt mask bits in the IMASK registers (for all message-buffer interrupts), in the CTRL register (for bus-off and error interrupts) and in the MCR for wake-up interrupt
6. Clear the HALT bit in the MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

Chapter 27
General Purpose Input/Output Module (GPIO)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

27.1 Overview

The general purpose input/output (GPIO) module provides 32 dedicated general-purpose one-bit contacts that can be individually configured as either inputs or outputs. Contacts configured as outputs reflect internal register values, and those configured as inputs can be detected by reading internal registers.

Figure 27-1 is a top-level diagram of the GPIO module.

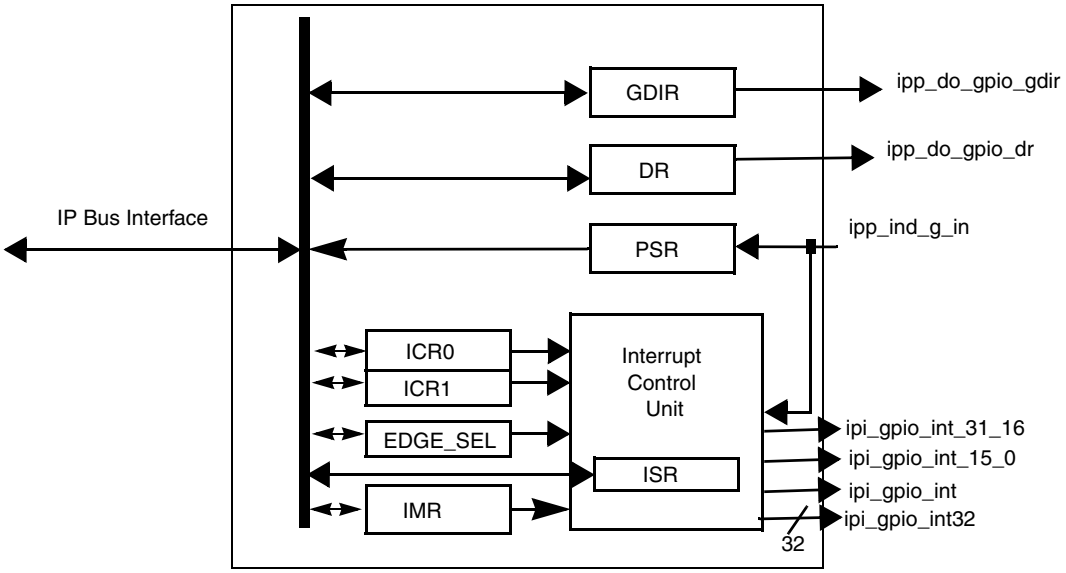


Figure 27-1. GPIO Module Diagram

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic. The eight registers, described in detail in Section 27.3.3, "Register Descriptions," include the following:

- Data register (DR)
- GPIO direction register (GDIR)
- Pad sample register (PSR)
- Interrupt control registers (ICR1, ICR2)

- Edge select register (EDGE_SEL)
- Interrupt mask register (IMR)
- Interrupt status register (ISR)

Each GPIO input has a dedicated edge-detect circuit that can be configured through software to detect rising edges, falling edges, logic low-levels or logic high-levels on the input signals. GPIO edge detection is described further in Section 27.4.2, "Interrupt Control Unit." The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (IMR). These qualified outputs are ORed together to generate three one-bit interrupt lines as follows:

- ipi_gpio_int_31_16: OR of 16 high interrupts
- ipi_gpio_int_15_0: OR of 16 low interrupts
- ipi_gpio_int: OR of all 32 interrupts

In addition, the 32-bit signal ipi_gpio_int32 gives the user access to all 32 individual interrupts.

27.1.1 Features

The GPIO includes the following features:

- General purpose input/output logic capabilities:
  - Drives specific data to output using the data register (DR)
  - Controls the direction of the signal using the GPIO direction register (GDR)
  - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (PSR).
- GPIO interrupt capabilities:
  - Supports up to 32 interrupts
  - Identifies interrupt edges
  - Generates three active-high interrupts to the SoC interrupt controller

27.2 External Signal Description

In addition to the IP bus interface, the GPIO has the signals listed in Table 27-1.

Table 27-1. External Signal Properties

Name	Function	Reset	Pull Up
ipp_do_gpio_gdir	GPIO direction signal. Controls the direction of the GPIO signal if the I/O multiplexer (IOMUX) is in GPIO mode. 0 GPIO signal is configured as input. 1 GPIO signal is configured as output.	0	—
ipp_do_gpio_dr	GPIO data signal—reflects GPIO data register values. If the IOMUX is in GPIO mode and the corresponding direction bit is set to output, this signal is driven to the I/O interface.	0	—
ipp_ind_g_in	GPIO input line from the IOMUX.	0	—

Table 27-1. External Signal Properties (continued)

Name	Function	Reset	Pull Up
ipi_gpio_int31_16	GPIO outputs functioning as interrupt—MSB OR'ed interrupts. One-bit interrupt OR of 16 high interrupts.	0	—
ipi_gpio_int15_0	GPIO outputs functioning as interrupt—LSB OR'ed interrupts. One-bit interrupt OR of 16 low interrupts.	0	—
ipi_gpio_int	GPIO outputs functioning as interrupt—OR'ed interrupts. One-bit interrupt OR of 32 interrupts.	0	—
ipi_gpio_int32	GPIO outputs functioning as interrupts—32-bit, all interrupt out.	0	—
ipg_clk_s	Unique GPIO input clock, derived from ipg_clk and gated by the module enable signal. The clock is only active when GPIO is accessed.	—	—

27.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers. There are eight 32-bit GPIO registers. All registers are accessible from the IP interface. Only 32-bit accesses are supported: non–32-bit read accesses return 32 bits, and non–32-bit write accesses are ignored.

27.3.1 Memory Map

Table 27-2 is the GPIO memory map. For the base address of a particular module instantiation, see the system memory map.

Table 27-2. GPIO Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (DR)	GPIO data register	R/W	0X0000_0000	27.3.3.1/27-5
0x0004 (GDIR)	GPIO direction register	R/W	0X0000_0000	27.3.3.2/27-6
0x0008 (PSR)	GPIO pad status register	Read-only	0X0000_0000	27.3.3.3/27-7
0x000C (ICR1)	GPIO interrupt configuration register1	R/W	0X0000_0000	27.3.3.4/27-8
0x0010 (ICR2)	GPIO interrupt configuration register2	R/W	0X0000_0000	27.3.3.5/27-9
0x0014 (IMR)	GPIO interrupt mask register	R/W	0X0000_0000	27.3.3.6/27-10
0x0018 (ISR)	GPIO interrupt status register	R/W	0X0000_0000	27.3.3.7/27-10
0x001C (EDGE_SEL)	GPIO edge select register	R/W	0X0000_0000	27.3.3.8/27-11

27.3.2 Register Summary

The following definitions serve as a key for the register summary and individual register diagrams.

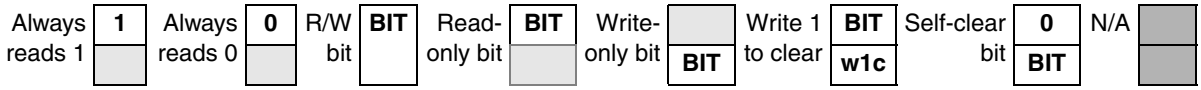


Figure 27-2. Key to Register Fields

Table 27-3 provides a key for register figures and the register summary table.

Table 27-3. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero (previously labeled slfclr).
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 27-4 shows the GPIO register summary.

Table 27-4. GPIO Register Summary

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (DR)	R	DR[31:16]															
	W																
	R	DR[15:0]															
	W																
0x0004 (GDIR)	R	GDIR[31:16]															
	W																
	R	GDIR[15:0]															
	W																

Table 27-4. GPIO Register Summary (continued)

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0008 (PSR)	R	PSR[31:16]															
	W																
	R	PSR[15:0]															
	W																
0x000C (ICR1)	R	ICR15			ICR14			ICR13			ICR12			ICR11			ICR10
	W																
	R	ICR9			ICR8			ICR7			ICR6			ICR5			ICR4
	W																
0x0010 (ICR2)	R	ICR3			ICR2			ICR1			ICR0			ICR31			ICR30
	W																
	R	ICR29			ICR28			ICR27			ICR26			ICR25			ICR24
	W																
0x0014 (IMR)	R	ICR23			ICR22			ICR21			ICR20			ICR19			ICR18
	W																
	R	ICR17			ICR16			IMR[31:16]									
	W																
0x0018 (ISR)	R	IMR[15:0]															
	W																
	R	ISR[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	ISR[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x001C (EDGE_SEL)	R	EDGE_SEL[31:16]															
	W																
	R	EDGE_SEL[15:0]															
	W																

27.3.3 Register Descriptions

This section contains the detailed descriptions for the GPIO registers.

27.3.3.1 GPIO Data Register (DR)

The 32-bit GPIO DR register stores data that is ready to be driven to the output lines. If the I/O multiplexer (IOMUX) is in GPIO mode and a given GPIO direction bit is set, then the corresponding DR bit is driven to the output. If a given GPIO direction bit is cleared, then a read of DR reflects the value of the corresponding signal. Read accesses automatically include two wait states to guarantee synchronization.

The results of a read of a DR bit depends on the IOMUX input mode settings and the corresponding GDIR bit as follows:

- If GDIR[*n*] is set to 1 and the IOMUX input mode is GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and the IOMUX input mode is GPIO, then reading DR[*n*] returns the corresponding input signal's value.
- If GDIR[*n*] is set to 1 and the IOMUX input mode is not GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and the IOMUX input mode is not GPIO, then reading DR[*n*] always returns zero.

Figure 27-3 shows the DR fields, and Table 27-5 shows the register's field descriptions.

Offset 0x0000 (DR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-3. GPIO Data Register (DR)

Table 27-5. DR Field Descriptions

Field	Description
31–0 DR	Data bits. This register defines the value of the GPIO output when the signal is configured as an output (GDIR[*n*]=1). Writes to this register are stored in a register. Reading DR returns the value stored in the register if the signal is configured as an output (GDIR[*n*]=1), or the input signal's value if configured as an input (GDIR[*n*]=0). **Note:** The IOMUX must be configured to GPIO mode for the DR value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

27.3.3.2 GPIO Direction Register (GDIR)

GDIR functions as direction control when the IOMUX is in GPIO mode. Each bit specifies the direction of a one-bit signal. The mapping of each DIR bit to a corresponding SoC signal is determined by the SoC's pin assignment and the IOMUX table—for more details consult the "External Signals and Multiplexing" chapter.

Figure 27-4 shows the GDIR fields, and Table 27-6 shows the register's field descriptions.

Offset 0x0004 (GDIR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GDIR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GDIR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-4. GPIO Direction Register (GDIR)

Table 27-6. GDIR Field Descriptions

Field	Description
31–0 GDIR	GPIO direction bits. Bit *n* of this register defines the direction of the GPIO[*n*] signal as follows: 0 GPIO[*n*] is configured as input. 1 GPIO[*n*] is configured as output. **Note:** GDIR only affects the direction of the I/O signal when the corresponding bit in the IOMUX is configured for GPIO.

27.3.3.3 GPIO Pad Status Register (PSR)

Each read-only bit of the PSR stores the value of the corresponding input signal (as configured in the IOMUX). This register is clocked with the ipg_clk_s clock, so that the input signal is sampled only when accessing this location. PSR reads automatically include two wait states to guarantee synchronization.

Figure 27-5 shows the PSR fields, and Table 27-7 shows the register's field descriptions.

Offset 0x0008 (PSR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-5. GPIO Pad Status Register (PSR)

Table 27-7. PSR Field Descriptions

Field	Description
31–0 PSR	GPIO pad status bits (status bits). Reading PSR returns the state of the corresponding input signal. **Note:** The IOMUX must be configured to GPIO mode for PSR to reflect the state of the corresponding signal. Otherwise the register shows the default values as set by the IOMUX.

27.3.3.4 GPIO Interrupt Configuration Register1 (ICR1)

ICR1 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal. Figure 27-6 shows the ICR1 fields, and Table 27-8 shows the register's field descriptions.

NOTE

The input signal configurations specified by ICR1 are overridden by setting bits in EDGE_SEL register. For more information, see Section 27.3.3.8, "GPIO Edge Select Register (EDGE_SEL)."

Offset 0x000C (ICR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR15		ICR14		ICR13		ICR12		ICR11		ICR10		ICR9		ICR8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR7		ICR6		ICR5		ICR4		ICR3		ICR2		ICR1		ICR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-6. GPIO Interrupt Configuration Register1 (ICR1)

Table 27-8. ICR1 Field Descriptions

Field	Description
31–0 ICR15–ICR0	Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for lines 15 to 0. Bits ICR*n*[1:0] determine the interrupt condition for signal *n* as follows: 00 Interrupt *n* is low-level sensitive. 01 Interrupt *n* is high-level sensitive. 10 Interrupt *n* is rising-edge sensitive. 11 Interrupt *n* is falling-edge sensitive.

27.3.3.5 GPIO Interrupt Configuration Register2 (ICR2)

ICR2 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal. Figure 27-7 shows the ICR2 fields, and Table 27-9 shows the register's field descriptions.

NOTE

The input signal configurations specified by ICR2 are overridden by setting bits in EDGE_SEL register. For more information, see Section 27.3.3.8, "GPIO Edge Select Register (EDGE_SEL)."

Offset 0x0010 (ICR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR31		ICR30		ICR29		ICR28		ICR27		ICR26		ICR25		ICR24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR23		ICR22		ICR21		ICR20		ICR19		ICR18		ICR17		ICR16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 27-7. GPIO Interrupt Configuration Register2 (ICR2)

Table 27-9. ICR2 Field Descriptions

Field	Description
31–0 ICR31–ICR16	Interrupt configuration 2 fields. These fields control the active condition of the interrupt function for lines 31 to 16. Bits ICR*n*[1:0] determine the interrupt condition for signal *n* as follows: 00 Interrupt *n* is low-level sensitive. 01 Interrupt *n* is high-level sensitive. 10 Interrupt *n* is rising-edge sensitive. 11 Interrupt *n* is falling-edge sensitive.

27.3.3.6 GPIO Interrupt Mask Register (IMR)

IMR contains masking bits for each interrupt line. Figure 27-8 shows the IMR fields, and Table 27-10 shows the register's field descriptions.

Offset 0x0014 (IMR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-8. GPIO Interrupt Mask Register (IMR)

Table 27-10. IMR Field Descriptions

Field	Description
31–0 IMR	Interrupt mask bits. This register is used to enable or disable the interrupt function on each of the 32 GPIO signals. Bit IMR[*n*] (n=0...31) controls interrupt *n* as follows: 0 Interrupt *n* is disabled. 1 Interrupt *n* is enabled.

27.3.3.7 GPIO Interrupt Status Register (ISR)

The ISR functions as an interrupt status indicator. Each bit indicates whether an interrupt condition has been met for the corresponding input signal. When an interrupt condition is met (as determined by the corresponding interrupt condition register field), the corresponding bit in this register is set. Read accesses automatically include two wait states to ensure synchronization. One wait state is included for reset. Figure 27-9 shows the ISR fields, and Table 27-11 shows the register's field descriptions.

Offset 0x0018 (ISR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ISR[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-9. GPIO Interrupt Status Register (ISR)

Table 27-11. ISR Field Descriptions

Field	Description
31–0 ISR	Interrupt status bits. Bit *n* of this register is set to 1 when the active condition (as determined by the corresponding ICR bit) is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in IMR. When the active condition has been detected, the corresponding bit remains set until cleared by software (by writing 1 to the bit).

27.3.3.8 GPIO Edge Select Register (EDGE_SEL)

EDGE_SEL can be used to override the interrupt configurations specified in the ICR1 and ICR2 registers. If the EDGE_SEL bit is set, then a rising edge or falling edge in the corresponding signal generates an interrupt. This register provides backward compatibility. On reset all bits are cleared, so that the ICR registers are not overridden.

Figure 27-10 shows the EDGE_SEL fields, and Table 27-12 shows the register's field descriptions.

Offset 0x001C (EDGE_SEL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDGE_SEL[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDGE_SEL[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 27-10. GPIO Edge Select Register (EDGE_SEL)

Table 27-12. EDGE_SEL Field Descriptions

Field	Description
31–0 EDGE_SEL	Edge select. When EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

27.4 GPIO Functional Description

27.4.1 Input/Output Signal Function

A GPIO signal can operate as a general-purpose input/output when the IOMUX is set to GPIO mode. Each GPIO signal can be independently configured as either an input or an output using the GPIO direction register (GDIR). When configured as an output (GDIR bit set to 1), the value in the data bit in the GPIO data register (DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GDIR bit is cleared), the state of the input can be read from the corresponding PSR bit.

27.4.2 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR bit is cleared). The interrupt control registers (ICR1 and ICR2) may be used to independently configure the interrupt condition of each input signal (low-to-high transition, high-to-low transition, low, or high). For information about the ICR1 and ICR2 settings, see Section 27.3.3.4, "GPIO Interrupt Configuration Register1 (ICR1)," and Section 27.3.3.5, "GPIO Interrupt Configuration Register2 (ICR2)."

The interrupt control unit is built of 32 interrupt control subunits, where each subunit handles a single interrupt line.

27.5 Initialization/Application Information

This section provides initialization and application information for the GPIO.

27.5.1 GPIO Read Mode

The programming sequence for reading input signals is as follows:

1. Configure IOMUX to select GPIO mode (using IOMUXC).
2. Configure the GPIO direction register (GDR) to input.
3. Read value from the data register or pad status register.

A pseudocode description to read [input3:input0] values is as follows:

```
write sw_mux_ctl_<input0>_<input1>_<input2>_<input3>, 32'h00000000 // SET INPUTS TO GPIO MODE.
write GDIR[31:4,input3_bit, input2_bit, input1_bit, input0_bit,] 32'hxxxxxxx0 // SET GDIR TO
INPUT.
read DR                                                          // READ INPUT VALUE FROM DR.
read PSR                                                        // READ INPUT VALUE FROM PSR.
```

NOTE

While the GPIO direction is set to input (GDIR[*n*] bit is cleared), a read access to DR[*n*] does not return DR[*n*] data. Instead, it returns the PSR[*n*] data, which is the corresponding input signal value.

27.5.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (using IOMUXC).
2. Configure GPIO direction register (GDR) to output.
3. Write value to data register (DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
write sw_mux_ctl_<output0>_<output1>_<output2>_<output3>, 32'h00000000         // SET OUTPUTS
TO GPIO MODE.
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF // SET GDIR
TO OUTPUT.
write DR, 32'hxxxxxxx5                                                  // WRITE OUTPUT VALUE TO DR.
```

```
read_cmp PSR, 32'hxxxxxxx5                    // READ OUTPUT VALUE FROM PSR ONLY.
```

NOTE

While the GPIO direction is set to output (GDIR[*n*] bit is set to 1), the real internal value can only be verified through the PSR[*n*], and not through the DR[*n*].

Chapter 28
General Purpose Timer (GPT)

Figure 28-1 shows the block diagram of the general purpose timer (GPT).

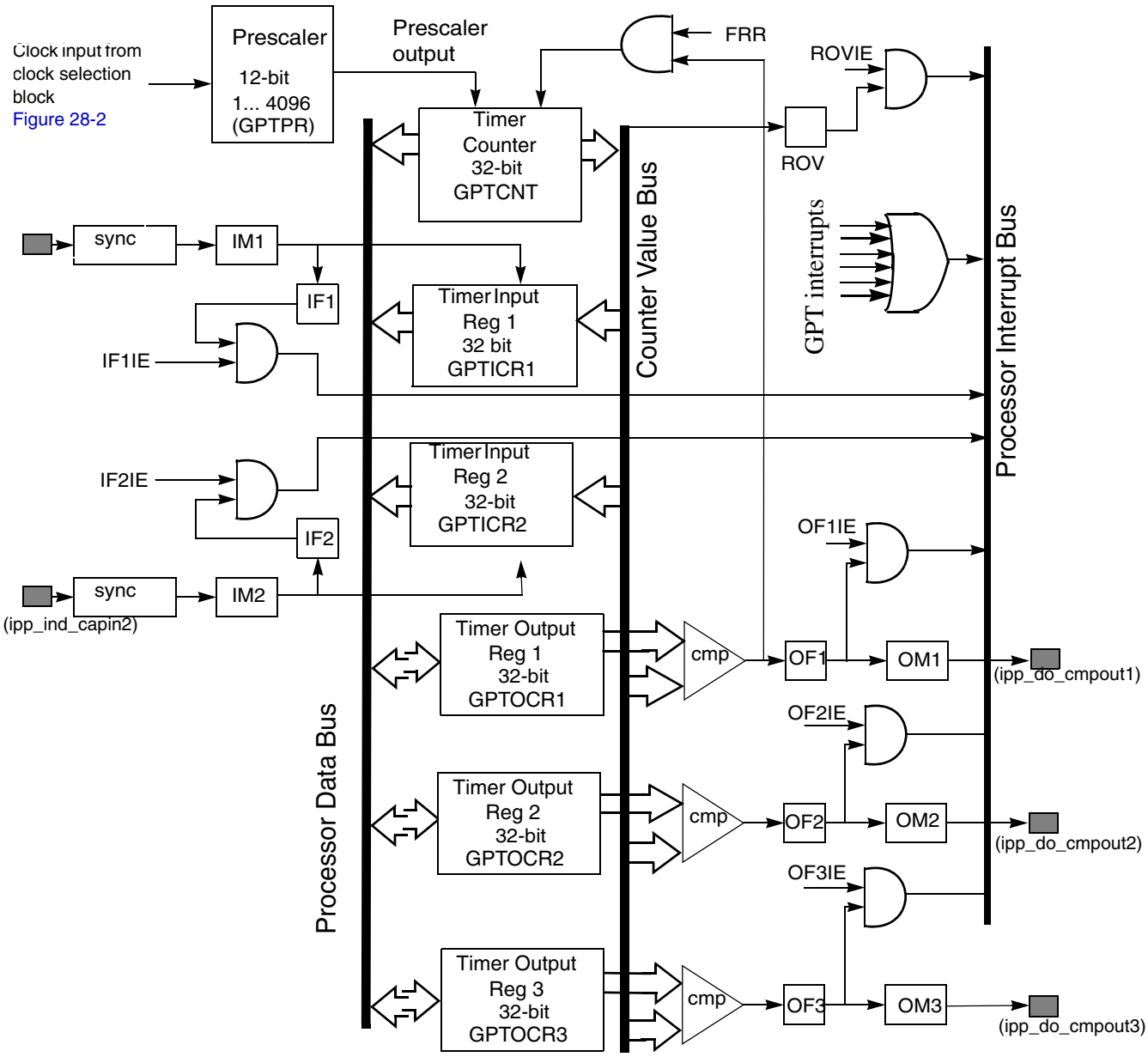


Figure 28-1. General Purpose Timer (GPT) Block Diagram

Figure 28-2 shows the GPT functional clocking scheme.

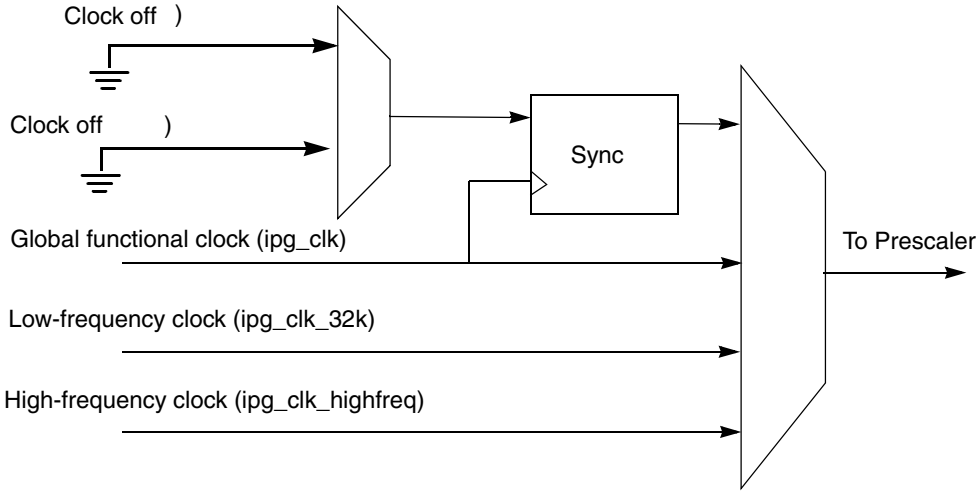


Figure 28-2. GPT Counter Clocks Diagram

28.1 Overview

The general purpose timer (GPT) has a 32-bit upcounter, whose value can be captured when triggered by an external event. The capture trigger can be programmed to be a rising or/and falling edge. The GPT can also assert multiple compare event signals and interrupts when the timer reaches a programmed value. A 12-bit prescaler provides a programmable clock frequency derived from multiple clock sources.

28.2 Features

- One 32-bit upcounter with clock source selection.
- Two input capture channels with programmable trigger edge.
- Three output compare channels with programmable output mode. Forced compare feature also available.
- Can be programmed to be active in low-power mode
- Interrupt generation at capture, compare, rollover events.
- Free-running or restart modes for counter operation.

28.3 Modes of Operation

The GPT can be programmed (using the control register) to run in free-running or restart modes:

- In free-running mode, the counter is not reset when a compare event occurs on any of the three compare channels (ipp_do_cmp1–3). The counter continues up to 0xFFFF_FFFF, then rolls over to 0x0000_0000.
- In restart mode compare channels 2 and 3 behave as in free-running mode, but channel 1 behaves differently. When the counter reaches the compared value for channel 1 it resets and restarts from 0x0000_0000. Any write access to the compare register of channel 1 results in the GPT counter being reset.

28.4 External Signals

Table 28-1 describes all GPT signals that connect off-chip. These signals are described in the following subsections.

Table 28-1. External Signal Description

Name	Direction	Function	Reset State	Pull Up
ipp_ind_capin1	Input	Capture event for capture channel 1	—	Passive
ipp_ind_capin2	Input	Capture event for capture channel 2	—	Passive
ipp_do_cmpout1	Output	Indicator for compare event occurrence in compare channel 1	0	Passive
ipp_do_cmpout2	Output	Indicator for compare event occurrence in compare channel 2	0	Passive
ipp_do_cmpout3	Output	Indicator for compare event occurrence in compare channel 3	0	Passive

Note: Not all of the capin or cmpout signals are implemented in all GPT instances. Users must check which signals are present in the IOMUX table.

28.4.1 Input Capture Trigger Signals (ipp_ind_capin[1:2])

The GPT counter value can be stored into a register with an event from outside the chip. A positive or/and negative edge on these signals can trigger this capture event. These signals are treated as asynchronous to ipg_clk. Only those transitions which occur at least a single clock cycle (clock selected to run the counter) after the previous recorded transition are guaranteed to trigger a capture event.

28.4.2 Output Compare Signals (ipp_do_cmpout[1:3])

These signals show the occurrence of output compare event through a specified transition.

28.5 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers.

The GPT has 10 user-accessible 32-bit registers that are used to configure, operate and monitor the state of the GPT.

28.5.1 Memory Map

Table 28-2 shows the GPT register memory map. For the base address of a particular module instantiation, see the system memory map.

Table 28-2. GPT Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (GPTCR)	GPT control register	R/W	0x0000_0000	28.5.2.1/28-6
0x0004 (GPTPR)	GPT prescaler register	R/W	0x0000_0000	28.5.2.2/28-9
0x0008 (GPTSR)	GPT status register	R/W	0x0000_0000	28.5.2.3/28-9

Table 28-2. GPT Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x000C (GPTIR)	GPT interrupt register	R/W	0x0000_0000	28.5.2.4/28-10
0x0010 (GPTOCR1)	GPT output compare register 1	R/W	0xFFFF_FFFF	28.5.2.5/28-11
0x0014 (GPTOCR2)	GPT output compare register 2	R/W	0xFFFF_FFFF	28.5.2.6/28-12
0x0018 (GPTOCR3)	GPT output compare register 3	R/W	0xFFFF_FFFF	28.5.2.7/28-13
0x001C (GPTICR1)	GPT input capture register 1	Read-only	0x0000_0000	28.5.2.8/28-13
0x0020 (GPTICR2)	GPT input capture register 2	Read-only	0x0000_0000	28.5.2.9/28-14
0x0024 (GPTCNT)	GPT counter register	Read-only	0x0000_0000	28.5.2.10/28-14

28.5.2 Register Summary

Figure 28-3 shows the key to the register fields and Table 28-3 shows the register figure conventions.

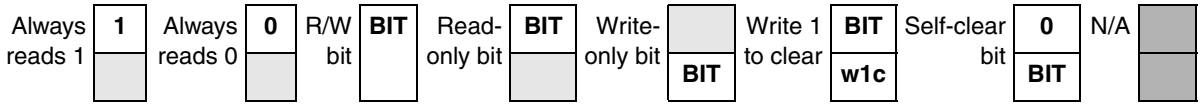


Figure 28-3. Key to Register Fields

Table 28-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 28-4 summarizes the GPT registers.

Table 28-4. GPT Register Summary

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (GPTCR)	R	0	0	0	OM3			OM2			OM1			IM2		IM1	
	W	FO3	FO2	FO1													
	R	SWR	0	0	0	0	0	FRR	CLKSRC			STOP EN	0	WAIT EN	DBGE N	ENM OD	EN
	W																
0x0004 (GPTPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRESCALER[11:0]											
	W																
0x0008 (GPTSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	ROV	IF2	IF1	OF3	OF2	OF1
	W											w1c	w1c	w1c	w1c	w1c	w1c
0x000C (GPTIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	ROVI E	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
	W																
0x0010 (GPTOCR1)	R	COMP[31:16]															
	W																
	R	COMP[15:0]															
	W																
0x0014 (GPTOCR2)	R	COMP[31:16]															
	W																
	R	COMP[15:0]															
	W																
0x0018 (GPTOCR3)	R	COMP[31:16]															
	W																
	R	COMP[15:0]															
	W																
0x001C (GPTICR1)	R	CAPT[31:16]															
	W																
	R	CAPT[15:0]															
	W																
0x0020 (GPTICR2)	R	CAPT[31:16]															
	W																
	R	CAPT[15:0]															
	W																

Table 28-4. GPT Register Summary (continued)

Offset (and Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0024 (GPTCNT)	R	COUNT[31:16]															
	W																
	R	COUNT[15:0]															
	W																

28.5.2.1 GPT Control Register (GPTCR)

The GPTCR is used to program and configure the GPT for appropriate operation and use of its features.

Figure 28-4 shows the register; Table 28-5 provides its field descriptions.

Offset 0x0000 (GPTCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	OM3			OM2			OM1			IM2		IM1	
W	FO3	FO2	FO1													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SWR	0	0	0	0	0	FRR	CLKSRC			STOP EN	0	WAIT EN	0	ENM OD	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-4. GPT Control Register

Table 28-5. Register Field Descriptions

Field	Description
31 FO3	Force output compare channel 3. The bit causes the pin action programmed for the timer output compare 3 pin (according to the bits OM3 in this register). The OF3 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes pin action on timer output compare 3 pin, OF3 flag is not set
30 FO2	Force output compare channel 2. The bit causes the pin action programmed for the timer output compare 2 pin (according to the bits OM3 in this register). The OF2 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect 1 Causes pin action on timer output compare 2 pin, OF2 flag is not set
29 FO1	Force output compare channel 1. The bit causes the pin action programmed for the timer output compare 1pin (according to the bits OM1 in this register). The OF1 flag in the status register is not affected. This bit is self negating and always read as zero. 0 Writing a 0 has no effect 1 Causes pin action on timer output compare 1pin, OF1 flag is not set

Table 28-5. Register Field Descriptions (continued)

Field	Description
28-26 OM3	Output compare channel 3 operating mode. This bit field specifies the response that a compare event generates on the output pin of output compare channel 3. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM3 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT control register 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate a active low pulse of one input clock wide on output
25-23 OM2	Output compare channel 2 operating mode. This bit field specifies the response that a compare event generates on the output pin of output compare channel 2. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM2 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT Control Register. 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate a active low pulse of one input clock wide on output
22-20 OM1	Output compare channel 1 operating mode. This bit field specifies the response that a compare event generates on the output pin of output compare channel 1. The toggle, clear and set options cause a change in the output pin only on occurrence of a compare event. When OM1 is programmed as 1xx (active low pulse), the output pin is set to one, immediately on the next input clock, and the low pulse occurs when there is a compare event. This low pulse width is input clock wide. Here input clock means clock selected by CLKSRC bits of GPT Control Register. 000 Output disconnected. No response on pin 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate a active low pulse of one input clock wide on output
19-18 IM2	Input capture channel 2 operating mode. This bit field determines the transition on the input pin for Input capture channel 2 which triggers a capture event. 00 Capture disabled 01 Capture on rising edge only 10 Capture on falling edge only 11 Capture on both edges
17-16 IM1	Input capture channel 1 operating mode. This bit field determines the transition on the input pin for Input capture channel 1 which triggers a capture event. 00 Capture disabled 01 Capture on rising edge only 10 Capture on falling edge only 11 Capture on both edges
15 SWR	Software reset. This is the software reset of the GPT module. It is a self clearing bit. This bit is set when the module is in reset state and is cleared when the reset procedure is over. Setting this bit resets all the registers to their default reset values except for the EN, ENMOD, STOPEN, and WAITEN bits in this control register. 0 GPT is not in reset state 1 GPT is in reset state

Table 28-5. Register Field Descriptions (continued)

Field	Description
14-10 RESERVED	Reserved
9 FRR	Free-running or restart mode. This bit determines the behavior of the GPT on compare event in channel 1. In restart mode the counter resets to 0x0000_0000 and resumes counting after the occurrence of a compare event. In freerun mode, after a compare event the counter continues counting till 0xFFFF_FFFF and then rolls over to 0. 0 Restart mode 1 Freerun mode
8-6 CLKSRC	Clock source select. These bits selects which clock goes to the prescaler and subsequently be used to run the GPT counter. This bit field value should only be changed after disabling the GPT by clearing the EN bit in this register. For other programming requirements while changing clock source, see Section 28.7.1, “Change of Clock Source”. 000 No clock 001 ipg_clk 010 ipg_clk_highfreq 011 Reserved 1xx ipg_clk_32k
5 STOPEN	GPT stop mode enable. This read/write control bit enables the operation of the GPT during stop mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled in stop mode 1 GPT is enabled in stop mode
4 RESERVED	Reserved.
3 WAITEN	GPT wait mode enable. This read/write control bit enables the operation of the GPT during wait mode. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled in wait mode 1 GPT is enabled in wait mode
2 RESERVED	Reserved
1 ENMOD	GPT enable mode. When GPT is disabled (EN=0), then both main counter and prescaler counter freeze their current count values. ENMOD bit determines the value of GPT counter when EN bit is set and Counter is enabled again. If ENMOD bit is set, then both the main counter and prescaler counter value is reset to 0 on enabling GPT again (EN=1). If ENMOD bit is programmed to 0, then both the main counter & prescaler counter restart counting from their frozen values on enabling GPT again (EN=1). If GPT is programmed to be disabled in a low-power mode (STOP/WAIT), then both the main counter and the prescaler counter freeze at their current count values when GPT enters low-power mode. When GPT exits the low-power mode, both main counter and prescaler counter start counting from their frozen values irrespective of the ENMOD bit. Setting the SWR bit clears the counter value regardless of the value of EN or ENMOD bits. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT counter retain its value when it is disabled 1 GPT counter value is reset to 0 when it is disabled
0 EN	GPT Enable. This bit is the module enable bit. It is recommended that all registers be properly programmed before setting this bit. This bit is reset by a hardware reset. A software reset does not affect this bit. 0 GPT is disabled 1 GPT is enabled

28.5.2.2 GPT Prescaler Register (GPTPR)

The GPTPR contains bits that determine the divide value of the clock that runs the counter. Figure 28-5 shows the register; Table 28-6 provides its field descriptions.

Offset 0x0004 (GPTPR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	PRESCALER											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-5. GPT Prescaler Register

Table 28-6. Register Field Descriptions

Field	Description
31-12 RESERVED	Reserved
11-0 PRESCALER	Prescaler bits. The clock selected by the CLKSRC field is divided by [PRESCALER + 1] and then used to run the counter. A change in the value of the PRESCALER bits result in Prescaler counter to reset & new count period to start immediately. See Figure 28-14 for timing diagram. 0x000 Divide by 1 0x001 Divide by 2 .... 0xFFF Divide by 4096

28.5.2.3 GPT Status Register (GPTSR)

The GPTSR contains bits that indicate the occurrence of rollover of the counter and any event on input capture and output compare channels. The bits are write one to clear.

Figure 28-6 shows the register; Table 28-7 provides its field descriptions.

Offset 0x0008 (GPTSR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ROV	IF2	IF1	OF3	OF2	OF1
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-6. GPT Status Register

Table 28-7. Register Field Descriptions

Field	Description
31-6 RESERVED	Reserved
5 ROV	Rollover Flag. This bit indicates that the counter has reached its maximum possible value and rolled over to 0 from which it continues counting. This bit is only set if the counter has reached 0xFFFF_FFFF in both restart and free-running modes. 0 Rollover not occurred 1 Rollover occurred
4 IF2	Input capture 2 flag. This bit indicates that a capture event has occurred on input capture channel 2. 0 Capture event not occurred 1 Capture event occurred
3 IF1	Input capture 1 flag. This bit indicates that a capture event has occurred on input capture channel 1. 0 Capture event not occurred 1 Capture event occurred
2 OF3	Output compare 3 flag. This bit indicates that a compare event has occurred on output compare channel 3. 0 Compare event not occurred 1 Compare event occurred
1 OF2	Output compare 2 flag. This bit indicates that a compare event has occurred on output compare channel 2. 0 Compare event not occurred 1 Compare event occurred
0 OF1	Output compare 1 flag. This bit indicates that a compare event has occurred on output compare channel 1. 0 Compare event not occurred 1 Compare event occurred

28.5.2.4 GPT Interrupt Register (GPTIR)

The GPTIR contains bits that control the generation of interrupt on rollover, input capture and output compare events.

Figure 28-7 shows the register; Table 28-8 provides its field descriptions.

Offset 0x000C (GPTIR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ROVIE	IF2IE	IF1IE	OF3IE	OF2IE	OF1IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-7. GPT Interrupt Register

Table 28-8. Register Field Descriptions

Field	Description
31-6 RESERVED	Reserved
5 ROVIE	Rollover interrupt enable. This bit controls the occurrence of rollover interrupt. 0 Interrupt disabled 1 Interrupt enabled
4 IF2IE	Input capture 2 interrupt enable. This bit controls the occurrence of input capture channel 2 interrupt. 0 Interrupt disabled 1 Interrupt enabled
3 IF1IE	Input capture 1 interrupt enable. This bit controls the occurrence of input capture channel 1 interrupt. 0 Interrupt disabled 1 Interrupt enabled
2 OF3IE	Output compare 3 interrupt enable. This bit controls the occurrence of output compare channel 3 interrupt. 0 Interrupt disabled 1 Interrupt enabled
1 OF2IE	Output compare 2 interrupt enable. This bit controls the occurrence of output compare channel 2 interrupt. 0 Interrupt disabled 1 Interrupt enabled
0 OF1IE	Output compare 1 interrupt enable. This bit controls the occurrence of output compare channel 1 interrupt. 0 Interrupt disabled 1 Interrupt enabled

28.5.2.5 GPT Output Compare Register 1 (GPTOCR1)

The GPTOCR1 holds the value that determines when a compare event is generated on output compare channel 1. Any write access to the compare register of channel 1 while in restart mode (FRR=0) results in the GPT counter being reset.

Figure 28-8 shows the register; Table 28-9 provides its field descriptions.

Offset 0x0010 (GPTOCR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMP[31:16]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 28-8. GPT Output Compare Register 1

Table 28-9. Register Field Descriptions

Field	Description
31-0 COMP	Compare value. When the counter value equals this bit field value a compare event is generated on output compare channel 1.

28.5.2.6 GPT Output Compare Register 2 (GPTOCR2)

The GPTOCR2 holds the value that determines when a compare event is generated on output compare channel 2.

Figure 28-9 shows the register; Table 28-10 provides its field descriptions.

Offset 0x0014 (GPTOCR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMP[31:16]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 28-9. GPT Output Compare Register 2

Table 28-10. Register Field Descriptions

Field	Description
31-0 COMP	Compare value. When the counter value equals this bit field value a compare event is generated on output compare channel 2.

28.5.2.7 GPT Output Compare Register 3 (GPTOC3)

The GPTOCR3 holds the value that determines when a compare event is generated on output compare channel 3.

Figure 28-10 shows the register; Table 28-11 provides its field descriptions.

Offset 0x0018 (GPTOCR3) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMP[31:16]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 28-10. GPT Output Compare Register 3

Table 28-11. Register Field Descriptions

Field	Description
31-0 COMP	Compare value. When the counter value equals this bit field value a compare event is generated on output compare channel 2.

28.5.2.8 GPT Input Capture Register 1 (GPTICR1)

The GPTICR1 is a read-only register that contains the counter's value during the last capture event on input capture channel 1.

Offset 0x001C (GPTICR1) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CAPT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-11. GPT Input Capture Register 1

Table 28-12. Register Field Descriptions

Field	Description
31-0 CAPT	Capture value. On occurrence of a capture event on Input capture channel 1, the current value of the counter is loaded into this register.

28.5.2.9 GPT Input Capture Register 2 (GPTICR2)

The GPTICR2 is a read-only register that holds the value that was in the counter during the last capture event on input capture channel 2.

Figure 28-12 shows the register; Table 28-13 provides its field descriptions.

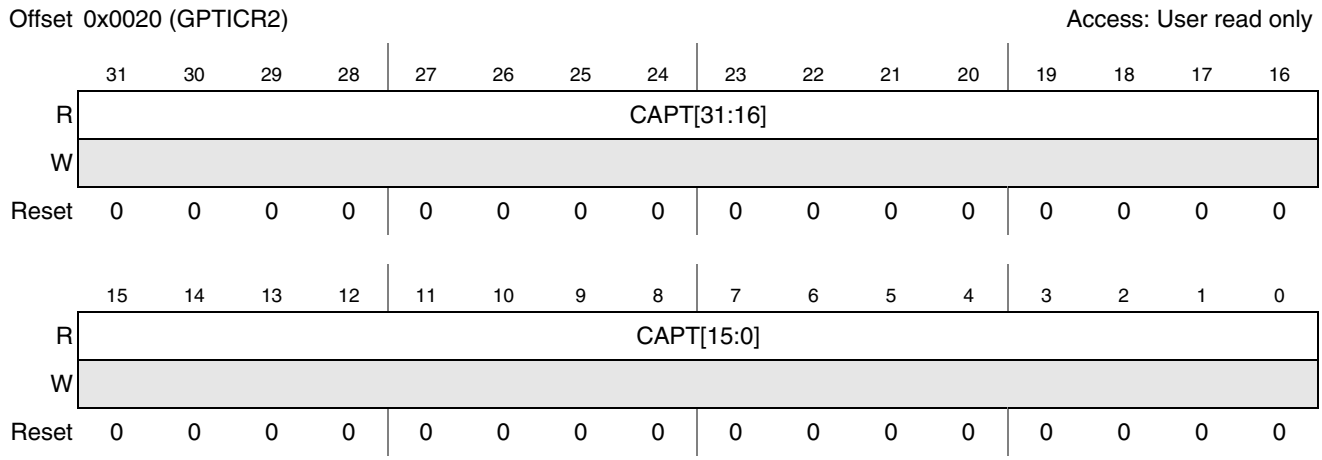


Figure 28-12. GPT Input Capture Register 2

Table 28-13. Register Field Descriptions

Field	Description
31-0 CAPT	Capture value. On occurrence of a capture event on input capture channel 1, the current value of the counter is loaded into this register.

28.5.2.10 GPT Counter Register (GPTCNT)

The GPTCNT register is the main counter register. It is read-only and can be read without affecting the counting process of the GPT.

Figure 28-13 shows the register; Table 28-14 provides its field descriptions.

Offset 0x0024 (GPTCNT) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 28-13. GPT Counter Register

Table 28-14. Register Field Descriptions

Field	Description
31-0 COUNT	Counter value. These bits show the current count value in the counter register.

28.6 Functional Description

28.6.1 Operation

The EN bit of the GPT control register (GPTCR) enables the GPT's 32-bit free-running upcounter. When the EN bit is set to 1, the counter starts immediately: the counter's clock source is the output of the prescaler shown in Figure 28-1.

The STOPEN and WAITEN If GPT is programmed to be disabled in a low-power mode (stop or wait mode), then both the main counter and the prescaler counter freeze at their current count values when GPT enters the low-power mode. When GPT exits low-power mode, both main and prescaler counters start counting from their frozen values, irrespective of the ENMOD bit. GPTCNT can be read at any time by the processor. Both input capture channels use the same counter (GPTCNT).

The SWR bit in the GPTCR can be set to 1 to produce a software reset. All GPT register bits are reset except for EN, ENMOD, STOPEN, and WAITEN bits, which are not affected by a software reset. A software reset can be invoked even when the GPT is disabled.

A hardware reset resets all the GPT registers to their respective reset values. All registers reset to 0 except for the output compare registers (GPTOCR1—GPTOCR3), which are reset to 0xFFFF_FFFF.

28.6.1.1 Clocks

The clock that feeds the prescaler can be selected from the following clock inputs.

- High-frequency reference clock (ipg_clk_highfreq)

This clock is provided (and optionally gated) by the clock controller module (CCM). This clock should be available in low-power mode when the global functional clock (ipg_clk) is turned off. thus allowing the module to run using this clock in low-power mode. In normal mode, the CCM synchronizes this clock to the main high frequency bus clock (AHB clock) and switches to an unsynchronized version in low-power mode.

- Low-frequency reference clock (ipg_clk_32k)

  This 32 kHz clock is provided by the CCM. This clock should be available in low-power mode when the global functional clock is turned off, thus allowing the module to run using this clock in low-power mode. In normal mode, the CCM synchronizes this clock to the main high frequency bus clock (AHB clock) and switches to an unsynchronized version in low-power mode.
- Global functional clock (ipg_clk) This clock is typically used in normal operations. In low-power modes, if the module is programmed to be disabled (by clearing the STOPEN or WAITEN bits), then the peripheral clock can be switched off.

The clock input source is determined by the clock source (CLKSRC) field in the control register. The clock input to the prescaler can also be disabled by programming the CLKSRC bits of control register to 000.

NOTE

The CLKSRC value should be changed only after disabling the GPT by clearing the EN bit in the GPTCR. For other programming requirements while changing clock source, see Section 28.7.1, "Change of Clock Source."

The PRESCALER field is used to select the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by any value from 1–4096, and can be changed any time. A change in the value of the PRESCALER field is immediately reflected on its output clock frequency. Figure 28-14 shows the timing associated with a change in PRESCALER's value

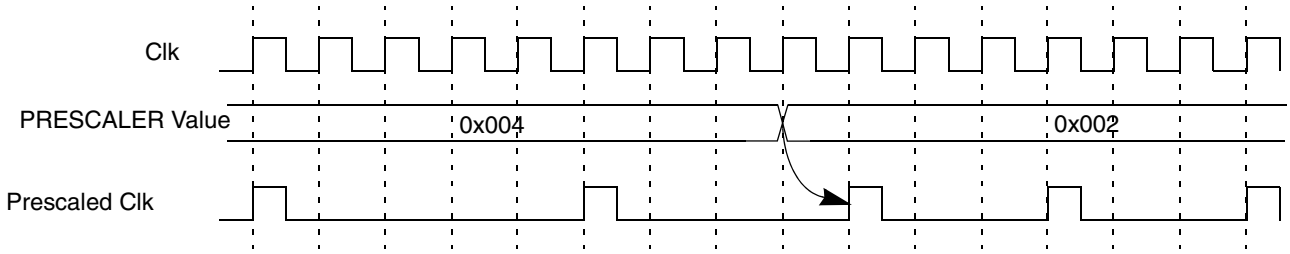
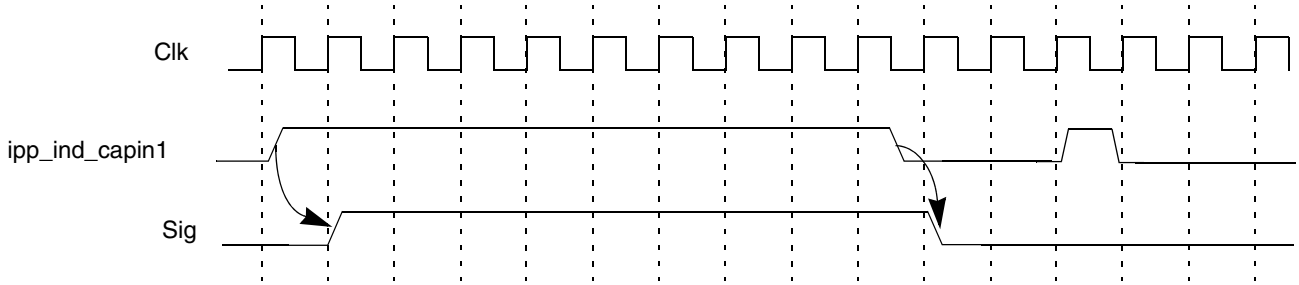


Figure 28-14. PRESCALER Value Change Timing

28.6.1.2 Input Capture

There are two input capture channels and each input capture channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to make an interrupt service request. When a selected edge transition occurs on an input capture pin, the contents of the GPTCNT is captured on the corresponding capture register and sets the appropriate interrupt status flag. An interrupt request can be generated when the transition is detected if its corresponding enable bit is set in the interrupt register. The capture can be programmed to occur on the input pin rising edge, falling edge, on both edges or can be disabled completely. The events

are synchronized with the clock selected to run the counter. Only those transitions which occur at least a single clock cycle (clock selected to run the counter) after the previous recorded transition are guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in latching of the input transition. The input capture registers can be read at any time without affecting their values.



Clk - The clock selected from CLKSRC bit field setting
ipp_ind_capin1 - Pad signal for capture channel 1
Sig - Capture signal as sensed by the module

Figure 28-15. Input Capture Event Timing Diagram

28.6.1.3 Output Compare

The three output compare channels use the same counter (GPTCNT) as the input capture channels. When the programmed content of an output compare register matches the value in GPTCNT, an output compare status flag is set and an interrupt is generated if the corresponding bit is set in the interrupt register. Consequently, the output compare timer pin is set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period (this is subject to the restriction on the maximum frequency allowed on the pad) according to the mode bits programmed. There also exists a forced-compare feature allowing the software to generate compare event when required without the condition of the counter value being equal to the compare value. The action taken as a result of a forced compare is the same as when an output compare match occurs, except that the status flags are not set and no interrupt can be generated. Forced channels take programmed action immediately after the write to the force-compare bits. These bits are self-negating and always read as zeros.

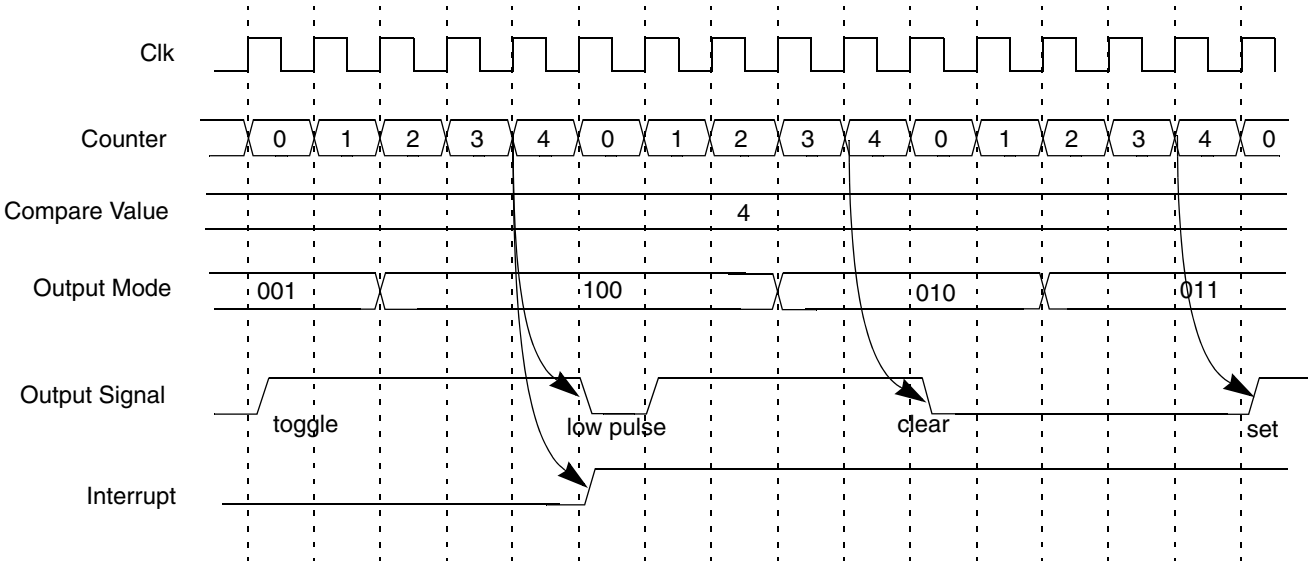


Figure 28-16. Output Compare and Interrupt Timing Diagram

28.6.1.4 Interrupts

The GPT can generate six different interrupts. All interrupts can be generated in low-power and debug modes if the selected clock for running the counter is available.

- Rollover interrupt

  This interrupt is generated when the GPT counter reaches 0xFFFF_FFFF and resets to 0x0000_0000 and continues counting. The interrupt is enabled by the ROVIE bit in GPTIR and its associated status bit is ROV bit in GPTSR.

- Input capture interrupts (1,2)

  On occurrence of a capture event, interrupts can be generated by each input capture channel. These interrupts are enabled by IF2IE and IF1IE bits in GPTIR, and their associated status bits are IF2 and IF1 in GPTSR register. The capture of the counter value due to a capture event is not affected by a pending capture interrupt. The capture register is updated with the new counter value on occurrence of capture event irrespective of whether that capture channel's interrupt has been serviced or not.

- Output compare interrupts (1,2,3)

  On occurrence of a compare event, interrupts can be generated by each output compare channel. These interrupts are enabled by the OF3IE, OF2IE and OF1IE bits in GPTIR and their associated status bits are OF3, OF2 and OF1 in GPTSR. No interrupt is generated due to a forced compare.

A cumulative interrupt line is also present which is asserted whenever any of the above interrupts are posted. This has no associated enable or status bits.

28.6.1.5 Low-Power Mode Behavior

In low-power modes (stop mode and wait mode) the counter continues to run if the clock from the selected clock source is available and if the control bit for that mode is set. Otherwise the main counter and the

prescaler counter freeze at their current values and resume counting from their frozen values when the low-power mode is exited.

28.7 Initialization/ Application Information

28.7.1 Change of Clock Source

the CLKSRC field in GPTCR determines the clock source. This field value should be changed only after disabling the GPT (EN=0). Below is the software sequence to be followed when changing clock source.

1. Disable GPT by setting EN=0 in GPTCR.
2. Disable GPT interrupt register (GPTIR).
3. Program OM3,OM2,OM1 to 00 in GPTCR.
4. Change clock source CLKSRC to desired value in GPTCR.
5. Clear the GPT status register (GPTSR) by writing 1’s.
6. Enable GPT interrupt register (GPTIR).
7. Set ENMOD=1 in GPTCR, to bring GPT counter to 0x0000_0000.
8. ENABLE GPT (EN=1) in GPTCR.

Chapter 29
Inter IC Module (I2C)

This chapter describes module-level operation and programming of I^2C module. The chapter is intended for a module driver software developer. To understand how the module is integrated at the SoC level, a system software developer can refer to discussions of the module in the appropriate SoC-level chapter(s).

References: This document assumes an understanding of the following reference:

1. Philips I^2C Bus Specification, Version 2.1

29.1 Overview

The Inter IC (I^2C) module provides functionality of a standard I^2C slave and master. The I^2C module is designed to be compatible with the standard Philips I^2C bus protocol.

I^2C is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange, minimizing the interconnection between devices. This bus is suitable for applications requiring occasional communications over a short distance between many devices. The flexible I^2C standard allows additional devices to be connected to the bus for expansion and system development. See the connection diagram in Figure 29-1.

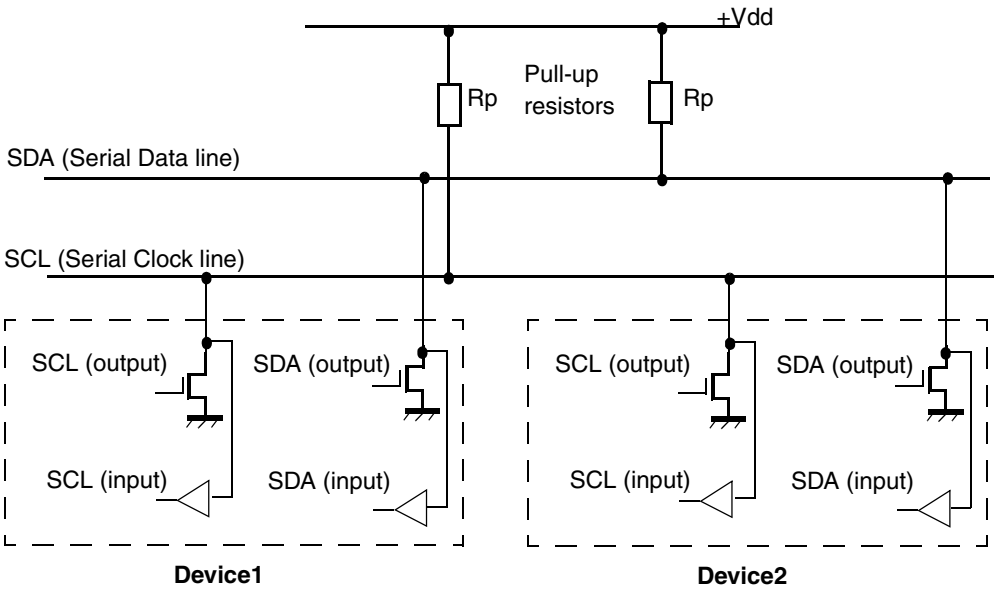


Figure 29-1. Connection of Devices to I^2C Bus

The I^2C interface operates up to 400 kbps, but it depends on the pin loading and timing characteristics. For pin requirement details, see Philips I^2C Bus Specification, Version 2.1. The I^2C system is a true multiple-master bus including arbitration and collision detection that prevents data corruption if multiple

devices attempt to control the bus simultaneously. This feature supports complex applications with multiprocessor control and can be used for rapid testing and alignment of end products through external connections to an assembly-line computer. Figure 29-2 shows the block diagram of I^2C module.

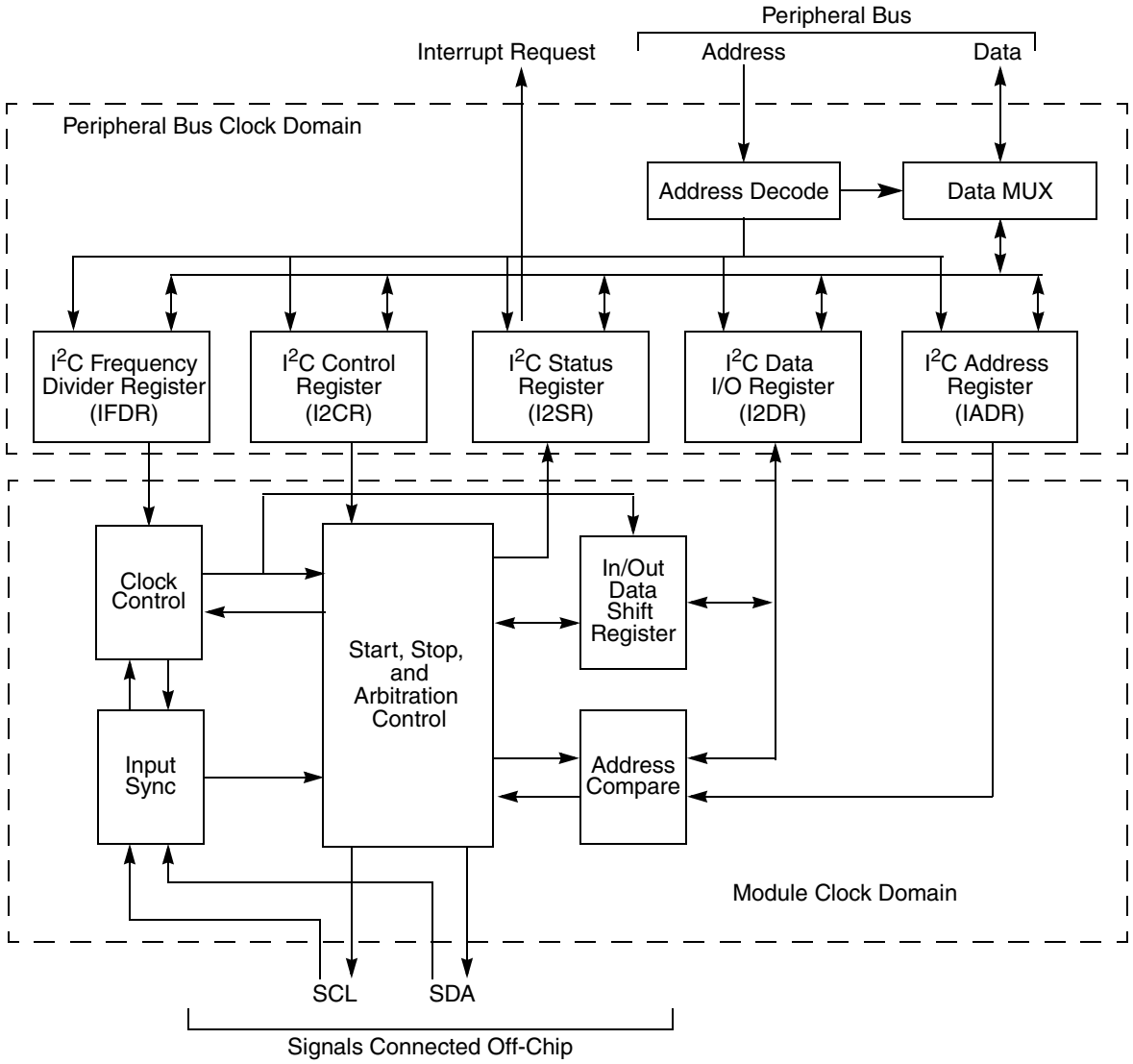


Figure 29-2. I^2C Block Diagram

29.1.1 Features

The I^2C module has the following key features:

- Compatibility with I^2C bus standard
- Multiple-master operation
- Software-programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

29.1.2 Modes and Operations

The I^2C module primarily operates in two different functional modes.

29.1.2.1 Standard Mode

In Standard mode, I^2C module supports the data transfer rates up to 100 Kbits/s.

29.1.2.2 Fast Mode

In Fast Mode, data transfer rates up to 400 Kbits/s can be achieved.

As per module operation, there is no special configuration required for Fast and Standard mode. It is the data transfer rate which distinguishes Standard and Fast mode.

29.2 External Signals

Table 29-1 describes all I2C signals that connect off-chip.

For I^2C compliance, all devices connected to the SCL and SDA signals must have open-drain or open-collector outputs. The logic AND function is exercised on both lines with external pull-up resistors.

NOTE

The reset state values and pull-up/down requirements provided in Table 29-1 are from the module-level perspective. To understand how the module is integrated at the SoC level, the system software developer must see discussions of the module in the appropriate SoC-level chapter(s). For example, a module signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

Table 29-1. Off-Chip Module Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/ Down
SCL	I/O	Serial Clock	1	Active
SDA	I/O	Serial Data	1	Active

1 The reset state values and pull-up/down requirements provided in this table are from the module-level perspective. To understand how the module is integrated at the SoC level, the system software developer must see discussions of the module in the appropriate SoC-level chapter(s). For example, a module signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

29.3 Memory Map and Register Definition

The I^2C module contains five 16-bit registers. Section 29.3.3, "Register Descriptions" provides the detailed descriptions for all of the I^2C registers.

29.3.1 Memory Map

Table 29-2 is the module memory map. For the base address of a particular module instantiation, see the system memory map.

Table 29-2. Module Memory Map

Base Address Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x0000 (IADR)	I^2C Address Register	R/W	0x0000	29.3.3.1/29-6
0x0004 (IFDR)	I^2C Frequency Divider Register	R/W	0x0000	29.3.3.2/29-6
0x0008 (I2CR)	I^2C Control Register	R/W	0x0000	29.3.3.3/29-7
0x000C (I2SR)	I^2C Status Register	R/W	0x0081	29.3.3.4/29-9
0x0010 (I2DR)	I^2C Data I/O Register	R/W	0x0000	29.3.3.5/29-10

NOTE

Registers at offsets 0x0002, 0x0006, 0x000A, and 0x000E are reserved for future additions.

29.3.2 Register Summary

Table 29-3 is the register summary table.

Table 29-3. Module Register Summary

Base Address Offset (Register Abbreviation)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (IADR)	R	0	0	0	0	0	0	0	0	ADR							0
	W																

Table 29-3. Module Register Summary (continued)

Base Address Offset (Register Abbreviation)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0004 (IFDR)	R	0	0	0	0	0	0	0	0	0	0	IC					
	W																
0x0008 (I2CR)	R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	1	0	0
	W														RSTA		
0x000C (I2SR)	R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
	W																
0x0010 (I2DR)	R	0	0	0	0	0	0	0	0	DATA							
	W																

29.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 29-3 and Table 29-4 explain conventions used in register diagrams and tables.

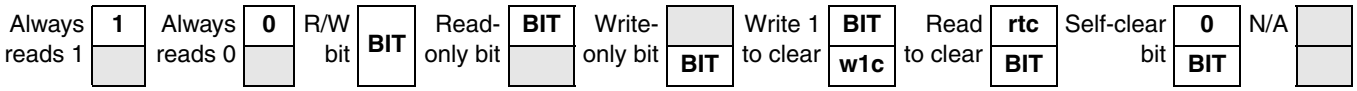


Figure 29-3. Register Field Conventions

Table 29-4. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).

Table 29-4. General Register Conventions (continued)

Convention	Description
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

29.3.3.1 I²C Address Register (IADR)

Figure 29-4 shows the I²C Address Register. Table 29-5 describes the register fields.

0x0000 (IADR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ADR							0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-4. I²C Address Register

Table 29-5. I²C Address Register Field Descriptions

Field	Description
15–8	Reserved
7–1 ADR	Slave address. Contains the specific slave address to be used by the I²C module. Slave mode is the default I²C mode for an address match on the bus. **Note:** The IADR holds the address the I²C responds to when addressed as a slave. The slave address is not the address sent on the bus during the address transfer. The register is not reset by a software reset.
0	Reserved

29.3.3.2 I²C Frequency Register (IFDR)

The IFDR provides a programmable prescaler to configure the clock for bit-rate selection. The register does not get reset by software reset. Figure 29-5 shows the I²C Frequency Register. Table 29-6 describes the register fields.

0x0004 (IFDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	IC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-5. I²C Frequency Register

Table 29-6. I²C Frequency Register Field Descriptions

Field	Description
15–6	Reserved
5–0 IC	I²C clock rate. Pre scales the clock for bit-rate selection. Due to potentially slow SCL and SDA rise and fall times, bus signals are sampled at the prescaler frequency. The serial bit clock frequency is equal to Module Clock divided by the divider shown in Table 29-7. **Note:** The IC value should not be changed during the data transfer, however, it can be changed before REPEAT START or START programming sequence in I²C module. The I²C protocol supports bit rates up to 400 kbps. The IC bits need to be programmed in accordance with this constraint.

Table 29-7. IFDR Register Field Values

IC	Divider	IC	Divider	IC	Divider	IC	Divider
0x00	30	0x10	288	0x20	22	0x30	160
0x01	32	0x11	320	0x21	24	0x31	192
0x02	36	0x12	384	0x22	26	0x32	224
0x03	42	0x13	480	0x23	28	0x33	256
0x04	48	0x14	576	0x24	32	0x34	320
0x05	52	0x15	640	0x25	36	0x35	384
0x06	60	0x16	768	0x26	40	0x36	448
0x07	72	0x17	960	0x27	44	0x37	512
0x08	80	0x18	1152	0x28	48	0x38	640
0x09	88	0x19	1280	0x29	56	0x39	768
0x0A	104	0x1A	1536	0x2A	64	0x3A	896
0x0B	128	0x1B	1920	0x2B	72	0x3B	1024
0x0C	144	0x1C	2304	0x2C	80	0x3C	1280
0x0D	160	0x1D	2560	0x2D	96	0x3D	1536
0x0E	192	0x1E	3072	0x2E	112	0x3E	1792
0x0F	240	0x1F	3840	0x2F	128	0x3F	2048

29.3.3.3 I²C Control Register (I2CR)

The I2CR is used to enable the I²C module and the I²C interrupt. It also contains bits that govern operation as a slave or a master. Figure 29-6 shows the I²C Control Register. Table 29-8 describes the register fields.

0x0008 (I2CR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	IEN	IIEN	MSTA	MTX	TXAK	0	0	0
W														RSTA		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-6. I²C Control Register

Table 29-8. I²C Control Register Field Descriptions

Field	Description
15–8	Reserved
7 IEN	I²C enable. Also controls the software reset of the entire I²C module. Resetting the bit generates an internal reset to the module. If the module is enabled in slave mode in the middle of a byte transfer between two other devices on the bus, then slave mode starts operating when the next start condition is detected. If the module is enabled in master mode, then the module is not aware that the bus is busy—so initiating a start cycle may corrupt the current bus cycle, ultimately causing either the current master or the I²C module to lose arbitration. After this, bus operation returns to normal. 0 The module is disabled, but registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other I2CR bits have any effect.
6 IIEN	I²C interrupt enable. 0 I²C module interrupts are disabled, but the status flag I2SR[IIF] continues to be set when an interrupt condition occurs. 1 I²C module interrupts are enabled. An I²C interrupt occurs if I2SR[IIF] is also set.
5 MSTA	Master/slave mode select bit. If the master loses arbitration, MSTA is cleared without generating a STOP signal. **Note:** Module clock should be on for writing to the MSTA bit. 0 Slave mode. Changing MSTA from 1 to 0 generates a STOP and selects slave mode. **Note:** The MSTA bit is cleared by software to generate a STOP condition; it can also be cleared by hardware when the I²C loses the bus arbitration. 1 Master mode. Changing MSTA from 0 to 1 signals a START on the bus and selects master mode.
4 MTX	Transmit/receive mode select bit. Selects the direction of master and slave transfers. 0 Receive. When a slave is addressed, the software should set MTX according to the slave read/write bit in the I²C status register (I2SR[SRW]). 1 Transmit. In master mode, MTX should be set according to the type of transfer required. Therefore, for address cycles, MTX is always 1.
3 TXAK	Transmit acknowledge enable. Specifies the value driven onto SDA during acknowledge cycles for both master and slave receivers. **Note:** Writing TXAK applies only when the I²C bus is a receiver. 0 An acknowledge signal is sent to the bus at the ninth clock bit after receiving one byte of data. 1 No acknowledge signal response is sent (that is, the acknowledge bit = 1).
2 RSTA	Repeat start. Always reads as 0. Attempting a repeat start without bus mastership causes loss of arbitration. 0 No repeat start 1 Generates a repeated START condition
1–0	Reserved

29.3.3.4 I²C Status Register (I2SR)

The I2SR contains bits that indicate transaction direction and status. Figure 29-7 shows the I²C Status Register. Table 29-9 describes the register fields.

0x000C (I2SR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ICF	IAAS	IBB	IAL	0	SRW	IIF	RXAK
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 29-7. I²C Status Register

Table 29-9. I²C Status Register Field Descriptions

Field	Description
15–8	Reserved
7 ICF	Data transferring bit. While one byte of data is transferred, ICF is cleared. 0 Transfer is in progress. 1 Transfer is complete. This bit is set by the falling edge of the ninth clock of the last byte transfer. **Note:** If data is written during the START condition, that is, just after setting the I2CR[MSTA] and I2CR[MTX] bits, then the ICF bit is cleared at the falling edge of SCLK after START. If data is written after the START condition and falling edge of SCLK, then ICF bit is cleared as soon as data is written.
6 IAAS	I²C addressed as a slave bit. The CPU is interrupted if the interrupt enable (I2CR[IIEN]) is set. The CPU must check the slave read/write bit (SRW) and set its TX/RX mode accordingly. Writing to I2CR clears this bit. 0 Not addressed 1 Addressed as a slave. Set when its own address (IADR) matches the calling address.
5 IBB	I²C bus busy bit. Indicates the status of the bus. 0 Bus is idle. If a STOP signal is detected, IBB is cleared. 1 Bus is busy. When START is detected, IBB is set. **Note:** When I²C is enabled (I2CR[IEN] = 1), it continuously polls the bus data (SDAK) and clock (SCLK) signals to determine a START or STOP condition.
4 IAL	Arbitration lost. Set by hardware in the following circumstances (IAL must be cleared by software by writing a “0” to it at the start of the interrupt service routine): • SDA input sampled low when the master drives high during an address or data-transmit cycle. • SDA input sampled low when the master drives high during the acknowledge bit of a data-receive cycle. For the above two cases, the bit is set at the falling edge of 9th SCL clock during the ACK cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. **Note:** Software cannot set the bit. 0 No arbitration lost. 1 Arbitration is lost.
3	Reserved
2 SRW	Slave read/write. When the I²C is addressed as a slave, IAAS is set, and the slave read/write bit (SRW) indicates the value of the R/W command bit of the calling address sent from the master. SRW is valid only when a complete transfer has occurred, no other transfers have been initiated, and the I²C module is a slave and has an address match. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave

Table 29-9. I²C Status Register Field Descriptions (continued)

Field	Description
1 IIF	I²C interrupt. Must be cleared by the software by writing a “0” to it in the interrupt routine. **Note:** The software cannot set the bit. 0 No I²C interrupt pending. 1 An interrupt is pending. This causes a processor interrupt request (if the interrupt enable is asserted [IIEN = 1]). The interrupt is set when one of the following occurs: • One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock). • An address is received that matches its own specific address in slave-receive mode. • Arbitration is lost.
0 RXAK	Received acknowledge. This is the value received of the SDA input for the acknowledge bit during a bus cycle. 0 An “acknowledge” signal was received after the completion of an 8-bit data transmission on the bus. 1 A “No acknowledge” signal was detected at the ninth clock.

29.3.3.5 I²C Data Register (I2DR)

In master-receive mode, reading the data register (I2DR) allows a read to occur and initiates the next byte to be received. In slave mode, the same function is available after it is addressed. Figure 29-8 shows the I²C Data Register. Table 29-10 describes the register fields.

0x0010 (I2DR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DATA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29-8. I²C Data Register

Table 29-10. I²C Data Register Field Descriptions

Field	Description
15–8	Reserved
7–0 DATA	Data Byte. Holds the last data byte received or the next data byte to be transferred. Software writes the next data byte to be transmitted or reads the data byte received. If the core writes data to I2DR to transmit, the core is not allowed to read it back. If the data in I2DR is read from the I2C bus, the core can read it—the value is the last read byte. **Note:** The I2DR is actually implemented by two different registers: one register is written by IP interface and the data written is transmitted on I²C bus; the other register is written by the data received from I²C, and read by IP interface.

29.4 Functional Description

This section includes information on the following:

29.4.1 I^2C System Configuration

Out of a reset, the I^2C module defaults to slave receive operations. Thus, when not operating as a master or responding to a slave transmit address, the I^2C module will default to the slave receiver state. For exceptions, see Section 29.5.1, “Initialization Sequence.”

NOTE

The I^2C module is designed to be compatible with the Philips I^2C bus protocol. For information on system configuration, protocol, and restrictions, see the I^2C Bus Specification, Version 2.1. The I^2C module supports Standard and Fast modes only.

29.4.2 I^2C Protocol

The I^2C communication protocol consists of six components, as follows:

- START
- Data Source/Recipient
- Data Direction
- Slave Acknowledge
- Data Acknowledge
- STOP

See Figure 29-9 for the I^{2}C standard communication protocol, as defined in the following sections.

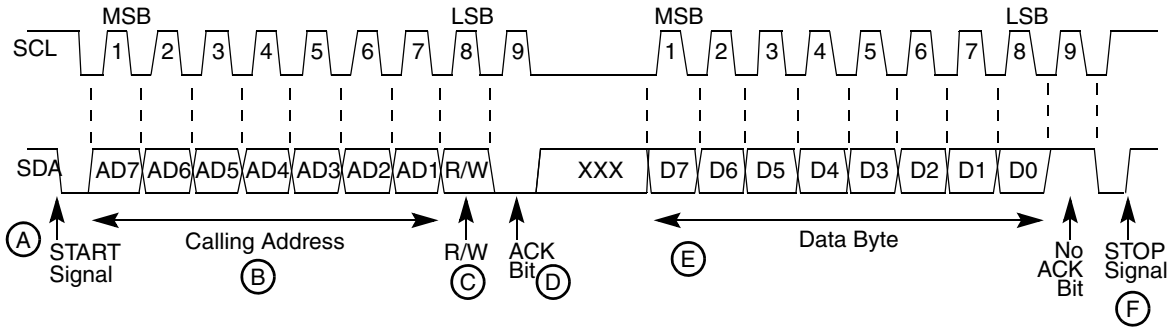


Figure 29-9. I^{2}C Standard Communication Protocol

29.4.2.1 START Signal

When no other device is a bus master (both SCL and SDA lines are at logic high), a device can initiate communication by sending a START signal (see A in Figure 29-9). A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a data transfer (each data transfer can be several bytes long) and awakens all slaves.

29.4.2.2 Slave Address Transmission

The master sends the slave address in the first byte after the START signal (B). After the seven-bit calling address, it sends the R/W bit (C), which tells the slave data transfer direction.

Each slave must have a unique address. An I^{2}C master must not transmit an address that is the same as its slave address; it cannot be master and slave at the same time.

The slave whose address matches that sent by the master pulls SDA low at the ninth clock (D) to return an acknowledge bit.

29.4.2.3 Data Transfer

When successful slave addressing is achieved, the data transfer can proceed (E) on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master.

Data can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 29-9. SCL is pulsed once for each data bit, most-significant bit first. The receiving device must acknowledge each byte by pulling SDA low at the ninth clock; therefore, a data byte transfer takes nine clock pulses.

If it does not acknowledge the master, the slave receiver must leave SDA high. The master can then generate a STOP signal to abort the data transfer or generate a START signal (a repeated start, as shown in Figure 29-10) to start a new calling sequence.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means end-of-data to the slave. The slave releases SDA for the master to generate a STOP or START signal.

NOTE

Writing to the data register triggers transmit operation.

Transmit data should always be written after MTX bit is programmed. Transmit data is not latched inside until the transfer is initiated on the interface bus.

After the transmit data write in I^2C, software can either wait for a transfer-done interrupt or it can poll for ICF bit for zero, if new data has to be written during the previous data transfer. The IIF bit may not be polled if the IIEN bit is set because the I^2C will generate an interrupt when IIF is set.

29.4.2.4 STOP Signal

The master can terminate communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical high (F).

NOTE

A master can generate a STOP even if the slave has made an acknowledgment; at which point, the slave must release the bus.

29.4.2.5 Repeat Start

Instead of signalling a STOP, the master can repeat the START signal, followed by a calling command (see A in Figure 29-10). A repeated START occurs when a START signal is generated without first generating a STOP signal to end the communication. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

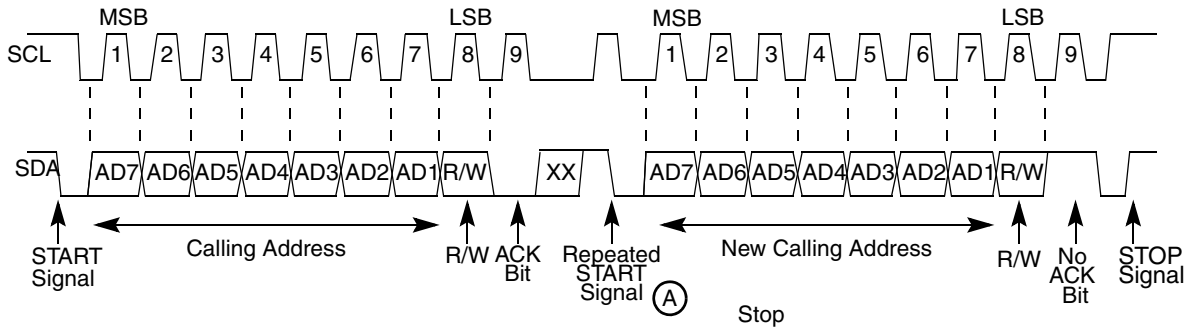


Figure 29-10. Repeated START

29.4.3 Arbitration Procedure

If multiple devices simultaneously request the bus, the bus clock is determined by a synchronization procedure in which the low period equals the longest clock-low period among the devices, and the high period equals the shortest. A data arbitration procedure determines the relative priority of competing devices. A device loses arbitration if it sends logic high while another sends logic low; it immediately switches to slave-receive mode and stops driving SDA. In this case, the transition from master to slave

mode does not generate a STOP condition. Meanwhile, hardware sets the IAL bit in the I^2C Status register (I2SR) to indicate loss of arbitration.

29.4.4 Clock Synchronization

Because wire-AND logic is used, a high-to-low transition on SCL affects devices connected to the bus. Devices start counting their low period when the master drives SCL low. When a device clock goes low, it holds SCL low until the clock high state is reached. However, the low-to-high change in this device clock may not change the state of SCL if another device clock is still in its low period. Therefore, the device with the longest low period holds the synchronized clock SCL low. Devices with shorter low periods enter a high wait state during this time (see Figure 29-11). When all devices involved have counted off their low period, the synchronized clock SCL is released and pulled high. There is then no difference between device clocks and the state of SCL, so all of the devices start counting their high periods. The first device to complete its high period pulls SCL low again.

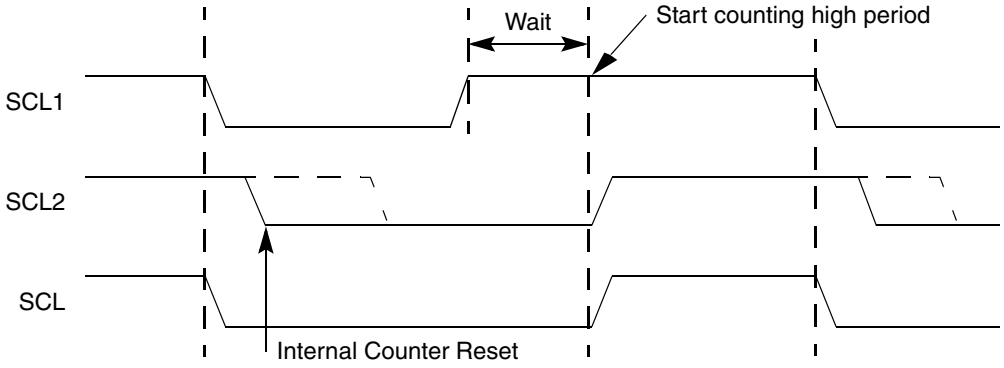


Figure 29-11. Synchronized Clock SCL

29.4.5 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. Slave devices can hold SCL low after completing one byte transfer (9 bits). In such a case, the clock mechanism halts the bus clock and forces the master clock into a wait state until the slave releases SCL.

29.4.6 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is longer than the master SCL low period, the resulting SCL bus signal low period is stretched.

29.4.7 Peripheral Bus Accesses

I^2C is a 16-bit module. Only half-word accesses should be performed to the module.

29.4.8 Generation of Transfer Error on IP Bus

If an address is received on the Peripheral slave bus interface that is not implemented, an access error is generated.

29.4.9 Clocks

There are two input clocks for I^2C module.

- Peripheral Clock—This is the system bus clock, and is used for peripheral bus register read/writes.
- Module Clock—This is the functional clock of the I^2C module. The serial bit clock frequency is obtained by dividing the module clock frequency by the divider (IFDR). The module clock and peripheral clocks are synchronous to each other. The frequency of the module clock must exceed 12.8 MHz for fast mode to achieve 400 Kbps operation.

29.4.10 Reset

The I^2C module can be reset in two ways.

- Global reset: A hard asynchronous reset of the whole I^2C module.
- Software reset: An internal reset for whole I^2C module, except IADR and IFDR registers, is initiated by deasserting the I2CR[IEN] bit.

29.4.11 Interrupts

There is only one interrupt from the module. The interrupt is enabled by setting the I2CR[IIEN] bit. The interrupt is generated in any one of the following conditions.

- One byte transfer is completed (the interrupt is set at the falling edge of the ninth clock).
- An address is received that matches its own specific address in slave-receive mode.
- Arbitration is lost.

29.4.12 Byte Order

The module only supports the little endian mode.

29.5 Initialization

This section includes information on the following:

29.5.1 Initialization Sequence

Before the interface can transfer serial data, registers must be initialized, as follows:

1. Set the data sampling rate (IFDR[IC] to obtain SCL frequency from the system bus clock.
2. Update the address in the (IADR) to define its slave address (address can range from 0 to 0x7f).
3. Set the I^2C enable bit (I2CR[IEN]) to enable the I^2C bus interface system.
4. Modify the bits in the I^2CR to select master/slave mode, transmit/receive mode, and interrupt-enable or not.

29.5.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the master transmitter mode. On a multiple-master bus system, the busy bus (I2SR[IBB]) must be tested to determine whether the serial bus is free. If the bus is free (IBB = 0), the START signal and the first byte (the slave address) can be sent. The data written to the data register comprises the address of the desired slave and the LSB indicates the transfer direction.

The free time between a STOP and the next START condition is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy (which indicates the data register can be written) after writing the calling address to the data register (I2DR) before proceeding to load data into the data register (I2DR).

29.5.3 Post-Transfer Software Response

Sending or receiving a byte sets the data transferring bit (I2SR[ICF]), which indicates one byte communication is finished. Upon completion, the interrupt status (I2SR[IIF]) is also set. An external interrupt is generated if the interrupt enable (I2CR[IIEN]) is set. The software must first clear the interrupt status (I2SR[IIF]) in the interrupt routine. (See the flowchart in Figure 29-12.) The data transferring bit (I2SR[ICF]) is cleared either by reading from I2DR in receive mode or by writing to this register in transmit mode.

If the I2C interrupt is disabled, software can detect the transfer completion by monitoring the interrupt status (I2SR[IIF]), which triggers an interrupt routine in the main program. Inside this polling loop, if the data transfer bit (I2SR[ICF]) is set to 1 then software reads the arbitration lost bit (I2SR[IAL]). If I2SR[IAL] is 0, then the transfer is continued—otherwise, software handles arbitration loss as described in Section 29.5.7, “Arbitration Lost.”

The timeout value will depend on the bus frequency at which I^2C is operating. The minimum timeout for polling the IIF bit at I^2C Max bus frequency of 400 kHz is 25 µs. This value can be interpolated for any bus frequency.

When an interrupt occurs at the end of the address cycle, the master is always in transmit mode; that is, the address is sent. If master receive mode is required, then I2CR[MTX] should be toggled and dummy read of I2DR register has to be done for triggering receive data.

During slave-mode address cycles (I2SR[IAAS] = 1), the slave read/write bit I2SR[SRW] is read to determine the direction of the next transfer. The transmit/receive bit (I2CR[MTX]) should also be

programmed accordingly. For slave-mode data cycles (IAAS = 0), SRW is invalid. MTX should be read to determine the current transfer direction.

29.5.4 Generation of STOP

A data transfer ends when the master signals a STOP, which can occur after all data is sent.

For a master receiver to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last data byte. This is done by setting the transmit acknowledge bit (I2CR[TXAK]) before reading the next-to-last byte. Before the last byte is read, a STOP signal must be generated.

29.5.5 Generation of Repeated START

After the data transfer, if the master still wants the bus, it can signal another START followed by another slave address without signalling a STOP.

29.5.6 Slave Mode

In the slave interrupt service routine (see Figure 29-12), the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (I2CR[MTX]) according to the I2SR[SRW]. Writing to the I2CR clears the IAAS automatically. The only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer can now be initiated by writing information to I2DR for slave transmits, or read from I2DR in slave-receive mode. A dummy read of I2DR in slave/receive mode releases SCL, allowing the master to send data.

In the slave transmitter routine, the receive acknowledge bit (I2SR[RXAK]) must be tested before sending the next byte of data. Setting RXAK means an end-of-data signal from the master receiver, after which the software must switch from transmitter to receiver mode. Reading the data register (I2DR) then releases SCL so that the master can generate a STOP signal.

29.5.7 Arbitration Lost

When several devices try to engage the bus at the same time, only one becomes master. Hardware immediately switches devices that lose arbitration to slave receive mode. Data output to SDA stops, but SCL is still generated until the end of the byte during which arbitration is lost. An interrupt occurs at the falling edge of the ninth clock of this transfer if the arbitration is lost (I2SR[IAL] = 1), and the slave mode is selected (I2CR[MSTA] = 0). See the flowchart in Figure 29-12.

If a device that is not a master tries to transmit or do a START, hardware inhibits the transmission, clears MSTA without signalling a STOP, generates an interrupt to the CPU, and sets I2SR[IAL] to indicate a failed attempt to engage the bus. When considering these cases, the slave service routine should first test I2SR[IAL], and the software should clear it if it is set.

For multi-master mode, when an I^2C module is enabled when the bus is busy and asserts START, the I2SR[IAL] bit gets set only for SDA=0, SCL=0/1, SDA=1, and SCL=0, but not for SDA=1 and SCA=1, which is the same as bus idle state.

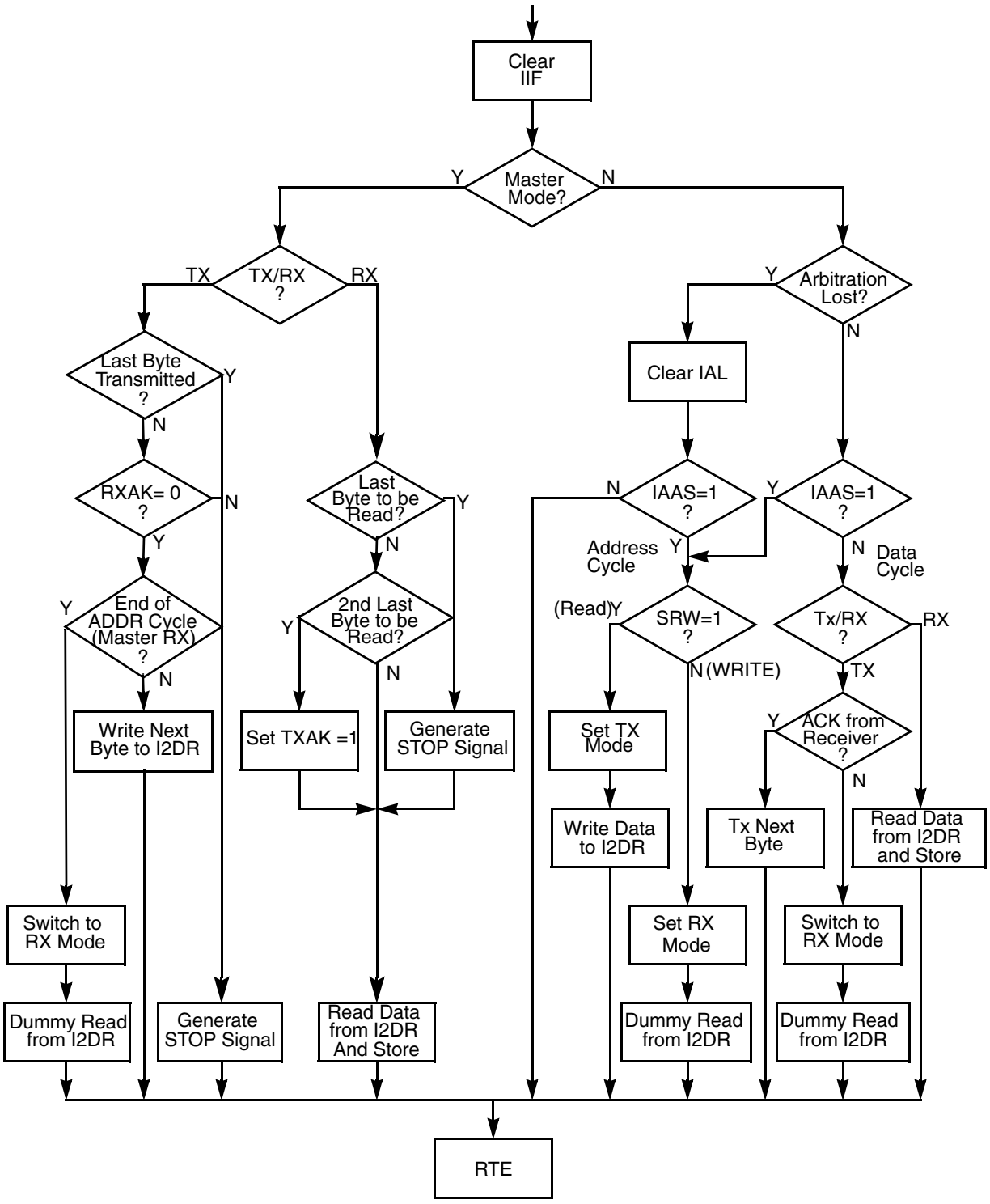


Figure 29-12. Flowchart for a Typical I^2C Interrupt Routine

NOTE

For a repeated start-only, the stop generation stage will not occur in master mode. A loop will repeat itself without stopping for the next start.

For master Rx mode, I^2C is programmed as master transmit during address mode and after slave address transfer, MTX bit should be cleared and a dummy read on I2DR register should be performed so that I^2C can read the next receive data.

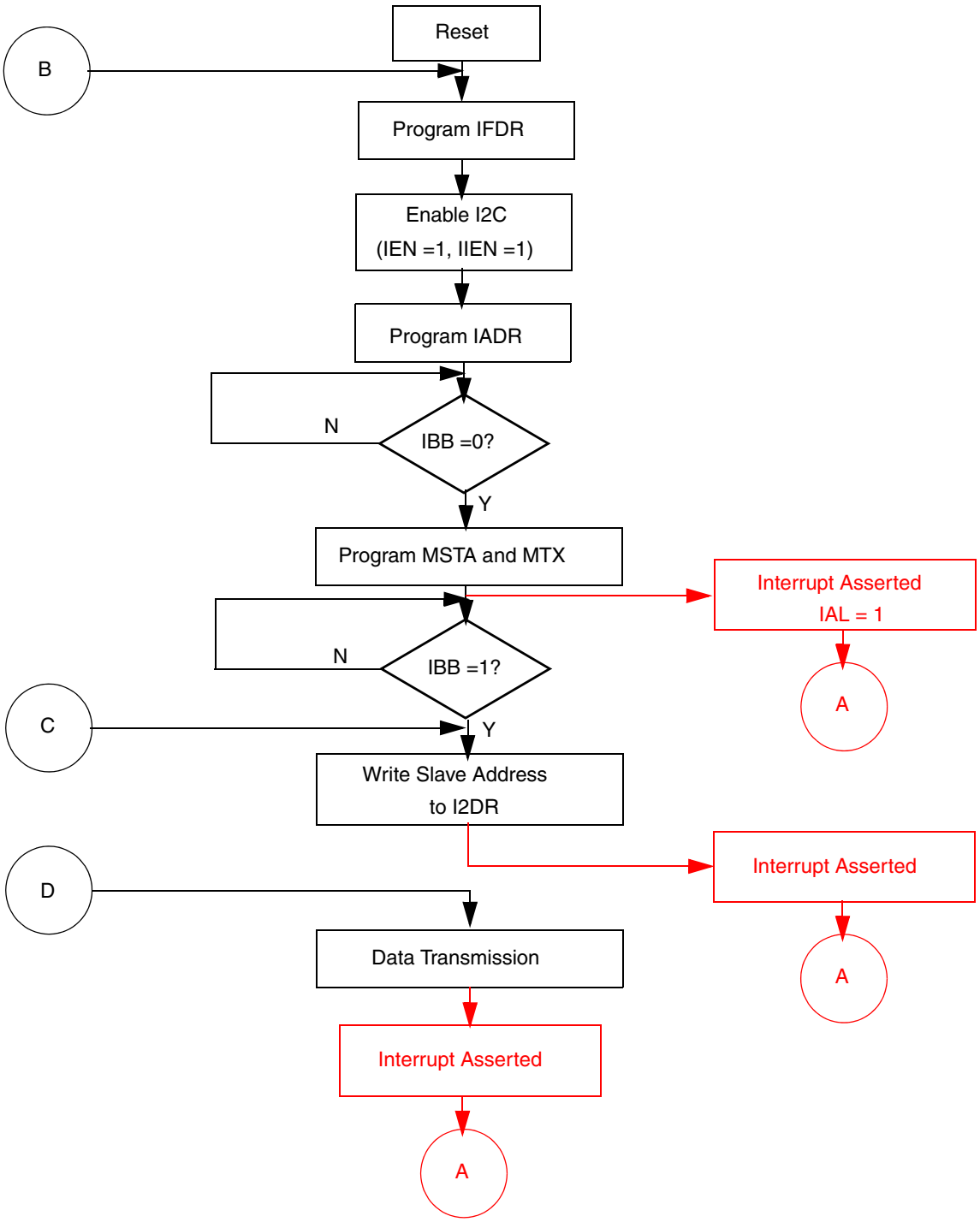


Figure 29-13. Detailed Flowchart of a Typical I^2C Master Tx Mode, Part 1

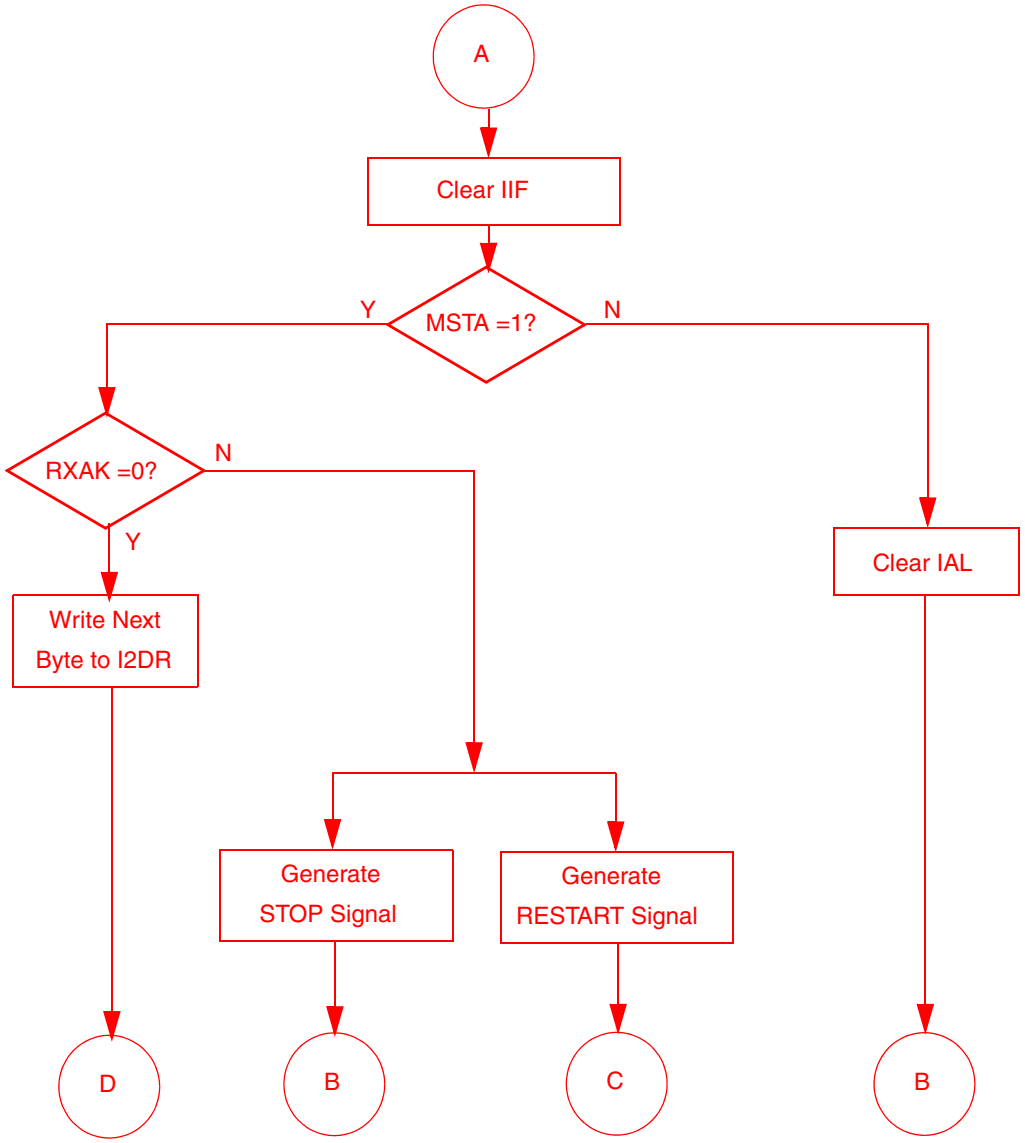


Figure 29-14. Detailed Flowchart of a Typical I^2C Master Tx Mode, Part 2

Figure 29-13 and Figure 29-14 show the Master Transmit mode operation with interrupt subroutine.

If an interrupt is generated while the MSTA bit is 0, then bus arbitration is lost and IAL is set. Software can then clear the IAL bit and reprogram the I^2C module.

An interrupt generated while MSTA bit is 1 is a transfer-done interrupt. Software can check the RXAK bit for data receive acknowledgement by the slave. If the RXAK bit is cleared, then software can generate STOP or RESTART by writing in I2CR register. Otherwise, the next data byte can be written to the I2DR register.

NOTE

The IBB bit is asserted by START condition on the bus, and it is negated by STOP condition on bus. Therefore, if arbitration is lost due to an unexpected STOP condition during transfer, then IBB is cleared. If arbitration is lost due to data mismatch, then it will not be cleared. Software should always clear the IEN bit and then set it if arbitration is lost.

29.6 Software Restriction

Software should take care that there is a delay of at least two Module Clock cycles after it sets the I2CR[RSTA] bit before writing to the I2DR register. Maximum possible clock period of Module Clock is 78 ns.

Chapter 30
IC Identification Module (IIM)

This chapter describes the features, registers, and operation of the IC identification module (IIM). The IIM provides an interface for reading, programming and overriding identification and control information stored in on-chip fuse elements.The IIM also provides a set of volatile software-accessible signals that can be used for software control of hardware elements that do not require non-volatility.

30.1 Overview

The IIM provides the primary user-visible mechanism for interfacing with on-chip fuse elements. Fuses are used to provide unique chip identifiers, mask revision numbers, cryptographic keys, and various control signals requiring permanent non-volatility. The IIM also provides up to 28 volatile control signals and the means to generate a second 168-bit key for the security controller (SCC).

The IIM consists of a master controller, a software fuse value shadow cache, and a set of registers to hold the values of signals visible outside the module and the hardware-visible shadow cache. Three banks of fuses (L-Fuses or eFUSEs) are associated with the IIM, but are instantiated outside it. Secure JTAG control and Device Unique ID are both in bank 0. The SCC key is in bank 1. The super-root key is in bank 2.

The IIM is accessible using an 8-bit IP bus interface that matches the natural width of the fuse arrays. All registers are 32-bit aligned, to allow the module to be instantiated on IP buses supporting only 32-bit peripherals. A subset of fuses, as well as the software-controlled volatile signals, are capable of driving top-level nets within the SoC. These signals are referred to as hardware-visible signals. These signals are intended for feature enabling and disabling and similar uses within the device.

IIM supports electrically-programmable poly fuses (eFUSEs). eFUSEs can be blown under software or JTAG control during IC final test, at the customer factory or in the field. eFUSEs include a mechanism to inhibit further blowing of fuses (write-protect), to support secure computing environments.

Fuse values can be overridden by software without modifying the fuse element. The override functionality can also be permanently disabled. Fuse banks can also be scan-inhibited on a per-bank basis to prevent reading and programming of fuses through the JTAG interface.

The fuses are divided into banks, with specific intended uses assigned to each bank. The different fuse banks are described in Section 30.4.1, “Fuse Functional Groups.”

30.1.1 Features

- Three independent fuse banks (Each bank is 256 bits. So the actual size is 256*3=768 bits.

- Support for driving secure JTAG challenge and response values to the secure JTAG controller (size of each field configurable using RTL parameter; challenge default size is 64 bits, response default size is 56 bits)
- Up to two distinct 168-bit 3DES keys provided from a single set of fuses
- Fuse values can be overridden by software, without affecting the fuse element—override capability can be permanently disabled on a per-bank basis
- eFUSEs can be write-protected on a per-bank basis
- Fuse scan protection (read and program) on a per-bank basis
- Fuses programmable by software, directly by JTAG or indirectly using JTAG using a processor
- Recommended signal assignments to maximize software reuse

30.1.2 Modes of Operation

The IIM is in its functional mode (all specified functionality available) any time it is out of reset and supplied with the proper clocks.

30.2 External Signal Description

The IIM has no external signals.

30.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers.

30.3.1 Memory Map

Table 30-1 shows the IIM memory map. For the base address of a particular module instantiation, see the system memory map.

All registers are 8 bits wide, but addressable on 32-bit boundaries. Reads of the upper 24 bits always return 0, and writes to these bits are ignored. Only the lower 8 bits (the usable bits) of each register are shown in subsequent diagrams.

Table 30-1. IIM Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (STAT)	Status register	R/W	0x– –	30.3.3.1/30-6
0x0004 (STATM)	Status IRQ mask register	R/W	0x00	30.3.3.2/30-6
0x0008 (ERR)	Module errors register	R/W	0x0 –	30.3.3.3/30-7
0x000C (EMASK)	Error IRQ mask register	R/W	0x0 –	30.3.3.4/30-8
0x0010 (FCTL)	Fuse control register	R/W	0x30	30.3.3.5/30-9
0x0014 (UA)	Upper address register	R/W	0x– 0	30.3.3.6/30-10
0x0018 (LA)	Lower address register	R/W	0x00	30.3.3.7/30-12

Table 30-1. IIM Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x001C (SDAT)	Explicit sense data register	R	0x00	30.3.3.8/30-12
0x0020 (PREV)	Product revision register	R	0x– –	30.3.3.9/30-13
0x0024 (SREV)	Silicon revision register	R	0x– –	30.3.3.10/30-13
0x0028 (PREG_P)	Program protection register	R/W	0x– –	30.3.3.11/30-14
Fusebox 0 (0x0800–0x087C)	Fuse Bank 0 (Secure JTAG control and Device Unique ID)	R/W	All 0's	30.3.3.16/30-15
Fusebox1 (0x0C00–0x0C7C)	Fuse Bank 1 (SCC key)	R/W	All 0's	30.3.3.17/30-15
Fusebox2 (0x1000–0x107C)	Fuse Bank 2 (Super-root key)	R/W	All 0's	30.3.3.18/30-15

30.3.2 Register Summary

Table 30-2 is the IIM register summary, and indicates the name and read/write accessibility for each bit.

NOTE

For bits denoted as write one to clear (w1c), writes of zero have no effect.

Table 30-2. IIM Register Summary

Bank	Offset (and Name Abbreviation)		7	6	5	4	3	2	1	0
Control and Status Registers	0x0000 (STAT)	R	BUSY						PRGD	SNSD
		W							w1c	w1c
	0x0004 (STATM)	R							PRGD_M	SNSD_M
		W								
	0x0008 (ERR)	R	PRGE	WPE	OPE	RPE	WLRE	SNSE	PARITYE	
		W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
	0x000C (EMASK)	R	PRGE_M	WPE_M	OPE_M	RPE_M	WLRE_M	SNSE_M	PARITY_M	
		W								
	0x0010 (FCTL)	R	DPC	PRG_LENGTH			0	0	0	0
		W					ESNS_N	ESNS_0	ESNS_1	PRG
	0x0014 (UA)	R			Address[13:8]					
		W								
	0x0018 (LA)	R	Address[7:0]							
		W								
	0x001C (SDAT)	R	Data[7:0]							
		W								
	0x0020 (PREV)	R	PRODUC_ REV[4:0]					PRODUCT_VT[2:0]		
		W								
	0x0024 (SREV)	R	SILICON_REV[7:0]							
		W								
	0x0028 (PREG_P)	R	PROTECTION_REG[7:0]							
		W								
FuseBox0	Fusebox 0 (0x0800–0x087C)	R	Words 0–31 of Bank0							
		W								
FuseBox1	Fusebox1 (0x0C00–0x0C7C)	R	Words 0–31 of Bank1							
		W								

Table 30-2. IIM Register Summary (continued)

Bank	Offset (and Name Abbreviation)		7	6	5	4	3	2	1	0
FuseBox2	Fusebox2 (0x1000–0x107C)	R	Words 0–31 of Bank2							
		W								

30.3.3 Register Descriptions

This section provides detailed descriptions of the IIM registers.

Register conventions: Figure 30-1 and Table 30-3 explain conventions used in register diagrams and tables.

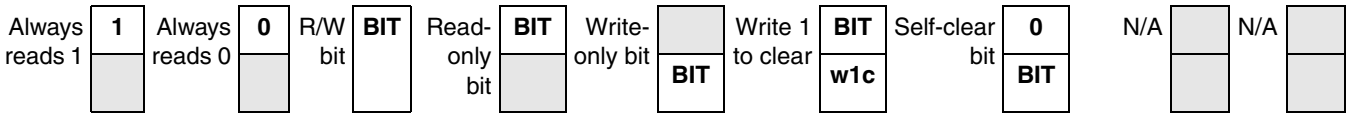


Figure 30-1. Key to Register Fields

Table 30-3. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that can be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one. Writing a zero to these bits has no effect.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

30.3.3.1 Status Register (STAT)

All module status information is read using the STAT register. Figure 30-2 shows the register, and Table 30-4 describes the register's fields.

Offset 0x0000 (STAT) Access: Supervisor read-only

	7	6	5	4	3	2	1	0
R	BUSY	0	0	0	0	0	PRGD	SNSD
W							w1c	w1c
Reset	0	0	—	—	—	—	—	0

Figure 30-2. Status Register

Table 30-4. Status Register Field Descriptions

Field	Description
7 BUSY	Indicates IIM is busy with a program or explicit sense cycle. Attempts to access IIM registers (other than STAT) while BUSY is asserted results in a bus error. 0 IIM is not busy with a program or explicit sense cycle 1 IIM is busy with a program or explicit sense cycle
6–2	Reserved
1 PRGD	Program done. Indicates eFUSE program operation is done. Assertion causes an interrupt request if PRGD_M is set in the status IRQ mask register. This bit is automatically set by hardware upon completion of an eFUSE program cycle. Software must clear the bit by writing 1 to it. 0 Program operation has not finished 1 Program operation has finished
0 SNSD	Explicit sense cycle done. Indicates that an explicit fuse sense cycle is done, and the data is available in the explicit sense data register. Assertion causes an interrupt request if SNSD_M is set in the status IRQ mask register. This bit is automatically set by hardware, Software must clear the bit by writing 1 to it. 0 No explicit sense cycle has finished 1 An explicit sense cycle has finished

30.3.3.2 Status IRQ Mask Register (STATM)

See Figure 30-3 for illustration of valid bits in the status IRQ mask register and Table 30-5 for description of the bit fields.

Offset 0x0004 (STATM) Access: Supervisor read/write

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PRGD_M	SNSD_M
W								
Reset	0	0	0	0	0	0	0	0

Figure 30-3. Status IRQ Mask Register

Table 30-5. Status IRQ Mask Register Field Descriptions

Field	Description
7–2	Reserved
1 PRGD_M	Program done mask. Masks or unmasks IRQ generation due to PRGD events. 0 PRGD events do not cause an IRQ 1 PRGD events cause an IRQ
0 SNSD_M	Explicit sense cycle done mask. Masks or unmasks IRQ generation due to SNSD events. 0 SNSD events do not cause an IRQ 1 SNSD events cause an IRQ

30.3.3.3 Module Errors Register (ERR)

See Figure 30-4 for illustration of valid bits in the module errors register and Table 30-6 for description of the bit fields.

Offset 0x0008 (ERR) Access: Supervisor read/write

	7	6	5	4	3	2	1	0
R	PRGE	WPE	OPE	RPE	WLRE	SNSE	PARITYE	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	0	0	0	0	0	0	—	u

Figure 30-4. Module Errors Register

Table 30-6. Module Errors Register Field Descriptions

Field	Description
7 PRGE	Program error. Indicates that an eFUSE programming operation ended in failure. Assertion causes an interrupt request if PRGE_M is set in the error IRQ mask register. This bit is automatically set by hardware, and must be cleared by software by writing 1 to it. 0 No fuse programming operation has finished with error 1 Fuse programming operation has finished with an error
6 WPE	Write protect error. Indicates that an eFUSE programming operation was attempted to a write-protected fuse bank, or a locked word, or when the value of PRG_P is not 0xAA. Assertion causes an interrupt request if WPE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No write-protect error has occurred 1 A write-protect error has occurred
5 OPE	Override protect error. Indicates that an attempt was made to override the values in an override-protected fuse bank or locked word. Assertion causes an interrupt request if OPE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No override-protect error has occurred 1 Override-protect error has occurred

Table 30-6. Module Errors Register Field Descriptions (continued)

Field	Description
4 RPE	Read protect error. Indicates that an attempt was made to read values from a read-protected fuse bank or SCC. Assertion causes an interrupt request if RPE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No read-protect error has occurred 1 Read-protect error has occurred
3 WLRE	Write to locked register error. Indicates an attempt was made to write to a locked SCS register. Assertion causes an interrupt request if WLRE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No write-to-locked-register error has occurred 1 Write-to-locked-register error has occurred
2 SNSE	Explicit sense cycle error. Indicates that an explicit fuse sense was refused because either the explicit sense protection is asserted for that fuse bank, or more than one of the fuse control register bits ESNS_N, ESNS_1, ESNS_0, PRG are set at the same time. Setting this bit causes an interrupt request, if SNSE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No explicit sense error has occurred 1 Explicit sense error has occurred
1 PARITYE	Parity error of cache. Indicate that an parity error was detected in hardware fuse cache or software fuse cache. Assertion causes an interrupt request if PARITYE_M is set in the error IRQ mask register. This bit is automatically set by hardware and must be cleared by software by writing 1 to it. 0 No parity error has been detected 1 Parity error has been detected
0	Reserved

30.3.3.4 Error IRQ Mask Register (EMASK)

See Figure 30-5 for illustration of valid bits in the error IRQ mask register, and Table 30-7 for description of the bit fields.

Offset 0x000C (EMASK) Access: Supervisor read/write

	7	6	5	4	3	2	1	0
R	PRGE_M	WPE_M	OPE_M	RPE_M	WLRE_M	SNSE_M	PARITYE_M	0
W								
Reset	0	0	0	0	0	0	0	u

Figure 30-5. Error IRQ Mask Register

Table 30-7. Error IRQ Mask Register Field Descriptions

Field	Description
7 PRGE_M	Program error mask. Masks or unmasks IRQ generation due to PRGE events. 0 PRGE events do not cause an IRQ 1 PRGE events cause an IRQ
6 WPE_M	Write protect error mask. Masks or unmasks IRQ generation due to WPE events. 0 WPE events do not cause an IRQ 1 WPE events cause an IRQ
5 OPE_M	Override protect error mask. Masks or unmasks IRQ generation due to OPE events. 0 OPE events do not cause an IRQ 1 OPE events cause an IRQ
4 RPE_M	Read protect error mask. Masks or unmasks IRQ generation due to RPE events. 0 RPE events do not cause an IRQ 1 RPE events cause an IRQ
3 WLRE_M	Write to locked register error mask. Masks or unmasks IRQ generation due to WLRE events. 0 WLRE events do not cause an IRQ 1 WLRE events cause an IRQ
2 SNSE_M	Explicit sense cycle error mask. Masks or unmasks IRQ generation due to SNSE events. 0 SNSE events do not cause an IRQ 1 SNSE events cause an IRQ
1 PARITYE_M	Parity error of cache mask. Masks or unmasks IRQ generation due to PARITYE events. 0 PARITYE events do not cause an IRQ 1 PARITYE events cause an IRQ
0	Reserved

30.3.3.5 Fuse Control Register (FCTL)

Bits [0] and [7:4] of the FCTL register can be written in supervisor mode only: write attempts in user mode are silently ignored. Reads from these bits in user mode are allowed. Bits [3:1] of the FCTL register can be accessed in user mode.

See Figure 30-6 for illustration of valid bits in the fuse control register and Table 30-8 for description of the bit fields.

Offset 0x0010 (FCTL) Access: Mixed read/ supervisor write

	7	6	5	4	3	2	1	0
R	DPC	PRG_LENGTH[2:0]			0	0	0	0
W					ESNS_N	ESNS_0	ESNS_1	PRG
Reset	0	0	1	1	0	0	0	0

Figure 30-6. Fuse Control Register

Table 30-8. Fuse Control Register Field Descriptions

Field	Description
7 DPC	Delayed program cycle. This bit is a control bit, selecting immediate or delayed program. When this bit is cleared, program cycles start immediately upon setting of PRG bit. When this bit is asserted, program cycles are delayed until the input signal is asserted. 0 Program cycles begin immediately upon setting of the PRG bit 1 Program cycles are delayed; they do not begin until delayed_pgm_start signal is asserted
6–4 PRG_LENGTH[2:0]	Program length. These bits define the length of program pulse, in units of 32 kHz clock cycles.
3 ESNS_N	Explicit sense - normal. Writing 1 to this bit initiates an unstressed (normal) explicit sense cycle. Writing 0 has no effect. Reads always return zero. This bit is cleared automatically by hardware when the sensing operation completed. Only one of ESNS_N, ESNS_0, ESNS_1, PRG can be set at one time—otherwise the module errors register's SNSE bit is set to indicate this error. 0 All reads return 0. Writes of 0 have no effect 1 Writing 1 Initiates an unstressed explicit sense cycle
2 ESNS_0	Explicit Sense (0-stressed). Writing 1 to this bit initiates a 0-stressed explicit sense cycle. Reads of this bit always return zero. This bit is cleared automatically by hardware when the sensing operation is completed. Only one of ESNS_N, ESNS_0, ESNS_1, PRG can be set at one time—otherwise the module errors register's SNSE bit is set to indicate this error. 0 All reads return 0. Writes of 0 have no effect 1 Writing 1 initiates an 0-stressed explicit sense cycle
1 ESNS_1	Explicit sense (1-stressed). Writing 1 to this bit initiates a 1-stressed explicit sense cycle. Reads of this bit always return zero. This bit is cleared automatically by hardware when the sensing operation is completed. Only one of ESNS_N, ESNS_0, ESNS_1, PRG can be set at one time—otherwise the module errors register's SNSE bit is set to indicate this error. 0 All reads return 0. Writes of 0 have no effect 1 Writing 1 initiates an 1-stressed explicit sense cycle
0 PRG	Fuse program. Writing 1 to this bit initiates a fuse programming cycle. Reads of this bit always return zero. This bit is cleared automatically by hardware when the programming operation is completed. Only one of ESNS_N, ESNS_0, ESNS_1, PRG can be set at one time—otherwise the module errors register's SNSE bit is set to indicate this error. 0 All reads return 0. Writes of 0 have no effect 1 Writing 1 initiates a fuse programming cycle

30.3.3.6 Upper Address Register (UA)

The UA register contains the upper part of the address of the eFUSE bit to be programmed or word to be sensed in an explicit sense cycle. Programming is done by bit, so the program address is a full-bit address. Sensing is done on 8-bit words, so the bottom three bits of the address are ignored.

See Figure 30-7 for illustration of valid bits in the upper address register and Table 30-9 for description of the bit fields.

Offset 0x0014 (UA) Access: Supervisor read/write

	7	6	5	4	3	2	1	0
R	0	0	A[13:8]					
W								
Reset	—	—	0	0	0	0	0	0

Figure 30-7. Upper Address Register

Table 30-9. Upper Address Register Field Descriptions

Field	Description
7–6	Reserved
5–0 A[13–8]	Upper six bits of the address of the eFUSE bit to be programmed or word to be sensed in an explicit sense cycle. The address must be written prior to initiating a programming/sensing operation (by setting the PRG or ESNS_*x* bit in the fuse control register). A[13:11] select the fuse bank, while A[10:8] provide the most significant portion of the row address within the bank.

30.3.3.7 Lower Address (LA)

The LA register contains the lower 8 bits of the address of the eFUSE bit to be programmed or word to be sensed in an explicit sense cycle.

See Figure 30-8 for illustration of valid bits in the lower address register and Table 30-10 for description of the bit fields.

Figure 30-8. Lower Address Register

Table 30-10. Lower Address Register Field Descriptions

Field	Description
7–0 A	Lower eight bits of the address of the eFUSE bit to be programmed or word to be sensed in an explicit sense cycle. The address must be written prior to initiating a programming or sensing operation (by setting the PRG or ESNS_*x* bit in the fuse control register). A[7:3] provides the least-significant portion of the row address. A[2:0] selects the bit position within the selected row (in sensing operations, A[2:0] is ignored).

30.3.3.8 Explicit Sense Data Register (SDAT)

See Figure 30-9 for illustration of valid bits in the explicit sense data register and Table 30-11 for description of the bit fields.

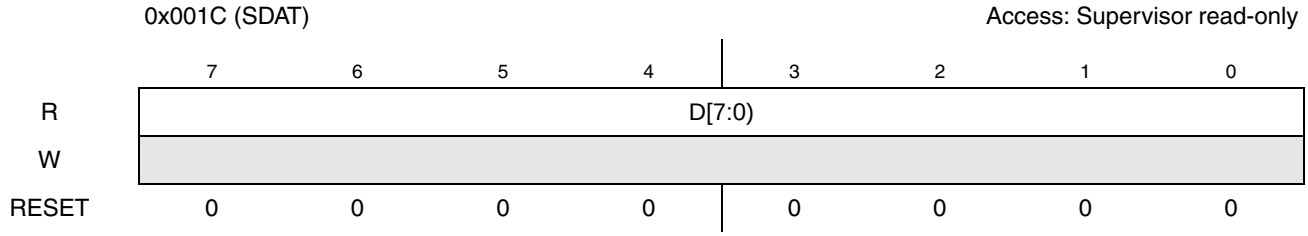


Figure 30-9. Explicit Sense Data Register

On an explicit sense cycle, the data sensed from the fuses is placed in this register at the conclusion of the sense cycle. Software can read the SNSD bit in the STAT register to determine when the cycle is complete.

Table 30-11. Explicit Sense Data Register Field Descriptions

Field	Description
7–0 D	Data sensed from the fuses.

30.3.3.9 Product Revision Register (PREV)

The PREV register contains the product revision which corresponds to the top eight bits of the deprecated HW_REV register.

See Figure 30-10 for illustration of valid bits in the product revision register and Table 30-12 for description of the bit fields.

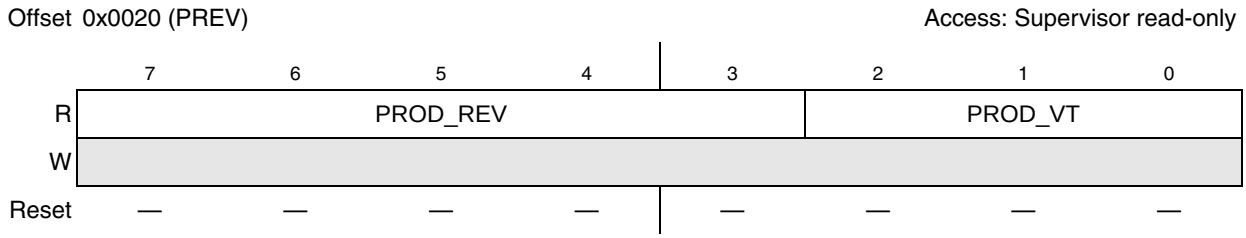


Figure 30-10. Product Revision Register

Table 30-12. Product Revision Register Field Descriptions

Field	Description
7–3 PROD_REV	Product revision. The product revision (specific to the product or product family). Setting according to product revision.
2–0 PROD_VT	Product vendor or technology. The product vendor and/or technology (specific to the product or product family). Setting according to product vendor/technology.

30.3.3.10 Silicon Revision Register (SREV)

The SREV register contains the silicon revision (also called mask revision). This corresponds to the bottom 8 bits of the deprecated HW_REV register.

See Figure 30-11 for illustration of valid bits in the silicon revision register and Table 30-13 for description of the bit fields.

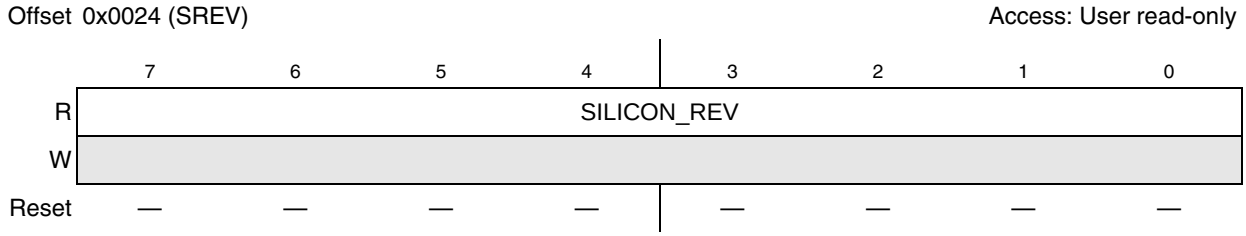


Figure 30-11. Silicon Revision Register

Table 30-13. Silicon Revision Register Field Descriptions

Field	Description
7–0 SILICON_REV	Mask set revision. The mask set revision used in fabrication of the part. The value changes with each change to the mask set. Setting according to mask set revision.

30.3.3.11 Program Protection Register (PRG_P)

This register is used to protect against accidental fuse programming. Fuses can be blown only when the value of this register is 0xAA. Software must only program this register to 0xAA while actively blowing fuses. After the programming operation is complete, this register must be immediately reprogrammed to a different value.

See Figure 30-12 for illustration of valid bits in the program protection register, and Table 30-14 for description of the bit fields.

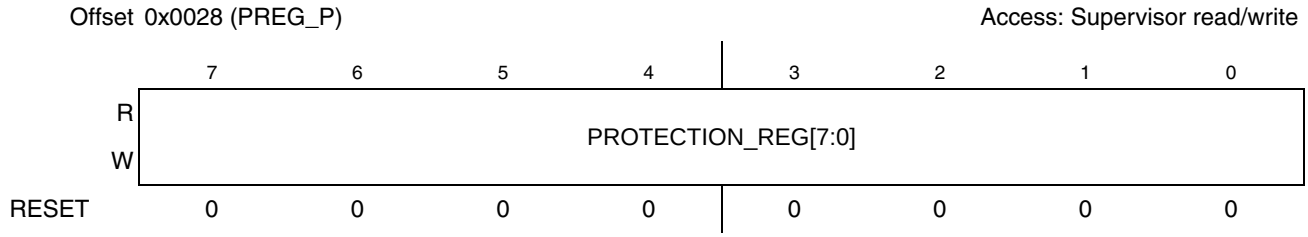


Figure 30-12. Program Protection Register

Table 30-14. Program Protection Register Field Descriptions

Field	Description
7–0 PROTECTION_REG	Fuses can be blown only when the value of this register is 0xAA. Otherwise, an attempt to program fuses terminates with error, and the WPE bit in the module errors register is asserted. 0xAA Fuses can be programmed All other values: Fuses cannot be programmed

30.3.3.12 Software-Controllable Signals Register 0 (SCS0)

This register is physically located in the hardware-visible signals submodule. It implements software-controlled, volatile signals that can drive SoC-level nets for feature enabling.

30.3.3.13 Software-Controllable Signals Register 1 (SCS1)

This register is physically located in the hardware-visible signals submodule. It implements software-controlled, volatile signals which can drive SoC-level nets for feature enabling.

30.3.3.14 Software-Controllable Signals Register 2 (SCS2)

This register is physically located in the hardware-visible signals submodule. It implements software-controlled, volatile signals which can drive SoC-level nets for feature enabling.

30.3.3.15 Software-Controllable Signals Register 3 (SCS3)

This register is physically located in the hardware-visible signals submodule. It implements software-controlled, volatile signals which can drive SoC-level nets for feature enabling.

30.3.3.16 Fusebox 0 (0x0800–0x087C)

See the fusemap for detailed definitions. Override is prohibited in user mode.

30.3.3.17 Fusebox1 (0x0C00–0x0C7C)

See the fusemap for detailed definitions. Override is prohibited in user mode.

30.3.3.18 Fusebox2 (0x1000–0x107C)

See the fusemap for detailed definitions. Override is prohibited in user mode.

30.4 Functional Description

The IIM is an 8-bit IP bus peripheral that implements interfaces for eFUSEs and contains hardware revision codes, cryptographic keys, and miscellaneous system-level feature enabling circuitry. Three banks, each with up to 2048 fuses, are supported.

30.4.1 Fuse Functional Groups

This section describes fuse functions that are distinctive to each of the three fuse banks. access protection is specified on a per-bank basis; this can limit the uses of the unspecified fuses.

30.4.1.1 Fuse Bank 0: Secure JTAG Control and Device Unique ID

30.4.1.1.1 Secure JTAG Control

Fuse bank 0 contains the secure JTAG control fuses, which are hardware-visible control signals. These fuses are used to control JTAG access to secured resources and provide a Device Unique ID.

30.4.1.1.2 Device Unique ID

Fuse Bank 2 contains the device’s Unique ID (UID) consisting of the fab ID, lot and wafer number, and die coordinates. The first 64 bits of this bank are usable by operating systems that require a unique device ID.

30.4.1.2 Fuse Bank 1: SCC Key

Fuse Bank 1 contains the first 168-bit SCC key.

30.4.1.2.1 SCC Key Checking

The SCC checks the key using the Hamming code.

30.4.1.3 Fuse Bank 2: Super-Root Key

Fuse Bank 2 is intended to hold data relevant to the super-root key (SRK).

30.4.1.4 Software-Controllable Volatile Signals

The IIM implements up to 28 software-controllable, hardware-visible, volatile control signals. These are implemented in four 8-bit registers, each with a lock bit to inhibit modification of the associated register until the IIM is reset. These 28 signals can all drive SoC-level nets. They are always software-readable, but can only be modified by software if the LOCK bit (bit 7) is not set in the register. The sequence for modifying these bits is shown in Figure 30-13.

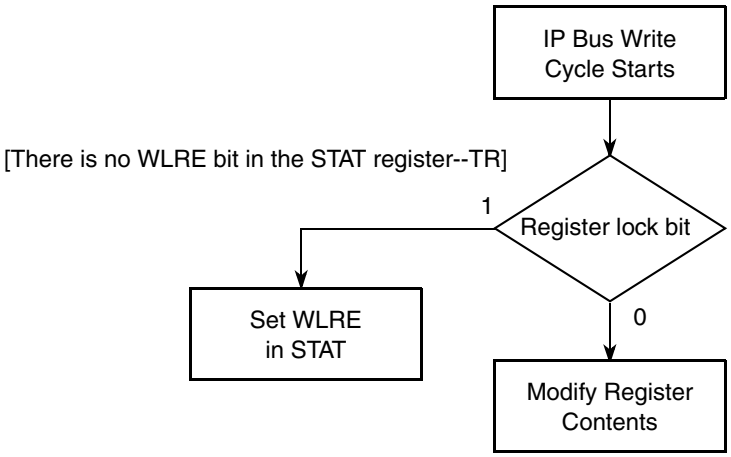


Figure 30-13. SCS Register Write Sequence

30.4.2 Fusebox Interface

The IIM is designed to mate with three fuseboxes, each consisting of up to 2048 eFUSEs.

30.4.2.1 Fusebox Signals

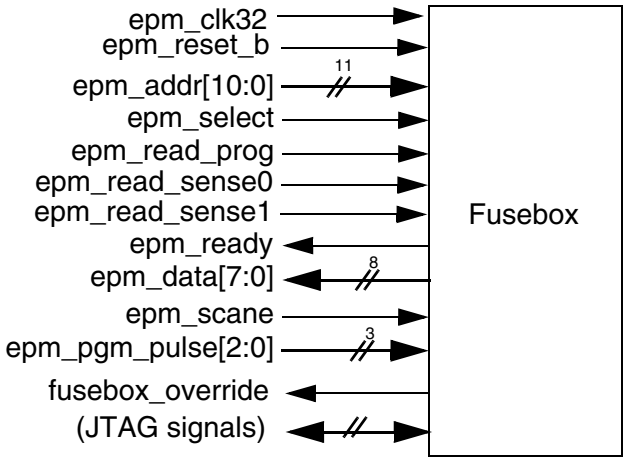


Figure 30-14. Fusebox Interface

The signals are summarized from the fusebox's perspective in Table 30-15.

Table 30-15. Fusebox Signal Overview

Signal Name	Signal Type	Signal Description
epm_clk32	Input	The 32.768 kHz clock used in the fusebox to time the program operation. This clock is not utilized in the IIM module.
epm_reset_b	Input	Reset signal. The reset is the early reset that negates 200 μs before system reset.
epm_addr[10:0]	Input	This bus provides the address for a read or program operation. Read operations are done on a word (8-bit) basis, so the word address is carried on address[10:3]. Program operations are done on a bit-basis, so the bit address is carried on address[10:0]. Address bits [2:0] carry the bit location in the byte.
epm_select	Input	This signal starts a read/program operation. The fusebox senses the rising edge of this signal to start the operation. On the rising edge, the address, operation to perform and program data are latched. This signal must remain asserted throughout the read/program operation. There is one select signal for each fuse bank.
epm_read_prog	Input	This signal selects whether the current operation is a read or program cycle. This signal is latched into the fusebox on the rising edge of epm_select.
epm_read_sense0	Input	This signal allows sensing of the 0 (unblown) state to be stressed. When asserted (high), the sense circuitry is configured to stress the sensing of the 0 (unblown) state. This can be useful in detecting marginally blown fuses.This signal is latched by the fusebox on the rising edge of epm_select.
epm_read_sense1	Input	This signal allows sensing of the 1 (blown) state to be stressed. When asserted (high), the sense circuitry is configured to stress the sensing of the 1 (blown) state. This can be useful in detecting marginally blown fuses. This signal is latched by the fusebox on the rising edge of epm_select.
epm_ready	Output	This signal indicates the progress of the current read/program cycle. De-asserted (low) indicates that the operation is in progress. Asserted (high) indicates that the operation is in progress. The IIM must wait for the fusebox to assert this signal before sending another command to the fusebox.
epm_data[7:0]	Output	This bus carries the read data from the fusebox during a read cycle.
epm_scane	Input	This signal indicates whether the fusebox is scannable using JTAG. De-asserting this signal (that is, tying it or driving it low) inhibits any scan access to the fusebox.
epm_pgm_length[2:0]	Input	These signals define the length of the program pulse.
fusebox_override	Output	Fuse box override signal to IIM
sjc_tdi	Input	TDI serial input from the JTAG
sjc_capture_dr	Input	Capture data control signal from JTAG
sjc_update_dr	Input	Update data control signal from JTAG
sjc_shift_dr	Input	Shift data control from JTAG
sjc_tck	Input	TCK clock input

Table 30-15. Fusebox Signal Overview (continued)

Signal Name	Signal Type	Signal Description
ipt_sjc_fusebox_serial_ac_en	Input	Fuse box channel access enable signal from JTAG
fusebox_tdo	Output	TDO output from the fusebox.

30.4.2.2 Fusebox Operations

All fusebox operations are self-timed and begin relative to the rising edge on the select line. Most interface signals are shared by all four fuseboxes. Only the fusebox selects (epm_select), read data (epm_data) and ready status (epm_ready) signals are unique to each fusebox. The 32 kHz clock is not used within the IIM. It is passed through to the fusebox to time program operations. The clock distributed to the fuseboxes can be gated on only during fusebox operations.

30.4.2.2.1 Word Read

Read operations are done on a word (8-bit) basis. The target word is specified by the address lines epm_addr[10:3]. The bottom three address lines (bit-select lines) are ignored in a read operation.

The timing for fuse word read cycles is shown in Figure 30-15. This timing diagram shows one read cycle followed by the start of a second read, demonstrating how quickly one read can follow another. Each read command is registered on the rising edge of epm_select[x]. Each fusebox has its own select line and the index [x] corresponds to the selected fusebox. The address (epm_addr[10:3]) and read command (epm_read_prog) must be stable t_{sels} before the rising edge of epm_select and must remain stable at least t_{selh} hold time beyond the rising edge. epm_ready is negated t_{nrdy} after the command is registered, indicating that the fusebox is busy. After a t_{dv} delay the data is made available on epm_data[7:0] and the epm_ready acknowledge signal is asserted t_{ready} after the initial assertion of epm_select[x]. The IIM synchronizes epm_ready and the data to the ipg_clk_s clock and drives the synchronized data onto the IP bus. This time is signified by t_{sync} in Figure 30-15 and is determined by the IIM design. Once the read data has been latched, epm_select is negated by the IIM. epm_select must remain negated a minimum cycle to cycle delay of t_{ccd} before the next cycle (read or program) can begin. epm_data remains at a steady state until replaced by data from a subsequent read cycle. See Table 30-16 for specific timing values.

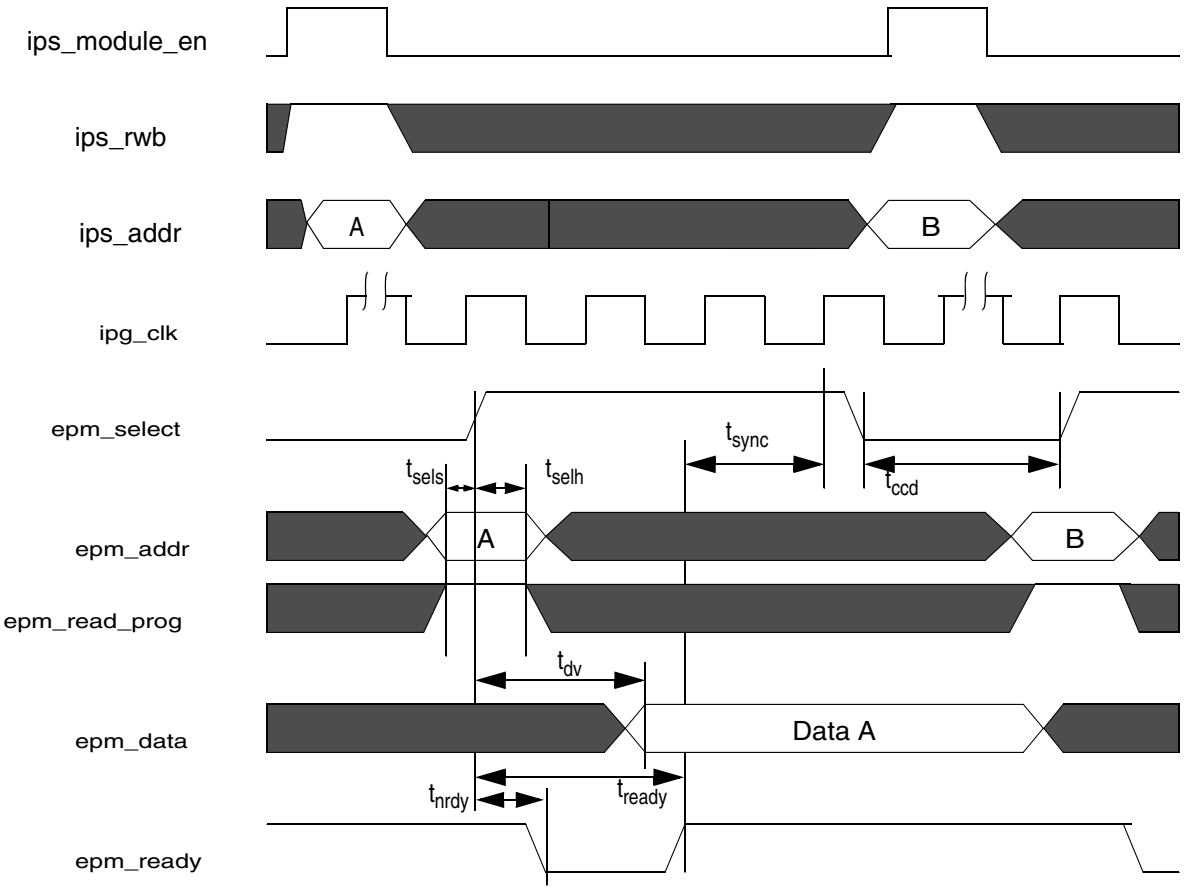


Figure 30-15. Fuse Box Read Cycle Timing

30.4.2.2.2 Fuse Programming

Fuses are programmed on a per-bit basis. The target bit is specified by the full address bus (epm_addr[10:0]). Program operations can only blow fuses (change them from logic 0 to logic 1), so no program data is required.

The timing for a program operation is shown in Figure 30-16. As in the read cycle, the address and program command are latched on the rising edge of epm_select. Both address and data must be setup t_{sels} ahead of the rising edge and remain held t_{selh} past the edge. Once the program command is registered, the fusebox negates the epm_ready signal after a delay of t_{nrdy}. On the next rising edge of the 32 kHz clock, the fusebox initiates the program operation. After the specified number of 32 kHz clocks (determined by epm_pgm_length[2:0]), the fusebox completes the program operation and assert the epm_ready signal. The IIM synchronizes the ready response to the ipg_clock during the period t_{sync}. After recognizing that the program operation has completed, the IIM negates the epm_select line. See Table 30-16 for specific timing values.

Table 30-16. Timing Values

Description	Designator	Timing Value
Setup time to select active edge	t_{sels}	2 ns
Hold time after select active edge	t_{selh}	2 ns
Delay from select to ready negation	t_{nrdy}	250 ps
Data valid after select active edge	t_{dv}	50–100 ns
Ready signal asserted after select active edge (read only)	t_{ready}	tdv + 2 ns
Ready signal asserted after select active edge (program only)	t_{ready}	(epm_program_length) * $period_{32\ kHz}$
Data and ready IP clock synchronization delay	t_{sync}	Average 2 IP bus clocks
Minimum cycle to cycle delay	t_{ccd}	5 ns

Figure 30-16 shows the timing of a fuse program operation.

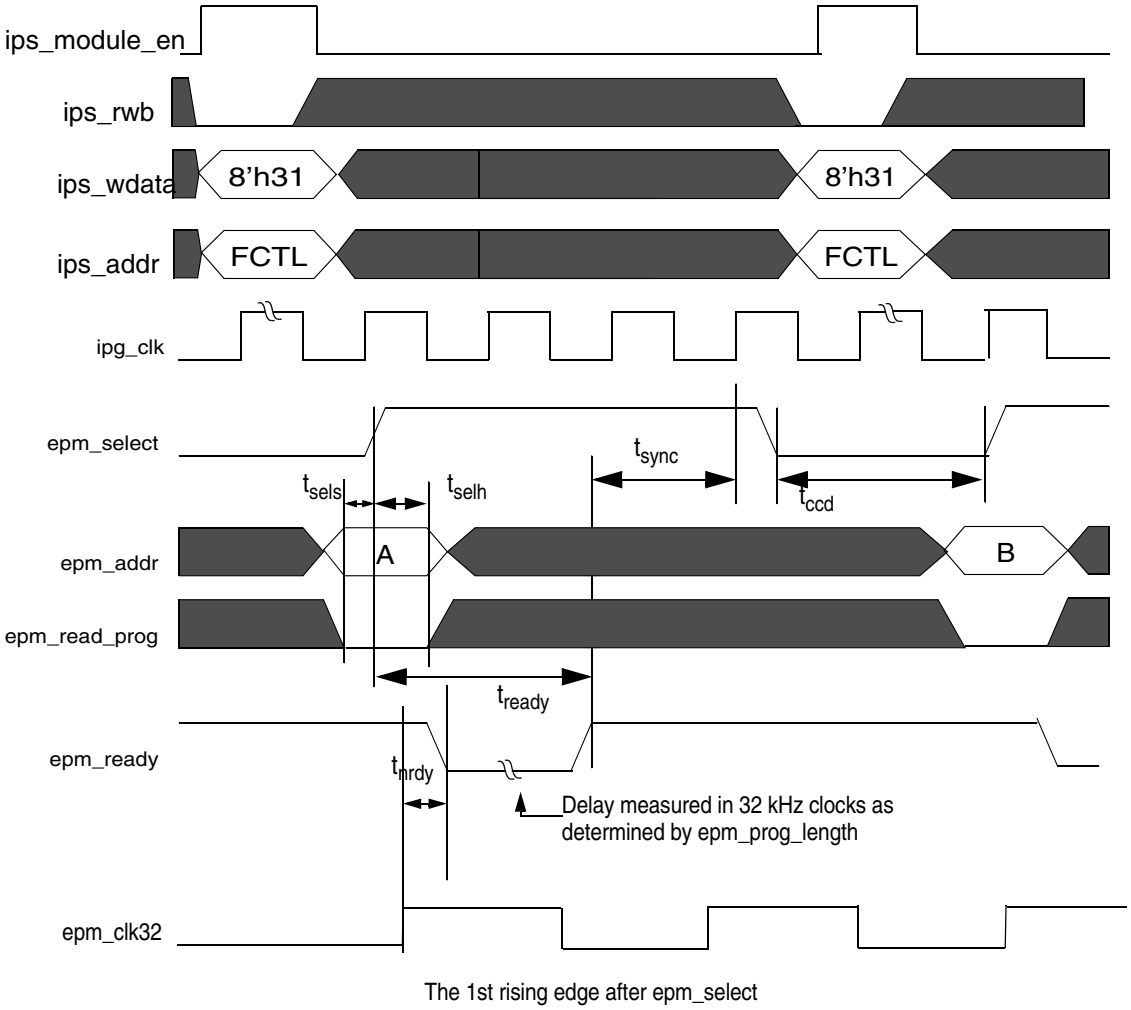


Figure 30-16. Fuse Program Cycle Timing

30.4.3 Fuse Value Caching

The values of fuses can be read from one of the following locations:

- The software fuse value shadow cache
- The hardware-visible fuse value shadow cache, driving SoC nets
- The fuse elements themselves

Fuse values are cached to reduce the risk of accidental programming of eFUSEs due to repeated reads, and to reduce power consumption associated with sense cycles.

The following two subsections describes the hardware-visible and software fuse value shadow caches, respectively.

30.4.3.1 Hardware-Visible Fuse Shadow Cache

The hardware-visible fuses include fuses in bank0 and bank1, and the fuse bank access control (FBAC) words of all three banks. When the IIM comes out of reset, it senses these fuses and writes their values to the appropriate registers. The IIM also ensures than any changes in fuse values are immediately reflected in the registers.

Figure 30-17 shows the hardware cache word format, which includes a parity bit. The overall hardware-visible word read sequence is shown in Section 30.4.5.1, “Read Sequence.”

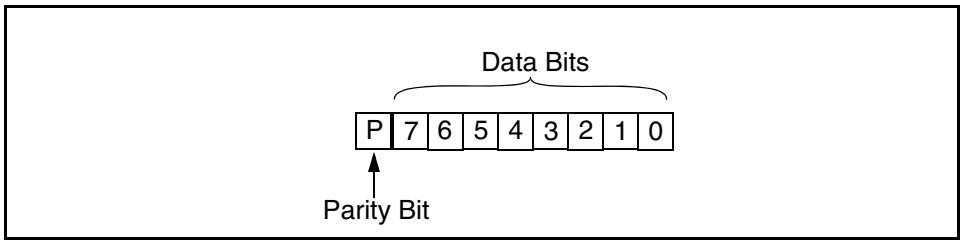


Figure 30-17. Hardware Cache Word Format

If a parity error is detected on the shadow cache, the IIM sets the error bit PARITYE and re-initializes the cache from fuses.

30.4.3.2 Software Fuse Value Shadow Cache

All fuses that are not hardware-visible have their values reflected in the software fuse value shadow class.

Figure 30-18 shows the software fuse value cache word format, which includes a valid bit and a parity bit. The parity bit reflects the parity of the data bits, and the valid bit is set if the data bits and parity bit have been set according to the sensed state of the corresponding fuses.

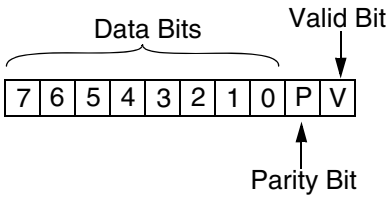


Figure 30-18. Software Fuse Value Cache Word Format

Cache words and fuse words are one-to-one mapped. All cache word bits are set to 0 when IIM comes out of reset.

When reading a non-hardware-visible fuse word, the IIM first attempts to read the word from the cache. A sense cycle is only run to the fuses if the valid bit is not set or the parity is incorrect. The overall cached fuse word read sequence is shown in Section 30.4.5.1, “Read Sequence.”

30.4.4 Fuse Protection

This section describes different features of the device which provide protection for the fuse banks.

30.4.4.1 Fuse Bank Protection Fuses

Each bank has fuses to protect the bank’s access to programming, override, read, scanning, and explicit sensing. Table 30-17 shows the fuse bank protection fuses and their functions.

Table 30-17. Fuse Bank Protection Fuses

Fuse Acronym	Fuse Name	Function
FBWP	Fuse bank write protect	Controls whether the bank’s fuses can be programmed
FBOP	Fuse bank override protect	Controls whether the bank’s fuses can be overridden
FBRP	Fuse bank read protect	Controls whether the bank’s fuses can be read
FBSP	Fuse bank scan protect	Controls whether the bank’s fuses can be scanned
FBESP	Fuse bank explicit sense protect	Controls whether the bank’s fuses can be explicit sensed

30.4.4.2 Word Lock Bits

The fuse bank protection fuses SJC_CHALL, SCC_KEY, and SJC_RESP all have corresponding word lock bits, which control override or programming capability. Table 30-18 shows the word lock bits and their location in the fuse bank access control words (FBAC).

Table 30-18. Fuse Bank Protection Fuses

Word	Word Lock Bit	Location	Comments
SJC_CHALL	TEST_LOCK	Bit 4 of FBAC0	—
SCC_KEY	SCC_LOCK	Bit 0 of FBAC1	SCC_KEY is non-readable by default
SJC_RESP	SJC_RESP_LOCK	Bit 1 of FBAC1	—

30.4.4.3 Scan-Out Protection

To prevent secret keys from being scanned out, specific logic is added to reset all the flip-flops in the IIM before entering scan mode. Figure 30-19 shows the scan-out protection circuit.

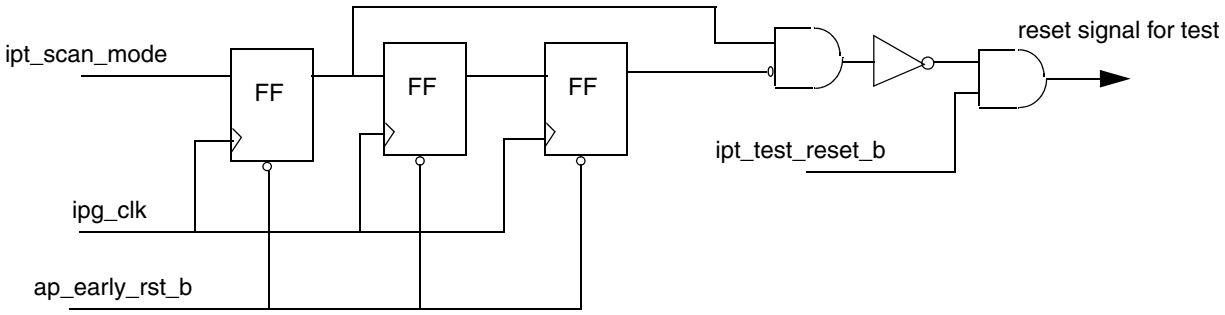


Figure 30-19. Scan-Out Protection Circuit

30.4.5 Fuse Bank Operations

30.4.5.1 Read Sequence

Fuses can be read using the IP bus interface from any bank which is not read-protected (that is, the bank's FBRP bit is unblown). Figure 30-20 and Figure 30-21 show read sequences for hardware-visible and software fuse words, respectively.

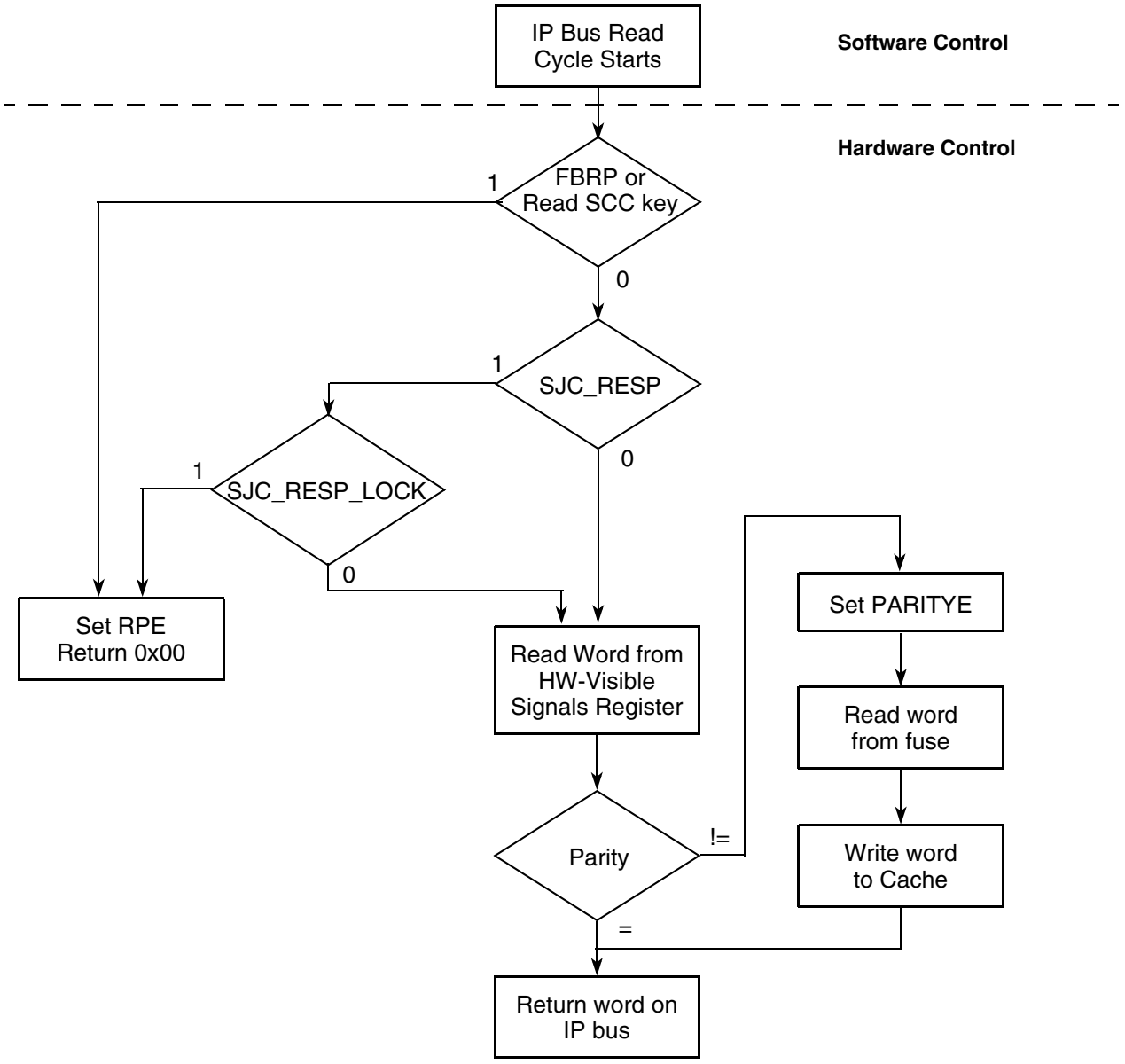


Figure 30-20. Hardware-Visible Fuse Read

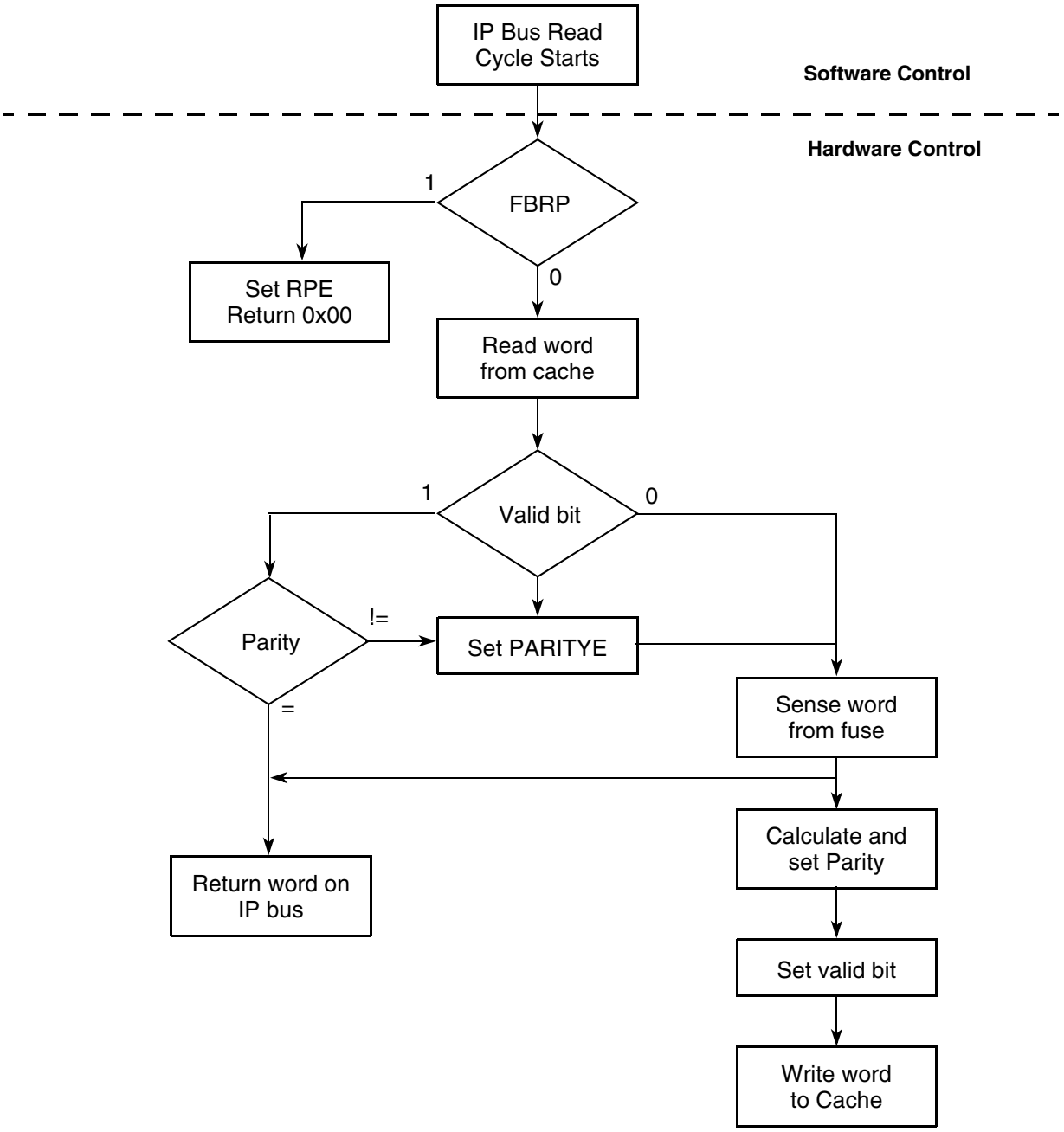


Figure 30-21. Software Fuse Read

30.4.5.2 Explicit Sense Sequence

The explicit sense sequence using the IP bus is depicted in Figure 30-22.

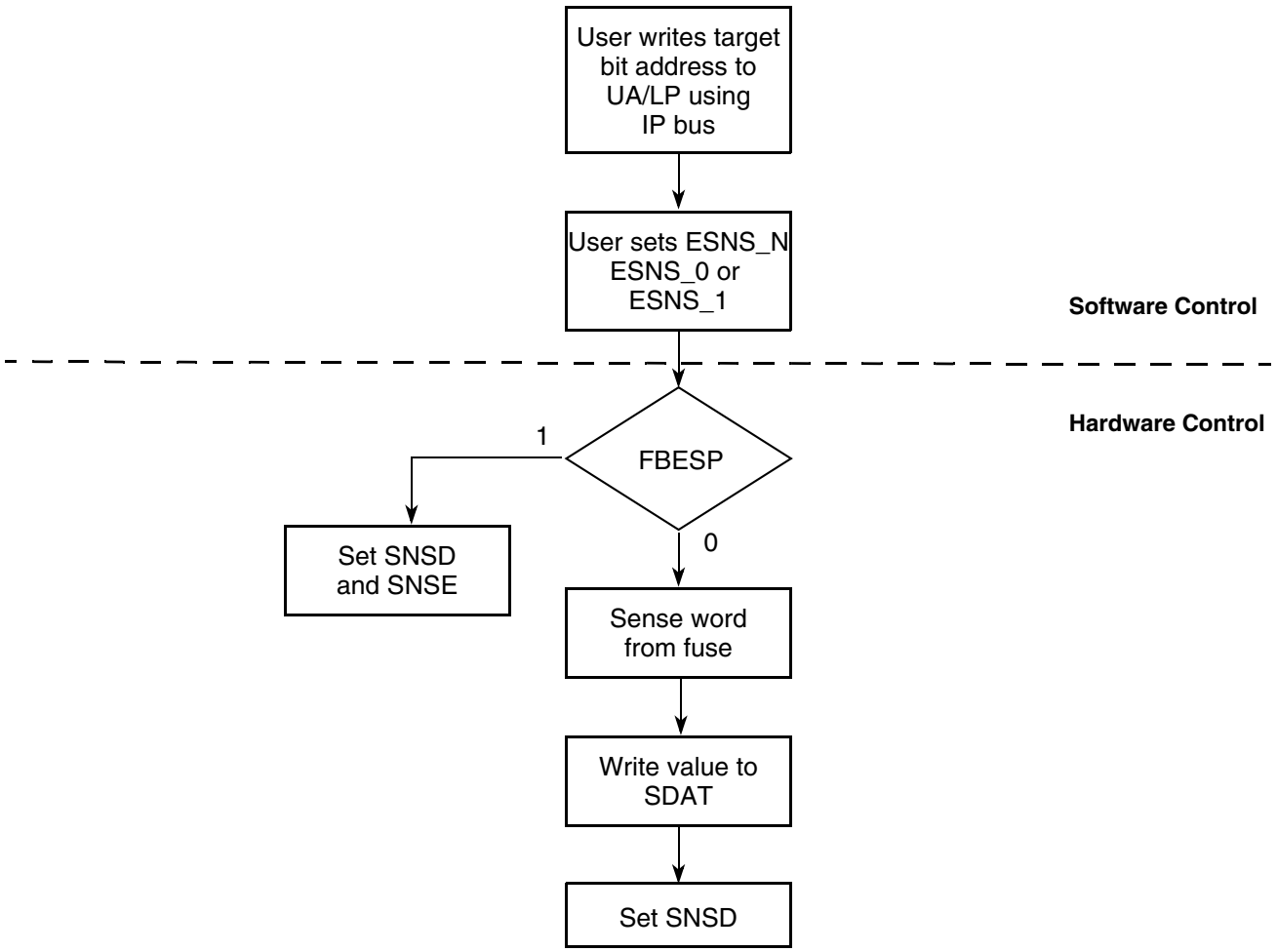


Figure 30-22. Explicit Sense Sequence

30.4.5.3 Programming Sequence

The software-controlled eFUSE programming sequence using the IP bus is depicted in Figure 30-23.

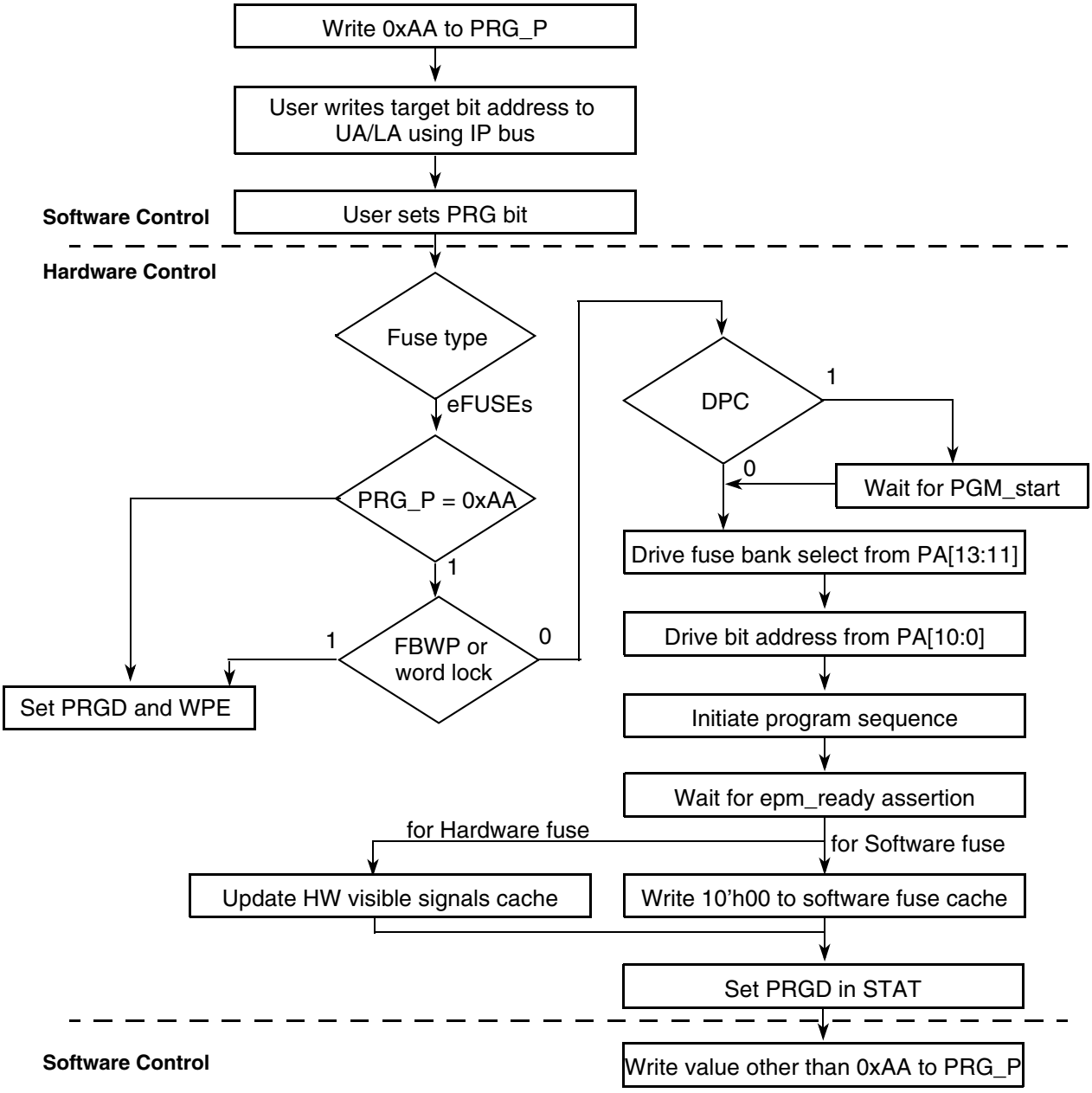


Figure 30-23. eFUSE Program Sequence

Fuses can also be programmed directly using the JTAG interface, as long as the IIM asserts the appropriate epm_scane signal. The IIM is not involved in this programming sequence aside from allowing or disallowing it using the epm_scane signal.

The minimum and maximum program voltages are 2.775 and 3.3 Volts, respectively.

30.4.5.4 Fuse Value Override Sequence

The fuse value override sequence is depicted in Figure 30-24.

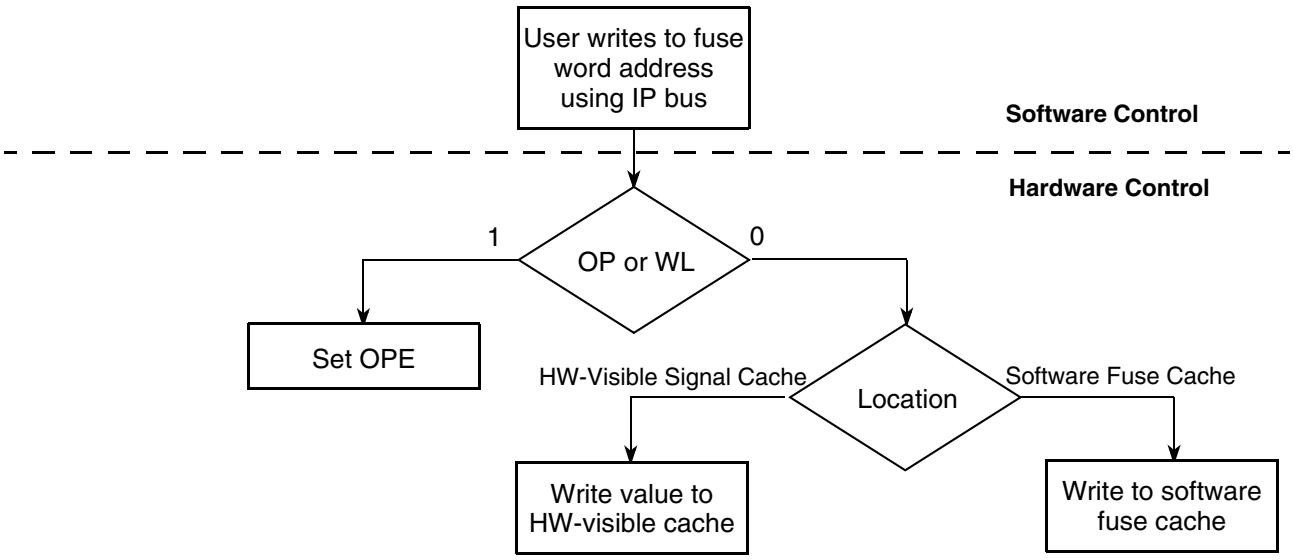


Figure 30-24. Fuse Value Override Sequence

30.5 Initialization/Application Information

30.5.1 Initialization

The IIM out-of-reset sequence proceeds as follows:

1. IIM automatically senses the hardware-visible fuses and writes their values to the hardware-visible cache. This action is completed within 200 µs to ensure that the hardware-visible fuses are loaded before the second reset at system level. Hardware-visible fuse word 2 is the first to be read.
2. The fuse bank access control (FBAC) words of all banks are sensed.
3. The remaining fuses are sensed.

The initialization time is determined by the number of hardware-visible fuses. That is, 32*3=96. (Each bank has 32 words.)

After the initialization is complete, the fuse_latched signal is asserted, as shown in Figure 30-25.

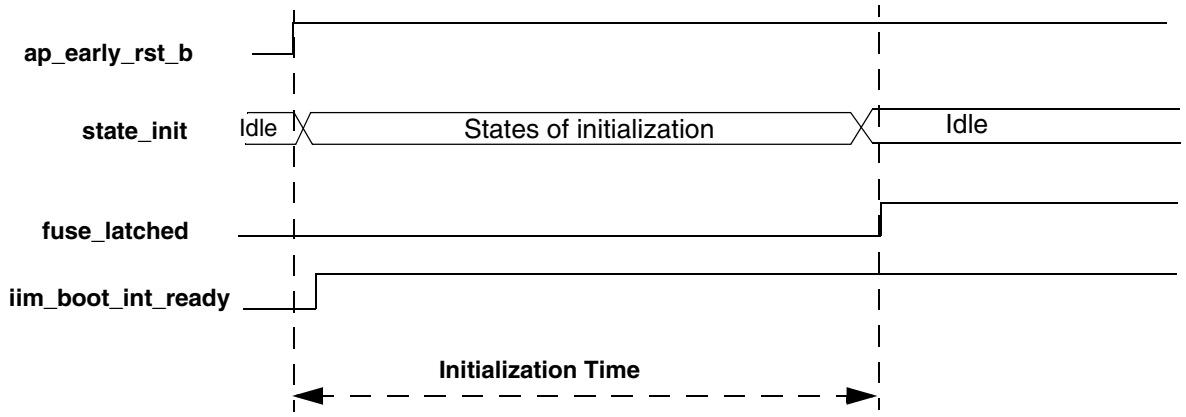


Figure 30-25. Initialization Timing Diagram

30.5.2 Programming

After JTAG programming, a reset must be applied to synchronize the fuse value to hardware-visible signal cache.

A wait period is required between program and read operations.

This section for the hardware design. Each project generates a parameter file according to this section.

The RTL design can be parameterized. The maximum number of fuse bank is 8. Maximum usable fuse bank size are 2048 bits. All the parameter are defined in file iim_para. The following parameters need to be defined.

30.5.2.1 Parameter for Number

- num_bank defines the number of fuse bank.
- num_ram_block defines the number of blocks of software fuses.
- num_ram defines the number of words in software fuses cache.
- num_hwv_block defines the number of blocks of hardware-visible fuses.
- num_hwv defines the number of words in hardware-visible fuses cache.

30.5.2.2 Parameter for Address

30.5.2.2.1 Address Boundary of Fuse Bank

al_bankx defines the low boundary of fuse bank x. au_bankx defines the up boundary of fuse bank x. The maximum number of fuse bank is 8. Assign 0 to unused banks.

Example 30-1. Address Boundary of Fuse Bank

```
parameter num_bank = 3;
```

parameter [13:0] al_bank0=14’h0800;

parameter [13:0] au_bank0=14’h083C;

parameter [13:0] al_bank1=14’h0C00;

parameter [13:0] au_bank1=14’h0C7C;

parameter [13:0] al_bank2=14’h0100;

parameter [13:0] au_bank2=14’h013C;

parameter [13:0] al_bank3=14’h0;

parameter [13:0] au_bank3=14’h0;

30.5.2.2.2 Address Boundaries of Software Fuse Blocks

al_ram_block*n* and au_ram_block*n* define respectively the lower and upper boundaries of software fuse block *n*. The fuse block which is defined in this section is mapped to Software Fuse Cache.The maximum number of blocks is 16. Assign 0 to unused blocks. If there is no fuse to be mapped to Software Fuse Cache, num_ram must be defined to 1 to avoid Verilog syntax error.

Example 30-2. Address Boundary of Software Fuse Block

parameter [4:0] num_ram_block =5’d1;

parameter [10:0] num_ram = 11’d15;

parameter [13:0] al_ram_block0=14’h1004;

parameter [13:0] au_ram_block0=14’h103C;

parameter [13:0] al_ram_block1=14’h0;

parameter [13:0] au_ram_block1=14’h0;

parameter [13:0] al_ram_block2=14’h0;

parameter [13:0] au_ram_block2=14’h0;

30.5.2.2.3 Address Boundaries of Hardware-Visible Signal Blocks

al_hwv_block*n* and al_hwv_block*n* define respectively the lower and upper boundaries of hardware-visible signal block *n*. The fuse block which is defined in this section is mapped to hardware-visible signal cache. The maximum number of blocks is 16. Assign 0 to unused block. Note that registers SCS0—SCS3 are assigned in block0

Example 30-3. Address Boundaries of Hardware-Visible Signal Blocks

parameter [4:0] num_hwv_block=5’d6;

parameter [10:0] num_hwv = 11’d53;

parameter [13:0] al_hwv_block0=14’h0024;

parameter [13:0] au_hwv_block0=14’h0030; // Definition for SCS0—SCS3

parameter [13:0] al_hwv_block1=14’h0808;

parameter [13:0] au_hwv_block1=14’h0808; // Definition for HWV2 (BOOT_INT)

parameter [13:0] al_hwv_block2=14’h0800;

parameter [13:0] au_hwv_block2=14’h0800;

parameter [13:0] al_hwv_block3=14’h0C00;

parameter [13:0] au_hwv_block3=14’h0C00;

parameter [13:0] al_hwv_block4=14’h1000;

parameter [13:0] au_hwv_block4=14’h1000; // Definition for FBAC words

parameter [13:0] al_hwv_block5=14’h0804;

parameter [13:0] au_hwv_block5=14’h0804;

parameter [13:0] al_hwv_block6=14’h080C;

parameter [13:0] au_hwv_block6=14’h083C;

parameter [13:0] al_hwv_block7=14’h0C04;

parameter [13:0] au_hwv_block7=14’h0C7C;

parameter [13:0] al_hwv_block8=14’h0;

parameter [13:0] au_hwv_block8=14’h0;

The initialization sequence follows the above order of definition: that is, when IIM comes out of reset it initializes block1 first, then block2, block3, and so on. The above order of parameter definitions is the same as order of the fuses in the hardware-visible signal cache. However, note that changing of the sequence of blocks which belong to first two banks need some changes in RTL(iim_hwv_reg.v). FBAC words of all banks must be placed together.

Fuses in Bank0, Bank1 and FBAC words of all banks are forced to be Hardware Visible Fuses, so they must be defined in hardware-visible signal block. that is, al_hwv_block*n*, au_hwv_block*n*.

30.5.2.3 Other Parameters

- chl_size defines the size of sjc_challenge, resp_size defines the size of sjc_response.
- hwv_addr_width defines the width of hardware-visible signal cache.
- ram_addr_width defines the width of software fuse cache.
- If the size of bank0 is larger than 128 fuses(16 words), the following statement must be added in the iim_para.
- ‘define bank0_para_size
- ‘define bank0_size_para

30.6 Fuse Map

See your Freescale representative for the current fuse map.

Chapter 31
IPMUX

31.1 Introduction

Figure 31-1 shows the IPMUX block diagram. It illustrates the IPMUX ports (inputs and outputs), the IPMUX three sub-blocks and their inter-connections.

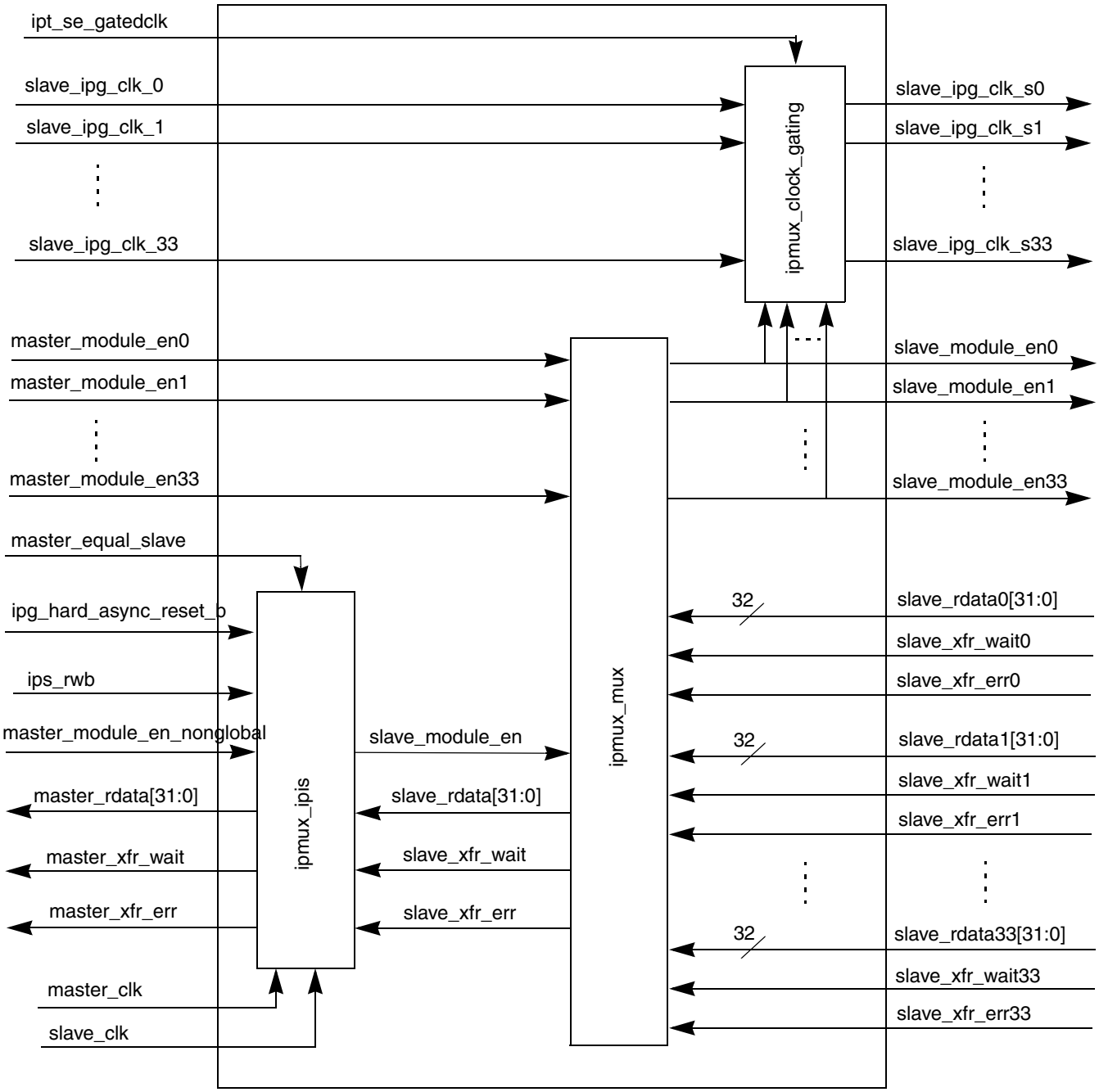


Figure 31-1. IPMUX Block Diagram

31.2 Overview

ARM/DSP platforms should be connected to most of the peripherals by the IP bus line (IPS) bus interface. The connection between the peripherals and the ARM/DSP platform can't be done directly, although the ARM9/ARM11/DSP platforms are converting the AHB-Lite/Q-Bus bus interface to IPS bus interface, by means of the AIPI/AIPS/Q2SB modules accordingly. Two main reasons for that: the platforms implement

only one full set of IPS signals (hence muxing is required) and the platforms may run in different frequency than the peripherals (hence synchronization is required). These two main reasons defined the necessity for a block that will be a bridge (gasket) between the ARM/DSP platform and the peripherals IPS bus interface. Also, to minimize the clock toggles on the IPS bus interface and by that reduce power, the IPMUX also implements the IPS bus gated clocks for the peripherals which toggle only when peripheral is actually accessed.

31.3 Features

The IPMUX includes the following features:

- Muxing of all the IP peripherals ips_rdata buses and response signals (ips_xfr_wait, ips_xfr_err) since ARM9 platform AIPI, ARM11 platform AIPS and DSP platform Q2SB can get only one set of these signals.
- Synchronizing the IPS protocol between the ARM/DSP platform (master, usually faster) and the peripherals (slaves, usually slower).
- Generating the peripherals bus clocks (ipg gated clocks active only on IPS accesses).

The IPMUX design is based on the assumption that the positive edges of both Master & Slave clocks are synchronized. Practically the above assumption means that master_frequency/slave_frequency is N or 1/N where N is a positive integer: N=1, 2, 3,…

31.4 Modes of Operation

The IPMUX has 2 modes of operations depends on the value of the master_equal_slave input

- IPS synchronization mode (master_equal_slave==1’b0)

  This mode should be used whenever the master clock and the slave clock are not equal and hence IPS protocol synchronization is required.

  To enter this mode the master_equal_slave input must be driven as 1’b0.

- IPS synchronization bypass mode (master_equal_slave==1’b1)

  This mode may be used whenever the master clock and the slave clock are equal and hence IPS protocol synchronization is not required. When not using this mode in this case the block will function correctly, however, cycles may be wasted in each IPS access.

  To enter this mode the master_equal_slave input must be driven as 1’b1.

31.5 External Signal Description

31.6 Overview

Table 31-1 list all the external signals (ports) of the IPMUX along with their direction and short description. Since the IPMUX is internally connected in the SOC (none of its signals is connected to pads) a simplified table format is used.

Table 31-1. IPMUX Signals

Port Name	Direction	Function
Global Signals		
ipg_hard_async_reset_b	input	Hardware Asynch. Reset; negates synchronously
master_clk	input	The IPS Master Clock
slave_clk	input	The IPS Slave (peripheral) Clock
ipt_se_gatedclk	input	scan enable control for clock gating logic
master_equal_slave	input	Notify that master clock frequency equals slave clock frequency. If asserted synchronization logic is bypassed
master_module_en_nonglobal	input	Indication from ARM platform that one of the 32 ips_module_en is asserted or: indication from DSP platform that one of the 33 ips_module_en is asserted
IPS Master Signals		
master_module_en0	input	ips_module_en from IPS Master for IPS Slave #0
master_module_en1	input	ips_module_en from IPS Master for IPS Slave #1
master_module_en2	input	ips_module_en from IPS Master for IPS Slave #2
master_module_en3	input	ips_module_en from IPS Master for IPS Slave #3
master_module_en4	input	ips_module_en from IPS Master for IPS Slave #4
master_module_en5	input	ips_module_en from IPS Master for IPS Slave #5
master_module_en6	input	ips_module_en from IPS Master for IPS Slave #6
master_module_en7	input	ips_module_en from IPS Master for IPS Slave #7
master_module_en8	input	ips_module_en from IPS Master for IPS Slave #8
master_module_en9	input	ips_module_en from IPS Master for IPS Slave #9
master_module_en10	input	ips_module_en from IPS Master for IPS Slave #10
master_module_en11	input	ips_module_en from IPS Master for IPS Slave #11
master_module_en12	input	ips_module_en from IPS Master for IPS Slave #12
master_module_en13	input	ips_module_en from IPS Master for IPS Slave #13
master_module_en14	input	ips_module_en from IPS Master for IPS Slave #14
master_module_en15	input	ips_module_en from IPS Master for IPS Slave #15
master_module_en16	input	ips_module_en from IPS Master for IPS Slave #16
master_module_en17	input	ips_module_en from IPS Master for IPS Slave #17
master_module_en18	input	ips_module_en from IPS Master for IPS Slave #18
master_module_en19	input	ips_module_en from IPS Master for IPS Slave #19
master_module_en20	input	ips_module_en from IPS Master for IPS Slave #20
master_module_en21	input	ips_module_en from IPS Master for IPS Slave #21
master_module_en22	input	ips_module_en from IPS Master for IPS Slave #22

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
master_module_en23	input	ips_module_en from IPS Master for IPS Slave #23
master_module_en24	input	ips_module_en from IPS Master for IPS Slave #24
master_module_en25	input	ips_module_en from IPS Master for IPS Slave #25
master_module_en26	input	ips_module_en from IPS Master for IPS Slave #26
master_module_en27	input	ips_module_en from IPS Master for IPS Slave #27
master_module_en28	input	ips_module_en from IPS Master for IPS Slave #28
master_module_en29	input	ips_module_en from IPS Master for IPS Slave #29
master_module_en30	input	ips_module_en from IPS Master for IPS Slave #30
master_module_en31	input	ips_module_en from IPS Master for IPS Slave #31
master_module_en32	input	ips_module_en from IPS Master for IPS Slave #32 dedicated slot for ips_module_en_glbl0 signal from ARM or ips_module_en[32] from DSP platform
master_module_en33	input	ips_module_en from IPS Master for IPS Slave #33 dedicated slot for ips_module_en_glbl1 signal from ARM
master_rdata[31:0]	output	ips_rdata read data bus to IPS Master
master_xfr_err	output	IPS ips_xfr_err protocol signal to IPS Master
master_xfr_wait	output	IPS ips_xfr_wait protocol signal to IPS Master
IPS Slave Signals		
slave_ipg_clk_0	input	ipg_clk clock related to IPS Slave #0
slave_ipg_clk_1	input	ipg_clk clock related to IPS Slave #1
slave_ipg_clk_2	input	ipg_clk clock related to IPS Slave #2
slave_ipg_clk_3	input	ipg_clk clock related to IPS Slave #3
slave_ipg_clk_4	input	ipg_clk clock related to IPS Slave #4
slave_ipg_clk_5	input	ipg_clk clock related to IPS Slave #5
slave_ipg_clk_6	input	ipg_clk clock related to IPS Slave #6
slave_ipg_clk_7	input	ipg_clk clock related to IPS Slave #7
slave_ipg_clk_8	input	ipg_clk clock related to IPS Slave #8
slave_ipg_clk_9	input	ipg_clk clock related to IPS Slave #9
slave_ipg_clk_10	input	ipg_clk clock related to IPS Slave #10
slave_ipg_clk_11	input	ipg_clk clock related to IPS Slave #11
slave_ipg_clk_12	input	ipg_clk clock related to IPS Slave #12
slave_ipg_clk_13	input	ipg_clk clock related to IPS Slave #13
slave_ipg_clk_14	input	ipg_clk clock related to IPS Slave #14
slave_ipg_clk_15	input	ipg_clk clock related to IPS Slave #15

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_ipg_clk_16	input	ipg_clk clock related to IPS Slave #16
slave_ipg_clk_17	input	ipg_clk clock related to IPS Slave #17
slave_ipg_clk_18	input	ipg_clk clock related to IPS Slave #18
slave_ipg_clk_19	input	ipg_clk clock related to IPS Slave #19
slave_ipg_clk_20	input	ipg_clk clock related to IPS Slave #20
slave_ipg_clk_21	input	ipg_clk clock related to IPS Slave #21
slave_ipg_clk_22	input	ipg_clk clock related to IPS Slave #22
slave_ipg_clk_23	input	ipg_clk clock related to IPS Slave #23
slave_ipg_clk_24	input	ipg_clk clock related to IPS Slave #24
slave_ipg_clk_25	input	ipg_clk clock related to IPS Slave #25
slave_ipg_clk_26	input	ipg_clk clock related to IPS Slave #26
slave_ipg_clk_27	input	ipg_clk clock related to IPS Slave #27
slave_ipg_clk_28	input	ipg_clk clock related to IPS Slave #28
slave_ipg_clk_29	input	ipg_clk clock related to IPS Slave #29
slave_ipg_clk_30	input	ipg_clk clock related to IPS Slave #30
slave_ipg_clk_31	input	ipg_clk clock related to IPS Slave #31
slave_ipg_clk_32	input	ipg_clk clock related to IPS Slave #32
slave_ipg_clk_33	input	ipg_clk clock related to IPS Slave #33
slave_ipg_clk_s0	output	ipg_clk_s gated clock (bus clock) to IPS Slave #0
slave_ipg_clk_s1	output	ipg_clk_s gated clock (bus clock) to IPS Slave #1
slave_ipg_clk_s2	output	ipg_clk_s gated clock (bus clock) to IPS Slave #2
slave_ipg_clk_s3	output	ipg_clk_s gated clock (bus clock) to IPS Slave #3
slave_ipg_clk_s4	output	ipg_clk_s gated clock (bus clock) to IPS Slave #4
slave_ipg_clk_s5	output	ipg_clk_s gated clock (bus clock) to IPS Slave #5
slave_ipg_clk_s6	output	ipg_clk_s gated clock (bus clock) to IPS Slave #6
slave_ipg_clk_s7	output	ipg_clk_s gated clock (bus clock) to IPS Slave #7
slave_ipg_clk_s8	output	ipg_clk_s gated clock (bus clock) to IPS Slave #8
slave_ipg_clk_s9	output	ipg_clk_s gated clock (bus clock) to IPS Slave #9
slave_ipg_clk_s10	output	ipg_clk_s gated clock (bus clock) to IPS Slave #10
slave_ipg_clk_s11	output	ipg_clk_s gated clock (bus clock) to IPS Slave #11
slave_ipg_clk_s12	output	ipg_clk_s gated clock (bus clock) to IPS Slave #12
slave_ipg_clk_s13	output	ipg_clk_s gated clock (bus clock) to IPS Slave #13
slave_ipg_clk_s14	output	ipg_clk_s gated clock (bus clock) to IPS Slave #14

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_ipg_clk_s15	output	ipg_clk_s gated clock (bus clock) to IPS Slave #15
slave_ipg_clk_s16	output	ipg_clk_s gated clock (bus clock) to IPS Slave #16
slave_ipg_clk_s17	output	ipg_clk_s gated clock (bus clock) to IPS Slave #17
slave_ipg_clk_s18	output	ipg_clk_s gated clock (bus clock) to IPS Slave #18
slave_ipg_clk_s19	output	ipg_clk_s gated clock (bus clock) to IPS Slave #19
slave_ipg_clk_s20	output	ipg_clk_s gated clock (bus clock) to IPS Slave #20
slave_ipg_clk_s21	output	ipg_clk_s gated clock (bus clock) to IPS Slave #21
slave_ipg_clk_s22	output	ipg_clk_s gated clock (bus clock) to IPS Slave #22
slave_ipg_clk_s23	output	ipg_clk_s gated clock (bus clock) to IPS Slave #23
slave_ipg_clk_s24	output	ipg_clk_s gated clock (bus clock) to IPS Slave #24
slave_ipg_clk_s25	output	ipg_clk_s gated clock (bus clock) to IPS Slave #25
slave_ipg_clk_s26	output	ipg_clk_s gated clock (bus clock) to IPS Slave #26
slave_ipg_clk_s27	output	ipg_clk_s gated clock (bus clock) to IPS Slave #27
slave_ipg_clk_s28	output	ipg_clk_s gated clock (bus clock) to IPS Slave #28
slave_ipg_clk_s29	output	ipg_clk_s gated clock (bus clock) to IPS Slave #29
slave_ipg_clk_s30	output	ipg_clk_s gated clock (bus clock) to IPS Slave #30
slave_ipg_clk_s31	output	ipg_clk_s gated clock (bus clock) to IPS Slave #31
slave_ipg_clk_s32	output	ipg_clk_s gated clock (bus clock) to IPS Slave #32
slave_ipg_clk_s33	output	ipg_clk_s gated clock (bus clock) to IPS Slave #33
slave_module_en0	output	ips_module_en to IPS Slave #0
slave_module_en1	output	ips_module_en to IPS Slave #1
slave_module_en2	output	ips_module_en to IPS Slave #2
slave_module_en3	output	ips_module_en to IPS Slave #3
slave_module_en4	output	ips_module_en to IPS Slave #4
slave_module_en5	output	ips_module_en to IPS Slave #5
slave_module_en6	output	ips_module_en to IPS Slave #6
slave_module_en7	output	ips_module_en to IPS Slave #7
slave_module_en8	output	ips_module_en to IPS Slave #8
slave_module_en9	output	ips_module_en to IPS Slave #9
slave_module_en10	output	ips_module_en to IPS Slave #10
slave_module_en11	output	ips_module_en to IPS Slave #11
slave_module_en12	output	ips_module_en to IPS Slave #12
slave_module_en13	output	ips_module_en to IPS Slave #13

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_module_en14	output	ips_module_en to IPS Slave #14
slave_module_en15	output	ips_module_en to IPS Slave #15
slave_module_en16	output	ips_module_en to IPS Slave #16
slave_module_en17	output	ips_module_en to IPS Slave #17
slave_module_en18	output	ips_module_en to IPS Slave #18
slave_module_en19	output	ips_module_en to IPS Slave #19
slave_module_en20	output	ips_module_en to IPS Slave #20
slave_module_en21	output	ips_module_en to IPS Slave #21
slave_module_en22	output	ips_module_en to IPS Slave #22
slave_module_en23	output	ips_module_en to IPS Slave #23
slave_module_en24	output	ips_module_en to IPS Slave #24
slave_module_en25	output	ips_module_en to IPS Slave #25
slave_module_en26	output	ips_module_en to IPS Slave #26
slave_module_en27	output	ips_module_en to IPS Slave #27
slave_module_en28	output	ips_module_en to IPS Slave #28
slave_module_en29	output	ips_module_en to IPS Slave #29
slave_module_en30	output	ips_module_en to IPS Slave #30
slave_module_en31	output	ips_module_en to IPS Slave #31
slave_module_en32	output	ips_module_en to IPS Slave #32
slave_module_en33	output	ips_module_en to IPS Slave #33
slave_rdata0[31:0]	input	ips_rdata bus from IPS Slave #0
slave_rdata1[31:0]	input	ips_rdata bus from IPS Slave #1
slave_rdata2[31:0]	input	ips_rdata bus from IPS Slave #2
slave_rdata3[31:0]	input	ips_rdata bus from IPS Slave #3
slave_rdata4[31:0]	input	ips_rdata bus from IPS Slave #4
slave_rdata5[31:0]	input	ips_rdata bus from IPS Slave #5
slave_rdata6[31:0]	input	ips_rdata bus from IPS Slave #6
slave_rdata7[31:0]	input	ips_rdata bus from IPS Slave #7
slave_rdata8[31:0]	input	ips_rdata bus from IPS Slave #8
slave_rdata9[31:0]	input	ips_rdata bus from IPS Slave #9
slave_rdata10[31:0]	input	ips_rdata bus from IPS Slave #10
slave_rdata11[31:0]	input	ips_rdata bus from IPS Slave #11
slave_rdata12[31:0]	input	ips_rdata bus from IPS Slave #12

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_rdata13[31:0]	input	ips_rdata bus from IPS Slave #13
slave_rdata14[31:0]	input	ips_rdata bus from IPS Slave #14
slave_rdata15[31:0]	input	ips_rdata bus from IPS Slave #15
slave_rdata16[31:0]	input	ips_rdata bus from IPS Slave #16
slave_rdata17[31:0]	input	ips_rdata bus from IPS Slave #17
slave_rdata18[31:0]	input	ips_rdata bus from IPS Slave #18
slave_rdata19[31:0]	input	ips_rdata bus from IPS Slave #19
slave_rdata20[31:0]	input	ips_rdata bus from IPS Slave #20
slave_rdata21[31:0]	input	ips_rdata bus from IPS Slave #21
slave_rdata22[31:0]	input	ips_rdata bus from IPS Slave #22
slave_rdata23[31:0]	input	ips_rdata bus from IPS Slave #23
slave_rdata24[31:0]	input	ips_rdata bus from IPS Slave #24
slave_rdata25[31:0]	input	ips_rdata bus from IPS Slave #25
slave_rdata26[31:0]	input	ips_rdata bus from IPS Slave #26
slave_rdata27[31:0]	input	ips_rdata bus from IPS Slave #27
slave_rdata28[31:0]	input	ips_rdata bus from IPS Slave #28
slave_rdata29[31:0]	input	ips_rdata bus from IPS Slave #29
slave_rdata30[31:0]	input	ips_rdata bus from IPS Slave #30
slave_rdata31[31:0]	input	ips_rdata bus from IPS Slave #31
slave_rdata32[31:0]	input	ips_rdata bus from IPS Slave #32
slave_rdata33[31:0]	input	ips_rdata bus from IPS Slave #33
slave_xfr_err0	input	ips_xfr_err signal from IPS Slave #0
slave_xfr_err1	input	ips_xfr_err signal from IPS Slave #1
slave_xfr_err2	input	ips_xfr_err signal from IPS Slave #2
slave_xfr_err3	input	ips_xfr_err signal from IPS Slave #3
slave_xfr_err4	input	ips_xfr_err signal from IPS Slave #4
slave_xfr_err5	input	ips_xfr_err signal from IPS Slave #5
slave_xfr_err6	input	ips_xfr_err signal from IPS Slave #6
slave_xfr_err7	input	ips_xfr_err signal from IPS Slave #7
slave_xfr_err8	input	ips_xfr_err signal from IPS Slave #8
slave_xfr_err9	input	ips_xfr_err signal from IPS Slave #9
slave_xfr_err10	input	ips_xfr_err signal from IPS Slave #10
slave_xfr_err11	input	ips_xfr_err signal from IPS Slave #11

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_xfr_err12	input	ips_xfr_err signal from IPS Slave #12
slave_xfr_err13	input	ips_xfr_err signal from IPS Slave #13
slave_xfr_err14	input	ips_xfr_err signal from IPS Slave #14
slave_xfr_err15	input	ips_xfr_err signal from IPS Slave #15
slave_xfr_err16	input	ips_xfr_err signal from IPS Slave #16
slave_xfr_err17	input	ips_xfr_err signal from IPS Slave #17
slave_xfr_err18	input	ips_xfr_err signal from IPS Slave #18
slave_xfr_err19	input	ips_xfr_err signal from IPS Slave #19
slave_xfr_err20	input	ips_xfr_err signal from IPS Slave #20
slave_xfr_err21	input	ips_xfr_err signal from IPS Slave #21
slave_xfr_err22	input	ips_xfr_err signal from IPS Slave #22
slave_xfr_err23	input	ips_xfr_err signal from IPS Slave #23
slave_xfr_err24	input	ips_xfr_err signal from IPS Slave #24
slave_xfr_err25	input	ips_xfr_err signal from IPS Slave #25
slave_xfr_err26	input	ips_xfr_err signal from IPS Slave #26
slave_xfr_err27	input	ips_xfr_err signal from IPS Slave #27
slave_xfr_err28	input	ips_xfr_err signal from IPS Slave #28
slave_xfr_err29	input	ips_xfr_err signal from IPS Slave #29
slave_xfr_err30	input	ips_xfr_err signal from IPS Slave #30
slave_xfr_err31	input	ips_xfr_err signal from IPS Slave #31
slave_xfr_err32	input	ips_xfr_err signal from IPS Slave #32
slave_xfr_err33	input	ips_xfr_err signal from IPS Slave #33
slave_xfr_wait0	input	ips_xfr_wait signal from IPS Slave #0
slave_xfr_wait1	input	ips_xfr_wait signal from IPS Slave #1
slave_xfr_wait2	input	ips_xfr_wait signal from IPS Slave #2
slave_xfr_wait3	input	ips_xfr_wait signal from IPS Slave #3
slave_xfr_wait4	input	ips_xfr_wait signal from IPS Slave #4
slave_xfr_wait5	input	ips_xfr_wait signal from IPS Slave #5
slave_xfr_wait6	input	ips_xfr_wait signal from IPS Slave #6
slave_xfr_wait7	input	ips_xfr_wait signal from IPS Slave #7
slave_xfr_wait8	input	ips_xfr_wait signal from IPS Slave #8
slave_xfr_wait9	input	ips_xfr_wait signal from IPS Slave #9
slave_xfr_wait10	input	ips_xfr_wait signal from IPS Slave #10

Table 31-1. IPMUX Signals (continued)

Port Name	Direction	Function
slave_xfr_wait11	input	ips_xfr_wait signal from IPS Slave #11
slave_xfr_wait12	input	ips_xfr_wait signal from IPS Slave #12
slave_xfr_wait13	input	ips_xfr_wait signal from IPS Slave #13
slave_xfr_wait14	input	ips_xfr_wait signal from IPS Slave #14
slave_xfr_wait15	input	ips_xfr_wait signal from IPS Slave #15
slave_xfr_wait16	input	ips_xfr_wait signal from IPS Slave #16
slave_xfr_wait17	input	ips_xfr_wait signal from IPS Slave #17
slave_xfr_wait18	input	ips_xfr_wait signal from IPS Slave #18
slave_xfr_wait19	input	ips_xfr_wait signal from IPS Slave #19
slave_xfr_wait20	input	ips_xfr_wait signal from IPS Slave #20
slave_xfr_wait21	input	ips_xfr_wait signal from IPS Slave #21
slave_xfr_wait22	input	ips_xfr_wait signal from IPS Slave #22
slave_xfr_wait23	input	ips_xfr_wait signal from IPS Slave #23
slave_xfr_wait24	input	ips_xfr_wait signal from IPS Slave #24
slave_xfr_wait25	input	ips_xfr_wait signal from IPS Slave #25
slave_xfr_wait26	input	ips_xfr_wait signal from IPS Slave #26
slave_xfr_wait27	input	ips_xfr_wait signal from IPS Slave #27
slave_xfr_wait28	input	ips_xfr_wait signal from IPS Slave #28
slave_xfr_wait29	input	ips_xfr_wait signal from IPS Slave #29
slave_xfr_wait30	input	ips_xfr_wait signal from IPS Slave #30
slave_xfr_wait31	input	ips_xfr_wait signal from IPS Slave #31
slave_xfr_wait32	input	ips_xfr_wait signal from IPS Slave #32
slave_xfr_wait33	input	ips_xfr_wait signal from IPS Slave #33

31.7 Detailed Signal Descriptions

Following is a detailed description of the above signals.

IPS bus interface signals are detailed in SRS 3.1 section 4.

All IPS bus interface signals should meet the timing requirements listed for IPS target/initiator in the SRS 3.1 documents.

31.7.1 ipg_hard_async_reset_b

Input hardware asynchronous reset, negates synchronously. Active low.

31.7.2 master_clk

Input IPS Master Clock. This clock is aligned with the clock which toggles the IPS Master F.F.s

31.7.3 slave_clk

Input IPS Slave (peripheral) Clock. This clock is aligned with the clock which toggles the IPS Slave F.F.s

31.7.4 ipt_se_gatedclk

Input Scan enable control for clock gating logic. Active high. Allows controllability of the clock gating in scan mode.

31.7.5 master_equal_slave

Input notify that the IPS Master clock frequency equals Slave clock frequency.

If asserted synchronization logic is bypassed to reduce clock cycles waste for synchronizing the IPS Master/Slave protocol.

When this signal is low (master_equal_slave==1'b0) the IPMUX is in: "IPS synchronization mode", when high (master_equal_slave==1'b1) the IPMUX is in: "IPS synchronization bypass mode"

31.7.6 master_module_en_nonglobal

Input indication from ARMARM platform that one of the 32 module_en is asserted or indication from DSP platform that one of the 33 module_en is asserted. This is actually an "OR" function of ip_module_en<x> signals. Note that for ARMARM platform when the ips_module_en_glbl are active, master_module_en_nonglobal will not be active. Also note that the name of DSP platform port that should be connected to this IPMUX port is: ip_module_en_global.

31.7.7 master_module_en0, master_module_en1,… master_module_en33

Input ips_module_en signal from IPS Master for IPS Slave #0 through IPS Slave #33 respectively.

When master_module_en<x> is asserted by the IPS Master, an access to IPS Slave (peripheral) #x is initiated.

master_module_en<x> inputs that are not used, should be tied LOW (0).

31.7.8 master_rdata[31:0]

Output IPS 32 bits read data bus to IPS Master.

31.7.9 master_xfr_err

Output IPS transfer error protocol signal (ips_xfr_err) to IPS Master.

When high indicates that the current access should be ignored.

31.7.10 master_xfr_wait

Output IPS transfer wait protocol signal (ips_xfr_wait) to IPS Master.

When high indicated that the Slave insert wait states: for write accesses the write data and controls signals should be kept valid by the Master, in read accesses indicates that the data is not ready yet - the Master must keep control signals valid

31.7.11 slave_ipg_clk_0, slave_ipg_clk_1,... slave_ipg_clk_33

Input clocks related to IPS Slave #0 through IPS Slave #33 respectively.

slave_ipg_clk<x> inputs that are not used, should be tied LOW (0).

31.7.12 slave_ipg_clk_s0, slave_ipg_clk_s1,... slave_ipg_clk_s33

Output gated clocks (bus clock) to IPS Slave #0 through IPS Slave #33 respectively.

This clock toggles only when the corresponding slave_module_en is asserted.

31.7.13 slave_module_en0, slave_module_en1,... slave_module_en33

Output IPS module enable signal to IPS Slave #0 through IPS Slave #33 respectively.

When slave_module_en<x> is asserted an access to IPS Slave (peripheral) #x is activated.

31.7.14 slave_rdata0[31:0], slave_rdata1[31:0],... slave_rdata33[31:0]

Input IPS 32 bits read data bus from IPS Slave #0 through IPS Slave #33 respectively.

slave_rdata<x> inputs that are not used should be tied LOW (0).

Slaves with lower number of ips_rdata bits (8/16 bits peripherals) should be connected to the LSB bits of the slave_rdata<x> bus, while other bits should be tied LOW (0).

31.7.15 slave_xfr_err0, slave_xfr_err1,... slave_xfr_err33

Input IPS transfer error protocol signal (ips_xfr_err) from IPS Slave #0 through IPS Slave #33 respectively.

slave_xfr_err<x> inputs that are not used should be tied HIGH (1)

31.7.16 slave_xfr_wait0, slave_xfr_wait1,... slave_xfr_wait33

Input IPS transfer wait protocol signal (ips_xfr_wait) from IPS Slave #0 through IPS Slave #33 respectively.

slave_xfr_wait<x> inputs that are not used should be tied LOW (0)

31.8 Memory Map/Register Definition

Not Available. IPMUX does not have status/configuration registers.

31.9 Functional Description

The IPMUX is consist of 3 sub blocks which implement the 3 different features of the block:

1. ipmux_mux—implements the muxing of all the IP peripherals ips_rdata buses and response signals (ips_xfr_wait, ips_xfr_err)
2. ipmux_ipis—implements the synchronization of the IPS protocol between the ARM/DSP platform (master) and the peripherals (slaves)
3. ipmux_clock_gating—implements the ipg clock gating for the peripherals (ipg bus clocks), means generating ipg gated clocks which are active only on IPS accesses

The next sub sections describes each sub block in details.

31.9.1 ipmux_mux (IPMUX MUX)

The ipmux_mux can get as inputs 34 sets of {ips_readta[31:0], ips_xfr_wait, ips_xfr_err} signals from 34 different IPS slaves peripherals and generates as output one set of those signals. It also routes the slave_module_en input from the ipmux_ipis sub-block to the activated peripheral.

The block diagram of the ipmux_mux is illustrated in Figure 31-2

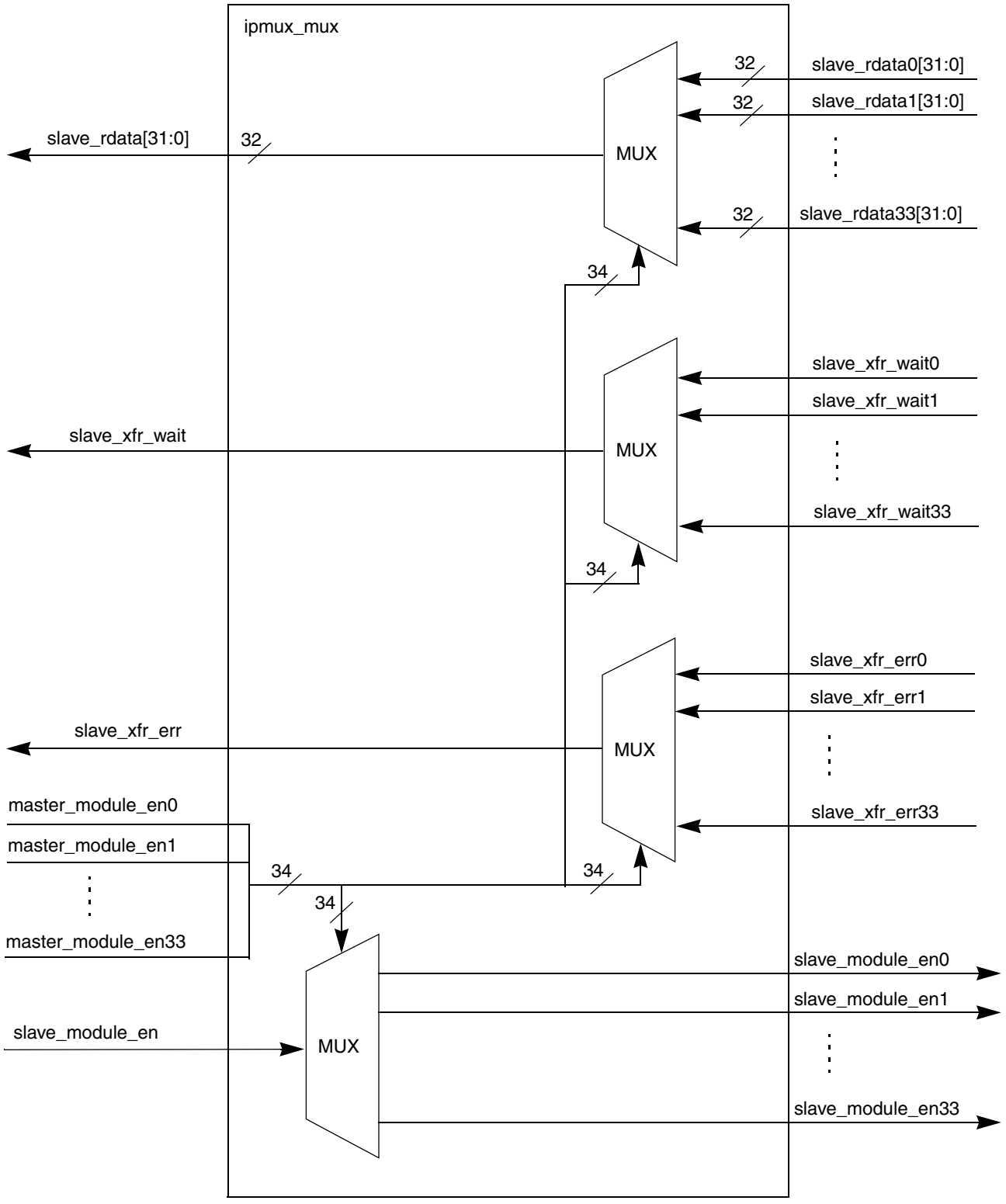


Figure 31-2. ipmux_mux Block Diagram

31.9.2 ipmux_ipis (IPMUX IPS Interface Synchronizer)

The ipmux_ipis is the IPMUX IPS (IP bus) interface synchronizing block which is responsible to synchronize the IPS bus protocol between IPS Master and IPS Slave which may toggle in different frequencies.

An assumption is made that both Master & Slave clocks are synchronized which practically means that master_frequency/slave_frequency is N or 1/N where N is a positive integer: N=1, 2, 3,...

The above assumption enables not using synchronizers (double F.F.s) on signals which cross Master/Slave clock domains but using simple synchronizing state machine and logic which saves clock cycles.

The ipmux_ipis block diagram is illustrated in Figure 31-3

The main sub-block is ipmux_ipis_master_neq_slave_logic which is responsible for the IPS interface synchronizing between IPS Master and IPS Slave in case these clocks are not synchronized. Figure 31-4 and Figure 31-5 illustrates this sub-block functionality, which is controlled by a simple FSM.

The synchronization is basically done by controlling the slave_module_en signal goes to the slave and the master_xfr_wait signal which goes to the master. Sampling of slave_xfr_err and slave_rdata bus from the slave completes the synchronization task.

It can be shown that the ips_module_en signal and the ips_xfr_wait together play main role in the IPI protocol and can be used directly to synchronize the handshake between master and slave.

Since IPS master and IPS slave may clock in different frequencies, the IPI protocol will operate at the lowest clock frequency. 3 cases should be investigated:

1. Master clock and Slave clock are equal
2. Master clock is faster than Slave clock
3. Master clock is slower than Slave clock

Each one of the cases will be detailed described in section Section 31.10.2, “Timing Diagrams.”

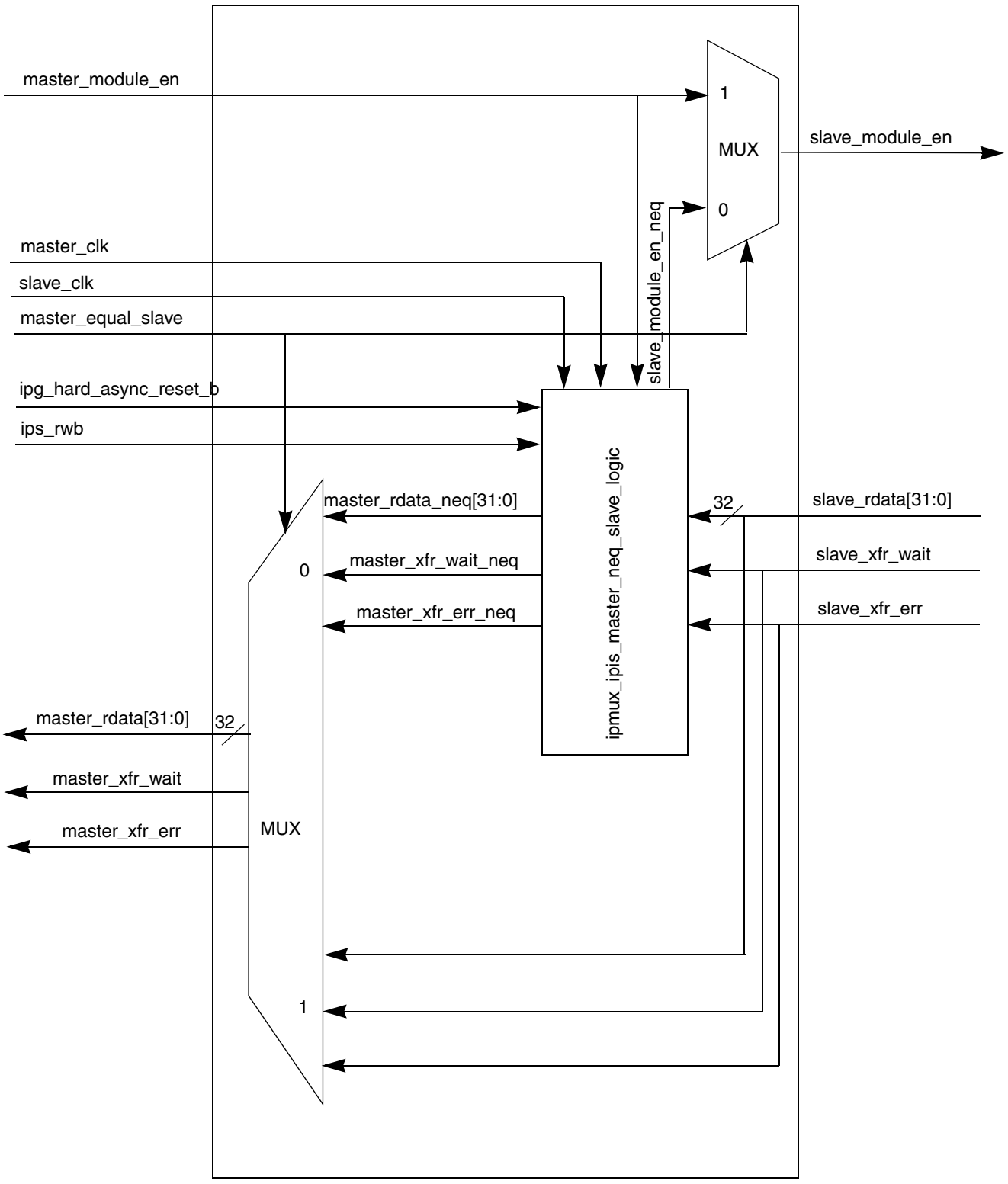


Figure 31-3. ipmux_ipis Block Diagram

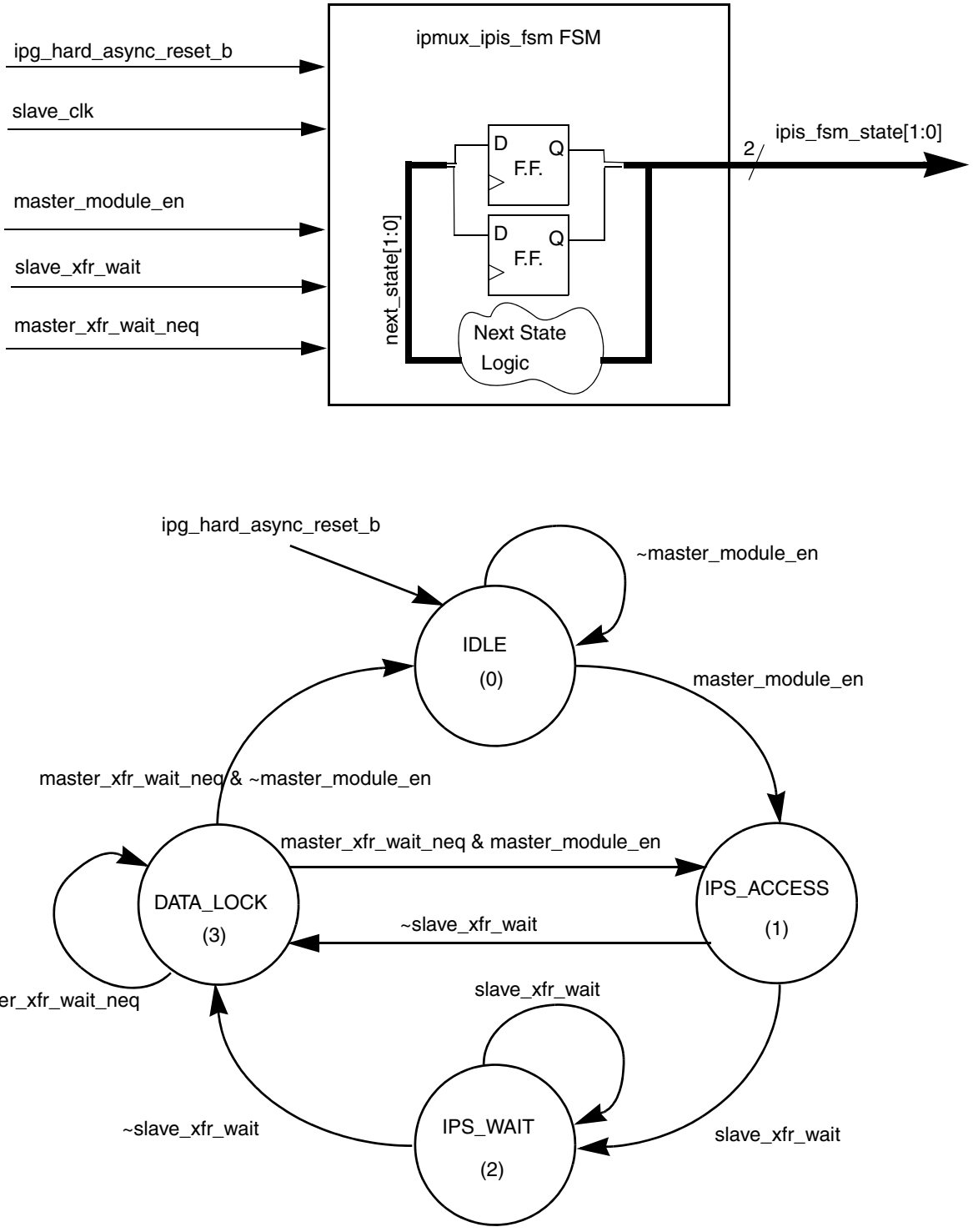


Figure 31-4. ipmux_ipis_master_neq_slave_logic block diagram part 1: ipmux_ipis_fsm

master_rdata_neq = slave_rdata sampled by slave_clk only if (ips_rwb == 1‘b1)&(master_module_en == 1‘b1)

master_xfr_err_neq = slave_xfr_err sampled by slave_clk only if (master_module_en == 1‘b1)

slave_module_en_neq = (state==IPS_ACCESS) | (state==IPS_WAIT)

master_xfr_wait_neq_not_sync = !(state==DATA_LOCK)

master_xfr_wait_neq = master_xfr_wait_neq_not_sync |
(!master_xfr_wait_neq_not_sync sampled with master_clk) <= see below figure

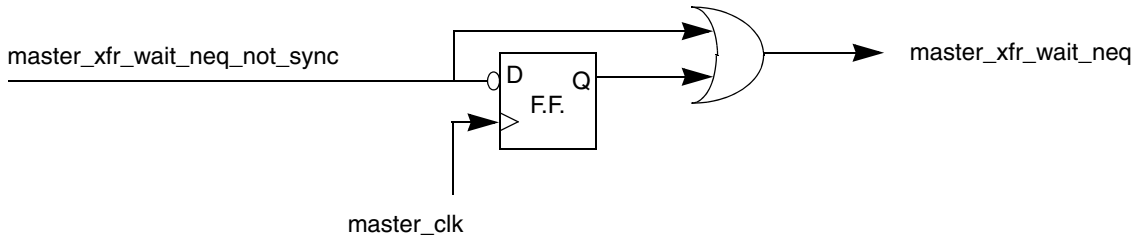


Figure 31-5. ipmux_ipis_master_neq_slave_logic block diagram part 1: Other Logic

31.9.3 ipmux_clock_gating (IPMUX Clock Gating)

The ipmux_clock_gating is the IPMUX clock gating sub-block which is responsible for generating the ipg gated clock, named also bus clock, for the 34 peripherals. Each gated clock will be toggling only when an IPS access is being activated for the specific peripheral. The gating which is done with the slave_module_en signal of each peripheral ensures minimum clock toggles for the proper functionality of the IPS bus interface and hence reduce power consumption.

The clock gating is done using the CMOS090LP library clock gating cells (sgclkn_seq_hivt_*). An ipt_se_gatedclk input signal to the ipmux_clock_gating sub-block will enable controllability of the clock gating during scan.

The ipmux_clock_gating block diagram is illustrated in Figure 31-6

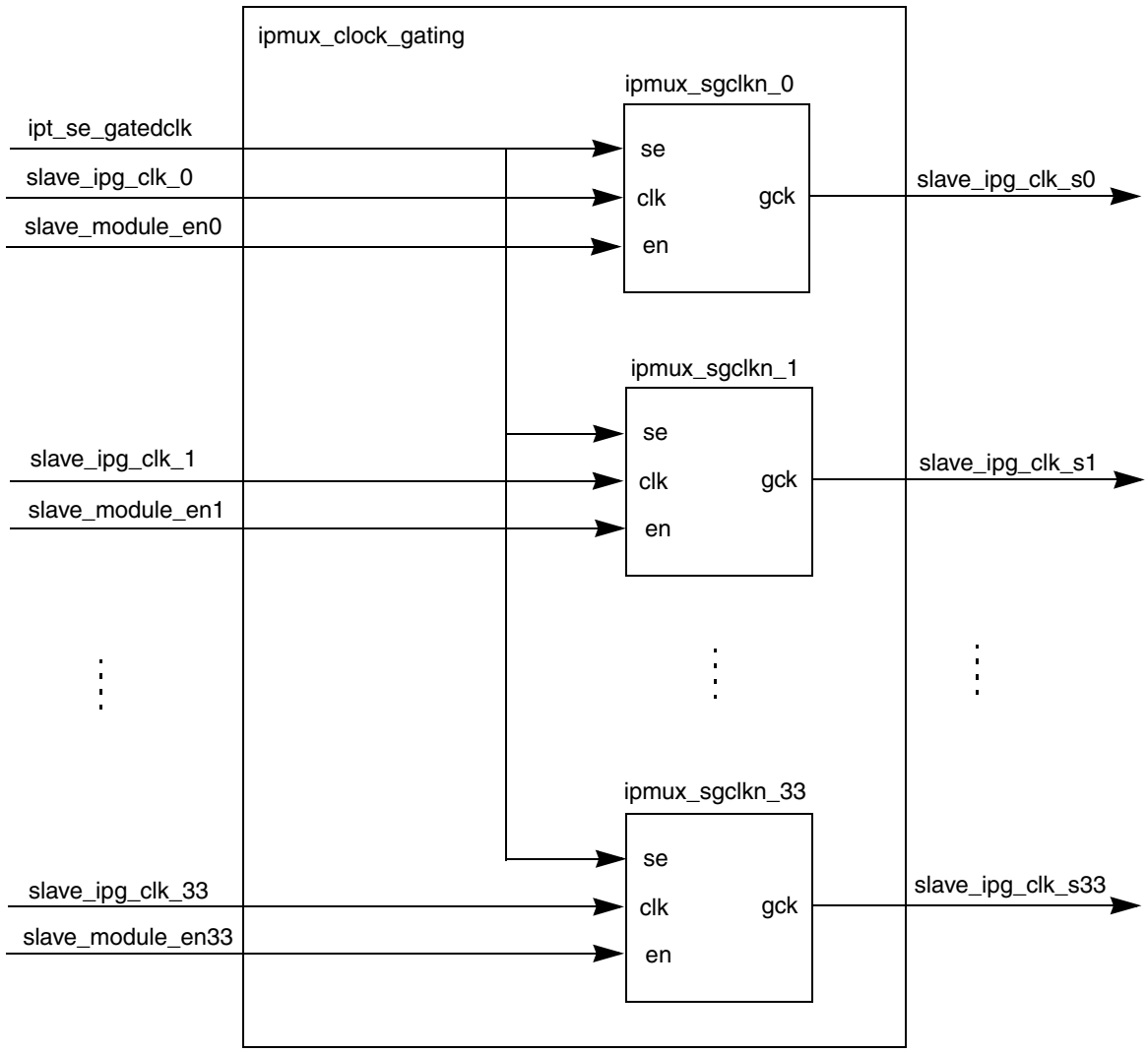


Figure 31-6. ipmux_clock_gating Block Diagram

31.10 Application Information

31.10.1 Connecting the IPMUX in the SOC

Figure 31-7 illustrates the connectivity of the IPMUX to the ARM platform and to the peripherals.

Each IPMUX can handle up to 34 peripherals (IPS Slaves) connections, up to 32 data bits each. master_module_en33 (slot #33) should be used for module_en_glbl1 signal from ARM platform master_module_en32 (slot #32) should be used for module_en_glbl0 signal from ARM platform or ips_module_en[32] from DSP platform,

When less than 34 peripherals are connected:

- master_module_en* inputs that are not used should be tied LOW (0)

- slave_xfr_wait* inputs that are not used should be tied LOW (0)
- slave_xfr_err* inputs that are not used should be tied HIGH (1)
- slave_rdata* inputs that are not used should be tied LOW (0)
- slave_ipg_clk* inputs that are not used should be tied LOW (0)

When less than 32 bit data peripheral is connected:

- The slave ips_rdata bus should be connected to the LSB of IPMUX slave_rdata bus
- The unused slave_rdata bits should be tied LOW (0)

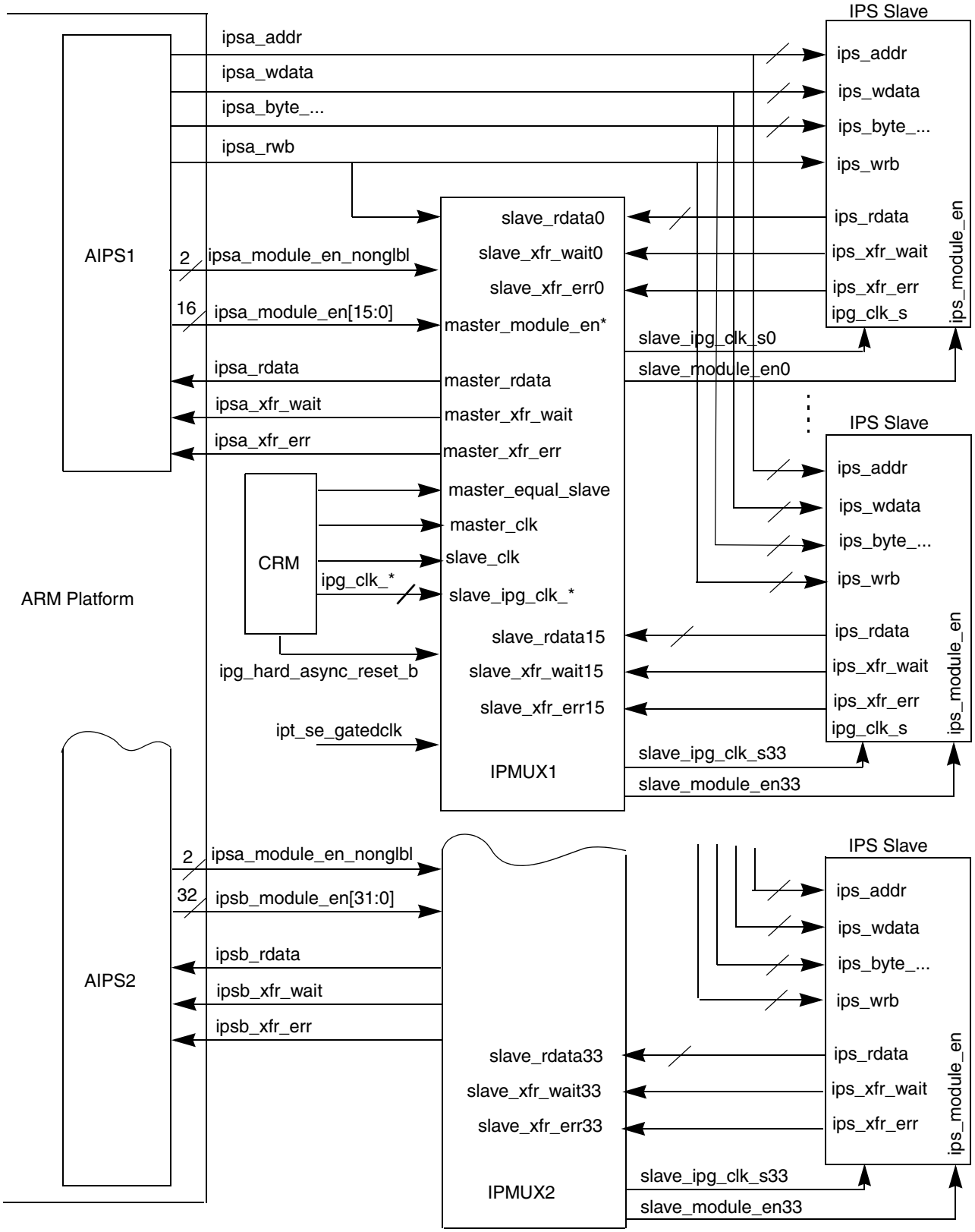


Figure 31-7. IPMUX Connectivity in SOC (IPMUX2 Connectivity is Partially Shown)

31.10.2 Timing Diagrams

In this section IPMUX timing diagrams will be described to expand the understanding of the IPMUX functionality. For better clarity 3 cases will be discussed separately according to the relation between the Master clock and the Slave clock:

- Master clock equal Slave clock
- Master clock is faster than Slave clock
- Master clock is slower than Slave clock

31.10.2.1 Master Clock Equals Slave Clock

The case that the Master clock equals the Slave clock is the regular IP Interface protocol.

Access to peripheral may take only 1 cycle or may be extended by the peripheral using the ips_xfr_wait indication.

Since the ARM/DSP Platform and all peripherals should be able, by definition, to communicate by the SkyBlue IPI protocol in this case we need just to directly transfer all of signals without any manipulation/processing on each of the signals involved. When master clock equal slave clock the master_equal_slave input of the IPMUX may be asserted high to bypass any synchronization logic and directly connect master signals to slave signals. If master_equal_slave input will remain low the IPMUX will functional correctly, however, each IPS access will take 2 clocks longer.

The IP Interface protocol is described in the SRS 3.1 IPI section, including signals detailed description, timing requirements and timing diagrams. See SRS 3.1 IP bus Interface for more details.

31.10.2.2 Master Clock Is Faster Than Slave Clock

In the case that the Master (initiator) clock is faster than the Slave (target) clock, wait states should be added in order to synchronize the Master fast clock to the Slave slow clock.

The ipmux_ipis pushes the master_xfr_wait to “1”, by default. This inserts the number of wait states required until the Slave will acknowledge that it is ready by slave_xfr_wait= “0”.

Figure 31-8 illustrates the timing diagram for 0 wait states read access in this case.

It should be cleared here:

1. that the signals: ips_rwb, ips_addr[11:0] and ips_wdata don’t require any special treatment and may be connected directly to the peripherals, since they are generated by the Master and held valid whenever the master_module_en is enabled.
2. the master_xfr_err signal is handled like the slave_rdata[31:0] signals as it can logically be treated as bit number 33 in the slave_rdata bus (in read cycles), indicates if the data is valid or not.

Hence, write accesses and the following signals are not shown in the timing diagram: ips_addr[11:0], ips_wdata

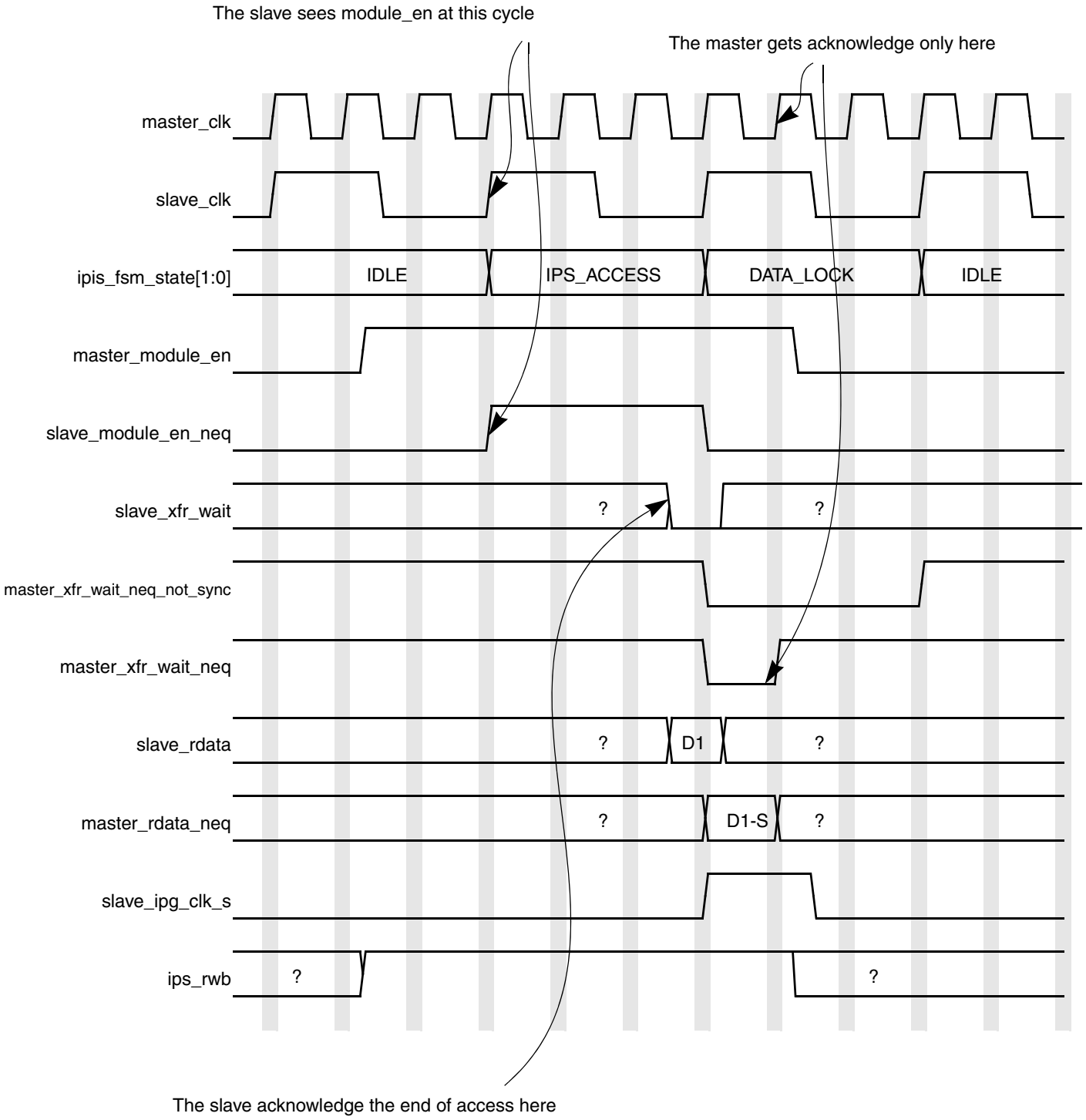


Figure 31-8. Master clock faster than Slave clock - Timing Diagram (read access)

31.10.2.3 Master clock Is Slower Than Slave Clock

In case that the Master (initiator) clock is slower than the Slave (target) clock, the danger is that the response from the slave (xfr_wait and xfr_err) and the slave_rdata information will be lost, since not captured by the Master.

The ipmux_ipis samples the response signals, slave_xfr_wait and slave_xfr_err and the slave_rdata into F.Fs until the slow Master will be able to accept the information.

After the Master capture the information all F.Fs will be enabled again, and new access can be initialized.

Figure 31-9 illustrates the timing diagram for 2 sequential read accesses in this case.

Again, it should be cleared here:

1. that the signals: ips_rwb, ips_addr[11:0] and ips_wdata don’t require any special treatment and may be connected directly to the peripherals, since they are generated by the Master and held valid whenever the master_module_en is enabled.
2. the master_xfr_err signal is handled like the slave_rdata[31:0] signals as it can logically be treated as bit number 33 in the slave_rdata bus (in read cycles), indicates if the data is valid or not.

Hence, write accesses and the following signals are not shown in the timing diagram: ips_addr[11:0], ips_wdata

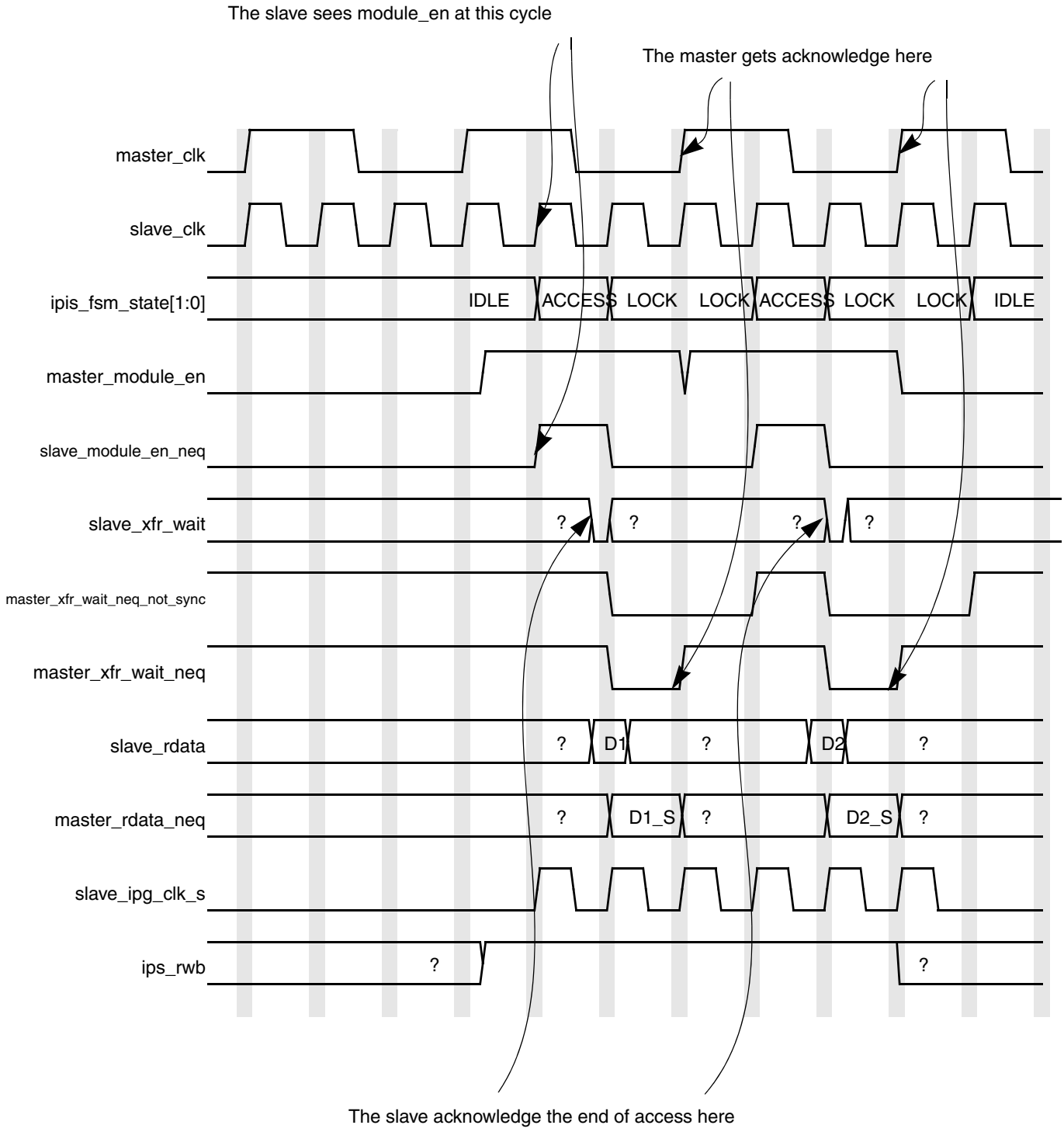


Figure 31-9. Master clock Slower Than Slave Clock—Timing Diagram (2 Read Accesses)

Chapter 32
Keypad Port (KPP)

32.1 Introduction

The kKeypad port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O). Figure 32-1 shows the KPP block diagram.

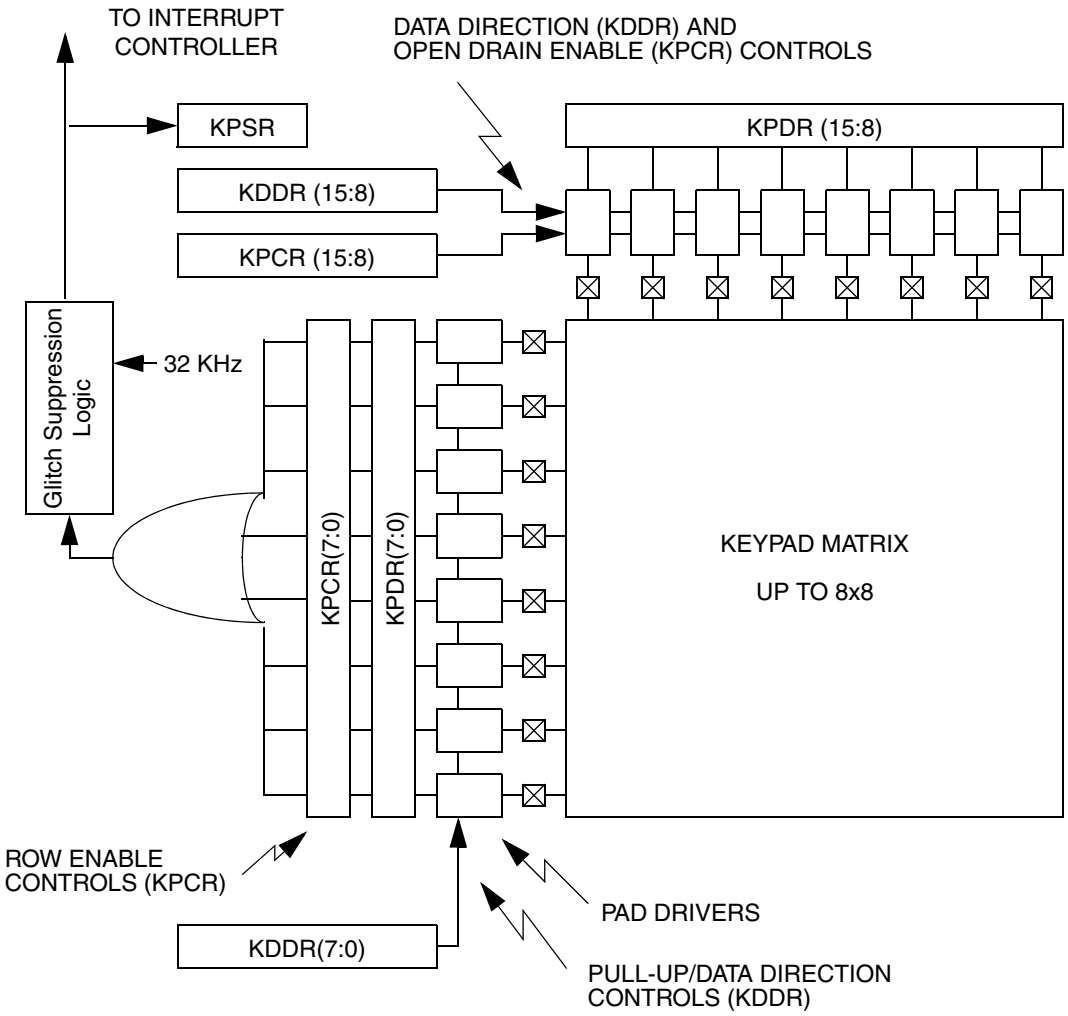


Figure 32-1. KPP Peripheral Block Diagram

32.2 Overview

The KPP is designed to interface with the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

32.2.1 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection
- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a two-point and three-point contact key matrix

32.2.2 Modes of Operation

This module supports the following modes of operation:

- Run mode—This is the normal functional mode in which the KPP can detect any key press event.
- Low power modes—The keypad can detect any key press even in low power modes (when there is no core clock).

32.3 External Signal Description

32.3.1 Overview

There are 16 pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide.

See Table 32-1 for the list of external signals.

Table 32-1. Signal Properties

Name	Port	Function	Reset State	Pull up
ipp_ind_col[7:0]	—	Column input pin, from chip	0	Active[1]
ipp_ind_row[7:0]	—	Row input pin, from chip	1	Active[1]

Table 32-1. Signal Properties (continued)

Name	Port	Function	Reset State	Pull up
ipp_do_col[7:0]	—	Column output pin going to chip	0	—
ipp_obe_col[7:0]	—	Enables the corresponding column outputs	0	—
ipp_ode_col[7:0]	—	Used to set the column outputs	0	—
ipp_do_row[7:0]	—	Row output pin going to chip	0	—
ipp_obe_row[7:0]	—	Enables the corresponding row output	0	—

1 The corresponding pads are required to be pull-up enabled.

32.3.1.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a “0” to the appropriate bits in the keypad data direction register (KDDR). Additionally, the least significant 8 bits (ROW inputs) corresponding to KDDR[7:0] have internal pull-ups, which are enabled when the pin is used as an input.

32.3.1.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the keypad data direction registerto a “1”. Additionally, the 8 most significant bits (15–8) can be designated as open drain outputs by writing a “1” to the appropriate bits in the keypad control register. See Table 32-2 for register settings and corresponding pin functions. The lower 8 bits (7–0) are always in “totem pole” style, driven when configured as outputs.

Table 32-2. Keypad Port Column Modes

KDDR (15:8)	KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a “1” during the scan routine. With this configuration, a time delay between the scanning of two subsequent columns is reduced.

32.4 Memory Map and Register Definition

The KPP module contains four registers. Section 32.4.3, “Register Descriptions,” provides detailed descriptions of the KPP registers.

32.4.1 KPP Memory Map

Table 32-3 shows the KPP memory map. For the base address of a particular module instantiation, see the system memory map.

Table 32-3. KPP Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (KPCR)	Keypad control register	R/W	0x4000	32.4.3.1/32-5
0x0002 (KPSR)	Keypad status register	R/W	0x0000	32.4.3.2/32-6
0x0004 (KDDR)	Keypad data direction register	R/W	0x0000	32.4.3.3/32-8
0x0006 (KPDR)	Keypad data register	R/W	0xXXXX	32.4.3.4/32-8

32.4.2 Register Summary

Figure 32-2 shows the key to the register fields and Table 32-4 shows the register figure conventions.

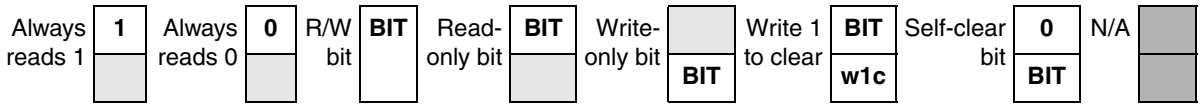


Figure 32-2. Key to Register Fields

Table 32-4. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 32-5 shows the KPP register summary.

Table 32-5. KPP Register Summary

Base Address Offset (Name Abbreviation)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (KPCR)	R W	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE0
0x0004 (KDDR)	R W	KCD D7	KCD D6	KCD D5	KCD D4	KCD D3	KCD D2	KCD D1	KCD D0	KRD D7	KRD D6	KRD D5	KRD D4	KRD D3	KRD D2	KRD D1	KRD D0
0x0006 (KPDR)	R W	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD0

32.4.3 Register Descriptions

This section consists of register descriptions; the registers are listed in address order.

32.4.3.1 Keypad Control Register (KPCR)

The Keypad control register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPCR register is byte- or half-word–addressable.

Figure 32-3 shows the valid bits in the KPCR register, and Table 32-6 provides its field descriptions.

Offset 0x0000 (KPCR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCO7	KCO6	KCO5	KCO4	KCO3	KCO2	KCO1	KCO0	KRE7	KRE6	KRE5	KRE4	KRE3	KRE2	KRE1	KRE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-3. KPCR Register

Table 32-6. Keypad Control Register Field Descriptions

Field	Description
15–8 KCO	Keypad column strobe open-drain enable. Setting a column open-drain enable bit (KCO7–KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input. 0 Column strobe output is totem pole drive. 1 Column strobe output is open drain. **Note:** Configuration of external port control logic (for example, GPIO) should be done properly so that the KPP module controls an open-drain enable of the pin.
7–0 KRE	Keypad row enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a “0” to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

32.4.3.2 Keypad Status Register (KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPSR register is byte- or half-word–addressable.

Figure 32-4 shows the KPSR register, and Table 32-7 provides its field descriptions.

Offset 0x0002 (KPSR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	KPP_EN	KRIE	KDIE	0	0	0	0	0	0	KPKR	KPKD
W													KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 32-4. KPSR Register Diagram

Table 32-7. Keypad Status Register Field Descriptions

Field	Description
15–11	Reserved
10 KPP_EN	Keypad clock gating enable. Signal generated using this bit can be used by “chip clock control module” for gating of module’s high frequency clock for register accesses and synchronization. Output of this bit is not used anywhere inside the KPP module. 0 Disable high frequency clock to keypad module. 1 Enable high frequency clock to keypad module.

Table 32-7. Keypad Status Register Field Descriptions (continued)

Field	Description
9 KRIE	Keypad release interrupt enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 KDIE	Keypad key depress interrupt enable. Software should ensure that the interrupt for a key release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4	Reserved.
3 KRSS	Key release synchronizer set. Self-clear bit. The key release synchronizer is set by writing1 to this bit. Reads return a value of 0. 0 No effect 1 Set bits which sets keypad release synchronizer chain
2 KDSC	Key depress synchronizer clear. Self-clear bit. The key depress synchronizer is cleared by writing 1 to this bit. Reads return a value of 0. 0 No effect 1 Set bits that clear the keypad depress synchronizer chain
1 KPKR	Keypad key release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. 0 No key release detected 1 All keys have been released Reset value of register is “0” as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become “1”.
0 KPKD	Keypad key depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the 32 KHz clock) elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0 No key presses detected 1 A key has been depressed

32.4.3.3 Keypad Data Direction Register (KDDR)

The bits in the KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KDDR register is byte- or half-word–addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in row DDR is cleared.

Figure 32-5 shows the valid bits in the KDDR register, and Table 32-8 provides its field descriptions.

Offset 0x0004 (KDDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCDD 7	KCDD 6	KCDD 5	KCDD 4	KCDD 3	KCDD 2	KCDD 1	KCDD 0	KRDD 7	KRDD 6	KRDD 5	KRDD 4	KRDD 3	KRDD 2	KRDD 1	KRD D0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 32-5. KDDR Register Diagram

Table 32-8. Keypad Data Direction Register Field Descriptions

Field	Description
15–8 KCDD	Keypad column data direction register. Setting any bit configures the corresponding pin as an output. 0 COL*n*[1] pin is configured as an input. 1 COL*n*[1] pin is configured as an output.
7–0 KRDD	Keypad row data direction. Setting any bit configures the corresponding pin as an output. 0 ROW*n*[1] pin configured as an input. 1 ROW*n*[1] pin configured as an output.

[1] *n*=7-0

32.4.3.4 Keypad Data Register (KPDR)

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPDR register is byte- or half-word–addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

Figure 32-6 shows the KPDR register, and Table 32-9 provides its field descriptions.

Offset 0x0006 (KPDR) Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD7	KCD6	KCD5	KCD4	KCD3	KCD2	KCD1	KCD0	KRD7	KRD6	KRD5	KRD4	KRD3	KRD2	KRD1	KRD 0
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 32-6. Keypad Data Register Diagram

Table 32-9. Keypad Data Register Field Descriptions

Field	Description
15-8 KCD	Keypad column data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to column ports 1 Read/Write “1” from/to column ports
7-0 KRD	Keypad row data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write “0” from/to row ports 1 Read/Write “1” from/to row ports

32.5 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes providing that a 32 KHz clock is on. The KPP may generate a CPU interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

32.5.1 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

32.5.2 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

32.5.3 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several (3 or 4) consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

32.5.4 Keypad Standby

There is no need for the CPU to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point, the CPU can attend to other tasks or revert to a low power standby mode. The KPP will interrupt the CPU if any key is pressed.

Upon receiving a keypad interrupt, the CPU should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

32.5.5 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the CPU. The circuit is a 4-state synchronizer clocked from a 32 KHz clock source. This clock must continue to run in any low power mode where the keypad is a wake-up source, as the CPU interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods (for 32 KHz clock: 93.75 µs) in duration. Noise filtering of the duration between three to four clock periods (for the 32 KHz clock: between 93.75 µs and 125 µs) cannot be guaranteed. The interrupt output is latched in an SR latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See Figure 32-7 for a functional diagram of the keypad synchronizer.

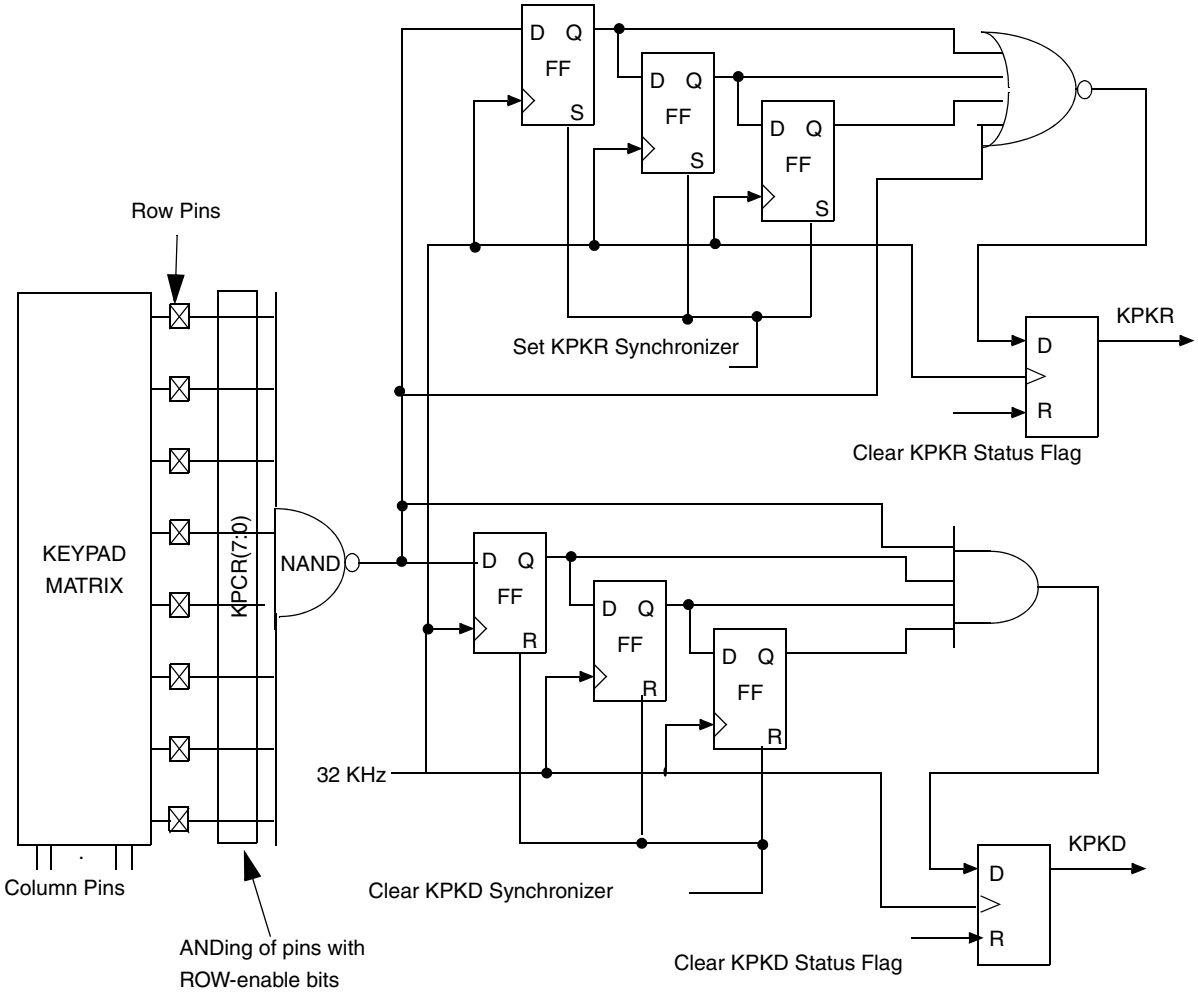


Figure 32-7. Keypad Synchronizer Functional Diagram

32.5.6 Multiple Key Closures

Using the key press and key release interrupts, the software can detect multiple keys or achieve N-key rollover. The key scanning routine can be programmed accordingly. See Section 32.6, "Initialization/Application Information," for more information.

See Figure 32-6 and Figure 32-9 for illustrations of the interfacing of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the keypad data register, pressed keys can be detected. Similarly, when keys present on same row line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

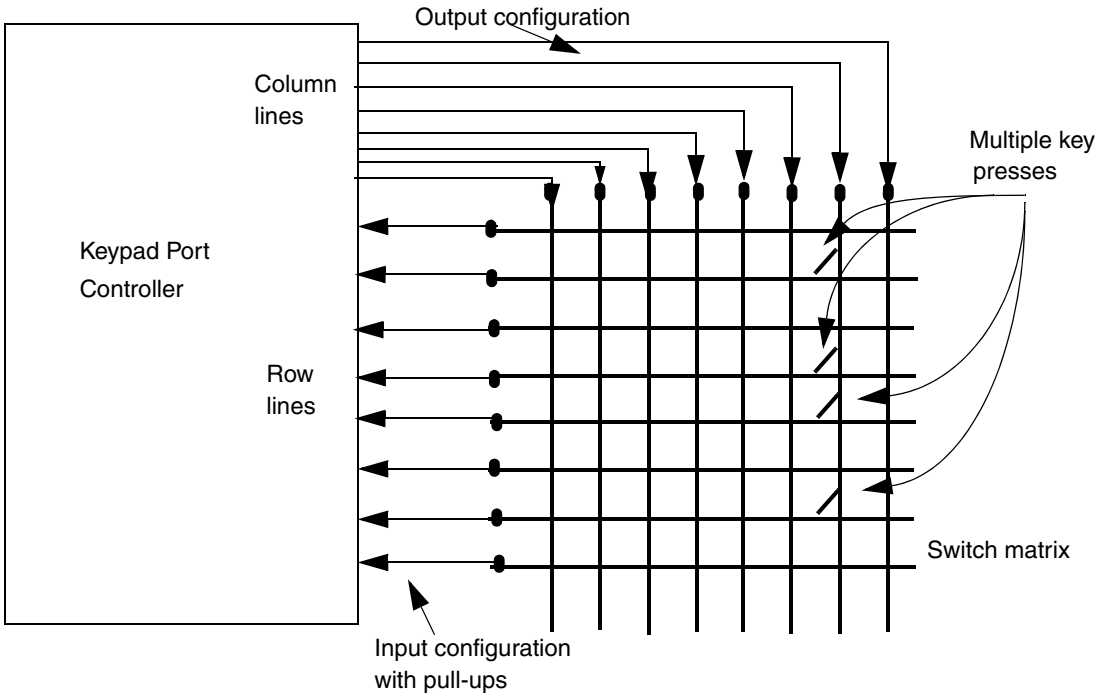


Figure 32-8. Multiple Key Presses on Same Column Line (Simplified View)

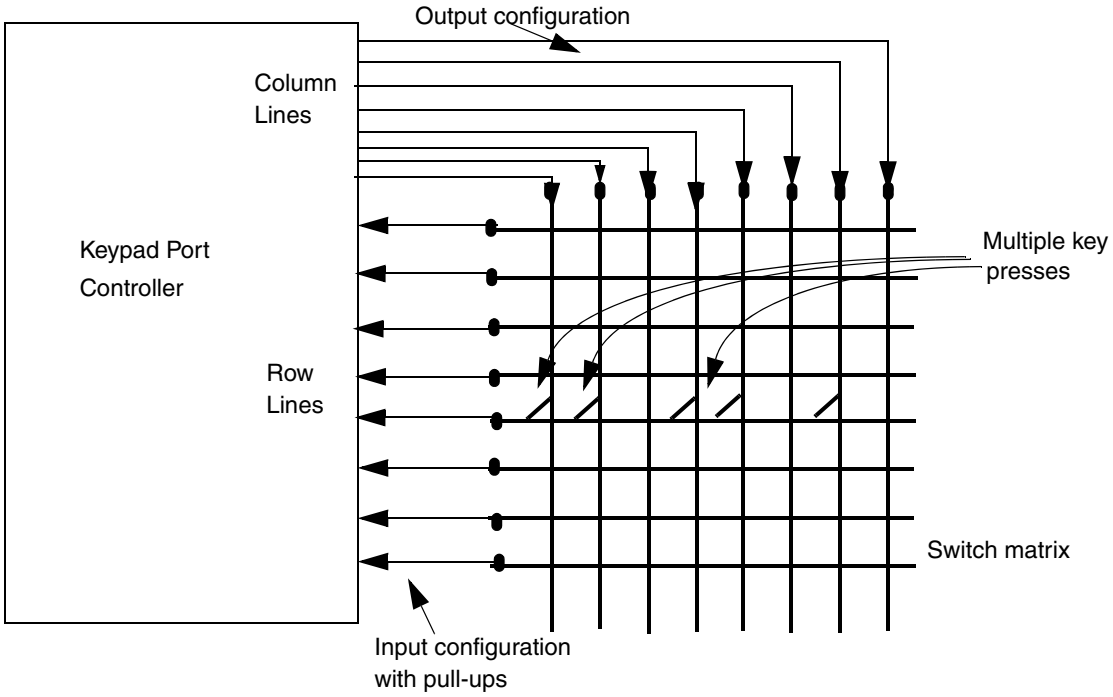


Figure 32-9. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An N-key rollover is a technique with which the system can recognize the order in which keys are pressed.

32.5.6.1 Ghost Key Problem and Correction

The KPP module detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of “ghost” key detection when three or more keys are pressed. However, this can be corrected by using a keypad matrix that provides “ghost” key protection. Such a matrix implements a one-way “diode” at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key .3-Point Contact Keys Support

The KPP module supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 32-10, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic). The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

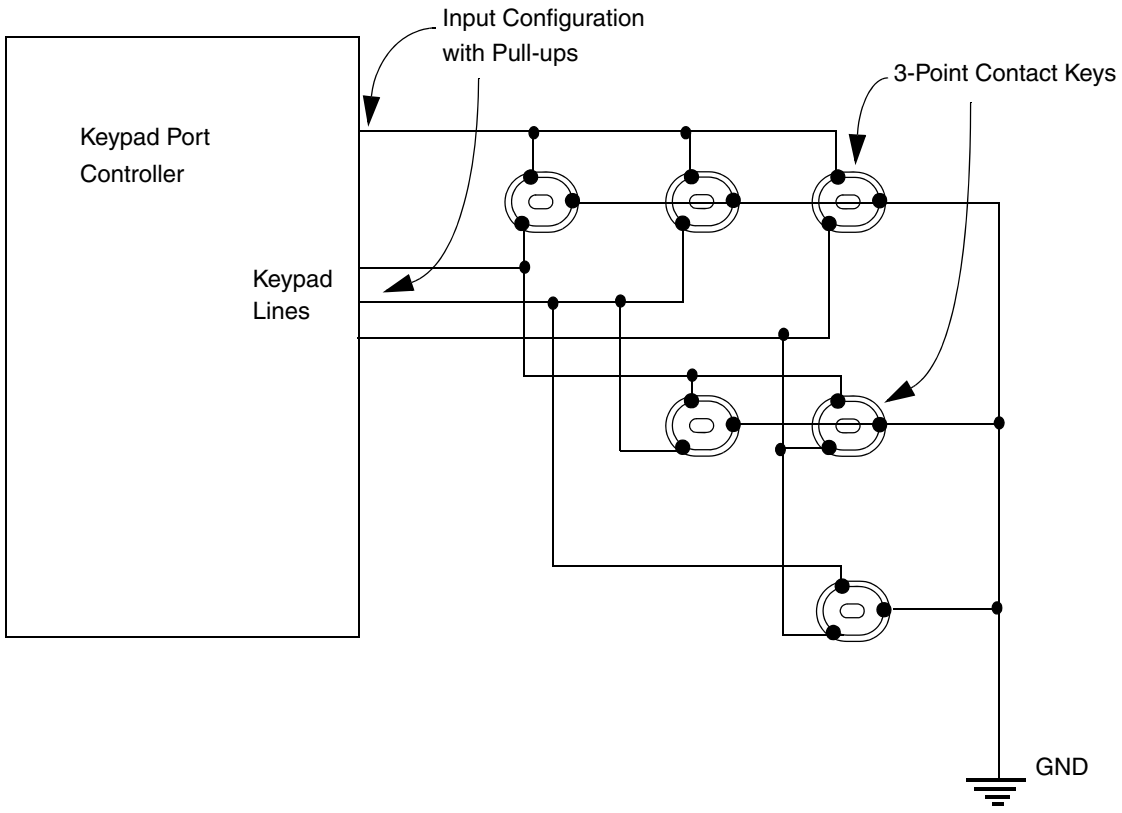


Figure 32-10. KPP Interface with 3-Point Contact Key Matrix (Simplified View)

32.6 Initialization/Application Information

32.6.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPCR[7:0]).
2. Write 0's to KPDR[15:8].
3. Configure the keypad columns as open-drain (KPCR[15:8]).
4. Configure columns as output and rows as input (KDDR[15:0]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).
7. (The system is now in standby mode, and awaiting a key press.)

32.6.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.
2. Write 1's to KPDR[15:8], setting column data to 1's.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).
4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2–6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing 1; set the KPKR synchronizer chain by writing a 1 to the KRSS status bit; and clear the KPKD synchronizer chain by writing 1 to the KDSC status bit (in the KPSR register).
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

32.6.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application's demands. The reason that such functionality cannot be put in the KPP module is that the module is limited by the number of external pins.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

Chapter 33
Liquid Crystal Display Controller (LCDC)

33.1 Introduction

The LCDC provides display data for external greyscale or color LCD panels. LCDC is capable of supporting black-and-white, greyscale, passive-matrix color (passive color or CSTN), and active-matrix color (active color or TFT) LCD panels. LCDC provides the following features:

- 32-bit AHB bus width
- Maximum screen resolution of 800 × 600
- Support for single-screen (non-split) monochrome or color LCD panels, and for self-refresh type LCD panels, with characteristics are shown in Table 33-1:

Table 33-1. Supported Panel Characteristics

Panel Type	Bits/Pixel	Panel Interface (Bits)	Number of Grey Levels/Colors	Palette Size	RGB Scheme
Monochrome	1	1, 2, 4, 8	black-and-white	—	—
	2		4	16	—
	4		16	16	—
CSTN	4 8 12	12	16 256 4096	4096	RGB444
TFT	4 8	18	16 256	256	RGB666
	12	12, 16, 18, 24	4096	4096	RGB444
	16[1]		64K	64K	RGB565
	18		256K	256K	RGB666
	24[1]		16384K	16384K	RGB888

[1] Supports 16- and 24-bpp AUO TFT panels

- For 4 and 8 bpp color, a palette table is used for remapping of data from memory, independent of the type of panel used. For 1, 2, 12, 16, 18, and 24 bpp, the palette table is bypassed.
- Supports timing requirements for Sharp 240 × 320 HR-TFT panel
- Hardware-generated cursor with blink, color, and size programmability
- Logical operation between color hardware cursor and background
- Hardware panning (soft horizontal scrolling)
- 8-bit pulse width modulator for software contrast control

- Graphic window support for viewfinder function in color display
- Graphic window color keying for graphical hardware cursor
- 256 transparency levels for alpha blending between graphic window and background plane

Figure 33-1 provides a top-level diagram of the LCDC.

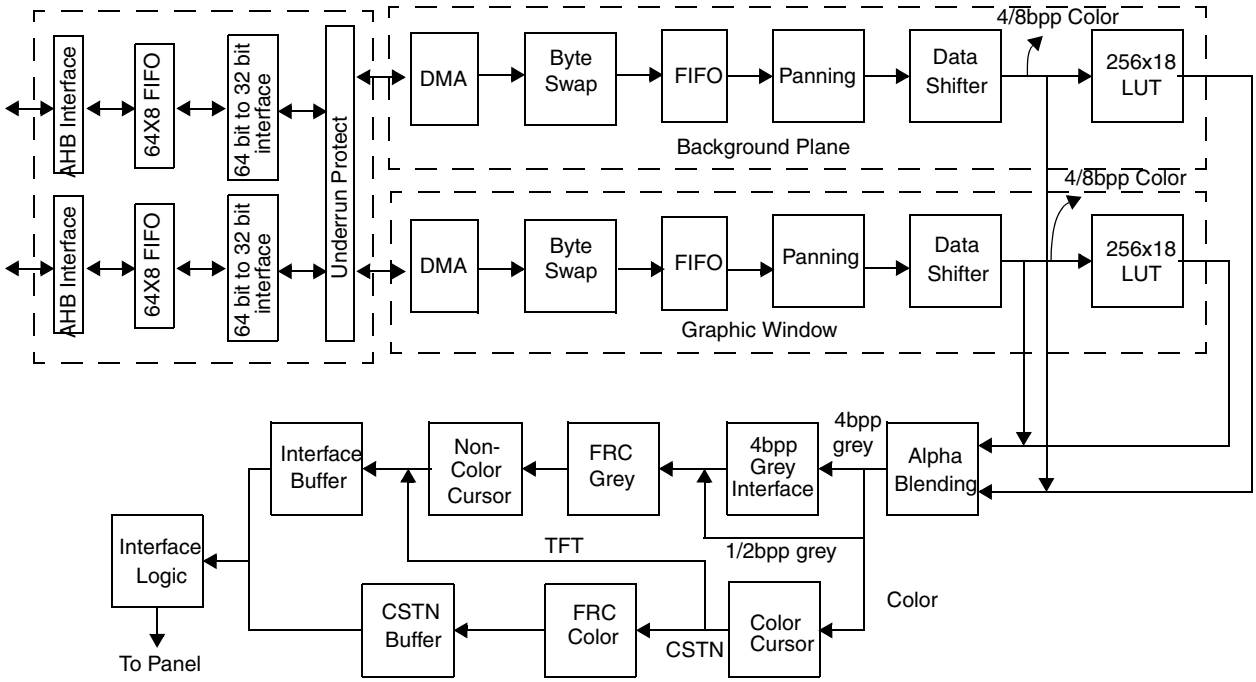


Figure 33-1. LCDC Block Diagram

33.2 LCDC Operation

The following sections describe the operation of LCDC with various industry standard LCD displays.

33.2.1 LCD Screen Format

Figure 33-2 shows the relationship between the screen size and memory window.

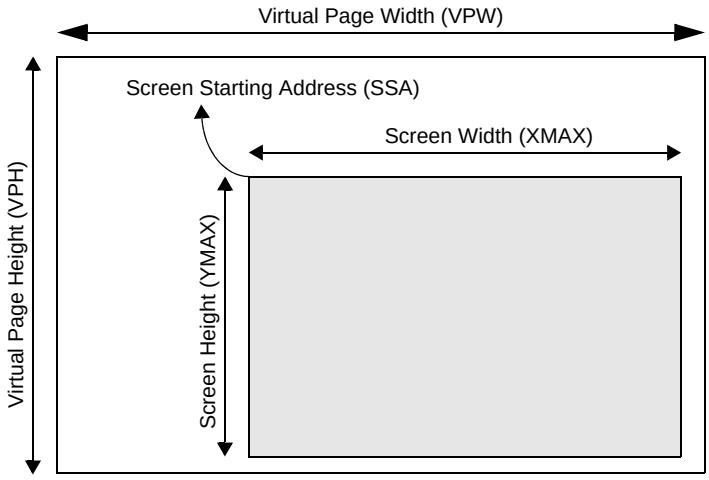


Figure 33-2. LCD Screen Format

The parameters shown in Figure 33-2 are programmable using LCDC registers, and have the following characteristics:

- Screen width (XMAX) and screen height (YMAX) specify the LCD panel size, in pixels.
- Virtual page width (VPW) specifies the maximum page width. VPW is used to calculate the RAM starting address representing the beginning of each displayed line.
- Virtual page height (VPH) is used by the programmer only for boundary checks. There is no VPH parameter internal to LCDC. VPH is limited only by memory size.
- Screen starting address (SSA) gives the location in display memory where LCDC begins scanning for display of the frame on the LCD panel. By changing the SSA register, a screen-sized window can be vertically or horizontally scrolled (panned) anywhere inside the virtual page boundaries. The software must control the SSA so that the scanning logic's system memory pointer (SMP) remains within the VPW and VPH limits: otherwise, artifacts may appear on-screen.

33.2.2 Graphic Window on Screen

A graphic window is supported in the LCD color panel screen for viewfinder and graphic hardware cursor functions. Figure 33-3 shows how the graphic window is configured and placed on the screen. The graphic window position on screen, virtual page width, start address, width and height are all programmable using LCDC registers.

Graphic window and background plane can be alpha blended with a constant alpha for the entire window, so that all pixels in the graphic window have the same transparency level (256 possible levels). In addition, a single pixel can be specified for color keying, so that the selected pixel color in the graphics window is made totally transparent: this feature can be used to construct a graphical hardware cursor.

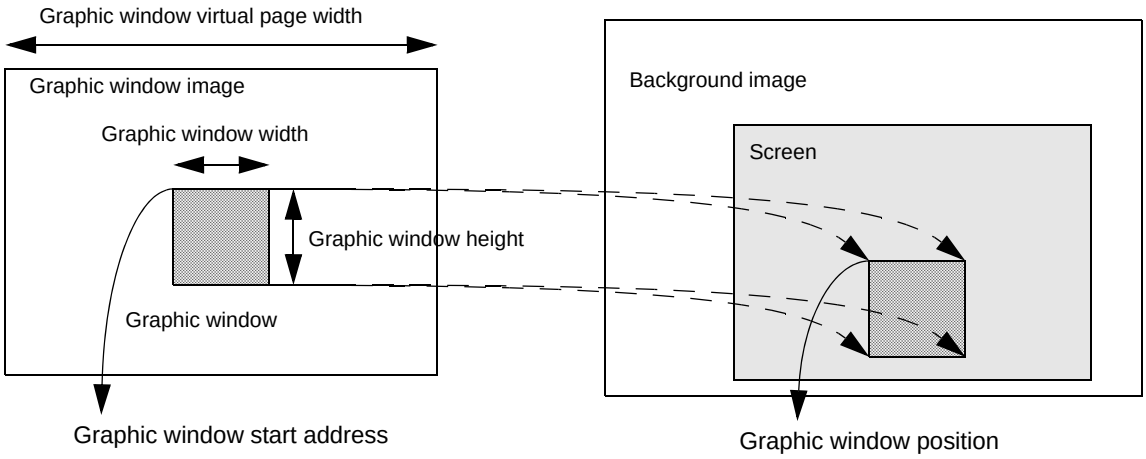


Figure 33-3. Graphic Window on Screen

NOTE

Graphic window and background image must have the same bpp setting.

33.2.3 Panning

Panning offset (POS) is expressed in bits rather than pixels, but must align to pixel boundaries. For example, in 12-bpp mode, pixels are aligned to 16-bit boundaries, so POS must be a multiple of 16. SSA and POS are located in isolated registers and are double buffered to accommodate real-time changes in parameter values. New values of POS (and SSA) do not take effect until the beginning of next frame. A typical panning algorithm includes an interrupt at the beginning of frame, and the interrupt service routine updates POS and/or SSA while old values are internally latched. The updates take effect on the next frame.

33.2.4 Display Data Mapping

LCDC supports 1-,2-,and 4-bpp in monochrome mode and 4-, 8-, 12-, 16-, 18-, and 24-bpp in color mode. System memory data is mapped to successive pixels as shown in Figure 33-4.

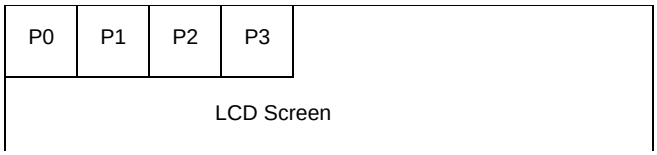


Figure 33-4. Pixel Location on Display Screen

Figure 33-5 through Figure 33-7 show data mapping to pixels in the various bpp modes. The number of bits allocated per pixel is always a power of 2, so that 12-,18-, and 24-bpp modes use 16, 32, and 32 bits of memory per pixel, respectively.

1-bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P24	P25	P26	P27	P28	P29	P30	P31
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P16	P17	P18	P19	P20	P21	P22	P23
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P8	P9	P10	P11	P12	P13	P14	P15
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0	P1	P2	P3	P4	P5	P6	P7

2-bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P12		P13		P14		P15	
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P8		P9		P10		P11	
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P4		P5		P6		P7	
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0		P1		P2		P3	

4-bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P6				P7			
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P4				P5			
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P2				P3			
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0				P1			

8-bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	P3							
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	P2							
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	P1							
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	P0							

12-bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
					Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green1 [3]	Green1 [2]	Green1 [1]	Green1[0]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
					Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green0 [3]	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
	Red1 [4]	Red1 [3]	Red1 [2]	Red1 [1]	Red1 [0]	Green1 [5]	Green1 [4]	Green1 [3]
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Green1 [2]	Green1 [1]	Green1 [0]	Blue1 [4]	Blue1 [3]	Blue1 [2]	Blue1 [1]	Blue1 [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Red0 [4]	Red0 [3]	Red0 [2]	Red0 [1]	Red0 [0]	Green0 [5]	Green0 [4]	Green0 [3]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Green0 [2]	Green0 [1]	Green0 [0]	Blue0 [4]	Blue0 [3]	Blue0 [2]	Blue0 [1]	Blue0 [0]

Figure 33-5. Display Data Mapping, 1-bpp–16-bpp Modes

Normal 18bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]		
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]		
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]		

Microsoft PAL_BGR 18bpp Mode

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]		
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]		
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]		

Figure 33-6. Display Data Mapping for 18-bpp Modes (Normal and Microsoft PAL_BGR)

Byte Address	Sample Bit-to-Pixel Mapping							
3	Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
2	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
	Blue[7]	Blue[6]	Blue [5]	Blue [4]	Blue [3]	Blue [2]	Blue [1]	Blue [0]
1	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Breen[7]	Green[6]	Green [5]	Green [4]	Green [3]	Green [2]	Green [1]	Green [0]
0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Red[7]	Red[6]	Red [5]	Red [4]	Red [3]	Red [2]	Red [1]	Red [0]

Figure 33-7. Display Data Mapping for 24-bpp AUS Mode (for AUO panels)

33.2.5 Black-and-White Operation

Black-and-white operation is equivalent to 1-bpp mode. Each pixel is always either fully on or fully off.

33.2.6 Greyscale Operation

In 2 bpp and 4-bpp modes, the LCDC displays 4 and 16 grey levels respectively. The shades of grey are obtained by controlling the number of frames in which the pixel is “on” over a period of 16 frames: this method is known as frame rate control (FRC). For more information on FRC, see Section 33.2.8, “Frame Rate Modulation Control (FRC).”

In 2-bpp mode the 2-bit pixel code is mapped to one of the four grey levels using internal logic, while in 4-bpp mode, the 4-bit pixel code is mapped to one of the 16 grey levels by writing to the mapping RAM. The greyscale mapping is shown in Figure 33-8.

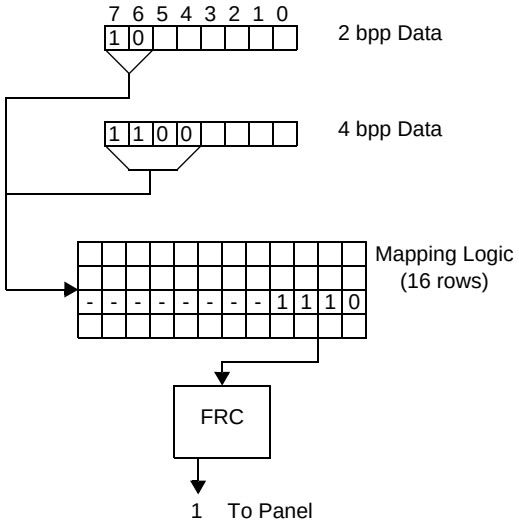


Figure 33-8. Greyscale Pixel Generation

Because crystal formulations and driving voltages vary, visual grey effects may or may not be linearly related to the frame rate. For example, for 2-bpp greyscale mode a logarithmic scale such as 0, 1/4, 1/2 and 1 might be more pleasing than a linearly spaced scale such as 0, 5/16, 11/16 and 1 for certain graphics.

33.2.7 Color Generation

The value corresponding to each color pixel on the screen is represented by a 4-, 8-, 12-, 16-, or 18-bit code in the display memory, depending on the bits per pixel (bpp) setting.

Figure 33-9 shows color pixel generation for passive matrix displays. The processing for the different bpp modes is as follows:

- In 4-bpp and 8-bpp modes for passive matrix displays, the LCDC's color mapping RAM is used to map the data to a 12-bit RGB code. This code is output to the FRC blocks that independently process the red, green and blue components of each pixel to generate the required shade and intensity.
- For 12-bit mode for passive matrix color display the mapping RAM is bypassed, and the RGB data is output directly to the FRC blocks.

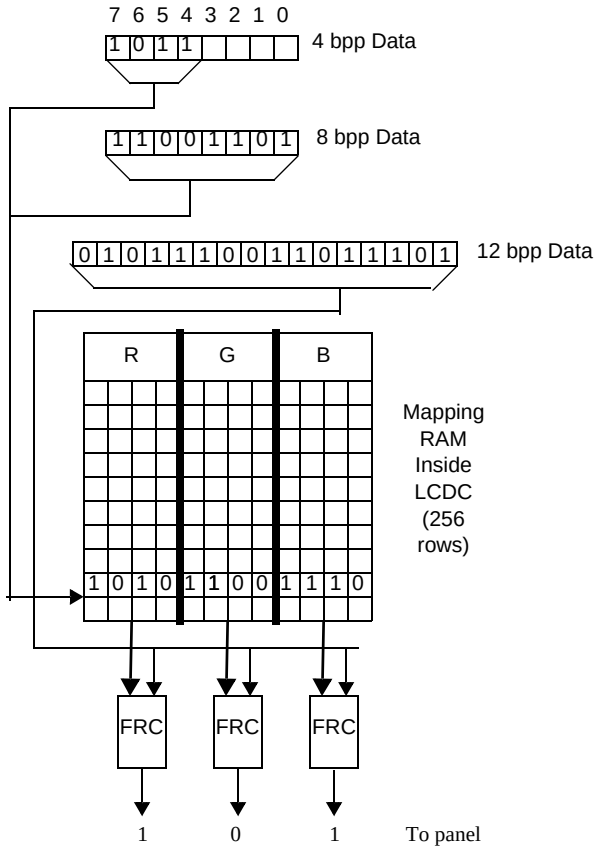


Figure 33-9. Passive Matrix Color Pixel Generation

Figure 33-10 shows color pixel generation for active matrix displays. The processing for the different bpp modes is as follows:

- In 4- and 8-bpp modes for active matrix displays, the data is mapped to an RGB code with 18 bits, which is output directly to the panel.
- For 12-, 16-, and 18-bpp modes for active matrix color displays, pixel data is simply moved from display memory to the LCDC output bus.

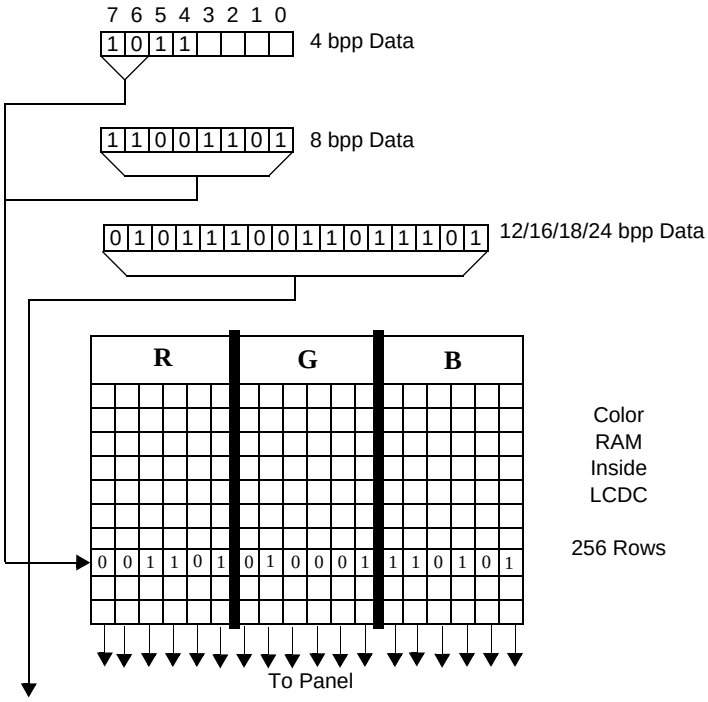


Figure 33-10. Active Matrix Color Pixel Generation

33.2.8 Frame Rate Modulation Control (FRC)

The FRC circuitry inside the LCDC generates intermediate grey or color levels on the panel by adjusting the density of zeroes and ones that appear over the frames. Table 33-2 shows the 16 LCDC greyscale levels.

Table 33-2. FRC Palette Density

Level Code (Hexadecimal)	Density	Density (Decimal)
0	0	0
1	1/8	0.125
2	1/5	0.2
3	1/4	0.25
4	1/3	$0.\overline{333}$
5	2/5	0.4
6	4/9	$0.\overline{444}$
7	1/2	0.5
8	5/9	$0.\overline{555}$
9	3/5	0.6
A	2/3	$0.\overline{666}$
B	3/4	0.75
C	4/5	0.8

Table 33-2. FRC Palette Density (continued)

Level Code (Hexadecimal)	Density	Density (Decimal)
D	7/8	0.875
E	14/15	$0.9\overline{33}$
F	1	1

Note: Overbars indicate repeating decimal numbers.

33.2.9 Panel Interface Signals and Timing

LCDC continuously provides pixel data to the LCD panel using the LCD panel interface. Figure 33-11 shows the panel interface signals.

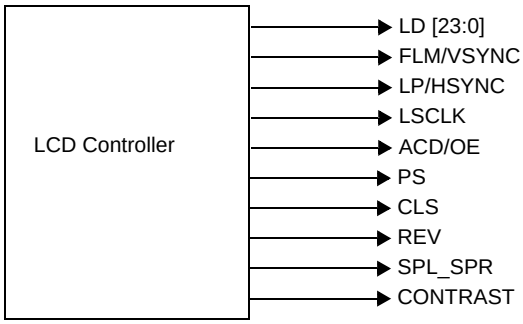


Figure 33-11. LCDC Interface Signals

Format, timing and polarity of panel interface signals are programmable using LCDC registers. The SPL_SPR, PS, CLS and REV signals shown in Figure 33-11 are dedicated for Sharp HR-TFT 240 × 320 panels only.

The following subsections discuss LCDC signal multiplexing and signal timing in the various operation modes (greyscale/color, passive/active).

33.2.9.1 Passive Matrix Panel Interface Timing

Figure 33-12 shows the LCD interface timing for monochrome panels, and Figure 33-13 shows LCD interface timing for passive matrix color panels. Signal polarities are shown as positive: however, polarities can be reversed by setting to 1 the corresponding bits in LCDC panel configuration register (LPCR).

Data bus timing for passive panels is controlled by shift clock (LSCLK), line pulse (LP), first line marker (FLM), alternate crystal direction (ACD) and line data (LD) signals. The panel interface signals operate as follows:

- LSCLK clocks the pixel data into the display driver's internal shift register.
- LP signifies the end of current line of serial data and latches the shifted pixel data into a wide latch.
- FLM marks the first line of the displayed page. LD (and the associated LP), enclosed by the FLM signal, marks the first line of the current frame.
- ACD toggles after a preprogrammed number of FLM pulses. This signal refreshes the LCD panel.

NOTE

The LD bus width is programmable to 1, 2, 4, or 8 bits in monochrome mode (that is, when the COLOR bit in Panel Configuration register is cleared). Data is justified to the least significant bits of the LD [17:0] bus. Passive color displays use a fixed 2-2/3 pixels of data per 8-bit vector as shown in Figure 33-13.

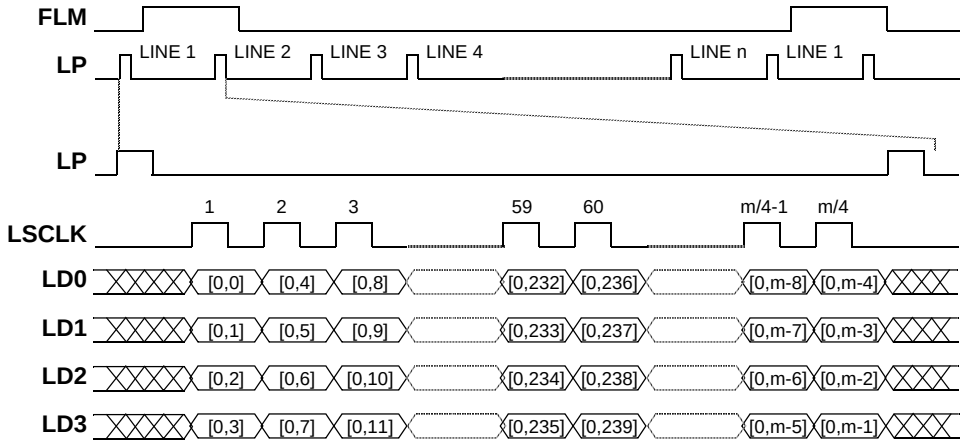


Figure 33-12. LCDC Interface Timing for 4-bit Data Width Greyscale Panels

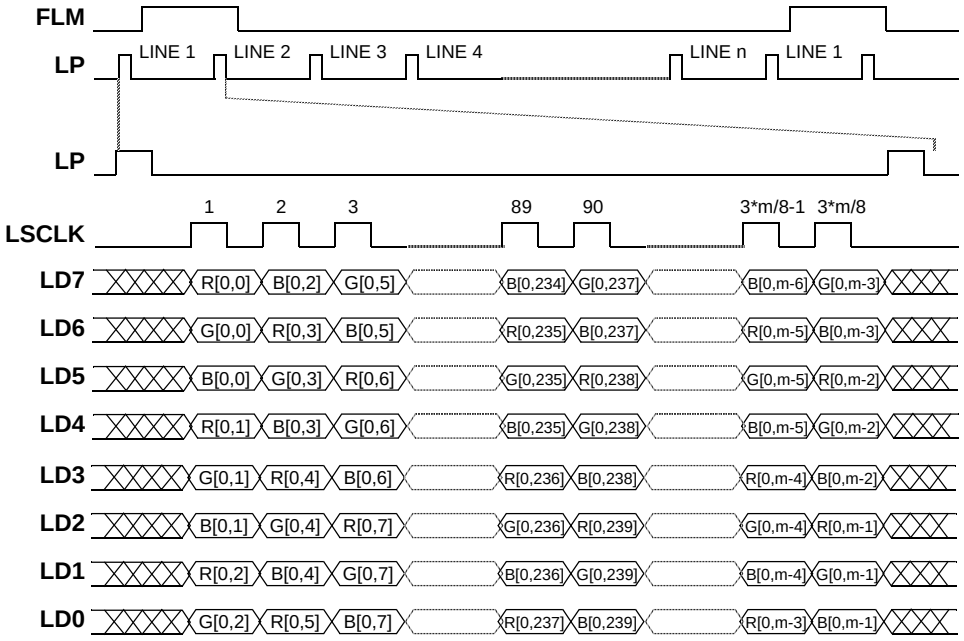
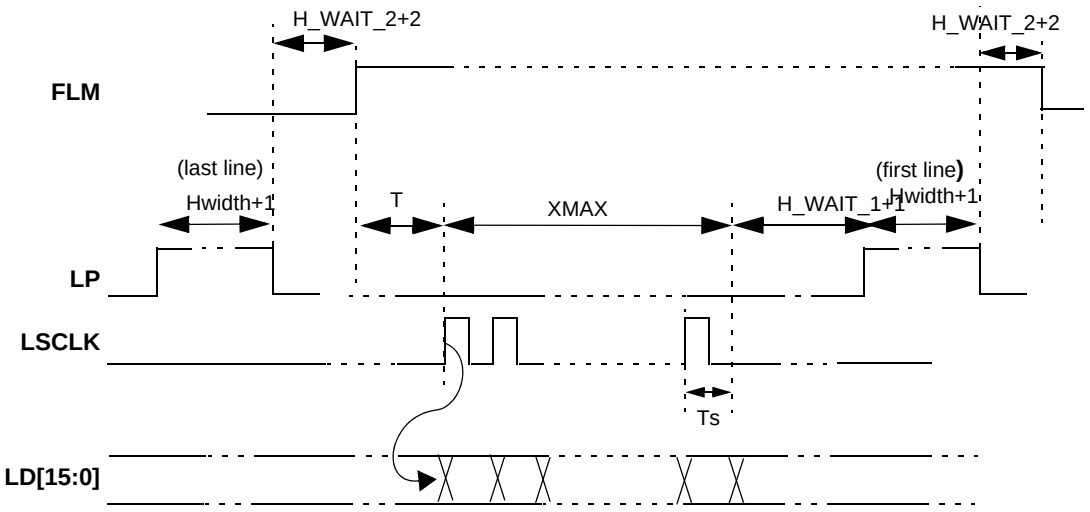


Figure 33-13. LCDC Interface Timing for 8-bit Data Width Passive Matrix Color Panels

33.2.9.1.1 Passive Panel Horizontal Synch Pulse Timing

Figure 33-14 shows passive panel horizontal timing (timing of one line), including both line pulse (LP) and data. The LP width and delays (both before and after LP) are programmable.



In CSTN mode or monochrome mode with bus width = 1, T = 1 SCLK period.
In monochrome mode with bus width = 2, 4, 8, T = 1, 2, 4 SCLK periods respectively.

Figure 33-14. Horizontal Sync Pulse Timing in Passive Mode

The parameters used for panel interface timing are:

- XMAX is the total number of pixels per line
- H_WAIT_1 is the delay (in pixel clocks) from the end of data output to the beginning of LP. This is also called the “horizontal front porch”.
- H_WIDTH is the width of the FLM pulse, in pixel clocks (must be at least 1).
- H_WAIT_2 is the delay (in pixel clocks) from end of LP to the beginning of data output. This is also called the “horizontal back porch”.

33.2.9.1.2 Passive Panel Vertical Synch Pulse Timing

Figure 33-15 shows the vertical sync pulse timings for passive panels (monochrome or CSTN).

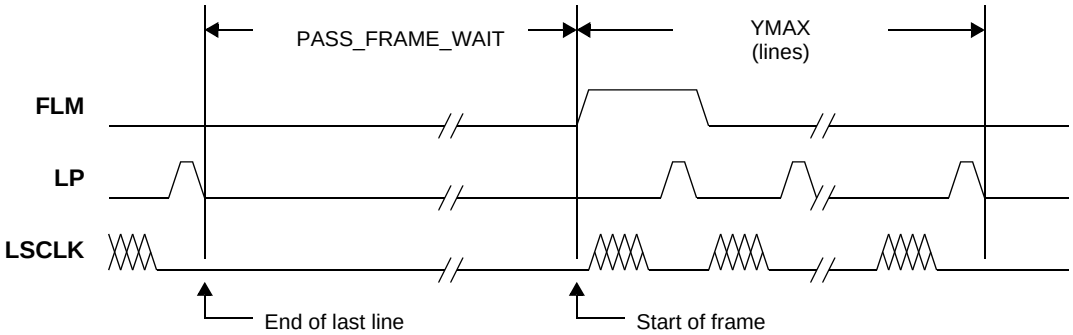


Figure 33-15. Vertical Sync Pulse Timing in Passive Mode

33.2.9.2 Active Matrix Panel Interface Signals

Figure 33-16 and Figure 33-17 shows the LCD interface timing for active matrix color TFT panels in 16- and 18-bpp modes respectively. Figure 33-18 and Figure 33-19 shows the LCD interface timing in 24-bpp and 16-bpp AUS (AUO panels) modes, respectively.

Signals are shown with negative polarity settings: the FLMPOL, LPPOL, CLKPOL, and OEPOL bits in the LCDC panel configuration register (LPCR) are all set to 1 in the LCDC panel configuration register. In TFT mode, LSCLK is automatically inverted. *m* represents the pixels per line, while *n* represents the lines per frame.

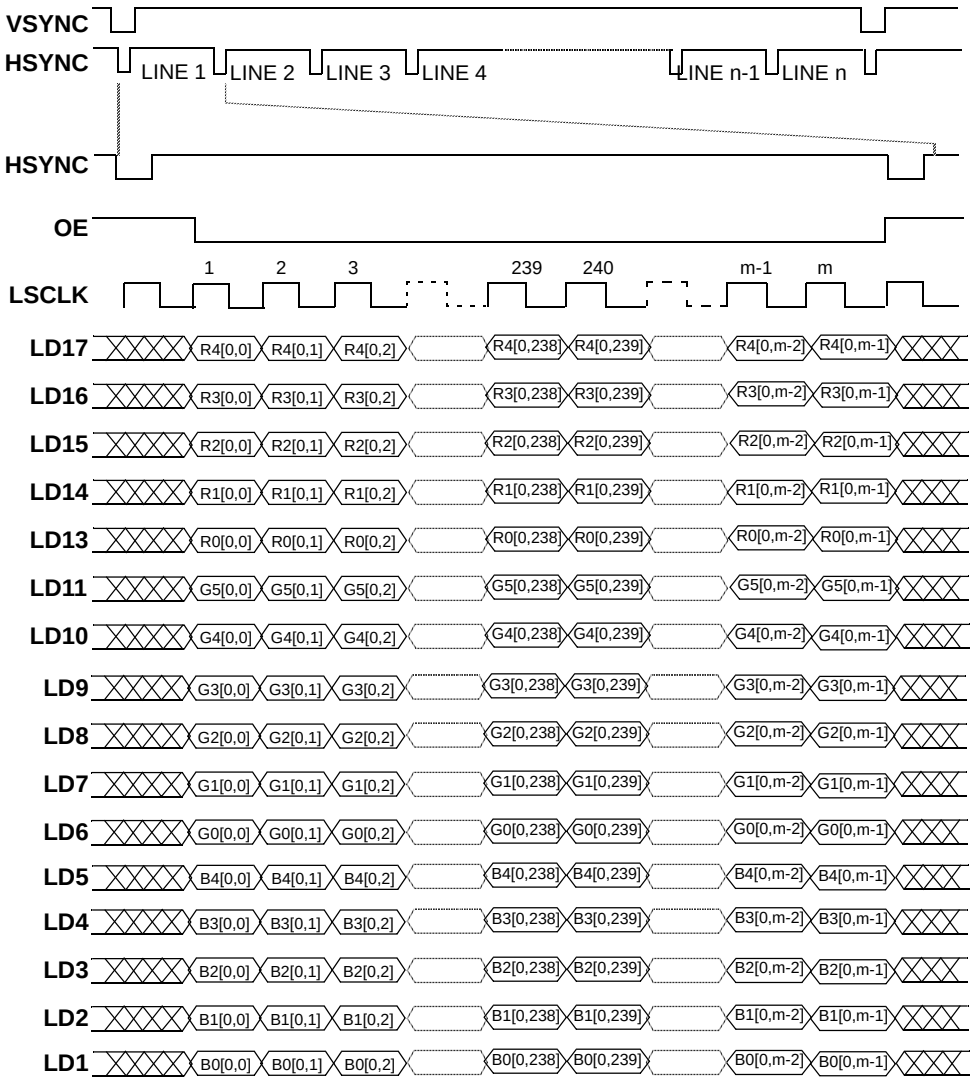


Figure 33-16. LCDC Interface 16-bit Timing for Active Matrix Color Panels

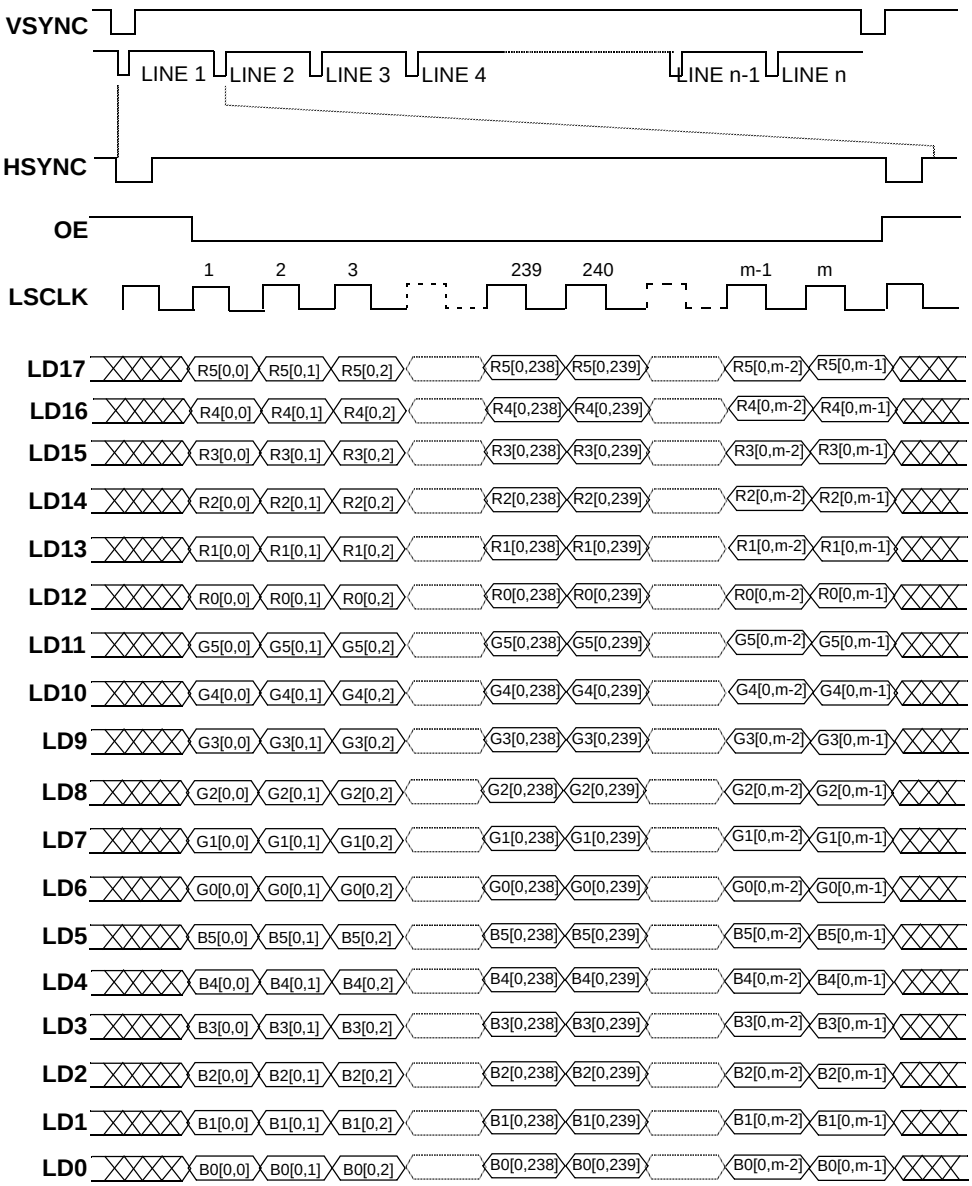


Figure 33-17. LCDC Interface 18-bit Timing for Active Matrix Color Panels

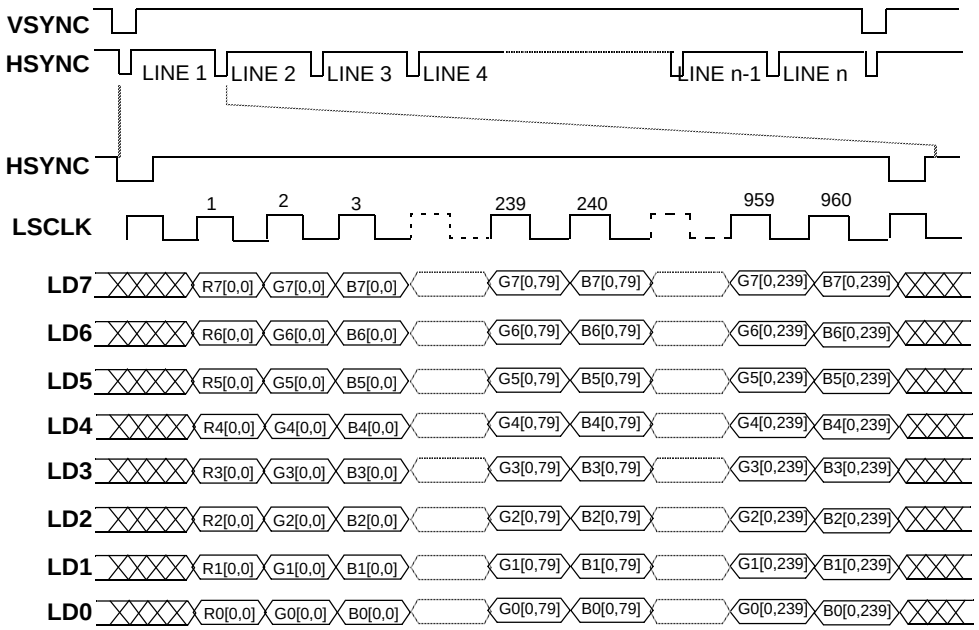


Figure 33-18. LCDC Interface Timing For 24-bpp AUS (AUO Panel) Mode

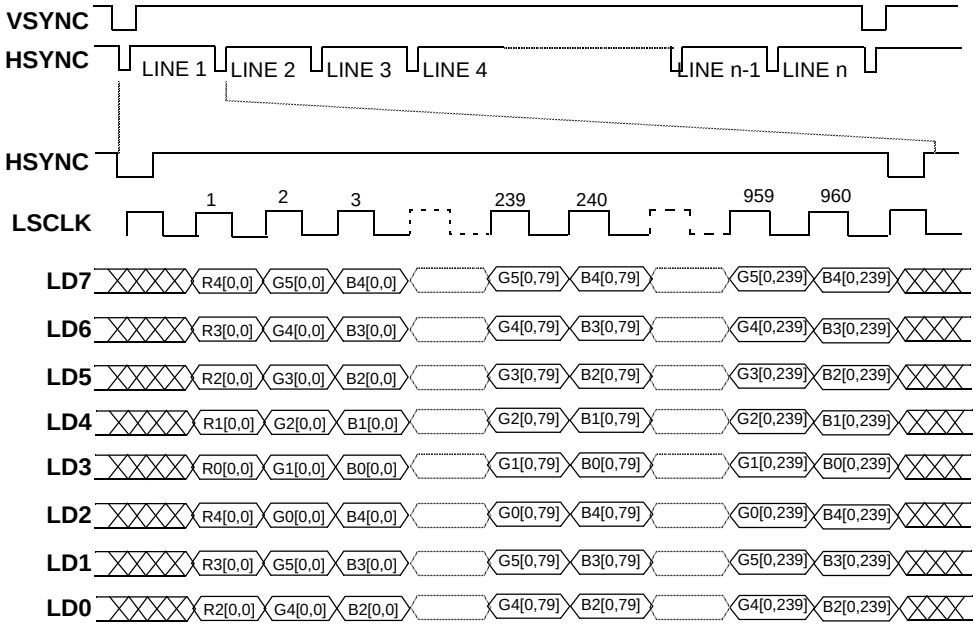


Figure 33-19. LCDC Interface Timing For 16-bpp AUS (AUO Panel) Mode

The panel interface timing for active matrix panels is sometimes referred to as a digital CRT, and is controlled by the shift clock (LSCLK), horizontal sync pulse (HSYNC, LP pin in passive mode), vertical sync pulse (VSYNC, FLM pin in passive mode), output enable (OE, ACD pin in passive mode), and line data (LD) signals.

The sequence of events for active matrix interface timing is as follows:

1. LSCLK latches data into the panel on its negative edge (when positive polarity is selected). In active mode, LSCLK runs continuously.
2. HSYNC causes the panel to start a new line.
3. VSYNC causes the panel to start a new frame. It always encompasses at least one HSYNC pulse.
4. OE functions as an output enable signal to the CRT. This output enable signal is similar to blanking output in a CRT and enables the data to be shifted onto the display. When disabled, data is invalid and the trace is off.

Table 33-3 shows the TFT color channel assignments for different bpp modes. R*n*, G*n*, and B*n* denote bits dedicated to red, green, and blue respectively. Unused bits are fixed at 0.

Table 33-3. TFT Color Channel Assignments

bpp Mode	LD 17	LD 16	LD 15	LD 14	LD 13	LD 12	LD 11	LD 10	LD 9	LD 8	LD 7	LD 6	LD 5	LD 4	LD 3	LD 2	LD 1	LD 0
4 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
8 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
12 bpp	R3	R2	R1	R0	—	—	G3	G2	G1	G0	—	—	B3	B2	B1	B0	—	—
16 bpp	R4	R3	R2	R1	R0	—	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	—
18 bpp	R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
16 bpp AUS (For AUO panels)	—	—	—	—	—	—	—	—	—	—	R4	R3	R2	R1	R0	R4	R3	R2[1]
	—	—	—	—	—	—	—	—	—	—	G5	G4	G3	G2	G1	G0	G5	G4[2]
	—	—	—	—	—	—	—	—	—	—	B4	B3	B2	B1	B0	B4	B3	B2[3]
24 bpp AUS (For AUO panels)	—	—	—	—	—	—	—	—	—	—	R7	R6	R5	R4	R3	R2	R1	R0[1]
	—	—	—	—	—	—	—	—	—	—	G7	G6	G5	G4	G3	G2	G1	G0[2]
	—	—	—	—	—	—	—	—	—	—	B7	B6	B5	B4	B3	B2	B1	B0[3]

[1] Data output at clock 3*N +1

[2] Data output at clock 3*N + 2

[3] Data output at clock 3*N + 3

Note: Need to add 24bpp paralled mode.

33.2.9.2.1 Active Panel Horizontal Timing

Figure 33-20 shows horizontal timing (timing of one line) for active matrix color TFT panels, including both horizontal sync pulse and data. The width of HSYNC and delay before and after HSYNC are all programmable. The timing signal parameters are defined as follows:

- H_WIDTH defines the width (in pixel clocks) of HSYNC pulse and must be at least 1.
- H_WAIT_2 defines the delay (in pixel clocks) from end of HSYNC to the beginning of the OE pulse.

- H_WAIT_1 defines the delay (in pixel clocks) from end of OE to the beginning of the HSYNC pulse.
- XMAX defines the (total) number of pixels per line.

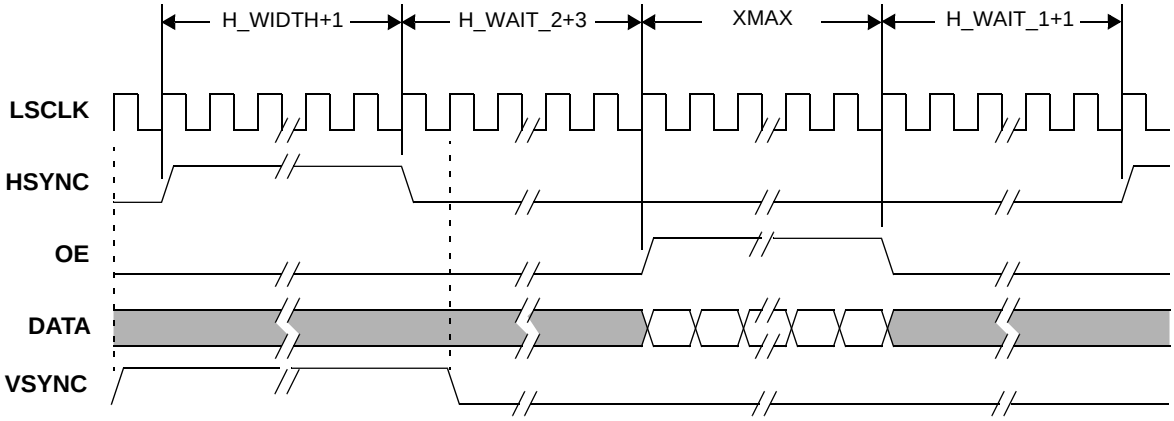


Figure 33-20. Horizontal Sync Pulse Timing in TFT Mode

33.2.9.2.2 Active Panel Vertical Timing

Figure 33-21 shows vertical timing (timing of one frame) for active matrix color TFT panels. The delay from end of one frame until the beginning of the next is programmable. The memory timing signal parameters are:

- V_WAIT_1 is a delay measured in lines. For V_WAIT_1 = 1 there is a delay of one HSYNC (time = one line period) before VSYNC. HSYNC pulse is output during the V_WAIT_1 delay. This is also called the “vertical front porch”.
- For V_WIDTH (vertical sync pulse width) = 0, VSYNC encloses one HSYNC pulse. For V_WIDTH = 2, VSYNC encloses two HSYNC pulses.
- V_WAIT_2 is a delay measured in lines. For V_WAIT_2 = 1, there is a delay of one HSYNC (time = one line period) after VSYNC. The HSYNC pulse is output during the V_WAIT_2 delay. This is also called “vertical back porch”.

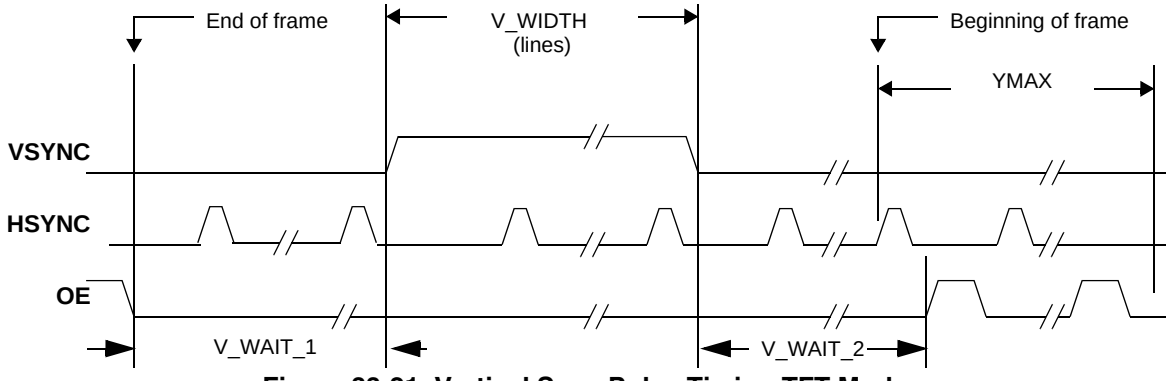


Figure 33-21. Vertical Sync Pulse Timing TFT Mode

33.3 Memory Map and Register Descriptions

This section gives the module memory map. It also includes detailed descriptions of all registers, as well as the mapping RAMs used to map a limited set of color/greyscale levels to a larger palette.

33.3.1 Memory Map

Table 33-4 is the module memory map. Only 32-bit word access is supported: byte and half-word access is undefined. For the base address of a particular module instantiation, see the system memory map.

The LCDC memory space contains 21 32-bit registers for display parameters, one read-only status register, and two 256×18 Color Mapping RAMs—one for graphic window and the other for background plane. The color mapping RAMs are physically located inside the palette lookup table module.

Table 33-4. Module Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (LSSAR)	LCDC Screen Start Address Register	RW	0x0000_0000	33.3.4/33-23
0x0004 (LSR)	LCDC Size Register	RW	0x0000_0000	33.3.5/33-23
0x0008 (LVPWR)	LCDC Virtual Page Width Register	RW	0x0000_0000	33.3.6/33-24
0x000C (LCPR)	LCDC Cursor Position Register	RW	0x0000_0000	33.3.7/33-25
0x0010 (LCWHB)	LCDC Cursor Width Height and Blink Register	RW	0x0101_00FF	33.3.8/33-26
0x0014 (LCCMR)	LCDC Color Cursor Mapping Register	RW	0x0000_0000	33.3.9/33-26
0x0018 (LPCR)	LCDC Panel Configuration Register	RW	0x0000_0000	33.3.10/33-28
0x001C (LHCR)	LCDC Horizontal Configuration Register	RW	0x0400_0000	33.3.11/33-30
0x0020 (LVCR)	LCDC Vertical Configuration Register	RW	0x0400_0000	33.3.12/33-31
0x0024 (LPOR)	LCDC Panning Offset Register	RW	0x0000_0000	33.3.13/33-32
0x0028 (LSCR)	LCDC Sharp Configuration Register	RW	0x400C_0373	33.3.14/33-32
0x002C (LPCCR)	LCDC PWM Contrast Control Register	RW	0x0000_0000	33.3.15/33-34
0x0030 (LDCR)	LCDC DMA Control Register	RW	0x8004_0060	33.3.16/33-35
0x0034 (LRMCR)	LCDC Refresh Mode Control Register	RW	0x0000_0000	33.3.17/33-36
0x0038 (LICR)	LCDC Interrupt Configuration Register	RW	0x0000_0000	33.3.18/33-37
0x003C (LIER)	LCDC Interrupt Enable Register	RW	0x0000_0000	33.3.19/33-38
0x0040 (LISR)	LCDC Interrupt Status Register	R	0x0000_0000	33.3.20/33-39
0x0050 (LGWSAR)	LCDC Graphic Window Start Address Register	RW	0x0000_0000	33.3.21/33-41
0x0054 (LGWSR)	LCDC Graphic Window Size Register	RW	0x0000_0000	33.3.22/33-41
0x0058 (LGWVPWR)	LCDC Graphic Window Virtual Page Width Register	RW	0x0000_0000	33.3.23/33-42
0x005C (LGWPOR)	LCDC Graphic Window Panning Offset Register	RW	0x0000_0000	33.3.24/33-43
0x0060 (LGWPR)	LCDC Graphic Window Position Register	RW	0x0000_0000	33.3.25/33-44

Table 33-4. Module Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0064 (LGWCR)	LCDC Graphic Window Control Register	RW	0x0000_0000	33.3.26/33-44
0x0068 (LGWDCR)	LCDC Graphic Window DMA Control Register	RW	0x8004_0060	33.3.27/33-45
0x0080 (LAUSCR)	LCDC AUS Mode Control Register	RW	0x0000_0000	33.3.28/33-47
0x0084 (LAUSCCR)	LCDC AUS mode Cursor Control Register	RW	0x0000_0000	33.3.29/33-48
0x0800–0x0BFC (BGLUT)	Background Lookup Table	RW	—	33.3.30/33-48
0x0C00–0x0FFC (GWLUT)	Graphic Window Lookup Table	RW	—	33.3.30/33-48

33.3.2 Register Summary

Table 33-5 is the register summary table.

Table 33-5. LCDC Register Summary

Offset (and Name Abbreviation)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (LSSAR)	Screen Start Address High – SSA H															
	Screen Start Address Low – SSA L															
0x0004 (LSR)				GWLPM			Screen Width – XMAX									
							Screen Height – YMAX									
0x0008 (LVPWR)																
							Virtual Page Width – VPW									
0x000C (LCPR)	CC			OP			Cursor X Position – CXP									
							Cursor Y Position – CYP									
0x0010 (LCWHB)	BK_EN			Cursor Width – CW								Cursor Height – CH				
									BD							
0x0014 (LCCMR)															CUR_COL_R[5:4]	
	Cursor Red – CUR_COL_R[3:0]				Cursor Green – CUR_COL_G						Cursor Blue – CUR_COL_B					
0x0018 (LPCR)	TFT	COLOR	PBSIZ		BPIX		PIXPOL	FLMPOL	LPPOL		CLKPOL	OEPOL	SCLKIDLE	END_SEL	END_BYTE_SWAP	REV_VS
	ACDSEL	Crystal Direction Toggle – ACD							SCLKSEL	SHARP	Pixel Clock Divider – PCD					

Table 33-5. LCDC Register Summary (continued)

Offset (and Name Abbreviation)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x001C (LHCR)	Horizontal Sync Width – H_WIDTH															
	Horizontal Wait 1 – H_WAIT_1								Horizontal Wait 2 – H_WAIT_2							
0x0020 (LVCR)	Vertical Sync Width – V_WIDTH															
	Vertical Wait 1 – V_WAIT_1								Vertical Wait 2 – V_WAIT_2							
0x0024 (LPOR)																
												Panning Offset – POS				
0x0028 (LSCR)	PS_RISE_DELAY								CLS_RISE_DELAY							
					REV_TOGGLE_DELAY				Grey 2				Grey 1			
0x002C (LPCCR)								CLS High Width								
	LD MSK					SCR		CC _EN	Pulse Width – PW							
0x0030 (LDCR)	BURST								DMA High Mark – HM							
									DMA Trigger Mark – TM							
0x0034 (LRMCR)																
																SELF _REF
0x0038 (LICR)																
												GW_ INT_ CON		INT SYN		INT CON
0x003C (LIER)																
									GW_ UDR_ ERR_ EN	GW_ ERR_ RES_ EN	GW_ EOF _EN	GW_ BOF _EN	UDR_ ERR_ EN	ERR_ RES_ EN	EOF_ EN	BOF_ EN
0x0040 (LISR)																
									GW_ UDR_ ERR	GW_ ERR_ RES	GW_ EOF	GW_ BOF	UDR _ERR	ERR _RES	EOF	BOF
0x0050 (LGWSAR)	Graphic Window Start Address High – GWSA H															
	Graphic Window Start Address Low – GWSA L															
0x0054 (LGWSR)							Graphic Window Width – GWW									
							Graphic Window Height – GWH									

Table 33-5. LCDC Register Summary (continued)

Offset (and Name Abbreviation)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0058 (LGWVPWR)																
							Graphic Window Virtual Page Width – GWVPW									
0x005C (LGWPOR)																
												Graphic Window Panning Offset – GWPO				
0x0060 (LGWPR)							Graphic Window X Position – GWXP									
							Graphic Window Y Position – GWYP									
0x0064 (LGWCR)	Graphic Window Alpha Value – GWAV								GWCKE	GWE	GW_RVS				Graphic Window Color Key Red – GWCKR[5:4]	
	Graphic Window Color Key Red – GWCKR[3:0]				Graphic Window Color Key Green – GWCKG						Graphic Window Color Key Blue – GWCKB					
0x0068 (LGWDCR)	GWBT								Graphic Window High Mark – GWHM							
									Graphic Window Low Mark – GWLM							
0x0080 (LAUSCR)	AUS Mode								Graphic Window Color Key Red – AGWCKR[7:0] (AUS mode only)							
	Graphic Window Color Key Green – AGWCKG[7:0] (AUS mode only)								Graphic Window Color Key Blue – AGWCKB[3:0] (AUS mode only)							
0x0084 (LAUSCCR)									Cursor Red – ACUR_COL_R [7:0]] (AUS mode only)							
	Cursor Green – ACUR_COL_G [7:0] (AUS mode only)								Cursor Blue– ACUR_COL_B [7:0] (AUS mode only)							
0x0800–0x0BFC (BGLUT)															(R[5:4])	
	First RAM Location of Background LUT (R [3:0], G [5:0], B [5:0])															
															(R[5:4])	
	Last RAM Location of Background LUT (R [3:0], G [5:0], B [5:0])															
x0C00–0x0FFC (GWLUT)																
	First RAM Location of Graphic Window LUT (R[3:0], G[5:0], B[5:0])															
															(R[5:4])	
	Last RAM location of Graphic Window LUT (R[3:0], G[5:0], B[5:0])															

33.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers.

Figure 33-22 and Table 33-6 explain conventions used in register diagrams and tables.

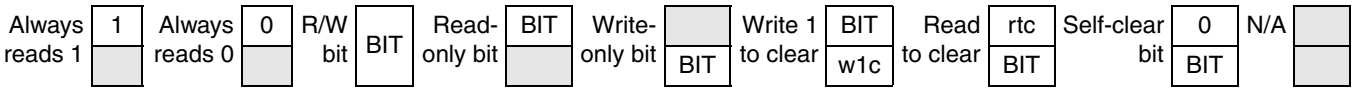


Figure 33-22. Register Field Conventions

Table 33-6. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

33.3.4 LCDC Screen Start Address Register (LSSAR)

The LSSAR register specifies the LCD screen start address (SSA). Figure 33-2 shows how the SSA determines the screen's position. Figure 33-23 shows the register, and Table 33-7 describes the register's fields.

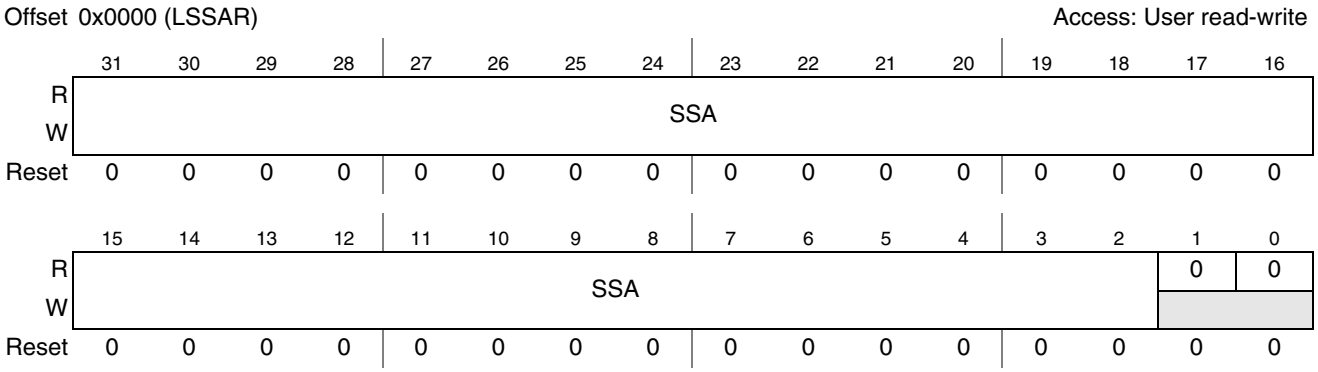


Figure 33-23. LCD Screen Start Address Register Diagram

Table 33-7. LCD Screen Start Address Register Description

Field	Description
31–2 SSA	Screen Start Address of LCD Panel. Holds pixel data for a new frame from SSA address. This field must start at a location that enables a complete picture to be stored in a 4 Mbyte memory boundary (A [21:0]). A [31:22] has a fixed value for a picture's image.
1–0	Reserved.

33.3.5 LCDC Size Register (LSR)

The LSR register defines the height and width of the LCD screen.Figure 33-24 shows the register, and Table 33-8 describes the register's fields.

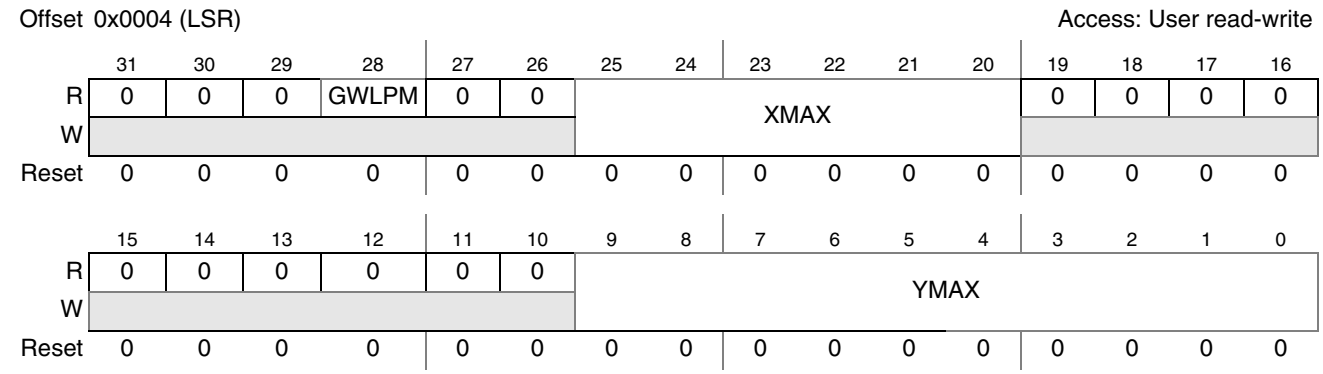


Figure 33-24. LCDC Size Register Diagram

Table 33-8. LCDC Size Register Description

Field	Description
31–26	Reserved.
28	GWLPM. Used for LCDC graphic window low-power mode. 0 LCDC normal mode. 1 For LCDC graphic window low-power mode.
25–20 XMAX	Screen width (in pixels) divided by 16. For black-and-white panels (1 bpp), XMAX [20] is ignored, forcing the x-axis of the screen size to be a multiple of 32 pixels/line.
19–10	Reserved.
9–0 YMAX	Screen Height. Specifies the height of LCD panel in terms of pixels or lines. Lines are numbered from 1 to YMAX for a total of YMAX lines.

33.3.6 LCDC Virtual Page Width Register (LVPWR)

The LVPWR register defines the virtual page width for LCD panel, as shown in Figure 33-2. Figure 33-25 shows the register, and Table 33-9 describes the register's fields.

Offset 0x0008 (LVPWR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	VFW									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-25. LCDC Virtual Page Width Register Diagram

Table 33-9. LCDC Virtual Page Width Register Description

Field	Description
31–10	Reserved.
9–0 VPW	Virtual Page Width. Defines virtual page width of LCD panel. VPW bits represent the number of 32-bit words required to hold the data for one virtual line. VPW is used in calculating the starting address representing the beginning of each displayed line.

33.3.7 LCDC Cursor Position Register (LCPR)

The LCPR register is used to determine the starting position of the cursor on the LCD panel. Figure 33-26 shows the register, and Table 33-10 describes the register's fields.

Offset 0x000C (LCPR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CC		0	OP	0	0	CXP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CYP									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-26. LCDC Cursor Position Register Diagram

Table 33-10. LCDC Cursor Position Register Description

Field	Description
31–30 CC	Cursor Control. Controls the cursor format and the type of arithmetic operations. OP = 0: 00 Transparent, cursor is disabled 01 1 for non-color displays; color defined in LCDC Color Cursor Mapping Register for color displays 10 Reversed, INV background for non-color displays; INV color defined in LCDC Color Cursor Mapping Register for color displays 11 0 for non-color displays; AND between background and cursor for color displays OP = 1 (color mode only): 00 Transparent, cursor is disabled 01 OR between background and cursor 10 XOR between background and cursor 11 AND between background and cursor
29	Reserved.
28 OP	Arithmetic Operation Control. Enables/disables arithmetic operations between the background and cursor. 0 Disable Arithmetic Operation 1 Enable Arithmetic Operation
27–26	Reserved.
25–16 CXP	Cursor X Position. Represents the cursor's horizontal starting position X in pixel count (from 0 to XMAX).
15–10	Reserved.
9–0 CYP	Cursor Y Position. Represents the cursor's vertical starting position Y in line count (from 0 to YMAX).

33.3.8 LCDC Cursor Width, Height and Blink (LCWHBR)

The LCWHBR register is used to determine the cursor's width and height, and blink characteristics. Figure 33-27 shows the register, and Table 33-11 describes the register's fields.

Offset 0x0010 (LCWHB) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BK_EN	0	0	CW					0	0	0	CH				
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	BD							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 33-27. LCDC Cursor Width, Height, and Blink Register Diagram

Table 33-11. LCDC Cursor Width, Height and Blink Rate Register Description

Field	Description
31 BK_EN	Blink Enable. Determines whether the blink enable cursor will blink or remain steady. 0 Blink is disabled 1 Blink is enabled
30–29	Reserved.
28–24 CW	Cursor Width. Specifies the width of hardware cursor in pixels. This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
23–21	Reserved.
20–16 CH	Cursor Height. Specifies the height of hardware cursor in pixels. This field can be any value between 1 and 31 (setting this field to zero disables the cursor)
15–8	Reserved.
7–0 BD	Blink Divisor. Sets the cursor blink rate. A 32 Hz clock from RTC is used to clock the 8-bit up counter. When the counter value equals BD, the cursor toggles on/off. Hence larger the BD, slower the cursor will blink. The faster cursor blinking rate is when BD is 0.

33.3.9 LCDC Color Cursor Mapping Register (LCCMR)

The LCCMR register defines the cursor color in passive or TFT color modes. If the bpp setting is less than 18 bpp, the cursor color component bits should be put in the MSBs (LSBs are ignored). For example, to create a color cursor with (R,G,B) = (0x1A, 0x26, 0x05) in 18-bpp mode, then CUR_COL_R, CUR_COL_G and CUR_COL_B should be set to 0x34, 0x26 and 0x0A respectively.

Figure 33-28 shows the register, and Table 33-12 describes the register's fields.

Offset 0x0014 (LCCMR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CUR_COL_R[5:4]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CUR_COL_R[3:0]				CUR_COL_G						CUR_COL_B					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-28. LCDC Color Cursor Mapping Register Diagram

Table 33-12. LCDC Color Cursor Mapping Register Description

Field	Description
31–18	Reserved.
17–12 CUR_COL_R	Cursor Red Field. Defines the red component of the cursor color in color mode. 000000No red … 111111Full red
11–6 CUR_COL_G	Cursor Green Field. Defines the green component of the cursor color in color mode. 000000No green … 111111Full green
5–0 CUR_COL_B	Cursor Blue Field. Defines the blue component of the cursor color in color mode. 000000No blue … 111111Full blue

33.3.10 LCDC Panel Configuration Register (LPCR)

The LPCR register defines properties of the LCD screen. Figure 33-29 shows the register, and Table 33-13 describes the register's fields.

Offset 0x0018 (LPCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TFT	COLOR	PBSIZ		BPIX			PIX POL	FLM POL	LP POL	CLK POL	OE POL	SCLK IDLE	END_ SEL	SWAP _SEL	REV_ VS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ACD SEL	ACD							SCLK SEL	SHARP	PCD					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-29. LCDC Panel Configuration Register

Table 33-13. LCDC Panel Configuration Register Description

Field	Description
31 TFT	Interfaces to TFT Display. Controls the format and timing of output control signals. Active and passive displays use different signal timing formats as described in Section 33.2.9, "Panel Interface Signals and Timing." TFT also controls the use of FRC in color mode. 0 LCD panel is a passive display 1 LCD panel is an active display: digital CRT signal format, FRC is bypassed. TFT and COLOR bit settings are summarized as follows: **TFT** / **COLOR** / **LCD Display** 0 / 0 / Monochrome 0 / 1 / CSTN 1 / 0 / — 1 / 1 / TFT
30 COLOR	Interfaces to Color Display. Activates 3 channels of FRC in passive mode to allow the use of special 2 2/3 pixels per output vector format. 0 LCD panel is a monochrome display 1 LCD panel is a color display
29–28 PBSIZ	Panel Bus Width. Specifies the panel bus width. Applicable for monochrome monitors. For passive color panels, only 8-bit panel bus width is supported. 00 1-bit width 10 4-bit width 11 8-bit width

Table 33-13. LCDC Panel Configuration Register Description (continued)

Field	Description
27–25 BPIX	Bits per Pixel. Indicates the number of bits per pixel in memory. 000 1 bpp, FRC bypassed 001 2 bpp 010 4 bpp 011 8 bpp 100 12 bpp (16-bits of memory used) 101 16 bpp 110 18 bpp (32-bits of memory used) 111 24 bpp (32-bits of memory used) **Note:** To set normal 18 bpp mode: BPIX = 110, END_SEL = 0 and SWAP_SEL = X (don't care) **Note:** To set Microsoft PAL_BGR 18 bpp mode: BPIX = 110, END_SEL = 1 and SWAP_SEL = 1
24 PIXPOL	Pixel Polarity. Sets the pixels polarity. 0 Active high 1 Active low
23 FLMPOL	First Line Marker Polarity. Sets the polarity of first line marker symbol. 0 Active high 1 Active low
22 LPPOL	Line Pulse Polarity. Sets the polarity of line pulse signal. 0 Active high 1 Active low
21 CLKPOL	LCD Shift Clock Polarity. Sets the polarity of active edge of LCD shift clock. 0 Active on negative edge of LSCLK (in TFT mode, active on positive edge of LSCLK) 1 Active on positive edge of LSCLK (in TFT mode, active on negative edge of LSCLK)
20 OEPOL	Output Enable Polarity. Sets the output enable signal polarity. 0 Active high 1 Active low
19 SCLKIDLE	LSCLK Idle Enable. Enables/disables LSCLK when VSYNC is idle in TFT mode. 0 Disable LSCLK 1 Enable LSCLK
18 END_SEL	Endian Select. Selects the image download into memory as big or little endian format. 0 Little endian 1 Big endian
17 SWAP_SEL	Swap Select. LCDC operates in big endian mode internally. Swap Select controls the swap of data before operation in little endian mode. 0 16 bpp, 12 bpp modes 1 8 bpp, 4 bpp, 2 bpp, 1 bpp mode **Note:** When SWAP_SEL = 0, byte 3 (bits 31–24), byte 2 (bits 23–16), byte 1 (bits 15–8), byte 0 (bits 7–0) data swapped to byte 1, byte 0, byte 3 and byte 2 respectively. When SWAP_SEL = 1, byte 3, byte 2, byte 1, byte 0 data swapped to byte 0, byte 1, byte 2 and byte 3 respectively.
16 REV_VS	Reverse Vertical Scan. Selects the vertical scan direction as normal or reverse image flips along the x-axis). SSA register must be changed accordingly. 0 Vertical scan in normal direction 1 Vertical scan in reverse direction
15 ACDSEL	ACD Clock Source Select. Selects the clock source used by alternative crystal direction counter. 0 Use FRM as clock source for ACD count 1 Use LP/HSYN as clock source for ACD count

Table 33-13. LCDC Panel Configuration Register Description (continued)

Field	Description
14–8 ACD	Alternate Crystal Direction. Toggles ACD signal once every 1–16 FLM cycles based on the value specified in this field. The actual number of FLM cycles between toggles is the programmed value plus one. For active mode (TFT=1), this parameter is not used.
7 SCLKSEL	LSCLK Select. Selects whether to enable or disable LSCLK in TFT mode when there is no data output. 0 Disable OE and LSCLK in TFT mode when no data output 1 Always enable LSCLK in TFT mode even if there is no data output
6 SHARP	Sharp Panel Enable. Enables/disables signals for Sharp HR-TFT 240 x 320 panels. 0 Disable Sharp signals 1 Enable Sharp signals
5–0 PCD	Pixel Clock Divider. Holds clock divider value. LCDC_CLK (PERCLK3) is divided by N (PCD plus one) to yield the pixel clock rate. Values of 1 to 63 yield N=2 to 64. Pixel clock rate is faster than LSCLK by a factor equal to the data bus width for monochrome display. For all other displays, pixel clock rate is the same as LSCLK. PCD value must be set such that LSCLK frequency is no more than one third or one fourth of HCLK frequency in TFT and CSTN modes respectively, otherwise LD will be incorrect.

33.3.11 LCDC Horizontal Configuration Register (LHCR)

The LHCR register defines the horizontal sync pulse timing. Detailed settings are given in the i.MX25 data sheet. Figure 33-30 shows the register, and Table 33-14 describes the register's fields.

Offset 0x001C (LHCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	H_WIDTH						0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_WAIT_1								H_WAIT_2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-30. LCDC Horizontal Configuration Register

Table 33-14. LCDC Horizontal Configuration Register Description

Field	Description
31–26 H_WIDTH	Horizontal Sync Pulse Width. Specifies number of SCLK periods for which HSYNC is activated. The active time is equal to (H_WIDTH + 1) of the SCLK periods.
25–16	Reserved.

Table 33-14. LCDC Horizontal Configuration Register Description (continued)

Field	Description
15–8 H_WAIT_1	Wait Between OE and HSYNC. In TFT mode, it specifies the number of SCLK periods between the end of OE signal and the beginning of HSYNC. Total delay time equals (H_WAIT_1 + 1) of SCLK periods. In CSTN mode, it specifies the number of SCLK periods between the last display data and the beginning of HSYNC signal. Total delay time equals (H_WAIT_1 + 1) of SCLK periods.
7–0 H_WAIT_2	Wait Between HSYNC and start of next line. In TFT mode, it specifies the number of SCLK periods between the end of HSYNC and the beginning of OE signal. Total delay time equals (H_WAIT_2 + 3). In CSTN mode, it specifies the number of SCLK periods between the end of HSYNC and the first display data in each line. Total delay time equals (H_WAIT_2 + 2) of SCLK periods.

33.3.12 LCDC Vertical Configuration Register (LVCR)

The LVCR register defines the vertical sync pulse timing. Detailed settings are provided in the i.MX25 data sheet.

Figure 33-31 shows the register, and Table 33-15 describes the register's fields.

Offset 0x0020 (LVCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	V_WIDTH						0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	V_WAIT_1								V_WAIT_2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-31. LCDC Vertical Configuration Register

Table 33-15. LCDC Vertical Configuration Register Description

Field	Description
31–26 V_WIDTH	Vertical Sync Pulse Width. Specifies the width, in lines, of the VSYNC pulse for active mode (TFT =1). If V_WIDTH = *N*, then the vertical sync pulse encompasses *N* HSYNC pulses. For passive mode (TFT=0) and non-color mode, see Figure 33-14.
25–16	Reserved.
15–8 V_WAIT_1	Wait Between Frames 1. Defines the delay, in lines, between the end of OE pulse and the beginning of VSYNC pulse for active mode (TFT=1). This field has no meaning in passive non-color mode. The actual delay is (V_WAIT_1). In passive color mode, this field is the delay, measured in virtual clock periods, between the last line of the frame to the beginning of next frame.
7–0 V_WAIT_2	Wait Between Frames 2. Defines the delay, in lines, between the end of VSYNC pulse and the beginning of OE pulse of the first line in active mode (TFT=1). The actual delay is V_WAIT_2) lines. Set this field to zero for passive non-color mode. The minimum value of this field is 0x01.

33.3.13 LCDC Panning Offset Register (LPOR)

The LPOR register specifies panning for the image.Figure 33-32 shows the register, and Table 33-16 describes the register's fields.

Offset 0x0024 (LPOR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	POS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-32. LCDC Panning Offset Register

Table 33-16. LCDC Panning Offset Register Description

Field	Description
31–5	Reserved.
4–0 POS	Panning Offset. Defines the number of bits that the data from memory is panned to the left before processing. POS is read by the LCDC once at the beginning of each frame. For example, setting POS = 0b10000 in 4-bpp mode causes a 16-bit shift, which pans the image 4 pixels to the left. POS settings corresponding to different bpp modes are as follows. **Bits per Pixel** / **POS** / **Effective Number of Pixels Panned on Image** 1 / *N* / *N* 2 / 2N / *N* 4 / 4*N* / *N* 8 / 8*N* / *N* 12 or 16 / 16*N* / *N* **Note:** The LSSAR register should be used instead of POS in 18 bpp mode, and/or in case the shift is more than 32 bits.

33.3.14 LCDC Sharp Configuration Register (LSCR)

The LSCR register defines various configuration parameters, including parameters that are valid only for Sharp display panels.Figure 33-33 shows the register, and Table 33-17 describes the register's fields.

For 2 bpp modes, full black (0b00) and full white (0b11) are predefined display levels and cannot be programmed. The two intermediate greyscale shading densities can be adjusted using the GREY1 and GREY2 fields in this register.

For Sharp panels, LSCR controls the relative delay timing of the CLS, REV and PS signals. Figure 33-34 provides a TFT timing diagram that shows the relationship between these signals. Detailed settings for Sharp panels are given in the i.MX25data sheet.

Offset 0x0028 (LSCR) Access: User read-write

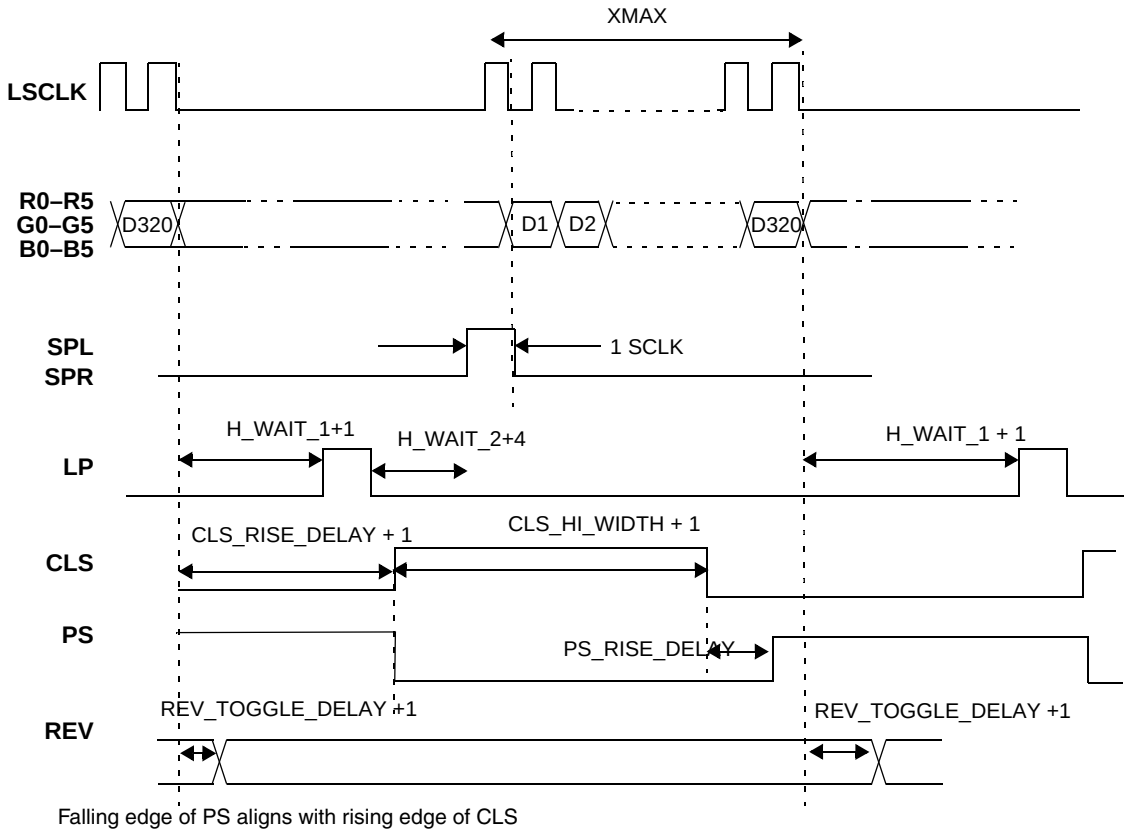
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PS_RISE_DELAY						0	0	CLS_RISE_DELAY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	REV_TOGGLE_DELAY				GREY 2				GREY 1			
W																
Reset	0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1

Figure 33-33. LCDC Sharp Configuration Register

Table 33-17. LCDC Sharp Configuration Register Description

Field	Description
31–26 PS_RISE_DELAY	PS Rise Delay. Controls the rising edge delay of PS relative to the falling edge of CLS. Total delay time equals to PS_RISE_DELAY SCLK periods. 000000 LSCLK period 000011 LSCLK period … 1111163 LSCLK periods
25–24	Reserved.
23–16 CLS_RISE_DELAY	CLS Rise Delay. Controls the rising edge delay of CLS relative to the last LD of the line. Total delay time equals to (CLS_RISE_DELAY + 1) SCLK periods. 0000000010 LSCLK periods 000000012 LSCLK period … 11111111256 LSCLK periods
15–12	Reserved.
11–8 REV_TOGGLE_DELAY	REV Toggle Delay. Controls the transition delay of REV relative to the last LD of the line. Total delay time equals to (REV_TOGGLE_DELAY + 1) SCLK periods. 00001 LSCLK period 00012 LSCLK period … 111116 LSCLK periods
7–4 GREY 2	Greyscale 2. Represents one of the two intermediate greyscale shading densities. This field is programmable to any value between 0x0 and 0xF inclusive, where 0x0 and 0xF correspond to full black and full white, respectively.
3–0 GREY 1	Greyscale 1. Represents the second intermediate greyscale shading density. This field is programmable to any value between 0x0 and 0xF inclusive, where 0x0 and 0xF correspond to full black and full white, respectively.



Falling edge of PS aligns with rising edge of CLS
The rising edge delay of PS is programmed by PS_RISE_DELAY
CLS_HI_WIDTH is equal to PWM_SCR0 • 256 + PWM_WIDTH in units of LSCLK.
SPL/SPR pulse width is fixed and aligned to the first data of the line.
REV toggles every LP period.

Figure 33-34. Horizontal Timing for Sharp Panels

33.3.15 LCDC PWM Contrast Control Register (LPCCR)

The LPCCR register is used to control the signal output at the contrast pin, which controls contrast of the LCD panel. Figure 33-35 shows the register, and Table 33-18 describes the register's fields.

Offset 0x002C (LPCCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	CLS_HI_WIDTH								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDMSK	0	0	0	0	SCR		CC_EN	PW							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-35. LCDC PWM Contrast Control Register

Table 33-18. LCDC PWM Contrast Control Register Description

Field	Description
31–25	Reserved.
24–16 CLS_HI_WIDTH	CLS High Pulse Width. Controls the pulse width of CLS in units of SCLK. The actual pulse width = CLS_HI_WIDTH +1.
15 LDMSK	LD Mask. Enables/disables the LD output to zero for Sharp TFT panel power-off sequence. 0 LD [15:0] is normal 1 LD [15:0] always equals 0
14–11	Reserved.
10–9 SCR	Source Select. Selects the input clock source for PWM counter. PWM output frequency is equal to the frequency of input clock divided by 256. 00 Line pulse 01 Pixel clock 10 LCD clock 11 Reserved
8 CC_EN	Contrast Control Enable. Enables/disables the contrast control function. 0 Contrast control is off 1 Contrast control is on
7–0 PW	Pulse Width. Controls the pulse width of the built-in PWM, which controls the contrast of LCD screen.

33.3.16 LCDC DMA Control Register (LDCR)

The LCDR register has a 128 × 32-bit line buffer that stores DMA data from system memory. LDCR controls the DMA burst length, and the triggering of DMA bursts in terms of number of data bytes left in the pixel buffer.

Figure 33-36 shows the register, and Table 33-19 describes the register's fields.

Offset 0x0030 (LDCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BURST	0	0	0	0	0	0	0	0	HM						
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	TM						
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 33-36. LCDC DMA Control Register

Table 33-19. LCDC DMA Control Register Description

Field	Description
31 BURST	Burst Length Mode. Determines whether the burst length is fixed or dynamic. 0 Dynamic burst length (recommended) 1 Fixed burst length (reset value) **Note:** The 0 setting (dynamic burst length) is recommended. The reset value is 1.
30–23	Reserved.
22–16 HM	DMA Dynamic High-Level Mark / Fixed Burst Length • In dynamic burst length mode, after the DMA request is made the data is loaded and pixel buffer, which continues to fill until the number of empty words left in the DMA FIFO is equal to HM – 2. The minimum HM setting in the dynamic case is 3. • In fixed burst length mode, the HM setting determines the burst length (in 32-bit words) of each request. The reset value of HM is 4.
15–7	Reserved.
6–0 TM	DMA Trigger Mark. Sets the low-level mark in the pixel buffer to trigger a DMA request. • In fixed burst length mode, when the number of non-empty words in the FIFO falls below the TM value then a DMA request is issued, with DMA burst length equal to HM. • In dynamic burst length mode, when the number of non-empty words in the FIFO falls below the TM value then DMA requests are issued until the number of empty words in the FIFO HM - 2. The reset value of TM is 96.

NOTE

Dynamic burst length mode (BURST = 0) is recommended for DMA, using the reset values for TM and HM. The same recommendation holds for the LCDC Graphic Window DMA Control Register (LGWDCR).

33.3.17 LCDC Refresh Mode Control Register (LRMCR)

The LRMCR register is used to control the refresh characteristics.Figure 33-37 shows the register, and Table 33-20 describes the register's fields.

Offset 0x0034 (LRMCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SELF_ REF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-37. LCDC Refresh Mode Control Register

Table 33-20. LCDC Refresh Mode Control Register Description

Field	Description
31–1	Reserved.
0 SELF_REF	Self-Refresh. Enables/disables self-refresh mode. 0 Disable self-refresh 1 Enable self-refresh

NOTE

- On entering self-refresh mode, LSCLK and LD [17:0] signals stay low. HYSN and VSYN operate normally.
- After entering self-refresh mode, all registers except SSA, BGLUT, and GWLUT must be reconfigured to avoid LCDC malfunction.
- SSA must always match the address range of the RAM selected. The LCDC must first be disabled before switching between different types of RAM.

33.3.18 LCDC Interrupt Configuration Register (LICR)

The LICR register is used to configure the interrupt conditions. Figure 33-38 shows the register, and Table 33-21 describes the register's fields.

Offset 0x0038 (LICR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	GW_INT_CON	0	INTSYN	0	INTCON
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-38. LCDC Interrupt Configuration Register

Table 33-21. LCDC Interrupt Configuration Register Description

Field	Description
31–5	Reserved.
4 GW_INT_CON	Graphic Window Interrupt Condition. Determines if an interrupt condition is set at the beginning or end of graphic window condition. 0 Interrupt flag is set when end of graphic window is reached 1 Interrupt flag is set when beginning of graphic window is reached
3	Reserved.

Table 33-21. LCDC Interrupt Configuration Register Description (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>2
INTSYN</td><td>Interrupt Source. Determines if an interrupt flag is set during last/first data of frame loading or on last/first data of frame output to the LCD panel.
Note: There is a latency between loading last/first data of the frame to output to the LCD panel.
0 Interrupt flag is set on loading the last/first data of frame from memory
1 Interrupt flag is set on output of the last/first data of frame to LCD panel
The INTSYN and INTCON settings are as follows:
<table>
<tr><th>INTSYN</th><th>INTCON</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Interrupt flag is set on loading last data of frame from memory.</td></tr>
<tr><td>0</td><td>1</td><td>Interrupt flag is set on loading first data of frame from memory.</td></tr>
<tr><td>1</td><td>0</td><td>Interrupt flag is set on output of last data of frame to LCD panel.</td></tr>
<tr><td>1</td><td>1</td><td>Interrupt flag is set on output of first data of frame to LCD panel.</td></tr>
</table></td></tr>
<tr><td>1</td><td>Reserved.</td></tr>
<tr><td>0
INTCON</td><td>Interrupt Condition. Determines if an interrupt condition is set at the beginning or end of frame condition.
0 Interrupt flag is set when the end of frame (EOF) is reached
1 Interrupt flag is set when the beginning of frame (BOF) is reached
The INTCON settings are described also in the INTSYN bit description.</td></tr>
</table>

33.3.19 LCDC Interrupt Enable Register (LIER)

The LIER register is used to enable the LCDC interrupts generated to ARM926EJ-S interrupt controller (AITC). When an interrupt is masked by clearing its corresponding LIER bit, LCDC does not generate the interrupt request to AITC. However, its status can still be observed in the interrupt status register.

Figure 33-39 shows the register, and Table 33-22 describes the register's fields.

Offset 0x003C (LIER) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	GW_UDR_ERR_EN	GW_ERR_RES_EN	GW_EOF_EN	GW_BOF_EN	UDR_ERR_EN	ERR_RES_EN	EOF_EN	BOF_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-39. LCDC Interrupt Enable Register

Table 33-22. LCDC Interrupt Enable Register Description

Field	Description
31–8	Reserved.
7 GW_UDR_ERR_EN	Graphic Window Underrun Error Interrupt Enable. Used to enable or mask the graphic window underrun error interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
6 GW_ERR_RES_EN	Graphic Window Error Response Interrupt Enable. Used to enable or mask the graphic window error response interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
5 GW_EOF_EN	Graphic Window End of Frame Interrupt Enable. Used to enable or mask the graphic window end of frame interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
4 GW_BOF_EN	Graphic Window Beginning of Frame Interrupt Enable. Used to enable or mask the graphic window beginning of frame interrupt to AITC.
3 UDR_ERR_EN	Underrun Error Interrupt Enable. Used to enable or mask the background plane underrun error interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
2 ERR_RES_EN	Error Response Interrupt Enable. Used to enable or mask the background plane error response interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
1 EOF_EN	End of Frame Interrupt Enable. Used to enable or mask the background plane end of frame interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled
0 BOF_EN	Beginning of Frame Interrupt Enable. Used to enable or mask the background plane beginning of frame interrupt to AITC. 0 Interrupt is masked 1 Interrupt is enabled

33.3.20 LCDC Interrupt Status Register (LISR)

The read-only LISR register indicates whether an interrupt has occurred or not. The status bit is set whenever the interrupt condition is met. If any bit in this register is set and the corresponding bit in LIER is set, the LCDC interrupt pin is asserted to AITC. The status bit is cleared by reading the register.

Figure 33-40 shows the register, and Table 33-23 describes the register's fields.

Offset 0x0040 (LISR) Access: User read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	GW_UDR_ERR	GW_ERR_RES	GW_EOF	GW_BOF	UDR_ERR	ERR_RES	EOF	BOF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-40. LCDC Interrupt Status Register

Table 33-23. LCDC Interrupt Status Register Description

Field	Description
31–8	Reserved.
7 GW_UDR_ERR	Graphic Window Underrun Error. Indicates whether the LCDC FIFO in graphic window plan has hit an underrun condition. This is when the data output rate is faster than the data input rate to the FIFO of the graphic window plan. Underrun can cause erroneous data output to LD. LD data output rate must be adjusted to prevent this error. 0 Interrupt has not occurred 1 Interrupt has occurred
6 GW_ERR_RES	Graphic Window Error Response. Indicates whether the LCDC has issued a read data request in graphic window and has received a response from the memory controller not equal to 'OK.' It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred
5 GW_EOF	Graphic Window End of Frame. Indicates whether the end of graphic window has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred
4 GW_BOF	Graphic Window Beginning of Frame. Indicates whether the beginning of graphic window has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred
3 UDR_ERR	Underrun error. Indicates whether the LCDC FIFO has hit an underrun condition. This is when the data output rate is faster than the data input rate to the FIFO. Underrun can cause erroneous data output to LD. LD data output rate must be adjusted to prevent this error. 0 Interrupt has not occurred 1 Interrupt has occurred
2 ERR_RES	Error response. Indicates whether the LCDC has issued a read data request and has received a response from the memory controller not equal to 'OK'. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred

Table 33-23. LCDC Interrupt Status Register Description (continued)

Field	Description
1 EOF	End of frame. Indicates whether the end of frame has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred
0 BOF	Beginning of frame. Indicates whether the beginning of frame has reached. It is cleared by reading the status register, at power on reset, or when the LCDC is disabled. 0 Interrupt has not occurred 1 Interrupt has occurred

33.3.21 LCDC Graphic Window Start Address Register (LGWSAR)

The LGWSAR register defines the starting address of the graphic window image. Figure 33-3 shows the positioning of the graphic window on the LCD screen.

Figure 33-41 shows the register, and Table 33-24 describes the register's fields.

Offset 0x0050 (LGWSAR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GWSA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GWSA														0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-41. LCDC Graphic Window Start Address Register

Table 33-24. LCDC Graphic Window Start Address Register Description

Field	Description
31–2 GWSA	Graphic window start address on LCD screen. This field must start at a location that enables a complete graphic window picture to be stored in a 4 Mbyte memory boundary (A[21:0]). A[31:22] is fixed for the graphic window picture's image.
1–0	Reserved.

33.3.22 LCDC Graphic Window Size Register (LGWSR)

The LGWSR register defines the height and width of the graphic window on the LCD screen.

Figure 33-42 shows the register, and Table 33-25 describes the register's fields.

Offset 0x0054 (LGWSR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	GWW						0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	GWH									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-42. LCDC Graphic Window Size Register

Table 33-25. LCDC Graphic Window Size Register Description

Field	Description
31–26	Reserved.
25–20 GWW	Graphic window width (in pixels) divided by 16. For black-and-white panels (1 bpp), GWW [0] is ignored, which forces the window width to be a multiple of 32 pixels. **Note:** GWW cannot be set to 0.
19–10	Reserved.
9–0 GWH	Graphic window height (in lines). The lines are numbered from 1 to GWH for a total of GWH lines. **Note:** GWH cannot be set to 0.

33.3.23 LCDC Graphic Window Virtual Page Width Register (LGWVPWR)

The LGWVPWR register defines the virtual page width for the graphic window picture on the LCD screen.

Figure 33-43 shows the register, and Table 33-26 describes the register's fields.

Offset 0x0058 (LGWVPWR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	GWVPW									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-43. LCDC Graphic Window Virtual Page Width Register

Table 33-26. LCDC Graphic Window Virtual Page Width Register Description

Field	Description
31–10	Reserved.
9–0 GWVPW	Graphic window virtual page width. Represents the number of 32-bit words required to hold the data for one virtual line. GWVPW is used in calculating the starting address representing the beginning of each line of the graphic window picture.

33.3.24 LCDC Graphic Window Panning Offset Register (LGWPOR)

The LGWPOR register sets up panning for the image.

Figure 33-44 shows the register, and Table 33-27 describes the register's fields.

Offset 0x005C (LGWPOR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	GWPO				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-44. LCDC Graphic Window Panning Offset Register

Table 33-27. LCDC Graphic Window Panning Offset Register Description

Filed	Description
31–5	Reserved.
4–0 GWPO	Graphic window panning offset. Defines the number of bits that the data from memory is panned to the left before processing. POS is read by the LCDC once at the beginning of each frame. For example, setting GWPO = 16 in 4 bpp causes a 16-bit shift, which pans the image 4 pixels to the left. GWPO settings corresponding to different bpp modes are as follows: **Bits per Pixel** / **GWPO Setting** / **Effective Number of Pixels Panned on Image** 1 / *N* / *N* 2 / 2*N* / *N* 4 / 4*N* / *N* 8 / 8*N* / *N* 12/16 / 16*N* / *N* **Note:** The LGWSAR register should be used instead of POS in 18 bpp mode and/or in case the shift is more than 32 bits.

33.3.25 LCDC Graphic Window Position Register (LGWPR)

LGWPR is used to determine the starting position of the graphic window on the LCD panel.

Figure 33-45 shows the register, and Table 33-28 describes the register's fields.

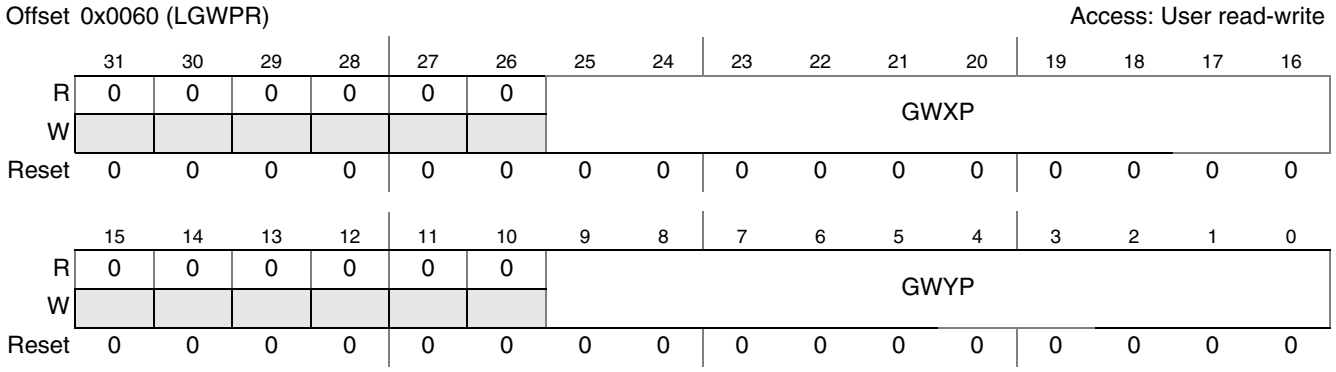


Figure 33-45. LCDC Graphic Window Position Register

Table 33-28. LCDC Graphic Window Position Register Description

Field	Description
31–26	Reserved.
25–16 GWXP	Graphic Window X Position. Represents the graphic window's horizontal starting position in pixel count (from 0 to XMAX)
15–10	Reserved.
9–0 GWYP	Graphic window Y position. Represents the graphic window's vertical starting position in lines (from 0 to YMAX).

33.3.26 LCDC Graphic Window Control Register (LGWCR)

The LGWCR register defines various characteristics of the graphic window.

Figure 33-46 shows the register, and Table 33-29 describes the register's fields.

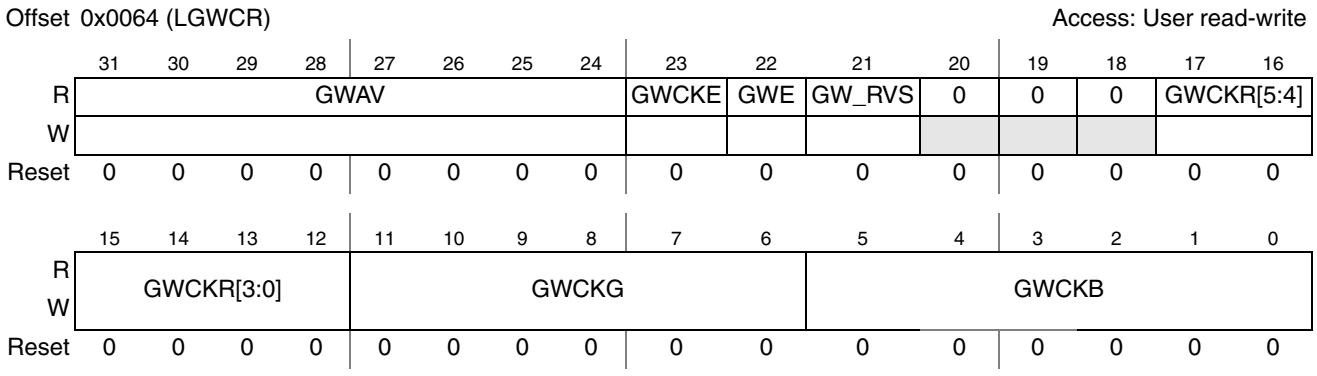


Figure 33-46. LCDC Graphic Window Control Register

Table 33-29. LCDC Graphic Window Control Register Description

Field	Description
31–24 GWAV	Graphic window alpha value. Defines the graphic window alpha value used for alpha blending between graphic window and background plane 0b00000000 Graphic window totally transparent—not displayed on LCD screen ... 0b11111111 Graphic window totally opaque—completely visible on LCD screen
23 GWCKE	Graphic window color keying enable. Enable/disable graphic window color keying. 0 Disable color keying of graphic window 1 Enable color keying of graphic window
22 GWE	Graphic window enable. Enable/disable graphic window displayed on screen. 0 Disable graphic window on screen 1 Enable graphic window on screen
21 GW_RVS	Graphic window reverse vertical scan. Selects the graphic window vertical scan direction as normal or reverse (graphic window image flips along the x-axis). LGWSAR must be changed accordingly. 0 Vertical scan in normal direction 1 Vertical scan in reverse direction
20–18	Reserved.
17–12 GWCKR	Graphic window color keying red component 000000No red … 111111Full red
11–6 GWCKG	Graphic window color keying green component 000000No green … 111111Full green
5–0 GWCKB	Graphic window color keying blue component 000000No blue … 111111Full blue

33.3.27 LCDC Graphic Window DMA Control Register (LGWDCR)

The LCDC has a 128×32-bit line buffer that stores graphic window data from system memory. LGWDCR controls the DMA burst length and when to trigger a DMA burst in terms of the number of data bytes left in the pixel buffer.

Figure 33-47 shows the register, and Table 33-30 describes the register's fields.

Offset 0x0068 (LGWDCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GWBT	0	0	0	0	0	0	0	0	GWHM						
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	GWTM						
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 33-47. LCDC Graphic Window DMA Control Register

Table 33-30. LCDC Graphic Window DMA Control Register Description

Field	Description
31 GWBT	Graphic window DMA burst mode. Determines whether burst length is fixed or dynamic in the graphic window plane. 0 Burst length is dynamic 1 Burst length is fixed
30–23	Reserved.
22–16 GWHM	Graphic window dynamic DMA high-level mark / Fixed burst length. • For dynamic burst length mode, once a DMA request is made, the data is loaded and the graphic window FIFO continues to be filled until the number of empty words left in the graphic window FIFO is equal to GWHM - 2. The minimum GWHM setting in dynamic burst mode is 3. • For fixed burst length mode, GWHM is the burst length (in 32-bit words) of each request.
15–7	Reserved.
6–0 GWTM	Graphic window DMA trigger mark. A DMA request is triggered f the number of non-empty words left in the graphic window FIFO hits or falls below the GWTM value.

NOTE

Dynamic burst length mode is recommended for graphic window DMA, using the reset values for TM and HM.

33.3.28 LCDC AUS Mode Control Register (LAUSCR)

The LAUSCR register determines LCDC characteristics in AUS mode.

Offset 0x0080 (LAUSCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	AUS Mode	0	0	0	0	0	0	0	Graphic Window Color Key Red – AGWCKR[7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Graphic Window Color Key Green – AGWCKG[7:0] (AUS mode only)								Graphic Window Color Key Blue – AGWCKB[7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-48. LCDC AUS Mode Control Register

Table 33-31. LCDC AUS Mode Control Register Description

Field	Description
31 AUS Mode	AUS Mode Control. Determines whether LCDC enters AUS mode or not. When BPIX = 101, supports 16-bpp AUO panels. When BPIX = 110, supports 24-bpp AUO panel. 0 Normal Mode 1 AUS Mode
30–24	Reserved.
23–16 AGWCKR	AUS graphic window color keying red component. Defines the red component of graphic window color keying. (AUS mode only)
15–8 AGWCKG	AUS graphic window color keying green component. Defines the green component of graphic window color keying. (AUS mode only)
7–0 AGWCKB	AUS graphic window color keying blue component. Defines the blue component of graphic window color keying.(AUS mode only)

33.3.29 LCDC AUS Mode Cursor Control Register (LAUSCCR)

The LAUSCCR register determines the color map of cursor in AUS mode.

Offset 0x0084 (LAUSCCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	Cursor Red – ACUR_COL_R [7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Cursor Green – ACUR_COL_G [7:0] (AUS mode only)								Cursor Blue – ACUR_COL_B [7:0] (AUS mode only)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 33-49. LCDC AUS Mode Cursor Control Register

Table 33-32. LCDC AUS Mode Cursor Control Register Description

Field	Description
31–24	Reserved.
23–16 ACUR_COL_R	AUS Cursor Red Field. Defines the red component of cursor color in color mode.(AUS mode only)
15–8 ACUR_COL_G	AUS Cursor Green Field. Defines the green component of cursor color in color mode.(AUS mode only)
7–0 ACUR_COL_B	AUS Cursor Blue Field. Defines the blue component of cursor color in color mode.(AUS mode only)

33.3.30 Mapping RAMs: Background Lookup Table (BGLUT) and Graphic Window Lookup Table (GWLUT)

There are 2 separate mapping RAMs in LCDC: the background lookup table (BGLUT) and the graphic window lookup table (GWLUT). These mapping RAMs are used for mapping 4-bit codes for greyscale to 16 grey shades, and for mapping 4-bit color and 8-bit color to 16 colors and 256 colors, respectively, out of a palette of 4096 (for passive panels) or 256K (active panels).

BGLUT is used for the background plane: in memory it occupies the offset range 0x0800–0x0BFC (measured from the module base address). GWLUT is for the graphic window, and occupies the offset range 0x0C00–0x0FFC (measured from the module base address).

Each mapping RAM contains 256 entries, where each RAM entry uses 4 bytes of address space. RAM is accessed with word transactions only, and the address must be word aligned. Unimplemented bits are read as 0. Byte or half-word accesses to the RAM corrupt the contents. All read and write data use the least significant 4, 12 or 18 bits, depending on the operating mode (greyscale, passive color, and active color, respectively).

In 4-bpp mode only the first sixteen RAM entries are used, while 8-bpp mode uses all 256 RAM entries. The color RAM is not initialized at reset. Only the following settings use the mapping RAMs:

- 4-bpp greyscale mode
- 4- and 8-bpp passive matrix color modes
- 4- and 8-bpp active matrix color modes

33.3.30.1 4-bpp Greyscale Mode

In 4-bpp greyscale mode, a 4-bit code represents a greyscale level. The first 16 mapping RAM entries must be written to define the codes for all 16 combinations.

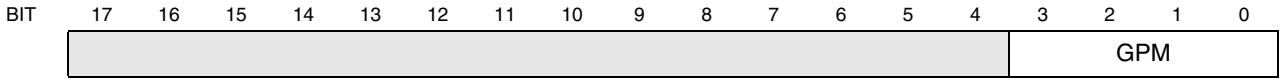


Figure 33-50. Mapping RAM Entry in 4-bpp Greyscale Mode

Table 33-33. 4-bpp Greyscale Mode

Field	Description
17–4	Reserved.
3–0 GPM	Grey Palette Map. Represents the greyscale level for a given pixel code.

33.3.30.2 4-bpp Passive Matrix Color Mode

In 4-bpp passive matrix color mode, a 4-bit code represents a 12-bit color. Because just 4-bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 4096. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

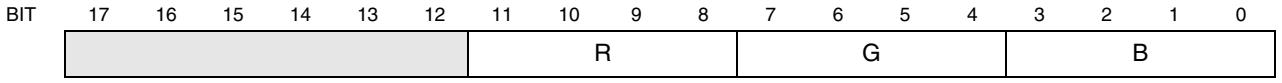


Figure 33-51. Mapping RAM Entry in 4-bpp Passive Matrix Color Mode

Table 33-34. 4-bpp Passive Matrix Color Mode

Field	Description
17–12	Reserved.
11–8 R	Red Level (color display). Represents the red component level in the color.
7–4 G	Green Level (color display). Represents the green component level in the color.
3–0 B	Blue Level (color display). Represents the blue component level in the color.

33.3.30.3 8-bpp Passive Matrix Color Mode

In 8-bpp passive matrix color mode, an 8-bit code represents a 12-bit color. Because 8 bits are used to encode the color, a maximum of 256 colors can be selected out of a palette of 4096. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

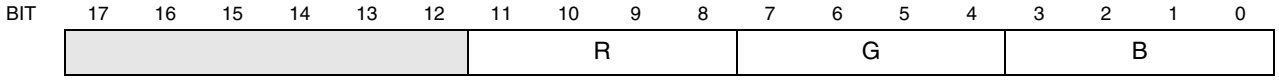


Figure 33-52. Mapping RAM Entry in 8-bpp Passive Matrix Color Mode

Table 33-35. Eight Bits per Pixel Passive Matrix Color Mode

Field	Description
17–12	Reserved.
11–8 R	Red Level (color display). Represents the red component level in the color.
7–4 G	Green Level (color display). Represents the green component level in the color.
3–0 B	Blue Level (color display). Represents the blue component level in the color.

33.3.30.4 4-bpp Active Matrix Color Mode

In 4-bpp active matrix color mode, a 4-bit code represents a 18-bit color. Since 4 bits are used to encode the color, a maximum of 16 colors can be selected out of a palette of 256K. The first 16 mapping RAM entries must be written to define the codes for the 16 available combinations.

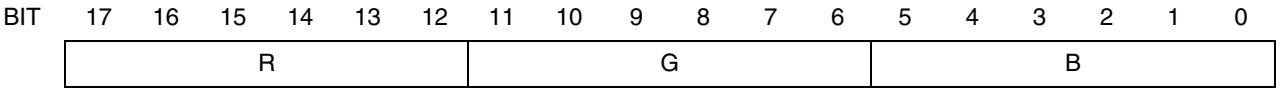


Figure 33-53. Mapping RAM Entry in 4-bpp Active Matrix Color Mode

Table 33-36. Four Bits per Pixel Active Matrix Color Mode

Field	Description
17–12 R	Red Level (color display). Represents the red component level in the color.
11–6 G	Green Level (color display). Represents the green component level in the color.
5–0 B	Blue Level (color display). Represents the blue component level in the color.

33.3.30.5 8-bpp Active Matrix Color Mode

In 8-bpp active matrix color mode, an 8-bit code represents an 18-bit color, so a maximum of 256 colors can be selected out of a palette of 256K. All 256 mapping RAM entries must be written to define the codes for the 256 available combinations.

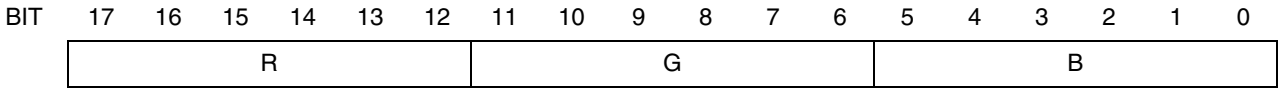


Figure 33-54. Mapping RAM Entry in 8-bpp Active Matrix Color Mode

Table 33-37. Eight Bits per Pixel Active Matrix Color Mode

Field	Description
17–12 R	Red Level (color display). Represents the red component level in the color.
11–6 G	Green Level (color display). Represents the green component level in the color.
5–0 B	Blue Level (color display). Represents the blue component level in the color.

Chapter 34
ARM9 Platform Multi-Layer AHB Crossbar Switch (MAX)

34.1 Introduction

34.1.1 Overview

This section provides an overview of the MAX The purpose of the MAX is to concurrently support up to five simultaneous connections between master ports and slave ports. The MAX supports a 32-bit address bus width and a 32-bit data bus width at all master and slave ports.

NOTE

The ARM9 platform implements a 5 master by 5 slave configuration.

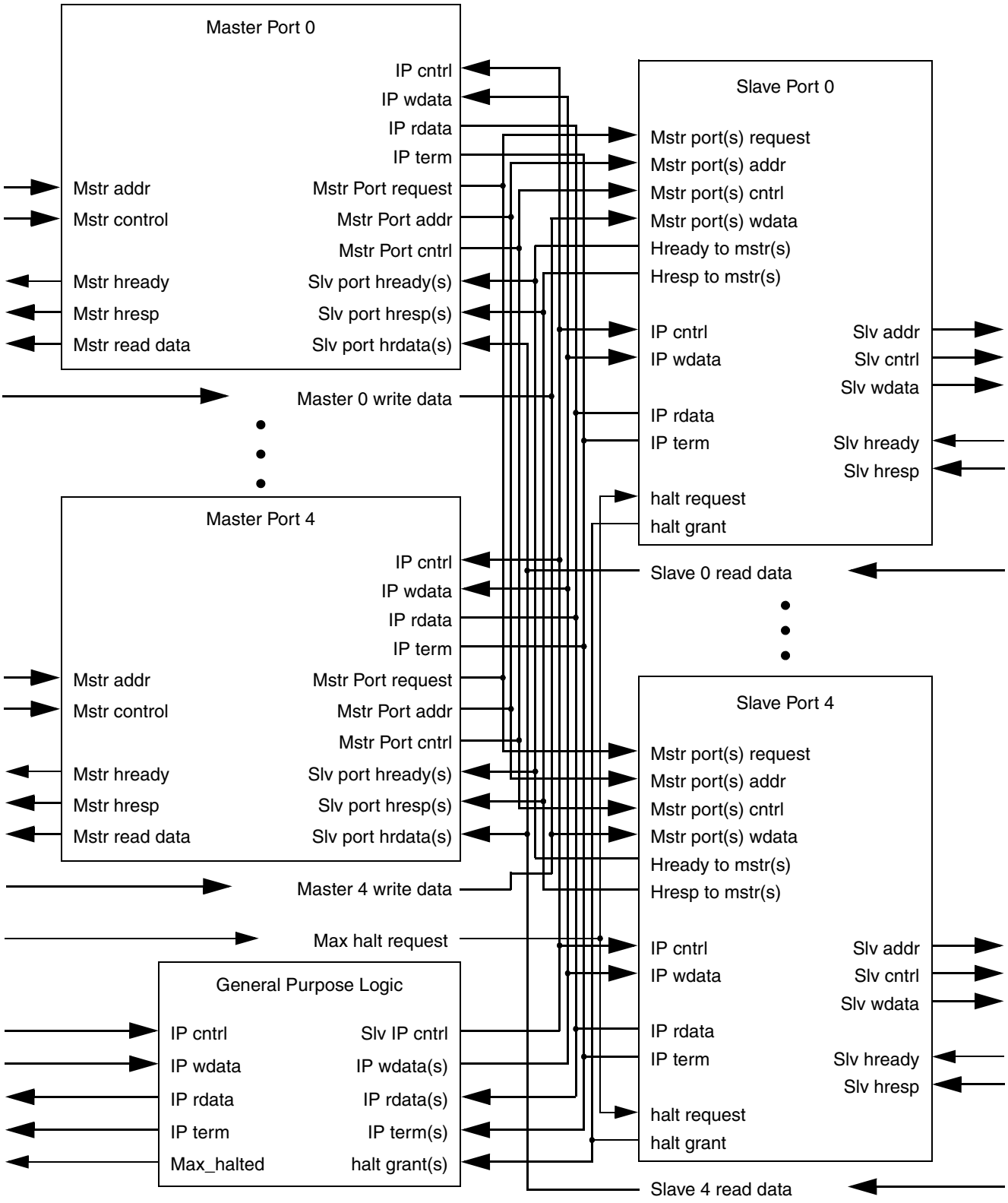


Figure 34-1. MAX Block Diagram

34.1.2 Features

The MAX has the ability to gain control of all the slave ports and prevent any masters from making accesses to the slave ports. This feature is useful when the user wishes to turn off the clocks to the system and needs to ensure that no bus activity is interrupted.

The MAX can put each slave port into a low power park mode so that the slave port does not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port.

Each slave port can also support multiple master priority schemes. Each slave port has a hardware input which selects the master priority scheme so the user can dynamically change master priority levels on a slave port by slave port basis.

The MAX allows for concurrent transactions to occur from any master port to any slave port. It is possible for four master ports and all slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions.

34.1.3 Limitations

The MAX routes bus transactions initiated on the master ports to the appropriate slave ports. There is no provision included to route transactions initiated on the slave ports to other slave ports or to master ports. Simply put, the slave ports do not support the bus request/bus grant protocol, the MAX assumes it is the sole master of each slave port.

Since the MAX does not support the bus request/bus grant protocol, if multiple masters are to be connected to a single master port, an external arbiter needs to be used.

Each master and slave port is fully AHB-Lite + AMBA V6 extensions-compatible. The ports are not fully AHB-compatible because the MAX does not support SPLITs or RETRYs.

34.1.4 General Operation

When a master makes an access to the MAX the access is immediately taken by the MAX. If the targeted slave port of the access is available then the access is immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the MAX. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted (**hready** held negated) until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding peripheral's access time.

Since the MAX appears to be just another slave to the master device, the master device has no knowledge of whether or not it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting it will simply be wait stated.

A master will be given control of the targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to one slave port that has a long response time, has a

pending access to a different slave port, and a lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a locked or fixed length burst transfer it retains control of the slave port until that transfer is completed. Based on the AULB bit in the MGPCR (Master General Purpose Control Register) the master either retains control of the slave port when doing undefined length incrementing burst transfers, or loses the bus to a higher priority master.

The MAX terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave busses). Additionally, when no master is requesting access to a slave port, the MAX drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port. When the MAX is controlling the slave bus (i.e. during low power park or halt mode) the **hmaster** field indicates 4'b0000.

When a slave bus is being IDLEd by the MAX it can park the slave port on the master port indicated by the PARK bits in the SGPCR (Slave General Purpose Control Register). This can be done in an attempt to save the initial clock of arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

34.2 MAX Registers

This section provides information on MAX registers.

34.2.1 Register Summary

There are four registers that reside in each slave port of the MAX and one register that resides in each master port of the MAX. These registers are IP bus-compatible registers. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

The registers are fully decoded and an error response is returned if an unimplemented location is accessed within the MAX.

The slave registers also feature a bit, which when written with a 1, prevents the registers from being written to again. The registers are still readable, but future write attempts have no effect on the registers and are terminated with an error response.

The memory map for the MAX program-visible registers is shown in Table 34-1. Table 34-2 shows the MAX register summary.

Table 34-1. MAX Register Configuration Summary

MAX Base Offset	Register	Use
0x000	MPR0	Master Priority Register for Slave port 0
0x010	SGPCR0	General Purpose Control Register for Slave port 0

Table 34-1. MAX Register Configuration Summary (continued)

MAX Base Offset	Register	Use
0x100	MPR1	Master Priority Register for Slave port 1
0x110	SGPCR1	General Purpose Control Register for Slave port 1
0x200	MPR2	Master Priority Register for Slave port 2
0x210	SGPCR2	General Purpose Control Register for Slave port 2
0x300	MPR3	Master Priority Register for Slave port 3
0x310	SGPCR3	General Purpose Control Register for Slave port 3
0x400	MPR4	Master Priority Register for Slave port 4
0x410	SGPCR4	General Purpose Control Register for Slave port 4
0x800	MGPCR0	General Purpose Control Register for Master port 0
0x900	MGPCR1	General Purpose Control Register for Master port 1
0xA00	MGPCR2	General Purpose Control Register for Master port 2
0xB00	MGPCR3	General Purpose Control Register for Master port 3
0xC00	MGPCR4	General Purpose Control Register for Master port 4

KEY:

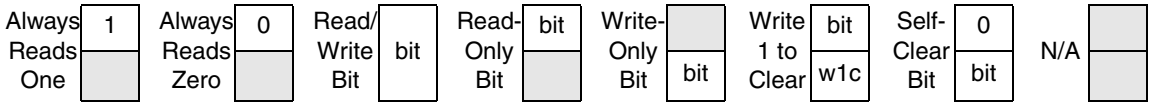


Table 34-2. MAX Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPRn ($BASE + 0x000 + n*0x100)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	MSTR_4		
	W																
	R	0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
	W																
SGPCRn ($BASE + 0x010 + n*0x100)	R	RO		HLP	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ARB		0	0	PCTL		0	PARK		
	W																
MGPCRm ($BASE + 0x800 + m*0x100	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
	W																

Note: for n = 0 to 4
for m = 0 to 4

34.2.2 MAX Register Descriptions

The following paragraphs provide detailed descriptions of the various MAX registers.

Table 34-3 provides a key to the terms found in MAX registers.

Table 34-3. Register Terms

Term	Description
Grey bit	Unimplemented bit; always reads as zero; writing has no effect.
Access	
S	Supervisor mode only
-	Supervisor or user mode
Type	
r	Read only. Writing to this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change a bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than reset.
w1c	A status bit that can be read and cleared by writing a logic 1.
slfclr	Self-clearing bit. Writing a 1 has some effect on module, but it always reads as a 0.
Reset	
0	Resets to a logic 0.
1	Resets to a logic 1.
u	Unaffected by reset.
?	Reset state is unknown.

34.2.2.1 Master Priority Register

The master priority register (MPR), shown in Figure 34-2, sets the priority of each master port on a per slave port basis and resides in each slave port.

Address $BASE + 0x000 + n*100 Access: Read/Write
Wait State: 1

	31												19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MSTR_4			0	MSTR_3			0	MSTR_2			0	MSTR_1			0	MSTR_0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Figure 34-2. Master Priority Register n[1]

[1] For n = 0–4

Table 34-4. Master Priority Register Descriptions

Field	Description
31–19	Reserved
18–16 MSTR_4	Master 4 Priority These bits set the arbitration priority for master port 4 on the associated slave port. These bits are initialized by hardware reset. The reset value is 100 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
15	Reserved
14–11 MSTR_3	Master 3 Priority These bits set the arbitration priority for master port 3 on the associated slave port. These bits are initialized by hardware reset. The reset value is 011 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
11	Reserved
10–8 MSTR_2	Master 2 Priority These bits set the arbitration priority for master port 2 on the associated slave port. These bits are initialized by hardware reset. The reset value is 010 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
7	Reserved
6–4 MSTR_1	Master 1 Priority These bits set the arbitration priority for master port 1 on the associated slave port. These bits are initialized by hardware reset. The reset value is 001 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.
3	Reserved
2–0 MSTR_0	Master 0 Priority These bits set the arbitration priority for master port 0 on the associated slave port. These bits are initialized by hardware reset. The reset value is 000 000 This master has the highest priority when accessing the slave port. 111 This master has the lowest priority when accessing the slave port.

The master priority register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the Slave General Purpose Control Register the Master Priority Register can only be read from, attempts to write to it have no effect on the MPR and result in an error response.

Additionally, no two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level result in an error response and the MPR is not updated.

34.2.2.2 Slave General Purpose Control Register

The slave general purpose control register (SGPCR), shown in Figure 34-3, controls several features of each slave port.

The read only (RO) bit prevents any registers associated with this slave port from being written to once set. This bit can be written with 0 as many times as the user desires, but after it is written to a 1 only a reset condition allows it to be written again.

The halt low priority (HLP) bit sets the priority of the **max_halt_request** input to the lowest possible priority for initial arbitration of the slave ports. By default it is the highest priority. Setting this bit does not prevent the **max_halt_request** from attaining highest priority once it has control of the slave ports.

The PCTL bits determine how the slave port parks when no master is actively making a request. The available options are to park on the master identified by the PARK bits, park on the last master to use the slave port, or go into a low power park mode, which forces all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if a the slave port is not saturated; however, it forces an extra clock of latency whenever any master tries to access it when it is not in use because it is not parked on any master.

The PARK bits determine which master the slave parks on when no master is making an active request and the **max_halt_request** input is negated. Use caution to only select master ports that are actually present in the design. If the user programs the PARK bits to a master not present in the current design implementation undefined behavior can result.

Address $BASE + 0x010 + n*100 Access: Read/Write
Wait State: 1

	31	30	29																	10	9	8	7	6	5	4	3	2	0
R	RO[1]	HLP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ARB		—		PCTL		—	PARK
W																													
Reset	All zeros																												

Figure 34-3. Slave General Purpose Control Register n[2]

[1] Once the RO bit is written to a 1, only hardware reset returns it to a 0.
[2] For n = 0–4

Table 34-5. Slave General Purpose Control Register Descriptions

Bits	Description
31 RO	Read Only This bit is used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset. This bit is initialized by hardware reset. 0 All this slave port's registers can be written. 1 All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response).
30 HLP	Halt Low Priority This bit is used to set the initial arbitration priority of the max_halt_request input. This bit is initialized by hardware reset. 0 The max_halt_request input has the highest priority for arbitration on this slave port 1 The max_halt_request input has the lowest initial priority for arbitration on this slave port.
29–10	Reserved
9–8 ARB	Arbitration Mode These bits are used to select the arbitration policy for the slave port. These bits are initialized by hardware reset. 00 Fixed Priority. 01 Round Robin (rotating) Priority. 10 Reserved 11 Reserved
7–6	Reserved
5–4 PCTL	Parking Control These bits determine the parking control used by this slave port. These bits are initialized by hardware reset. 00 When no master is making a request the arbiter parks the slave port on the master port defined by the PARK bit field. 01 When no master is making a request the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master is making a request the arbiter parks the slave port on no master and drives all outputs to a constant safe state. 11 Reserved
3	Reserved
2–0 PARK	PARK These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00. These bits are initialized by hardware reset. 000 Park on Master Port 0 001 Park on Master Port 1 010 Park on Master Port 2 011 Park on Master Port 3 100 Park on Master Port 4 101 Reserved 110 Reserved 111 Reserved

The SGPCR can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the SGPCR the SGPCR can only be read, attempts to write to it has no effect on the SGPCR and result in an error response.

34.2.2.3 Master General Purpose Control Register

The master general purpose control register (MGPCR) presently controls only whether or not the master's undefined length burst accesses is allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters.

The AULB (Arbitrate on Undefined Length Bursts) bit field determines whether (and when) or not the MAX arbitrates away the slave port the master owns when the master is performing undefined length burst accesses.

Offset $BASE + 0x800 + n*100 Access: Read/Write
Wait State: 1

	31																										3	2	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AULB	
W																													
Reset	All zeros																												

Figure 34-4. Master General Purpose Control Register n[1]

1 For n = 0–4

for n = 0 to 4

Table 34-6. Master General Purpose Control Register Descriptions

Bits	Description
31–3	Reserved
2–0 AULB	Arbitrate on Undefined Length Bursts These bits are used to select the arbitration policy during undefined length bursts by this master. 000 No arbitration is allowed during an undefined length burst. 001 Arbitration is allowed at any time during an undefined length burst. 010 Arbitration is allowed after four beats of an undefined length burst. 011 Arbitration is allowed after eight beats of an undefined length burst. 100 Arbitration is allowed after 16 beats of an undefined length burst. 101 Reserved 110 Reserved 111 Reserved

The MGPCR can only be accessed in supervisor mode with 32-bit accesses.

34.2.3 Coherency

Since the content of the registers has a real time effect on the operation of the MAX it is important for the user to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave port related AHB accesses but instead track only with IP bus accesses.

The exception to this rule are the AULB bits in the MGPCR. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the IP bus cycle to write them has long since terminated successfully. If the AULB bits in the MGPCR are written in between two burst accesses the new AULB encodings does not take effect until an IDLE cycle has been initiated by the master on that master port.

34.3 Function

This section describes in more detail the functionality of the MAX.

34.3.1 Arbitration

The MAX supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

34.3.1.1 Arbitration During Undefined Length Bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR AULB field setting. When a defined length is imposed on the burst using the AULB bits the undefined length burst is treated as a single or series of single back to back fixed length burst accesses.

Example: A master runs an undefined length burst and the AULB bits in the MGPCR indicate arbitration occurs after the fourth beat of the burst. The master runs two sequential beats and then starts what is a 12-beat undefined length burst access to a new address within the same slave port region as the previous access. The MAX does not allow an arbitration point until the fourth overall access (second beat of the second burst). At that point, all remaining accesses are open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. Once the master regains control of the slave port no arbitration point are available until after the master has run four more beats of its burst. After the fourth beat of the (now continued) burst (ninth beat of the second burst from the master's perspective) is taken all beats of the burst are once again open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third (now continued) burst (10th beat of the second burst from the master's perspective). Once the master regains control of the slave port it is allowed to complete its final two beats of its burst without facing arbitration.

Fixed length burst accesses are not affected by the AULB bits. All fixed length burst accesses lock out arbitration until the last beat of the fixed length burst.

34.3.1.2 Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the MPR (Master Priority Register). If two masters both request access to a slave port, the master with the highest priority in the selected priority register gains control over the slave port.

Any time a master makes a request to a slave port the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case, the new requesting master has to wait until the end of the burst transfer or locked transfer before it is granted control of the slave port. If the master is running an undefined length burst transfer the new requesting master must wait until an arbitration point for the undefined length burst transfer before it is granted control of the slave port. Arbitration points for an undefined length burst are defined in the MGPCR for each master.

If the new requesting master's priority level is lower than that of the master that currently has control of the slave port the new requesting master is forced to wait until the master that currently has control of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

34.3.1.3 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number. This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus as the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master (ID is defined by master port number, not the **hmaster** field).

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next assertion of **sX_hready**, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the MAX is implemented with master ports 0, 1, 2, 3, 4 and 5. If the last master of the slave port was master 1, and master 0, 4 and 5 make simultaneous requests, (master ports 2 and 3 make no requests), they are serviced in the order 4, 5 and then 0.

Parking may still be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer is reset to point at master port 0, giving it the highest priority.

34.3.2 Priority Assignment

Each master port needs to be assigned a unique three bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register the MAX responds with an error and the registers are not updated.

34.3.3 Slave Port Functionality

34.3.3.1 General

Each slave port consists of a register slice, a bank of muxes and a state machine.

The register slice contains the registers associated with the specific slave port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The muxes are a series of 6 to 1 muxes that take in all the address, control and write data information from each of the master ports and then pass the correct master's signals to the slave port. The state machine controls all the muxes.

The state machine is where the main slave port arbitration occurs, it decides which master is in control of the slave port and which master is in control of the slave port in the next bus cycle.

A block diagram of a slave port can be seen in Figure 34-5.

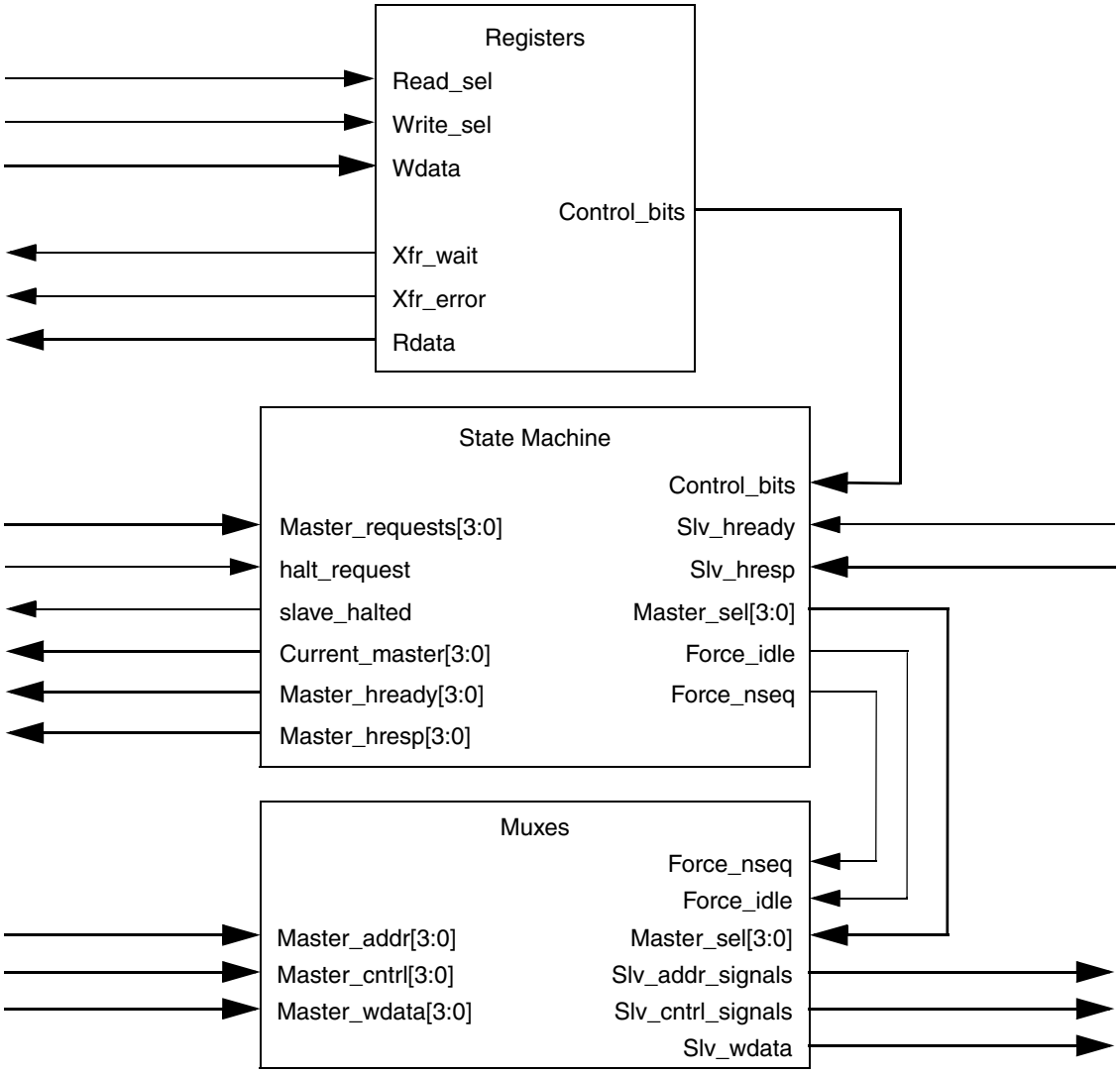


Figure 34-5. MAX Slave Port Block Diagram

34.3.3.2 Slave Port Muxes

The block diagram (Figure 34-5) shows only one block for all the muxes. In reality that block instantiates many 6 to 1 muxes, one for each master-to-slave signal in fact. All the muxes are designed in an AND - OR fashion, so that if no master is selected the output of the muxes is zero. (This is an important feature for low power park mode.)

The muxes also have an override signal which is used by the slave port to asynchronously force IDLE cycles onto the slave bus. When the state machine forces an IDLE cycle it zeros out **htrans** and **hmastlock**, making sure the slave bus sees a valid IDLE cycle being run by the MAX.

The enable to the mux controlling **htrans** also contains an additional control signal from the state machine so that a NSEQ transaction can be forced. This is done any time the slave port switches masters to ensure

that no IDLE-SEQ, BUSY-SEQ or NSEQ-SEQ transactions are seen on the slave port when they shouldn't be. If the state machine indicates to run both an IDLE and an NSEQ cycle, the IDLE directive has priority.

NOTE

IDLE-SEQ is in fact an illegal access, but a possible scenario given the multi-master environment in the MAX unless corrected by the MAX.

34.3.3.3 Slave Port Registers

There is a register control block at the same level of the master port and slave port instantiations in the MAX. This control block ensures that all accesses are 32-bit supervisor accesses before passing them on to the master and slave ports.

The registers in the slave port are only those registers associated with this particular slave port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

The register outputs are connected directly to the slave state machine. The registers can be read from an unlimited number of times. The registers can only be written to as long as the RO bit is written to 0 in the SGPCR, once it is written to a 1 only a hardware reset allows the registers to be written again.

34.3.3.4 Slave Port State Machine

34.3.3.4.1 Slave Port State Machine States

At the heart of the slave port is the state machine. The state machine is simplicity itself, requiring only four states - **steady state**, **transition state, transition hold state** and **hold state**. Either the slave port is owned by the same master it was in the last clock cycle (either by active use or by parking), it is transitioning to a new master (either for active use or parking), it is transitioning to a new master during wait states or it is being held on the same master pending a transition to a new master.

34.3.3.4.2 Slave Port State Machine Arbitration

The real work in the state machine is determining which master port is in control of the slave port in the next clock cycle, the arbitration. Each master is programmed with a fixed 3 bit priority level. The MAX uses these bits in determining priority levels when programmed for fixed priority mode of operation.

Arbitration always occurs on a clock edge, but only occurs on edges when a change in mastership does not violate AHB-Lite protocols. Valid arbitrations points include any clock cycle in which **sX_hready** is asserted (provide the master is not performing a burst or locked cycle) and any wait state in which the master owning the bus indicates a transfer type of IDLE (provided the master is not performing a locked cycle).

Since arbitration can occur on every clock cycle the slave port masks off all master requests if the current master is performing a locked transfer or a protected burst transfer, guaranteeing that no matter how low its priority level it is allowed to finish its locked or protected portion of a burst sequence.

34.3.3.4.3 Slave Port State Machine Master Handoff

The only times the slave port switches masters when programmed for fixed priority mode of operation is when a higher priority master makes a request or when the current master is the highest priority and it gives up the slave port by either running and IDLE cycle to the slave port or running a valid access to a location other than the slave port.

If the current master loses control of the slave port because a higher priority master takes it away, the slave port does not incur any wasted cycles. The current master gets its current cycle terminated by the slave port at the same time the new master's address and control information is recognized by the slave port. This looks like a seamless transition on the slave port.

If the current master is being wait stated when the higher priority master makes its request, then the current master is allowed to make one more transaction on the slave bus before giving it up to the new master. Figure 34-6 illustrates the effect of a higher priority master taking control of the bus when the slave port is programmed for a fixed priority mode of operation.

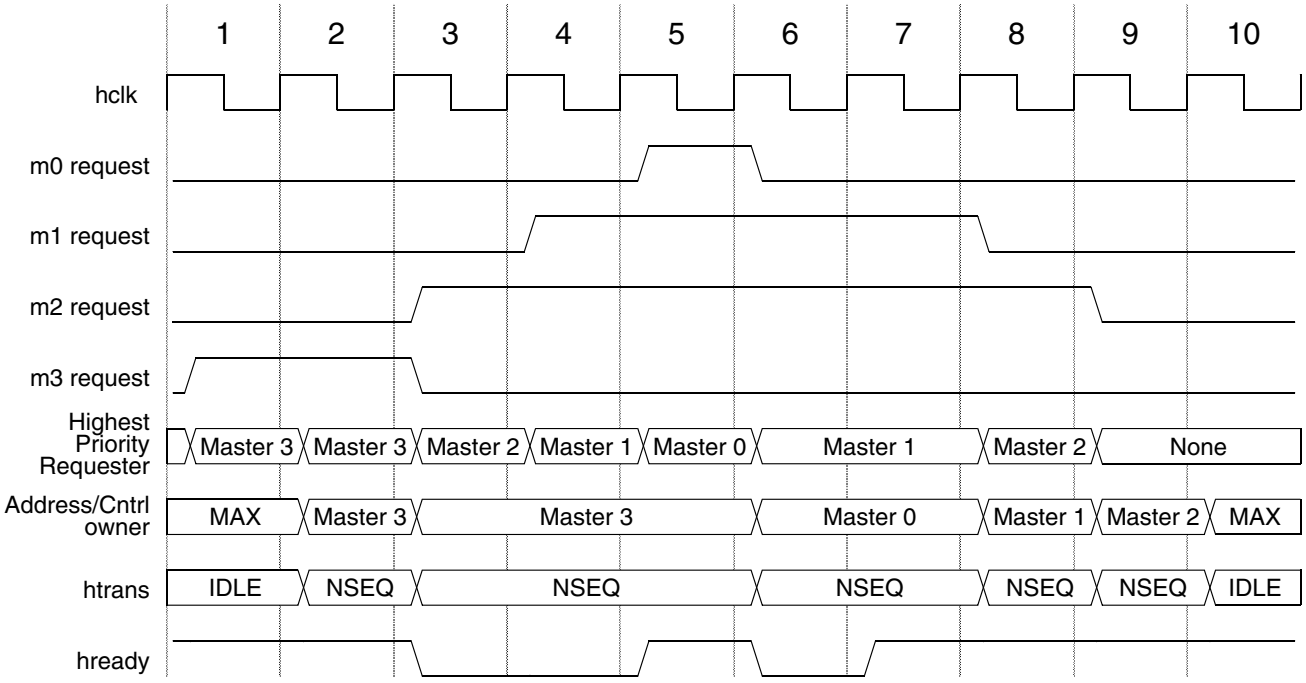


Figure 34-6. Low to High Priority Mastership Change

If the current master is the highest priority master and it gives up the slave port by running an IDLE cycle or by running a valid cycle to another location other than the slave port the next highest priority master gains control of the slave port. If the current access incurs any wait states then the transition is seamless and no bandwidth is lost; however, if the current transaction is terminated without wait states then one IDLE cycle is forced onto the slave bus by the MAX before the new master is able to take control of the slave port. If no other master is requesting the bus then IDLE cycles is run by the MAX but no bandwidth is truly lost since no master is making a request. Figure 34-7 illustrates the effect of a higher priority master giving up control of the bus.

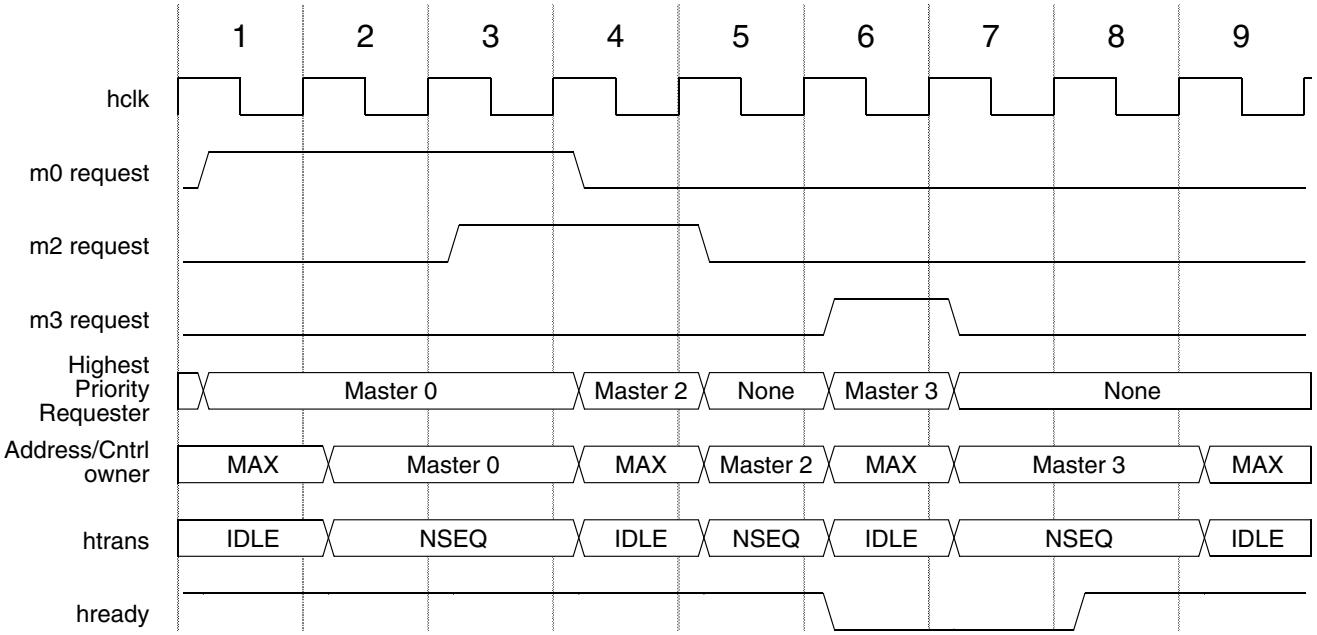


Figure 34-7. High to Low Priority Mastership Change

When the slave port is programmed for round-robin mode of arbitration then the slave port switches masters any time there is more than one master actively making a request to the slave port. This happens because any master other than the one which presently owns the bus is considered to have higher priority. Figure 34-8 shows an example of round-robin mode of operation.

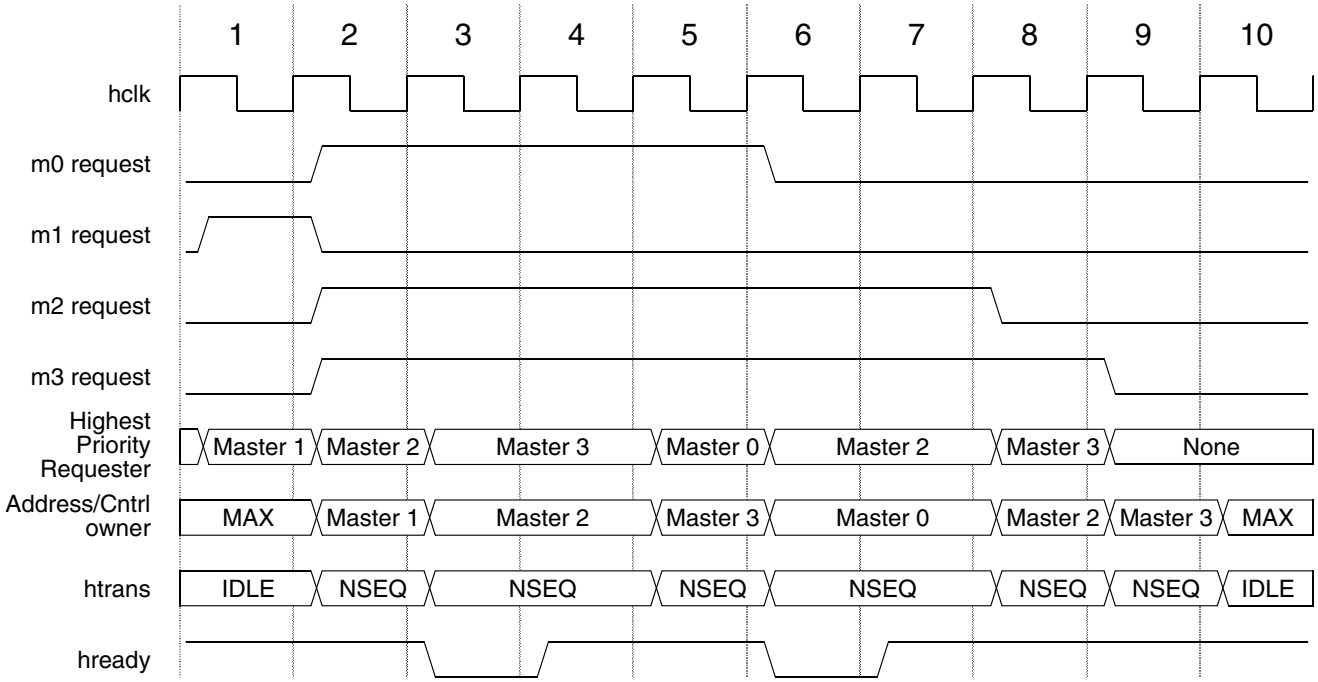


Figure 34-8. Round-Robin Mastership Change

34.3.3.4.4 Slave Port State Machine Parking

If no master is currently making a request to the slave port then the slave port is parked. It parks in one of four places, dictated by the PCTL and PARK bits in the GPCR and the locked state of the last master to access it.

If the last master to access the slave port ran a locked cycle and continues to run locked cycles even after leaving the slave port the slave port parks on that master without regard to the bit settings in the GPCR and without regard to pending requests from other masters. This is done so a master can run a locked transfer to the slave port, leave it, and return to it and be guaranteed that no other master has had access to it (provided the master maintains all transfers are locked transfers). If locking is not an issue for parking, the GPCR bits dictate the parking method.

If the PCTL bits are set for “low power park” mode then the slave port enters low power park mode. It does not recognize any master as being in control of it and it does not select any master’s signals to pass through to the slave bus. In this case, all slave bus activity effectively halts because all slave bus signals being driven from the MAX are 0. This of course can save quite a bit of power if the slave port is not in use for some time. The down side is that when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

If the PCTL bits are set to “park on last” mode then the slave port will park on the last master to access it, passing all that masters signals through to the slave bus. The MAX will asynchronously force **htrans[1:0]**, **hmaster[3:0]**, **hburst[2:0]** and **hmastlock** to 0 for all access that the master does not run to the slave port. When that master access the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed.

If the PCTL bits are set to “use PARK” mode then the slave port will park on the master designated by the PARK bits. The behavior here is the same as for the “park on last” mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK bits tries to access the slave port it will not pay an arbitration penalty while any other master will pay a one clock penalty. Figure 34-9 illustrates parking on a specific master.

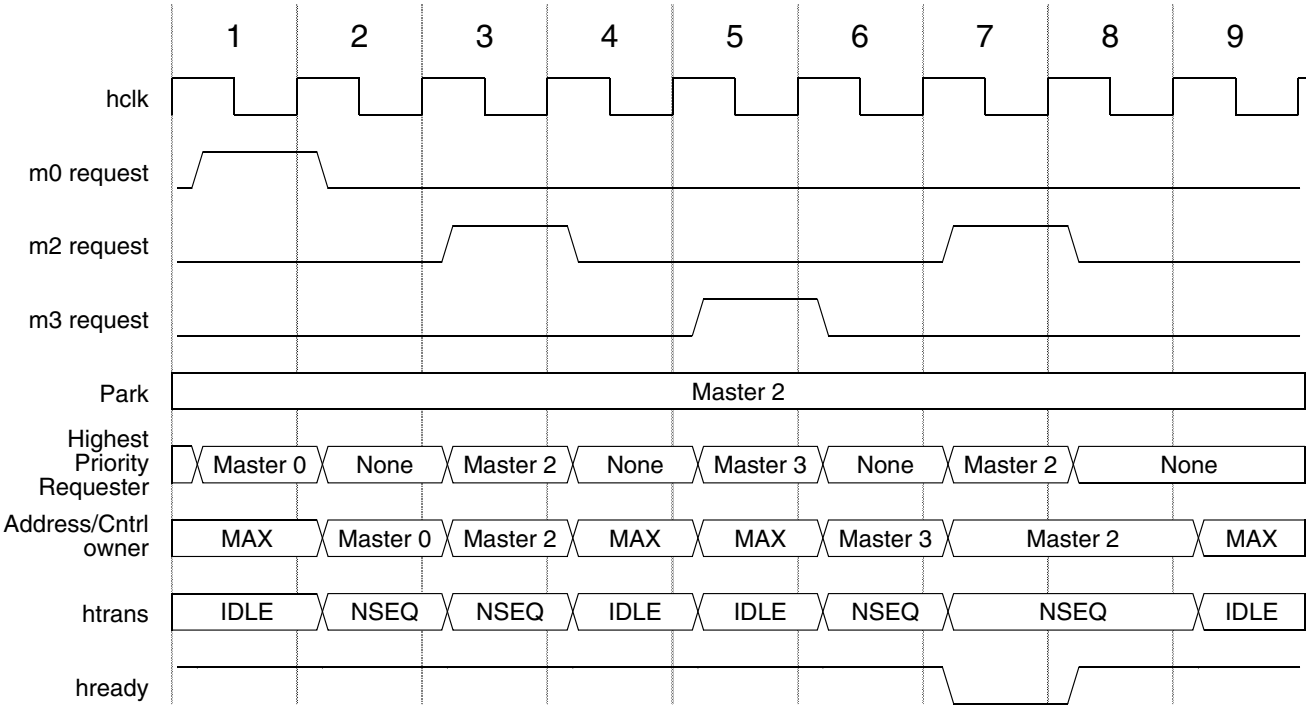


Figure 34-9. Parking on a Specific Master

Figure 34-10 illustrates parking on the last master. Note that in cycle 6 simultaneous requests are made by master 2 and master 3. Although master 2 has higher priority, the slave bus is parked on master 4 so master 4's access will be taken first. The slave port parks on master 2 once it has given control to master 2. This same situation can occur when parking on a specific master as well.

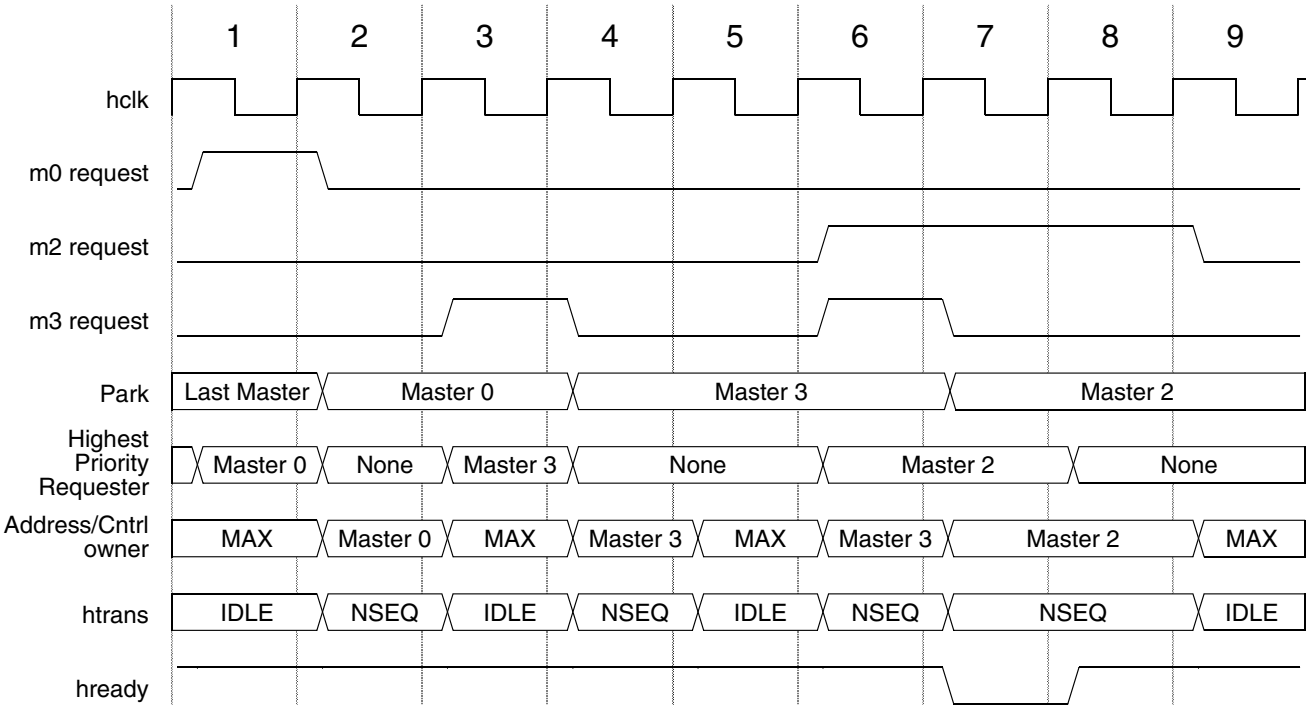


Figure 34-10. Parking on Last Master

34.3.3.4.5 Slave Port State Machine Halt Mode

If the **max_halt_request** input is asserted the slave port will eventually halt all slave bus activity and go into halt mode, which is almost identical to low power park mode. The HLP bit in the GPCR controls the priority level of the **max_halt_request** in the arbitration algorithm. If the HLP bit is cleared then the **max_halt_request** will have the highest priority of any master and will gain control of the slave port at the next arbitration point (most likely the next bus cycle, unless the current master is running a locked or fixed length burst transfer). If the HLP bit is set then the slave port will wait until no masters are actively making requests before moving to halt mode.

Regardless of the state of the HLP bit, once the slave port has gone into halt mode as a result of **max_halt_request** being asserted, it will remain in halt mode until **max_halt_request** is negated, regardless of the priority level of any masters that may make requests.

In halt mode no master is selected to own the slave port so all the outputs of the slave port are set to 0.

34.4 Initialization/Application Information

No initialization is required by or for the MAX. Hardware reset ensures all the register bits used by the MAX are properly initialized.

34.5 Interface

The main goal of the MAX is to increase overall system performance by allowing multiple masters to communicate in parallel with multiple slaves. In order to maximize data throughput it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the MAX will stall the masters or insert bubbles on the slave side.

34.5.1 Master Ports

Master accesses will receive one of four responses from the MAX. They will either be terminated, taken, stalled or responded to with an error.

34.5.1.1 Terminated Accesses

A master access will be terminated if the transfer type is IDLE. The MAX will terminated the access and it will not be allowed to pass through the MAX.

34.5.1.2 Taken Accesses

A master access will be taken if the transfer type is non IDLE and the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case the MAX will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred.

34.5.1.3 Stalled Accesses

A master access will be stalled if the transfer type is non IDLE and the access decodes to a slave port that is busy serving another master, parked on another master or is in low power park mode. The MAX will indicate to the master that the address phase of the access has been taken but will then queue the access to the appropriate slave port to enter into arbitration for access to that slave port.

If the slave port is currently parked on another master or is in low power park mode and no other master is requesting access to the slave port then only one clock of arbitration will be incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters then the master will gain control over the slave port as soon as the data phase of the current access is completed (burst and locked transfers excluded). If the slave port is currently servicing another master of a higher priority then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

34.5.1.4 Error Response Terminated Accesses

A master access will be responded to with an error if the transfer type is non IDLE and the access decodes to a location not occupied by a slave port. This is the only time the MAX will respond with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the MAX.

34.5.2 Slave Ports

The goal of the MAX with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. In order to do this the MAX must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the MAX will force a bubble onto the slave bus when a master is actively making a request. This occurs when a higher priority master has control of the slave port and is running single clock (zero wait state) accesses while a lower priority master is stalled waiting for control of the slave port. When the higher priority master either leaves the slave port or runs an IDLE cycle to the slave port the MAX will take control of the slave bus and run a single IDLE cycle before giving the slave port to the lower priority master that was waiting for control of the slave port.

The only other times the MAX will have control of the slave port is when the MAX is halting or when no masters are making access requests to the slave port and the MAX is forced to either park the slave port on a specific master or put the slave port into low power park mode.

In most instances when the MAX has control of the slave port it will indicate IDLE for the transfer type, negate all control signals and indicate ownership of the slave bus using the **hmaster** encoding of 4’b0000. One exception to this rule is when a master running locked cycles has left the slave port but continues to run locked cycles. In this case the MAX will control the slave port and will indicate IDLE for the transfer type but it will not affect any other signals.

NOTE

When a master runs a locked cycle through the MAX, the master will be guaranteed ownership of all slave ports it accesses while running locked cycles for one cycle beyond when the master finishes running locked cycles.

Chapter 35
Multi-Master Memory Interface (M3IF)

The Multi-Master Memory Interface (M3IF) controls memory accesses (read/write/erase/program) from one or more masters through different port interfaces to three different external memory controllers:

- Enhanced SDRAM mobile / low power DDR and DDR2 controller (ESDRAMC/MDDRC, subsequently abbreviated as ESDRAMC)
- NAND Flash controller (NFC)
- Wireless external interface module (WEIM).

Figure 35-1 shows the M3IF module's functional organization at top level.

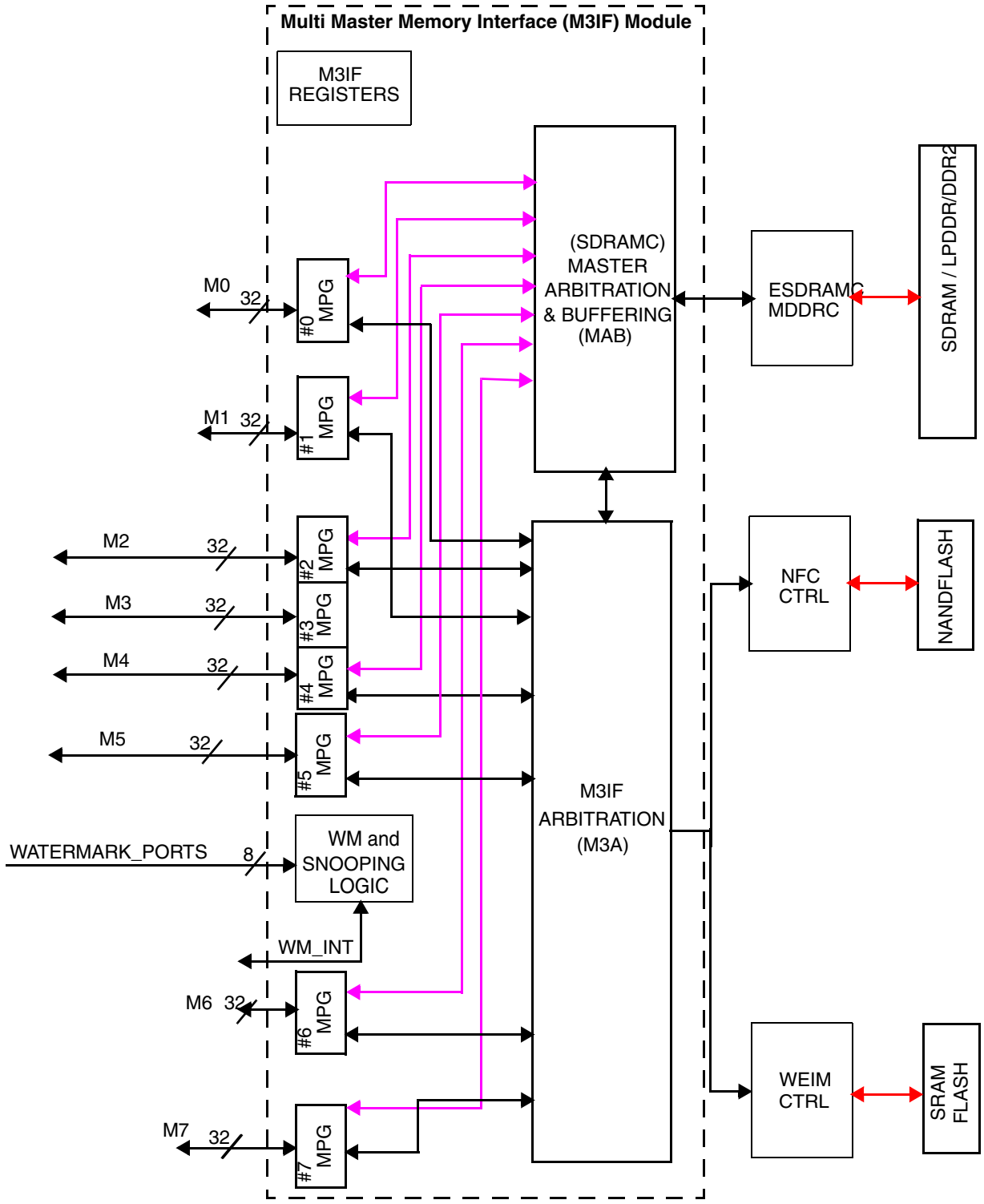


Figure 35-1. M3IF Schematics Connection.

35.1 Overview

The M3IF-ESDRAMC interface is optimized and designed to reduce access latency by generating multiple accesses through the dedicated ESDRAMC arbitration (MAB) module, which controls the access to/from the ESDRAMC. For the other port interfaces, the M3IF only arbitrates and forwards the master requests received through the Master Port Gasket (MPG) interface and M3IF Arbitration (M3A) module toward the respective memory controller.

When a master requests a memory access, the access is immediately taken by the M3IF if no other access is in progress. The M3IF forwards the access to the respective memory controller (slave), and depending on the state of the respective memory controller, a command to the memory is generated. If the access cannot be started due to a previous active access, the master request remains pending (HREADY held negated) until it is executed by the memory controller. When the access execution is complete, the HREADY is asserted and a new request can be processed.

Accesses to the SDRAM, LPDDR or DDR2 external devices are optimized through command anticipation (MIF2 strategy). For example, the next access control phase (memory address and command) is driven during the previous access data phase (data flow to/from the memory). This creates an overlap between accesses that partially or fully hides the latency.

35.1.1 M3IF Interfaces

The interface between the M3IF and the controllers can be divided into two different types: M3IF-ESDRAMC, and M3IF-all others. The M3IF-to-ESDRAMC interface reduces access latency by generating multiple accesses using the dedicated ESDRAMC arbitration (MAB) module.

For the other port interfaces, the M3IF arbitrates and forwards the masters' requests received through the Master Port Gasket (MPG) interfaces and the M3IF arbitration (M3A) module toward the respective memory controller. To support multiple accesses to the ESDRAMC, the MAB includes a FIFO which controls the access traffic from/to the ESDRAMC.

The M3IF can be viewed as a device with multiple SDRAM/LPDDR/DDR2 controllers, plus one controller per non-SDRAM/LPDDR/DDR2 memory type. The M3A arbitrates memory access requests as follows:

- A round robin chooses the next master that is granted access to the bus, as follows:
  - If this master is requesting access to the ESDRAMC, the M3A waits until the previous access finishes, then passes the request.
  - If the master is requesting access to ESDRAMC-controlled memory, the M3IF arbitration passes the access to the MAB as follows:
    - If the previous access is to non–ESDRAMC-controlled memory, the M3IF arbitration passes the request after the previous access is completed.
    - If the previous access was to ESDRAMC-controlled memory, the M3IF arbitration passes the request immediately, without waiting for the previous access to complete.

The M3IF Arbitration (M3A) and the ESDRAMC Master Arbitration and Buffering (MAB) supports a round-robin arbitration scheme that can be programmed to non-equal probability. If two masters request access to the memory port on the same cycle, the master with the token gains control on the bus (for more

details, see Section 35.3.3.2, “M3A—Find First 1 (FF1) Algorithm”). To support multiple accesses to the ESDRAMC, the MAB includes a FIFO which controls the access traffic from/to the ESDRAMC.

35.1.2 Features

The M3IF Master Port Gasket (MPG) converts the master request (data write, data read, address, and controls) to a set of bus/signals that the M3IF arbitration, the ESDRAMC arbitration, and other memory controllers need. The MPG is also responsible to give the right response to the master after getting the response from the relevant memory controller. The M3IF supports both 32bits interfaces and 64bits interfaces. The number of gaskets and the bus width is set according to SOC requirements,.

The M3IF includes these distinctive features:

- Supports multiple requests up to 8 masters through input port interfaces:
  - Master Port Gasket (MPG) - ARM11 AMBA AHB lite bus protocol.
  - Master Port Gasket (MPG64) - AMBA AHB access with 64 bits data bus width.
- Arbitrates requests to three different memory controllers (that share some of their I/O pads)
  - Enhanced SDRAM/LPDDR/DDR2 controller (ESDRAMC)
  - NAND Flash Controller - (NFC)
  - Wireless External Interface Memory Controller - (WEIM)
- Multiple requests capabilities to ESDRAMC through a dedicated arbitration mechanism.
- Flexible round robin access arbitration, with equal priority or 50% priority to selective masters.
- Programmable master that controls (lock) accesses to SDRAM/DDR and programmable master that controls (lock) accesses to other memories (= general: NFC, WEIM).
- Support for multi-endian byte order to all memory controllers.
- Supports memory watermark protection for up to 8 different chip selects for preselected masters. Watermark is enabled according to SOC requirements.
  - Configurable protected memory region (base address with 1-Kbyte resolution) for each one of the 8 predefined chip selects.
  - One status bit for each memory region that indicates security violation.
  - Maskable watermark interrupt generation capability.
- Supports memory snooping, an example of which would be monitoring a region in external memory for write accesses:
  - The region’s location is specified by a base address (from 2 Kbytes up to 16 Mbytes), which is divided into 64 equal segments.
  - Each segment has an access status and enable bit in the M3IF register definition.
  - M3IF generates a one cycle DMA_ACCESS for each snooping detection.

35.2 Memory Map and Register Definition

The M3IF programming model consists of several classes of registers, M3IF control and lock registers, watermark configuration and status registers, and snooping configuration and status registers as shown in Table 35-1. The control and master lock general register defines the M3IF configurable logic functionality. The configuration and status registers set and monitor watermark and snooping activity respectively.

All M3IF registers are 32-bits in length with bit fields defined in Figure 35-3 to Figure 35-11. All implemented bits are fully readable and writable in supervisor mode only (an error response is generated in case of user mode access to M3IF registers). All M3IF (and ESDRAMC) registers can be accessed only by a SINGLE word (32-bit) access, through the AHB bus protocol. Accesses of any other size or type cause undetermined behavior.

All registers can be accessed by only one master at a time. Multi access to M3IF register causes undetermined behavior. The only exception is the M3IF Master Lock General register that can be accessed by more than one master at a time.

The reset state of each bit is shown underneath the bit field name. An asterisk indicates that the value is dependent on the operating mode selected during reset. Details are provided in the following bit field descriptions.

35.2.1 Memory Map

Table 35-1 shows the memory map for the M3IF registers. Table 35-2 shows the M3IF memory space allocated to the three different memory controllers (ESDRAMC, NFC, and WEIM).For the base address of a particular module instantiation, see the system memory map.

Table 35-1. M3IF Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (M3IFCTL)	M3IF Control Register	R/W	0x0000_0000	35.2.3.1/35-11
0x0004 (M3IFWCFG0)	M3IF Watermark Configuration Register 0	R/W[1]	0xFFFF_FFFF	35.2.3.2/35-13
0x0008 (M3IFWCFG1)	M3IF Watermark Configuration Register 1	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x000C (M3IFWCFG2)	M3IF Watermark Configuration Register 2	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x0010 (M3IFWCFG3)	M3IF Watermark Configuration Register 3	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x0014 (M3IFWCFG4)	M3IF Watermark Configuration Register 4	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x0018 (M3IFWCFG5)	M3IF Watermark Configuration Register 5	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x001C (M3IFWCFG6)	M3IF Watermark Configuration Register 6	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x0020 (M3IFWCFG7)	M3IF Watermark Configuration Register 7	R/W	0xFFFF_FFFF	35.2.3.2/35-13
0x0024 (M3IFWCSR)	M3IF Watermark Control and Status Register	R/W	0x0000_0000	35.2.3.3/35-14
0x0028 (M3IFSCFG0)	M3IF Snooping Configuration Register 0	R/W	0x0000_0000	35.2.3.4/35-15
0x002C (M3IFSCFG1)	M3IF Snooping Configuration Register 1	R/W	0x0000_0000	35.2.3.5/35-17
0x0030 (M3IFSCFG2)	M3IF Snooping Configuration Register 2	R/W	0x0000_0000	35.2.3.6/35-18

Table 35-1. M3IF Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0034 (M3IFSSR0)	M3IF Snooping Status Register 0	R/W	0x0000_0000	35.2.3.7/35-19
0x0038 (M3IFSSR1)	M3IF Snooping Status Register 1	R/W	0x0000_0000	35.2.3.8/35-20
0x0040 (M3IFMLWE0)	M3IF Master Lock WEIM CS0 Register	R/W	0x0000_0000	35.2.3.9/35-21
0x0044 (M3IFMLWE1)	M3IF Master Lock WEIM CS1 Register	R/W	0x0000_0000	35.2.3.9/35-21
0x0048 (M3IFMLWE2)	M3IF Master Lock WEIM CS2 Register	R/W	0x0000_0000	35.2.3.9/35-21
0x004C (M3IFMLWE3)	M3IF Master Lock WEIM CS3 Register	R/W	0x0000_0000	35.2.3.9/35-21
0x0050 (M3IFMLWE4)	M3IF Master Lock WEIM CS4 Register	R/W	0x0000_0000	35.2.3.9/35-21
0x0054 (M3IFMLWE5)	M3IF Master Lock WEIM CS5 Register	R/W	0x0000_0000	35.2.3.9/35-21

Note:

1 All 9 watermark registers are accessible (read/write) only by the predefined (hardware) master in a supervisor mode access, HPROT signal high.

Table 35-2. M3IF Memory Space Summary

Address	Use	Access
ESDRAMC Memory Space		
0x8000_0000–0x8FFF_FFFF	CSD0 SDRAM/LPDDR/DDR2 memory region (256MB)	Read/write
0x9000_0000–0x9FFF_FFFF	CSD1 SDRAM/LPDDR/DDR2 memory region (256MB)	Read/write
WEIM Memory Space		
0xA000_0000– 0xA7FF_FFFF	WEIM CS0 memory region (128MB)	Read/write
0xA800_0000–0xAFFF_FFFF	WEIM CS1 memory region (128MB)	Read/write
0xB000_0000–0xB1FF_FFFF	WEIM CS2 memory region (32MB)	Read/write
0xB200_0000– 0xB3FF_FFFF	WEIM CS3 memory region (32MB)	Read/write
0xB400_0000–0xB5FF_FFFF	WEIM CS4 memory region (32MB)	Read/write
0xB600_0000–0xB7FF_FFFF	WEIM CS5 memory region (32MB)	Read/write
NFC Memory Space		
0xBB00_0000–0xBB00_1FFF	NAND Flash memory region[1] (4KB)	Read/write

Note:

1 Can be used as a boot memory region.

35.2.2 Register Summary

Figure 35-2 shows the key to the register fields, and Table 35-3 shows the register figure conventions.

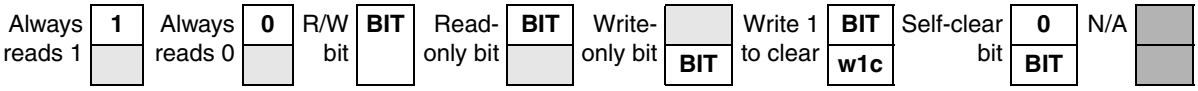


Figure 35-2. Key to Register Fields

Table 35-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 35-4 shows the M3IF register summary.

Table 35-4. M3IF Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (M3IFCTL)	R	SDA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	MLSD_EN	MLSD			MRRP							
	W																

Table 35-4. M3IF Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0004 (M3IFWCFG0)	R	WBA0															
	W																
	R	WBA0						1	1	1	1	1	1	1	1	1	1
	W																
0x0008 (M3IFWCFG1)	R	WBA1															
	W																
	R	WBA1						1	1	1	1	1	1	1	1	1	1
	W																
0x000C (M3IFWCFG2)	R	WBA2															
	W																
	R	WBA2						1	1	1	1	1	1	1	1	1	1
	W																
0x0010 (M3IFWCFG3)	R	WBA3															
	W																
	R	WBA3						1	1	1	1	1	1	1	1	1	1
	W																
0x0014 (M3IFWCFG4)	R	WBA4															
	W																
	R	WBA4						1	1	1	1	1	1	1	1	1	1
	W																
0x0018 (M3IFWCFG5)	R	WBA5															
	W																
	R	WBA5						1	1	1	1	1	1	1	1	1	1
	W																
0x001C (M3IFWCFG6)	R	WBA6															
	W																
	R	WBA6						1	1	1	1	1	1	1	1	1	1
	W																

Table 35-4. M3IF Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0020 (M3IFWCFG7)	R	WBA7															
	W																
	R	WBA7						1	1	1	1	1	1	1	1	1	1
	W																
0x0024 (M3IFWCSR)	R	WIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
0x0028 (M3IFSCFG0)	R	SWBA															
	W																
	R	SWBA					0	0	0	0	0	0	SWSZ				SE
	W																
0x002C (M3IFSCFG1)	R	SSE0															
	W																
	R	SSE0															
	W																
0x0030 (M3IFSCFG2)	R	SSE1															
	W																
	R	SSE1															
	W																
0x0034 (M3IFSSR0)	R	SSS0															
	W																
	R	SSS0															
	W																
0x0038 (M3IFSSR1)	R	SSS1															
	W																
	R	SSS1															
	W																

Table 35-4. M3IF Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0040 (M3IFMLWE0)	R	WEM A0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE0 _EN	MLWE0		
	W																
0x0048 (M3IFMLWE2)	R	WEM A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE1 _EN	MLWE1		
	W																
0x0048 (M3IFMLWE2)	R	WEM A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE2 _EN	MLWE2		
	W																
0x004C (M3IFMLWE3)	R	WEM A3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE3 _EN	MLWE3		
	W																
0x0050 (M3IFMLWE4)	R	WEM A4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE4 _EN	MLWE4		
	W																
0x0054 (M3IFMLWE5)	R	WEM A5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE5 _EN	MLWE5		
	W																

35.2.3 Register Descriptions

This section contains the detailed register descriptions for the M3IF registers.

35.2.3.1 M3IF Control Register (M3IFCTL)

The M3IFCTL register contains access status and provides access control for SDRAM/LPDDR/DDR2 memory devices, and sets arbitration priority for M3IF port masters. The field assignments for this register are shown in Figure 35-3 and the field descriptions are listed in Table 35-5.

Offset 0x0000 (M3IFCTL) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	MLSD_EN	MLSD			MRRP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-3. M3IF Control Register

Table 35-5. M3IF Control Register Field Descriptions

Field	Description
31 SDA	SDRAM/LPDDR/DDR2 Memory Active. This is a read-only status bit that, if set, indicates that an active/pending access to SDRAM/LPDDR memory exists. The SDA bit is set on one of the following conditions: • If MLSD_EN (bit 11) is cleared, then any active/pending access to SDRAM/LPDDR/DDR2 memory space sets the bit (until the access is completed). • If MLSD_EN is set, then any accesses to SDRAM/LPDDR/DDR2 memory space initiated previously to MLSD_EN assertion, keeps the SDA status bit set. The bit clears after all pending/active accesses are completed. Access from the master indicated by the MLSD field (bits 10:8) do not assert the status bit. **Note:** When the MLSD_EN is set, any new accesses (initiated after MLSD_EN assertion) to SDRAM/LPDDR/DDR2 not from MLSD master will be pending without setting SDA to 1. Only the SDRAM/LPDDR/DDR2 memory space region will be locked to the MLSD port. Accesses to M3IF/ESDRAMC registers are available to all masters in the system: the software is responsible not to access those registers during the lock period. 0 No active/pending access to SDRAM/LPDDR/DDR2 memory exists. 1 Indicates an active/pending access to SDRAM/LPDDR/DDR2 memory exists.
30–12	Reserved
11 MLSD_EN	Master Lock SDRAM/LPDDR/DDR2 Access. This bit enables the Master Control SDRAM/LPDDR/DDR2 access (MLSD). The reset value of this bit is “0”. 0 Master Control SDRAM/LPDDR/DDR2 access (MLSD) disabled. 1 Master Control SDRAM/LPDDR/DDR2 access (MLSD) enabled.

Table 35-5. M3IF Control Register Field Descriptions (continued)

Field	Description
10–8 MLSD	Master Lock SDRAM/LPDDR/DDR2 Access. This 3-bit field defines the master port number (MPG) that is the only master in the system to be served by the ESDRAMC. All accesses toward the SDRAM/LPDDR/DDR2 from the other masters will be postponed, until the MLSD master clears the MLSD_EN bit. The reset value of the MLSD is 0b000. **Note:** Accesses to ESDRAMC registers are not effected by the MLSD field. For example, they can be accessed by any master even if MLSD_EN is set. Prior to lock accesses, the MLSD master should perform the following steps: 1. Set the MLSD_EN bit and the MLSD field (with the desired value) in the M3IFCTL register. 2. Read M3IFCTL register and check: 3. SDA status bit is cleared (no pending/active accesses to SDRAM/LPDDR/DDR2 memory space exists). 4. MLSD_EN bit is set. 5. MLSD (value) points to the required port number (master port number that requires lock accesses). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7
7–0 MRRP	Master Round Robin Priority. MRRP field is an 8-bit field with one bit per master (bit #i to master #i). Masters with their MRRP bit set are added to a priority arbitration “list” so that together they have 50% probability to gain access through both M3A and MAB arbitration processes (50% probability for each one of the arbitration separately). Assertion of MRRP bit for an unused master is forbidden. If all MRRP bits are cleared, the masters have equal probability to pass the arbitration processes. For more details about the M3IF arbitration see Section 35.3.3.2, “M3A—Find First 1 (FF1) Algorithm.” 0 The respective master is not on the priority arbitration “list”. 1 Add the respective master to the priority arbitration “list” with a 50% probability to pass the arbitration processes.

35.2.3.2 M3IF Watermark Configuration Registers (M3IFWCFG0–M3IFWCFG7)

The M3IF Watermark Control Register contains the base address for each specific watermark space in the selected memory region. The first six registers are reserved for WEIM (CS0-CS5), and the last two registers are for SDRAM/LPDDR/DDR2 (CSD0-CSD1) system memory address space. The watermark feature is described in detail in Section 35.3.5.2, "Watermark Overview," on page 1-58. The field assignments for this register are shown in Figure 35-4 and the field descriptions are listed in Table 35-6.

Offset 0x0004 (M3IFWCFG0)
0x0008 (M3IFWCFG1)
0x000C (M3IFWCFG2)
0x0010 (M3IFWCFG3)
0x0014 (M3IFWCFG4)
0x0018 (M3IFWCFG5)
0x001C (M3IFWCFG6)
0x0020 (M3IFWCFG7)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WBAx															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WBAx						1	1	1	1	1	1	1	1	1	1
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 35-4. M3IF Watermark Configuration Register

Table 35-6. Watermark Configuration Register

Field	Description
31–10 WBAx	Watermark Memory Region (x) Base Address. The WBAx field gives the watermark base address for memory region x, x = 0…7. The watermark configuration registers are reserved for WEIM CS0 to CS5 and SDRAM/LPDDR/DDR2 CSD0 and SDRAM/LPDDR/DDR2 CSD1 system memory space address only. Writing an address which is not mapped to these respective address regions cause undefined functionality of watermark logic operation.
9–0	Reserved

35.2.3.3 M3IF Watermark Control and Status Register (M3IFWCSR)

The Watermark feature is described in detail in Section 35.3.5.2, "Watermark Overview." The field assignments for this register are shown in Figure 35-5 and the field descriptions are listed in Table 35-7.

Offset 0x0024 (M3IFWCSR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	WS7	WS6	WS5	WS4	WS3	WS2	WS1	WS0
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-5. M3IF Watermark Control and Status Register

Table 35-7. M3IF Watermark Control and Status Register Field Descriptions

Field	Description
31 WIE	Watermark Interrupt Enable. This bit enables the watermark interrupt generation. For example, if WIE is set and a watermark violation is detected (by the M3IF) a watermark interrupt is generated. 0 Watermark interrupt is disabled. 1 Watermark interrupt is enabled.
30–8	Reserved
7 WS7	Watermark Status 7. This bit indicates if watermark violation had occurred in the watermark memory region of SDRAM/LPDDR/DDR2 CS1 as defined by the M3IFWCFG7 register (WBA7). WS7 is cleared by writing a one to the bit. 0 SDRAM/LPDDR/DDR2 CSD1 watermark violation did not occur. 1 SDRAM/LPDDR/DDR2 CSD1 watermark violation had occurred.
6 WS6	WS6 - Watermark Status 6. This bit indicates if watermark violation had occurred in the watermark memory region of SDRAM/LPDDR/DDR2 CS0 as defined by the M3IFWCFG6 register (WBA6). WS6 is cleared by writing a one to the bit. 0 SDRAM/LPDDR/DDR2 CSD0 watermark violation did not occur. 1 SDRAM/LPDDR/DDR2 CSD0 watermark violation occurred.
5 WS5	Watermark Status 5. This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS5 as defined by the M3IFWCFG5 register (WBA5). WS5 is cleared by writing a one to the bit. 0 WEIM CS5 watermark violation has not occurred. 1 WEIM CS5 watermark violation occurred.
4 WS4	WS4 - Watermark Status 4. This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS4 as defined by the M3IFWCFG4 register (WBA4). WS4 is cleared by writing a one to the bit. 0 WEIM CS4 watermark violation has not occurred. 1 WEIM CS4 watermark violation has occurred.

Table 35-7. M3IF Watermark Control and Status Register Field Descriptions (continued)

Field	Description
3 WS3	WS3 - Watermark Status 3 This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS3 as defined by the M3IFWCFG3 register (WBA3). WS3 is cleared by writing a one to the bit. 0 WEIM CS3 watermark violation has not occurred. 1 WEIM CS3 watermark violation has occurred.
2 WS2	WS2 - Watermark Status 2 This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS2 as defined by the M3IFWCFG2 register (WBA2). WS2 is cleared by writing a one to the bit. 0 WEIM CS2 watermark violation has not occurred. 1 WEIM CS2 watermark violation has occurred.
1 WS1	WS1 - Watermark Status 1 This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS1 as defined by the M3IFWCFG1 register (WBA1). WS1 is cleared by writing a one to the bit. 0 WEIM CS1 watermark violation has not occurred. 1 WEIM CS1 watermark violation has occurred.
0 WS0	WS0 - Watermark Status 0 This bit indicates if watermark violation had occurred in the watermark memory region of WEIM CS0 as defined by the M3IFWCFG0 register (WBA0). WS0 is cleared by writing a one to the bit. 0 WEIM CS0 watermark violation had not occurred. 1 WEIM CS0 watermark violation has occurred.

35.2.3.4 M3IF Snooping Configuration Register 0 (M3IFSCFG0)

The M3IFSCFG0 register contains the snooping window base address, the size of snooping window and the snooping control bit fields which are used by the M3IF to monitor the write access. The snooping feature is described in detail in Section 35.3.5.3, “Snooping Overview.” The field assignments for this register are shown in Figure 35-6 and the field descriptions are listed in Table 35-8.

Offset 0x0028 (M3IFSCFG0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SWBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SWBA					0	0	0	0	0	0	SWSZ				SE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-6. M3IF Snooping Configuration Register 0 (M3IFSCFG0)

Table 35-8. M3IF Snooping Configuration Register 0 Field Descriptions

Field	Description
31–11 SWBA	Snooping Window Base Address. This field defines the snooping window base address to be monitored by the M3IF. M3IF monitors write accesses to the memory region above the base address window.
10–5	Reserved
4–1 SWSZ	Snooping Window Size. This field define the snooping window size as described in Table 35-9
0 SE	Snooping Enable. This bit enables snooping detection. The M3IF monitors and detects write accesses to the snooping window. 0 Snooping feature is disabled. 1 Snooping feature is enabled.

Table 35-9. SWSZ Field Descriptions

SWSZ	Snooping Window Size	Window Base Address Bits	Window Address Bits in Use
0000	2 Kbyte	[31:11]	[10:0]
0001	4 Kbyte	[31:12]	[11:0]
0010	8 Kbyte	[31:13]	[12:0]
0011	16 Kbyte	[31:14]	[13:0]
0100	32 Kbyte	[31:15]	[14:0]
0101	64 Kbyte	[31:16]	[15:0]
0110	128 Kbyte	[31:17]	[16:0]
0111	256 Kbyte	[31:18]	[17:0]
1000	512 Kbyte	[31:19]	[18:0]
1001	1 Mbyte	[31:20]	[19:0]
1010	2 Mbyte	[31:21]	[20:0]
1011	4 Mbyte	[31:22]	[21:0]
1100	8 Mbyte	[31:23]	[22:0]
1101	16 Mbyte	[31:24]	[23:0]
1110	Reserved	—	—
1111	Reserved	—	—

35.2.3.5 M3IF Snooping Configuration Register 1 (M3IFSCFG1)

The M3IFSCFG1 register contains the enable bits for the lower 32 segments [31:0] in M3IFSCFG0 register. The Snooping feature is described in detail in Section 35.3.5.3, “Snooping Overview.” The field assignments for this register are shown in Figure 35-7 and the field descriptions are listed in Table 35-10.

Offset 0x002C (M3IFSCFG1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSE0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSE0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-7. M3IF Snooping Configuration Register 1 (M3IFSCFG1)

Table 35-10. M3IF Snooping Configuration Register 1 Field Descriptions

Field	Description
31–0 SSE0	Snooping Segment Enable 0. This register contains the enable bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). If snooping is enabled for segment #*x* (respective SSE0 bit is high), than any write access detected to that segment sets the DMA_ACCESS for one cycle, and the respective snooping status bit is set. If the SSE0 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register is set, but the DMA_ACCESS is be generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.

35.2.3.6 M3IF Snooping Configuration Register 2 (M3IFSCFG2)

M3IFSCFG2 register contains the enable bits for the upper 32 segments [63:32] in the M3IFSCFG0 register.The Snooping feature is described in detail in Section 35.3.5.3, “Snooping Overview.” The field assignments for this register are shown in Figure 35-8 and the field descriptions are listed in Table 35-11.

Offset 0x0030 (M3IFSCFG2) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSE1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSE1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-8. M3IF Snooping Configuration Register 2 (M3IFSCFG2)

Table 35-11. M3IF Snooping Configuration Register 2 Field Descriptions

Field	Description
31–0 SSE1	Snooping Segment Enable 1. This register contains the enable bits for the higher 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). If snooping is enabled for segment #x (respective SSE1 bit is high), than any write access detected to that segment sets the DMA_ACCESS for one cycle, and the respective snooping status bit is set. If the SSE1 bit is low, and a write access to the respective segment is detected by the M3IF, only the relevant status bit in the snooping status register is set but the DMA_ACCESS is not generated. 0 Snooping segment #x is disabled. 1 Snooping segment #x is enabled.

35.2.3.7 M3IF Snooping Status Register 0 (M3IFSSR0)

The M3IFSSR0 register contains the snooping status bits for the lower 32 segments. The Snooping feature is described in detail in Section 35.3.5.3, "Snooping Overview." The field assignments for this register are shown in Figure 35-9 and the field descriptions are listed in Table 35-12.

Offset 0x0034 (M3IFSSR0) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSS0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSS0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-9. M3IF Snooping Status Register 0 (M3IFSSR0)

Table 35-12. M3IF Snooping Status Register 0 Field Descriptions

Field	Description
31–0 SSS0	Snooping Segment Status 0. This register contains the snooping status bits for the lower 32 segments [31:0] in the snooping window (defined by the M3IFSCFG0 register). A bit in the SSS0 register is asserted if snooping to the respective segment occurred. **Note:** If snooping occurred the status bit is updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS is asserted only if the respective snooping segment enable bit SSE0[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x occurred.

35.2.3.8 M3IF Snooping Status Register 1 (M3IFSSR1)

The M3IFSSR1 register contains the snooping status bits for the upper 32 segments. The Snooping feature is described in detail in Section 35.3.5.3, "Snooping Overview." The field assignments for this register are shown in Figure 35-10 and the field descriptions are listed in Table 35-13.

Offset 0x0038 (M3IFSSR1) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SSS1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SSS1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-10. M3IF Snooping Status Register 1 (M3IFSSR1)

Table 35-13. M3IF Snooping Status Register 0 Field Descriptions

Field	Description
31–0 SSS1	SSS1 - Snooping Segment Status 1. This register contains the snooping status bits for the upper 32 segments [63:32] in the snooping window (defined by the M3IFSCFG1 register). A bit in the SSS1 register is asserted if snooping to the respective segment occurred. **Note:** If snooping occurred the status bit is updated regardless of the respective snooping segment enable bit SSE0[x]. The DMA_ACCESS is asserted only if the respective snooping segment enable bit SSE1[x] is enabled. 0 Snooping for segment #x did not occur. 1 Snooping for segment #x has occurred.

35.2.3.9 M3IF Master Lock WEIM CS*x* Register (M3IFMLWE*x*)

The field assignments for this register are shown in Figure 35-11 and the field descriptions are listed in Table 35-14.

Offset 0x0040 (M3IFMLWE0)
0x0044 (M3IFMLWE1)
0x0048 (M3IFMLWE2)
0x004C (M3IFMLWE3)
0x0050 (M3IFMLWE4)
0x0054 (M3IFMLWE5)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WEMAx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	MLWE*x*_EN	MLWE*x*		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 35-11. M3IF Lock WEIM CS*x* Register (M3IFMLWE*x*)

Table 35-14. M3IF Lock WEIM CS*x* Register Field Descriptions

Field	Description
31 WEMAx	WEIM CSx (0-5) Memory Active. This is a read-only status bit, that if set indicates that an active/pending access to a WEIM CSx memory exists. The WEIMx bit is set on one of the following conditions: • MLWEx_EN cleared - any active/pending access to WEIM CSx memory space sets the bit (until the access is completed). • MLWEx_EN is set - any accesses to WEIM CSx memory space initiated previously to MLWEx_EN assertion, keeps the WEMAx status bit set. The bit clears after all pending/active accesses execution is completed. Access from master number equal to MLWEx field does not assert the status bit. **Note:** When MLWEx_EN is set, any new accesses (initiated after MLWEx_EN assertion) to WEIM CSx memories (or to M3IFMLWEx register**)** not from MLWEx master are pending without setting WEMAx to 1. Both the M3IFMLWEx register and the WEIM CSx space region are locked to the MLWEx port. 0 No active/pending access to WEIM CSx memory exists. 1 Indicates an active/pending access to WEIM CSx memory exists.
30–4	Reserved

Table 35-14. M3IF Lock WEIM CS*x* Register Field Descriptions (continued)

Field	Description
3 MLWEx_EN	Master Lock WEIM CS*x* Access Enable. This bit enables the Master Lock WEIM CSx access (MLWEx). The reset value of this bit is 0. **Note:** After MLWEx master does not need the lock any more, the master should clear MLWEx_EN bit, so WEIM CSx memory region is open to all masters. 0 Master Lock WEIM CSx access (MLWEx) disabled. 1 Master Lock WEIM CSx access (MLWEx) enabled
2–0 MLWE*x*	MLWEx - Master Lock WEIM CS*x* Access. This 3-bit field defines the master port number (MPG) that is the only one in the system served by the WEIM controller. All accesses to the WEIM CS*x* memory space from the other masters are postponed. The reset value of the MLWE*x* is 0. 1. Prior to lock accesses, the MLGE master should perform the following steps: 2. Set the MLWEx_EN bit and the MLWEx field (with the desired value) in the M3IFMLWEx register. 3. Read M3IFMLWEx register and check: WEMAx status bit is cleared (no pending/active accesses to WEIM CSx memory space exists). MLWEx_EN bit is set. MLWEx (value) points to the required port number (master port number that requires lock accesses). 000 Master Port Gasket 0 001 Master Port Gasket 1 010 Master Port Gasket 2 011 Master Port Gasket 3 100 Master Port Gasket 4 101 Master Port Gasket 5 110 Master Port Gasket 6 111 Master Port Gasket 7

35.3 Functional Description

This section provides the functional description for the M3IF module.

35.3.1 Master Port Gasket (MPG)

The MPG is a flexible port gasket. Up to 8 masters can be connected to the M3IF with any combination of the following MPG port types:

- MPG: Master port gasket for ARM11 AMBA AHB-Lite 32-bit data bus.
- MPG64: Master port gasket AMBA AHB-Lite 64-bit data bus.

The number and type of ports in use is system-dependent, and the unused ports are disconnected at the system level. Each of the MPG gaskets is assigned a single port type, and communicates with a single master.

35.3.1.1 Overview of MPG Operation

The MPG port gasket is used for those system masters that are 32-bit ARM11 AHB-Lite bus-compatible. The MPG port gasket appears as a slave to any master it connects to. Table 35-15 lists the access types supported by the MPG.

Table 35-15. MPG Supported Burst Accesses

HBURST	TYPE	M3IF SLAVES		
		ESDRAMC 32-Bit	WEIM 32-Bit	NFC[1] 16/32-Bit
000	SINGLE	YES[2]	YES	YES
001	INCR	YES	YES	YES
010	WRAP 4	YES	YES	NO
011	INCR 4	YES	YES	YES
100	WRAP 8	YES	YES	NO
101	INCR 8	YES	YES	YES
110	WRAP 16	NO	YES	NO
111	INCR 16	NO	YES	YES

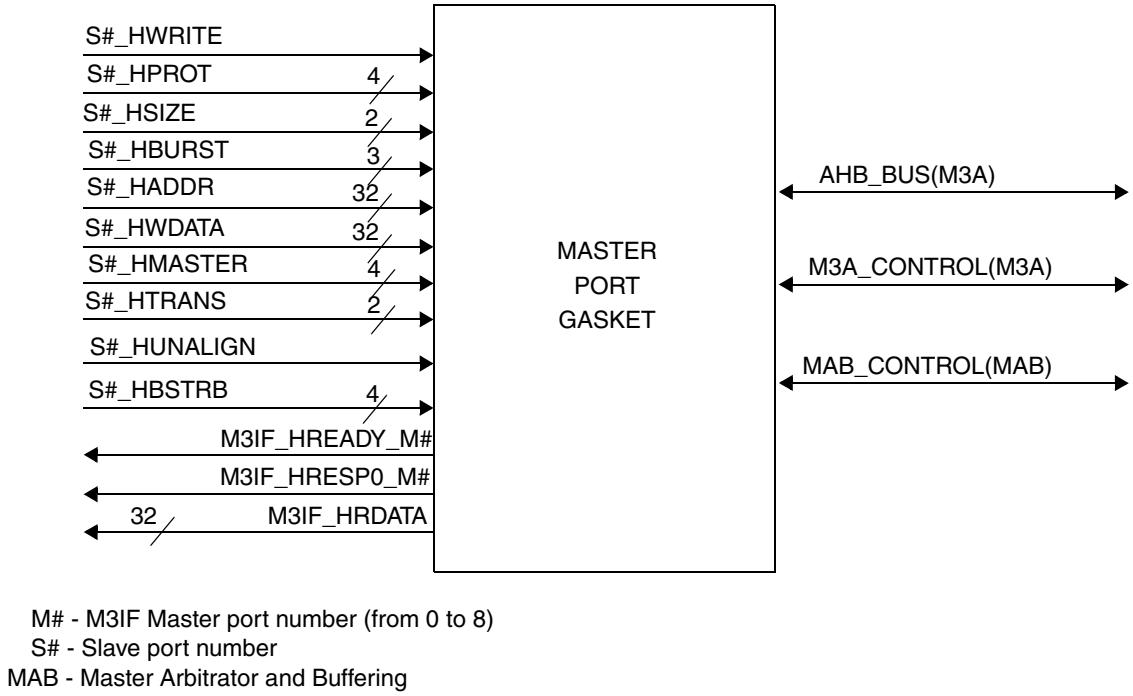
1 NFC does not support accesses of 8-bit data width.

2 ESDRAMC and WEIM supports only word-size bursts (32 bits). Since SINGLE access is not a burst type access, byte (8 bits) or half -word (16 bits) is supported as well.

NOTE

Unsupported access types produce undefined behavior, but an error response is not generated.

Figure 35-12 shows the MPG port interface diagram. The interface is ARM 11 AMBA AHB-Lite-compatible (does not support RETRY and SPLIT transfers).



M# - M3IF Master port number (from 0 to 8)
S# - Slave port number
MAB - Master Arbitrator and Buffering

Figure 35-12. Master Port Gasket (MPG) Interface Diagram

The MPG works with both M3A and MAB, and outputs AHB_BUS and CONTROL signals to/from M3A (including requests to ESDRAMC and other controllers). The MPG decodes AHB bus inputs, converts them to pass through the MAB_CONTROL bus as address, data, and control signals (for instance suspend and abort commands).

Once an accesses is initiated by one of the M3IF masters, the access reaches the respective MPG. The MPG asserts the request signal toward the M3A which starts the arbitration process. Once the arbitration is completed and the request can gain access to the bus, the request is accepted by the MPG and the handshake between the MPG and the M3A is completed for that access. If the access was not targeted toward the ESDRAMC, the master can start the access (by passing the master AHB bus) toward the respective slave (NFC, WEIM).

If the initiated access is targeted to the ESDRAMC after the M3A arbitration process is completed, the request is transferred toward the MAB, which arbitrates and schedules the access toward the ESDRAMC as a function of ESDRAMC state. An internal handshake between the MAB and ESDRAMC is used to schedule the new access, and once the handshake is completed, the MAB asserts the request accept signal toward the MPG. All ESDRAMC-related AHB signals are transferred from the master to the MPG which convert them to an internal protocol between the MPG and the MAB.

The MPG also converts MAB or M3A outputs to the AHB standard interface. Table 35-16 presents the signals name in both modules.

Table 35-16. MPG MAX Signals

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HWRITE (o)	M3IF_HWRITE_M# (I)	HWRITE high indicates a write transfer. HWRITE low indicates a read transfer.
S#_HPROT[3:0] (O)	M3IF_HPROT_M#[3:0] (I)	The protection control signals provide additional information about a bus access. For more information on this signal see Protection paragraph at the AHB specification. M3IF is using only HPROT[1] signal - user/supervisor access.This signal is used to protect both registers and restricted memory regions. An error response is generated in case of protection violation—for example, access supervisor registers/memory regions in user mode.
S#_HSIZE[1:0] (O)	M3IF_HSIZE_M#[1:0] (I)	Indicates the size of the transfer 00 8-bits (Byte) 01 16-bits (Half-word) 10 32-bits (Word) 11 Not define for MPG
S#_HMASTER[3:0] (O)	not defined	Not used by M3IF.
S#_HBURST[2:0] (O)	M3IF_HBURST_M#[2:0] (I)	Burst information is provided using HBURST signal, and the 8 possible types are: 000 SINGLE (Single transfer) 001 INCR (Incrementing burst of unspecified length) 010 WRAP4 (4-beat wrapping burst) 011 INCR4 (4-beat incrementing burst) 100 WRAP8 (8-beat wrapping burst) 101 INCR8 (8-beat incrementing burst) 110 WRAP16 (16-beat wrapping burst)[1] 111 INCR16 (16-beat incrementing burst)[1]
S#_HADDR[31:0] (O)	M3IF_HADDR_M#[31:0] (I)	Indicates the 32 bits memory ADDRESS bus.
S#_HWDATA[31:0] (O)	M3IF_HWDATA_M#[31:0] (I)	The write data bus is driven by the master during write transfers (on data phase). If the transfer is extended then the bus master hold the data valid until the transfer completes, as indicated by HREADY HIGH.

Table 35-16. MPG MAX Signals (continued)

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HTRANS[1:0] (O)	M3IF_HTRANS_M#[1:0] (I)	Each transfer can be classified into one of four different types, as indicated by the HTRANS[1:0] signals: 00 IDLE. Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. M3IF provides a zero wait state OKAY response to IDLE transfers. 01 BUSY. The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. M3IF provides a zero wait state OKAY response to IDLE transfers. When a master uses the BUSY transfer type the address and control signals reflects the next transfer in the burst. 10 NONSEQ. Indicates the first transfer of a burst or a single transfer. Single transfers on the bus are treated as bursts of one and therefore the transfer type is NONSEQUENTIAL. 11 SEQ. The remaining transfers in a burst are SEQUENTIAL and the address and control are related to the previous transfer. In the case of a wrapping burst the address of the transfer wraps at the boundary equal to the size (in bytes) multiplied by the number of beats in the transfer (4,8 or 16).
S#_HBSTRB[3:0] (O)	M3IF_HBSTRB_M#[3:0] (I)	Indicates which byte lanes are valid for each word transfer.[2]
S#_HUNALIGN (O)	M3IF_HUNALIGN (I)	Signal to indicate an unalign access requiring HBSTRB information.[2]
S#_HMASTLOCK (O)	M3IF_HMASTLOCK (I)	Indicates that the current master is performing a locked sequence of transfers.
S#_HREADY (I)	M3IF_HREADY_M# (O)	M3IF uses HREADY signal to insert the appropriate number of wait states in to the transfer (the M3IF adds wait states as long as the HREADY in signal is negated). The transfer completes with HREADY HIGH (and an OKAY response, which indicates the successful completion of the transfer). One wait state is added for every cycle that has HREADY negated.

Table 35-16. MPG MAX Signals (continued)

AHB Master—Signal Name	MPG—Signal Name	Description
S#_HRESP0 (I)	M3IF_HRESP0_M# (O)	The (HRESP0) response is used by the M3IF to indicate some form of error condition with the associated transfer. Since M3IF is AHB Lite-compatible (AHB SPLIT and RETRY protocols are not supported) means that only one response signal is needed. HREPS0 encoding is: 0 OKAY. When HREADY is HIGH this shows the transfer has completed successfully. The OKAY response is also used for any additional cycles that are inserted, with HREADY LOW. 1 ERROR. This (two cycle) response shows an error has occurred. The error condition is signalled to the bus master so it is aware the transfer has been unsuccessful. M3IF response with Error on cases as specified in Section 35.3.1.4, “MPG Transfer Response,” on page 1-40.
S#_HRDATA[31:0] (I)	M3IF_HRDATA[31:0] (O)	The read data bus is driven by the M3IF during read transfers. If M3IF extends the read transfer by holding HREADY low, then M3IF provides valid data at the end of the final cycle of the transfer, as indicated by HREADY high.

1 INCR16/WRAP16 are supported only for accesses addressed to the WEIM.

2 HUANLIGN and HBSTRB are supported only by ESDRAMC and WEIM.

A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provides information on the address, direction and width of the transfer, as well as indication if the transfer forms parts of a burst. Two different forms of burst transfers are allowed:

- Incrementing bursts, which do not wrap at address boundaries.
- Wrapping bursts, which wrap at particular address boundaries.

A write data bus is used to move data from the master to M3IF, while read data bus is used to move data from M3IF to the master.

Every transfer consists of:

- An address and control cycle (address phase)
- One or more cycles for the data (data phase)

Since the first address phase cannot be extended (since it always get HREADY asserted high) M3IF samples all control bus during first address phase, so if the master does not gain access immediately, the address phase information is saved. The data, however, can be extended by using M3IF_HREADY_MX signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for M3IF (ESDRAMC or memories) to provide or sample data. In this way, back to back access between different/same slave can be performed and MPG stores all needed bus/signals so that when the master gains access, all the needed bus/signals are available.

During a transfer the M3IF shows the status using only one response signal HRESP0 (since M3IF is only AHB Lit-compatible),

- 0 - OKAY—The OKAY response is used to indicate that the transfer is progressing normally and when M3IF_HREADY_MX goes high this shows the transfer has completed successfully.

- 1 - ERROR—The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.

35.3.1.2 MPG Basic Transfer

An AMBA AHB transfer consists of two distinct sections:

- The address phase.
- The data phase, which may require several cycles. This is achieved using the M3IF_HREADY_MX signal.

Figure 35-13 shows the simplest transfer, one data with no wait states.

- The AHB lite bus-compatible master drives the address and control signals onto the bus after the rising edge of the clock.
- M3IF then samples the address and control information in the next rising edge of the clock and access starts (memory is not busy).
- After M3IF has sampled the address and control (and derived the appropriate command to the memory) it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

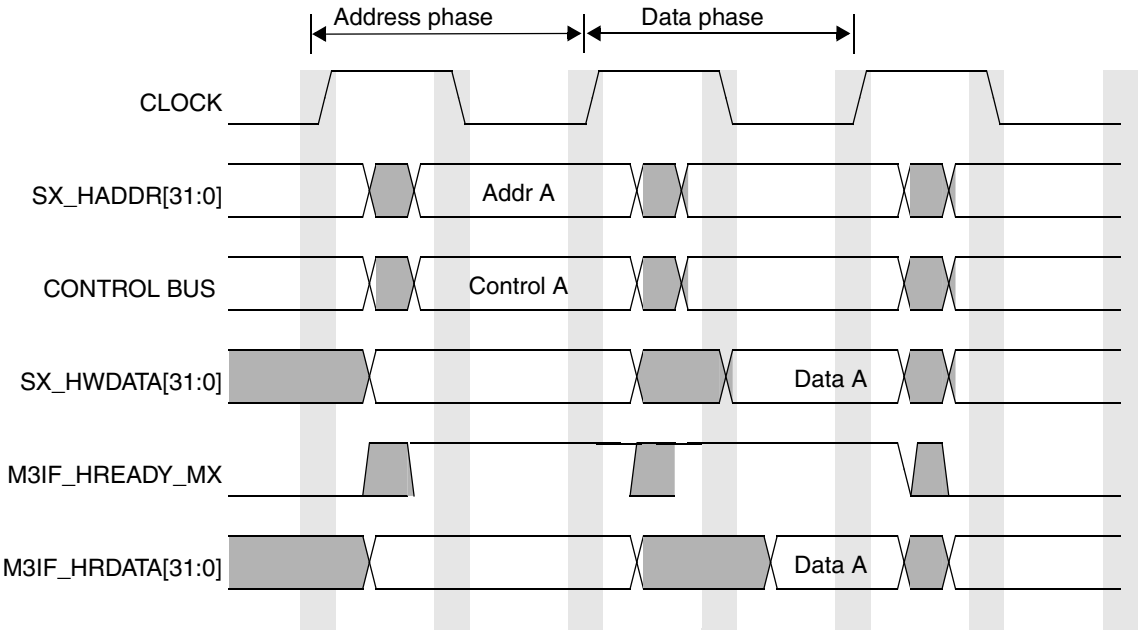


Figure 35-13. MPG Simple Transfer

The address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is at the pipelined nature of the AHB bus and allows for high performance operation.

M3IF may insert wait states into any transfer, as shown in Figure 35-14, which extends the transfer allowing additional time for completion.

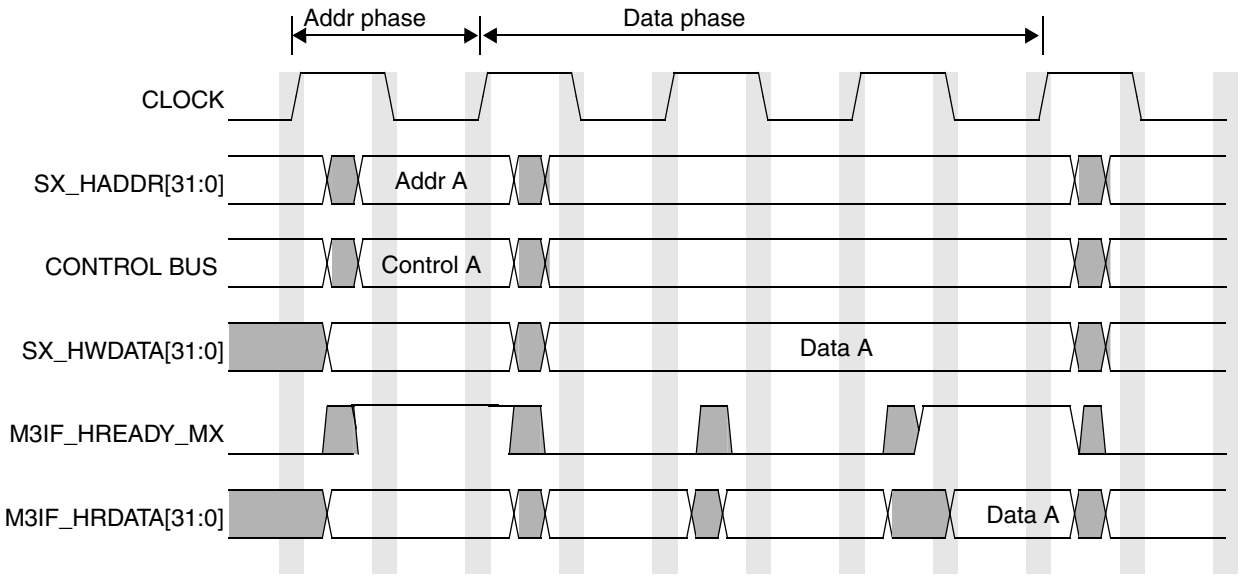


Figure 35-14. MPG with Wait States

- For write operations the bus master holds the data stable throughout the extended cycles.
- For read transfer the M3IF does not have to provide valid data until the transfer is about to complete.

When a transfer is extended in this way it has the side effect of extending the address phase of the following transfer. This is illustrated in Figure 35-15, which shows three transfers to unrelated addresses, A, B, and C.

- The transfers to addresses A and C are both zero state.
- The transfer to address B is one wait state.

Extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C.

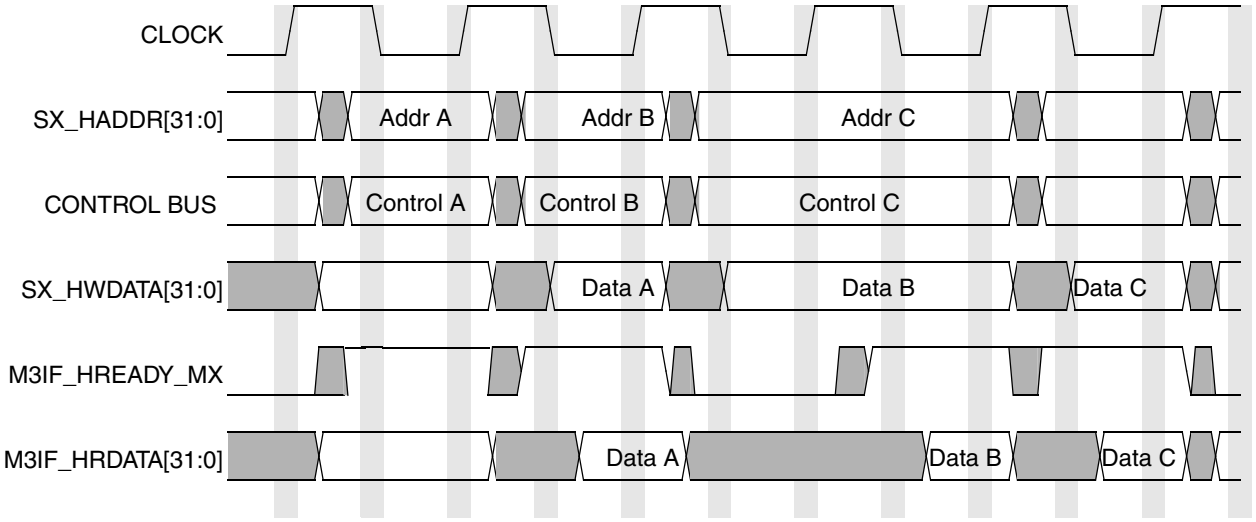


Figure 35-15. MPG Multiple Transfers

35.3.1.3 MPG Transfer Type

Every transfer can be classified into one of four different types, as indicated by SX_HTRANS[1:0] signals as described in Table 35-16. Figure 35-16 shows a number of different transfer types being used.

- The first transfer is the start of a burst and therefore is NON-SEQUENTIAL.
- The master is unable to perform the second transfer of the burst immediately and therefore the master uses BUSY transfer to delay the start of the next transfer (after M3IF sees BUSY with HREADY high it continues to give HREADY high until HTRANS bus changes from BUSY and then HREADY acts as usual). In this example the master requires only one cycle before it is ready to start the next transfer in the burst, which completes with no wait states.
- The master performs the third transfer of the burst immediately, but this time the M3IF is unable to complete and uses M3IF_HREADY_MX to insert a single wait state.
- The final transfer of the burst completes with zero wait states.

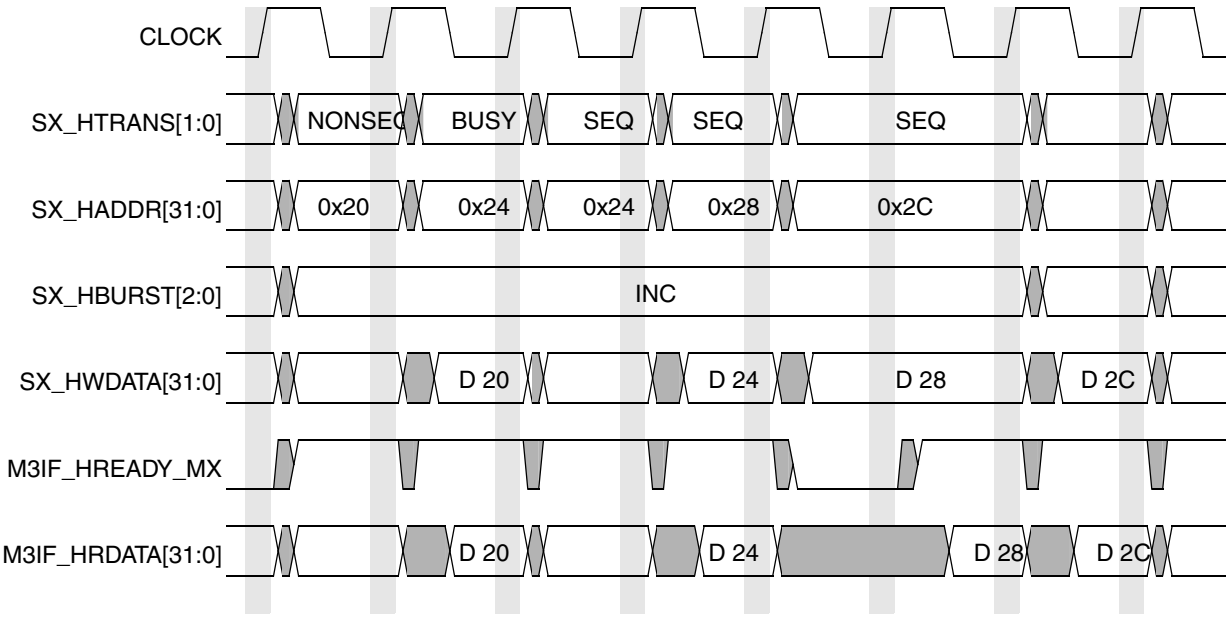


Figure 35-16. MPG - Transfer Type Examples

35.3.1.4 MPG Transfer Response

Whenever M3IF is accessed it provides a response which indicates the status of the transfer. The M3IF_HREADY_MX signal is used to extend the transfer and this works in combination with the response signals, M3IF_HRESP_MX, which provide the status of the transfer. M3IF can complete the transfer in a number of ways:

- Complete the transfer immediately.
- Insert one or more wait states to allow time to complete the transfer.
- Signal error to indicate that the transfer has failed.

The M3IF_HREADY_MX signal is used to extend the data portion/phase of a transfer. When LOW the M3IF_HREADY_MX indicates the transfer is to be extended and when HIGH indicates that data transfer had completed. Both M3IF_HREADY_MX and M3IF_HRESP0 encoding is described in Figure 35-13. It should be noted that M3IF does not support AMBA AHB, SPLIT and RETRY transfer response.

A transfer completes successfully (as defined by the AHB bus protocol) with M3IF_HREADY_MX HIGH and an OKAY response (M3IF_HRESP[1] LOW). A transfer completes unsuccessfully (ERROR response) with two consecutive cycles of M3IF_HRESP[1] HIGH, while during the first cycle M3IF_HREADY_MX is LOW and during the second cycle M3IF_HREADY_MX is HIGH (as defined by the AHB bus protocol). The ERROR response is used by the M3IF to indicate one of the following error types (which can be associated with the transfer):

- Master is trying to access a disabled CSD in the ESDRAMC system register.
- Master in user mode is trying to access a CSD that is configured to SUPERVISOR access only.
- System gave software reset command to ESDRAMC while access to ESDRAMC is in progress.
- A non predefined master that accesses a region that is marked as a Watermark region.

- Error response coming from all other memory controllers (except ESDRAMC).
- Access to ESDRAMC registers during an active access to SDRAM memory.

NOTE

The M3IF controls/handles ESDRAMC error response logic, and only transfer the error response signal from all other memory controllers (NFC and WEIM). For more details regarding the error response generation from the other memory controllers, consult the respective memory controller specification document available in the respective system architecture.

If an error response is generated by the MPG on the beginning of an access, the access is not executed and none of the data that is supposed to be read/written is transferred. However, if the error response has been given after few data transfers (in a burst access), the status of the first data transfer before the error response, for write access, is unknown (data maybe written or not) and the master should treat the data of the whole access as unknown data. In the case that this access was a read access the data that has been transferred until the error response is valid data and master can use it. If an error occurs during a burst access, the M3IF generates an (AHB) error response for all remaining beats from the burst.

35.3.1.5 MPG Burst Operation

Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as incremental undefined length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol. A detailed description of the supported access type by the MPG is shown at Table 35-15.

Burst information is provided using SX_HBURST[2:0] signal and the eight possible types are defined in Figure 35-13. It is acceptable to perform single transfers using an unspecified length incrementing burst which only has a burst length of one.

The burst size indicates the number of beats in the burst, not the number of bytes transferred. The total amount of data transferred in a burst is calculated by multiplying the number of beats by the amount of data in each beat, as indicated by SX_HSIZE[1:0]. SX_HSIZE[1:0] encoding is shown in Figure 35-13. The size is used in conjunction with the SX_HBURST[2:0] signals to determine the address boundary for wrapping bursts.

All transfers within a burst must be aligned to the address boundary equal to the size of the transfer (that must be a word as mentioned). For example, word transfers must be aligned to word address boundaries (that is A[1:0]=00). If an unalign access is being perform HUNALIGN signal must be asserted high and the respective HBSTRB bus must be given by the master.

NOTE

Unaligned burst crossing bus width boundary is supported only if the eventual number of transfers on the bus is not higher than the value implied by the HBURST.

Four-beat wrapping and incrementing burst are shown in Figure 35-17 and Figure 35-18 respectively.

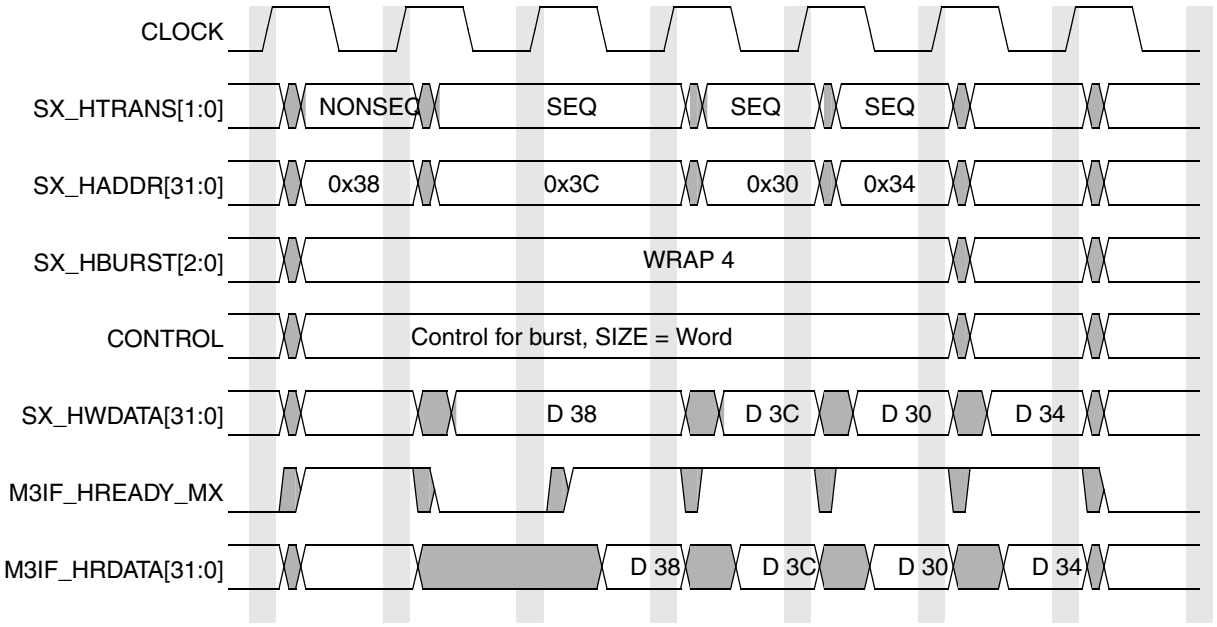


Figure 35-17. MPG Four Beat Wrapping Burst

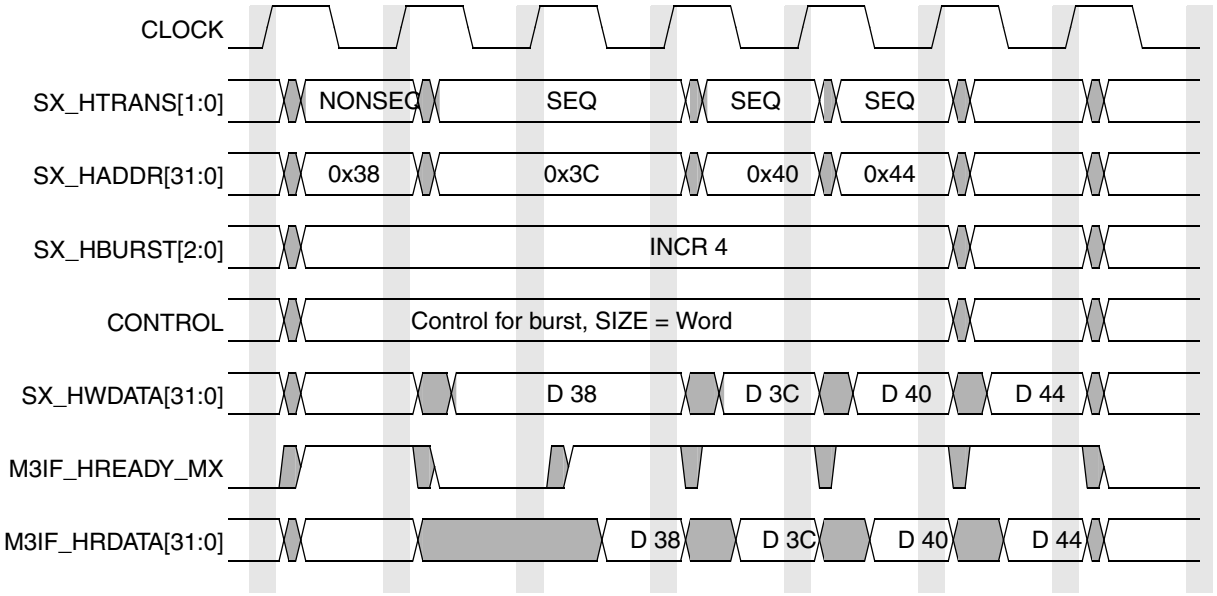


Figure 35-18. MPG Four Beat Incrementing Burst

35.3.1.6 MPG Early Burst Termination

M3IF can determine when a burst has terminated early by monitoring the SX_HTRANS[1:0] signals and ensuring that after the start of the burst every transfer is labelled as SEQUENTIAL or BUSY. If a

NON-SEQUENTIAL transfer occurs in middle of a burst it indicates that a new burst has started and therefore the previous one must be terminated immediately. If an IDLE transfer occurs in middle of a burst, it indicates the burst should be terminated immediately.

If a master cannot complete a burst because it loses ownership of the bus (for example, MAX slave port SX_HTRANS[1:0] is IDLE during a burst access due to MAX internal arbitration logic, means that the served MAX master port loses ownership of the bus) then it must rebuild the burst appropriately when it re-gains access to the bus. For example, if a master has only completed one beat of a four-beat burst then it must use an undefined-length burst to perform the remaining three transfers. Figure 35-19 shows incrementing bursts of undefined length that starts after aborting previous INCR 4 burst access.

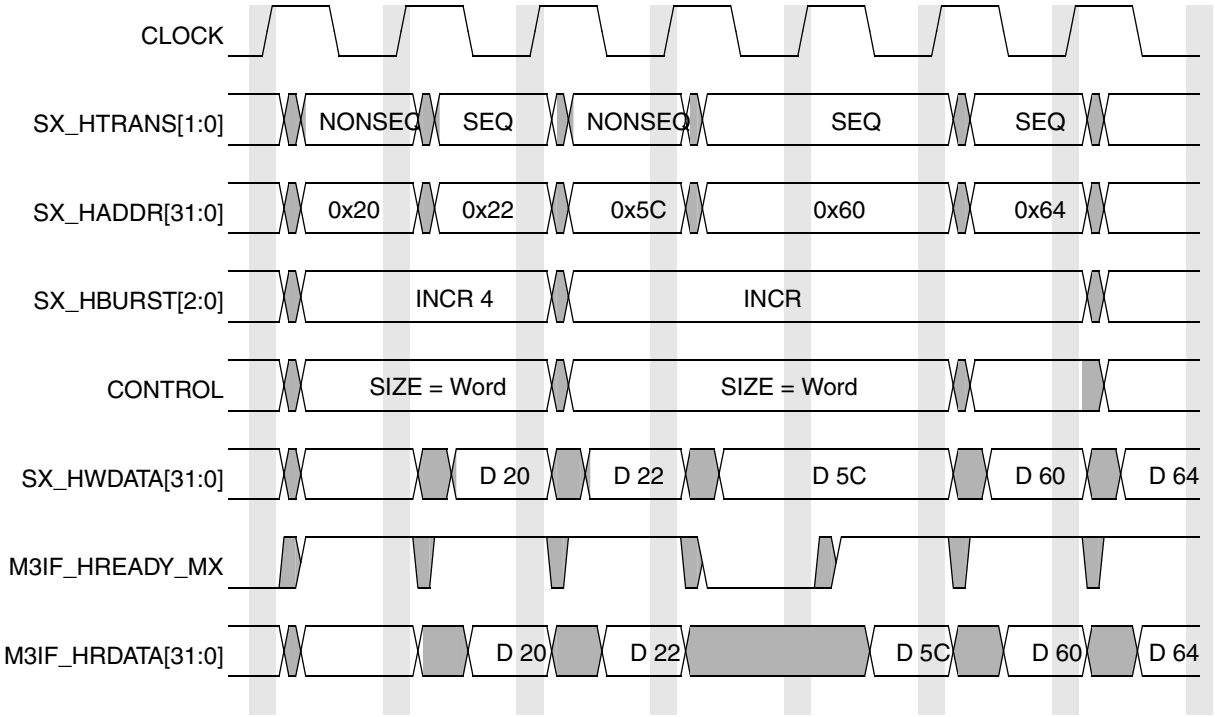


Figure 35-19. MPG Undefined Length Bursts

NOTE

To perform burst access length not equal to 4 or 8 words, it is possible to start INCR access of undefined length and to abort it after the desired words, or to start 4/8 burst length access and to abort it after the desired word., Both ways are supported by the M3IF and it is the master's decision which way to choose.

35.3.1.7 Multi-Endian Byte Order

M3IF supports multi-endian byte order. For example, there is an endian signal input to each one of the MPGs. The endian signal from each master should be static after reset, which means that all accesses from each master have the same byte order. If, in a given system, there are masters connected to the M3IF that do not drive endian signals (that is, they support only one endian type, big or little) the respective MPGs

big-endian signal should be static, meaning connected to 0 or 1 (depends on the byte order supported by the master connected to it). M3IF does NOT support shared external memory areas for masters with different endian modes. This feature should be handled by software or other additional hardware in the system.

35.3.2 Master Port Gasket 64 (MPG64)

35.3.2.1 Overview

The MPG64 port gasket is used for those system masters that have 64 bits data bus. Currently this gasket is used by the Layer2 Cache module. Table 35-17 presents the access types supported by the MPG.

Table 35-17. MPG64 Supported Burst Accesses

HBURST	TYPE	M3IF SLAVES					
		ESDRAMC		WEIM		NFC[1]	
		32 bit	64 bit	32 bit	64 bit	16/32 bit	64 bit
000	SINGLE	YES[2]	YES	YES[3]	YES	YES	YES
001	INCR	YES	YES	YES	YES	YES	YES
010	WRAP 4	YES	YES	YES	YES	NO	NO
011	INCR 4	YES	YES	YES	YES	YES	YES
100	WRAP 8	YES	NO	YES	NO	NO	NO
101	INCR 8	YES	YES	YES	YES	YES	YES
110	WRAP 16	NO	NO	YES	NO	NO	NO
111	INCR 16	NO	NO	YES	NO	YES	NO

1 NFC does not support accesses of 8 bit data width.

2 M3IF MPG64 supports only double word (64 bits) or word (32 bits) size bursts. Since SINGLE access is not a burst type access, byte (8 bits) or half word (16 bits) is supported as well.

NOTE

Unsupported accesses type cause undetermined behavior (that is, an error response is not generated).

For MPG64 brief overview description see Section 35.3.1.1, “Overview of MPG Operation.” Table 35-18 only shows the buses that have different widths, all other signals are the same as described in Table 35-17.

Table 35-18. MPG64 Additional Signals

64-Bit Master—Signal Name	MPG64—Signal Name	Description
S#_HBSTRB[7:0] (O)	M3IF_HBSTRB_M#[7:0] (I)	Indicates which byte lanes are valid. (each bit for each byte)—extended to 8 bits.[1]
S#_HWDATA[63:0] (o)	M3IF_HWDATA_M#[63:0] (I)	The write data bus extended to 64 bit width

Table 35-18. MPG64 Additional Signals (continued)

64-Bit Master—Signal Name	MPG64—Signal Name	Description
S#_HSIZE[1:0] (O)	M3IF_HSIZE_M#[1:0] (I)	Indicates the size of the transfer 00 - 8 bits (Byte) 01 - 16 bits (Halfword) 10 - 32 bits (Word) 11- 64 bits (Double-Word) (defined only for MPG64)
S#_HBURST[2:0] (O)	M3IF_HBURST_M#[2:0] (I)	Burst information is provided using HBURST signal, and the 8 possible types are; 000- SINGLE (Single transfer) 001- INCR (Incrementing burst of unspecified length) 010- WRAP4 (4-beat wrapping burst) 011 - INCR4 (4-beat incrementing burst) 100 - WRAP8 (8-beat wrapping burst) 101 - INCR8 (8-beat incrementing burst) 110 - WRAP16 (16-beat wrapping burst) 111 - INCR16 (16-beat incrementing burst)
S#_HRDATA[63:0] (I)	M3IF_HRDATA_M#[63:0] (O)	The read data bus extended to 64 bit width

1 HUANLIGN and HBSTRB are supported only by ESDRAMC and WEIM.

35.3.2.2 MPG64 Basic Transfer

All Basic transfer for 32 bits access are perform as AMBA-AHB usual access as described at Section 35.3.1.2, “MPG Basic Transfer.”

All 64 bits access are performed differently. Since the output data port is 32 bits wide, each 64 bit (double word) access is translated into 2 separated access, so a single read/write access of 64 bits is translated by the MPG64 into two single read/write 32 bits access. Burst length of 4 double words is being translated into 8 words (32 bits) burst length and 8 double words burst length is translated into 2 bursts of 8 words (32 bits) length. For 32-bit LPDDR there is no need for the MPG64 gasket to translate the access since 64 bits can be transferred by the LPDDR each cycle.

35.3.2.3 MPG64 Transfer Type

See Section 35.3.1.3, “MPG Transfer Type.”

35.3.2.4 MPG64 Transfer Response

See Section 35.3.1.4, “MPG Transfer Response.”

35.3.2.5 MPG64 Burst Operation

There are few things different than what is described in Section 35.3.1.5, “MPG Burst Operation.”

The size of each beat can be either 8, 16, 32 or 64-bits (byte/half word/word/double word), as shown in Table 35-17. If a burst access of double word is issued, WRAP8, INCR16 and WRAP16 are not supported by the M3IF.

35.3.2.6 MPG64 Early Burst Termination

See Section 35.3.1.6, “MPG Early Burst Termination.”

35.3.2.7 MPG64 Multi-Endian Byte Order

See Section 35.3.1.7, “Multi-Endian Byte Order.”

35.3.3 M3IF Arbitration (M3A)

35.3.3.1 Overview

The M3A is a programmable arbiter. All incoming requests from the different masters are on hold until access is granted by the arbiter. The arbitration is performed by a round robin algorithm which grants the access to the master that holds the token. In the case that a master holds the token but does not request access (to one of the M3IF slaves), the bus is granted to the nearest requesting master with a higher round robin number.

The internal signal bus free indicates the FF1 algorithm to choose a new master. The bus_free signal is asserted high by the M3A by monitoring HTRANS bus of the active master (the master that grants access). As soon as the M3A notices that the access has been accomplished (HTRANS equal to NONSEQ or IDLE with HREADY asserted high) it allows to a new master to gain access according to the round robin value. When a new master gains an access, the M3A transfers the AHB bus coming from this master to all memory controllers with an hsel signal to the specific memory controller (all others get low hsel).

Because MAB can get multiple access to the ESDRAMC and pass accesses to the ESDRAMC according to internal handshake between the ESDRAMC and the MAB, the M3A allow multiple access to pass to the MAB without waiting for previous accesses to be completed.

The M3A passes the request to the MAB if the access is to the ESDRAMC, and passes a new access to the MAB (before the previous/active master access is completed) if the master with the token is accessing the ESDRAMC If the request is for a different memory controller the M3A holds the access until all pending transfers in the MAB (ESDRAMC accesses) are finished, after which the bus_free signal asserts high to indicate the MAB has completed all incoming requests.

There are two different access paths, as follows:

- Access requests to SDRAM/LPDDR/DDR2. These involve the ESDRAMC memory controller. The access path is as follows:
  a) Master #*x* initiates access to memory by asserting the M#_ESDCTL_REQ signal.
  b) M3A arbitrates the master request.
  c) After successful arbitration, M3A passes the request to MAB by asserting the MASTER_REQ_EN signal.
  d) MAB and the respective MPG (that initiated the access) handle the access directly from/to the ESDRAMC, without involving the M3A.
  e) All data transfer is accomplished by the ESDRAMC and the external memory.

- Access requests to non–ESDRAMC-controlled memories. These accesses involve the respective memory controller. The access path is as follows:
  a) Master #*x* initiates access to non–ESDRAMC-controlled memory, by asserting the M#_GENERAL_REQ signal.
  b) M3A arbitrates the master request.
  c) After successful arbitration M3A passes the AHB bus (address, data, control signals) of the respective master to the relevant memory controller.
  d) M3A and the respective MPG (the one that initiated the access) handle the access from/to the relevant memory controller (MAB is not involved).
  e) All data transfer is accomplished by the relevant memory controller and the external memory.

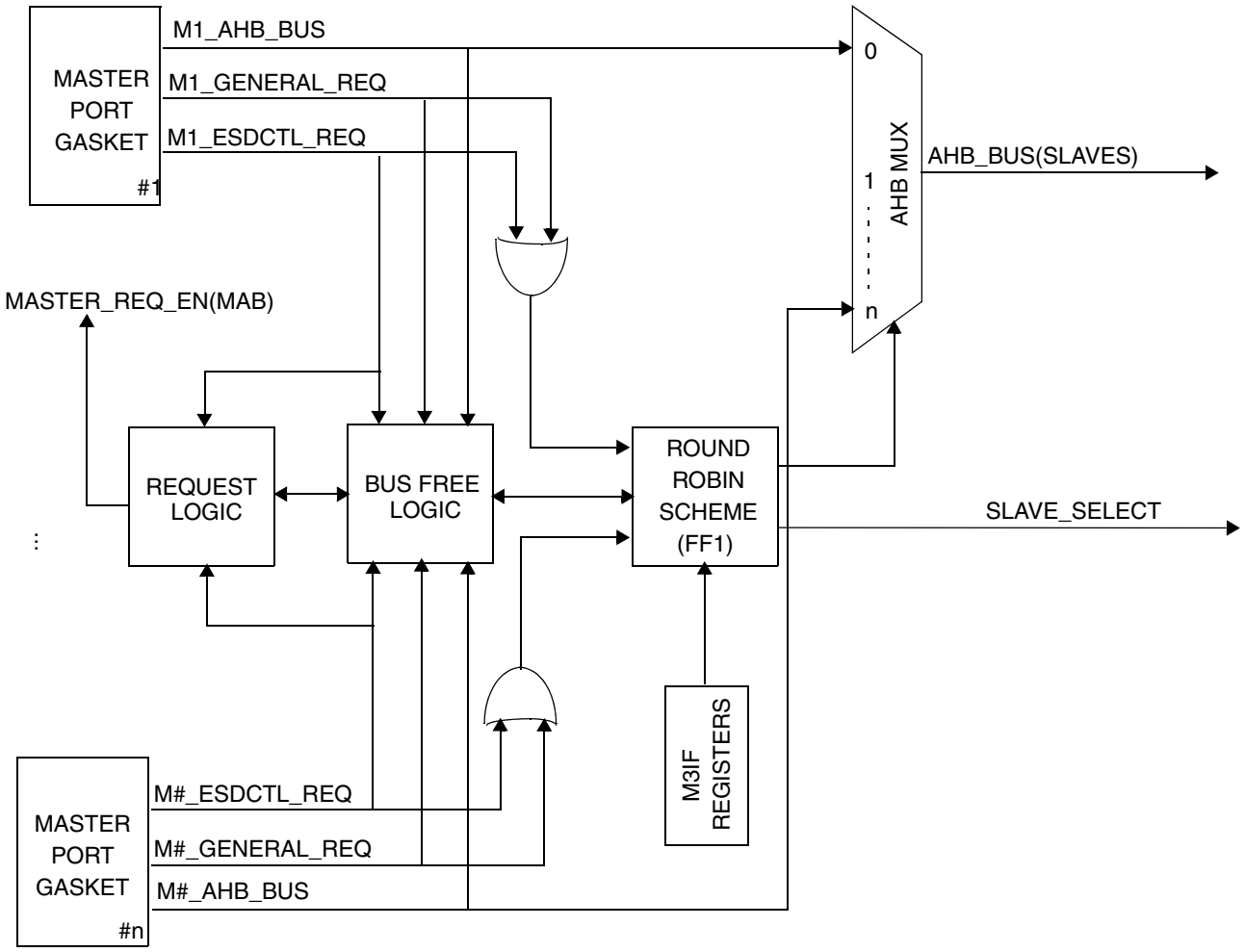


Figure 35-20. M3A Block Diagram

NOTE

Although the AHB bus indicates only one direction, the round robin scheme demultiplexes the AHB response signals (HREADY, HRESP, and HRDATA) from all slaves except for ESDRAMC (which directly to the MPG).

Figure 35-21 illustrates a simple transfer (all previous accesses are completed, and there are no other pending requests) between one of the M3IF masters and the WEIM module. There is one cycle penalty at the beginning of the access. The MPG samples the access relevant signals and de-asserts HREADY signal toward the master, until the target slave confirms the access (WEIM_HREADY high with WEIM_NONSEQ cycle). After the target slave (WEIM in this example) confirms the request (HREADY high with WEIM_NONSEQ cycle) the access traffic (control and data) is direct between the master and the target slave (WEIM).

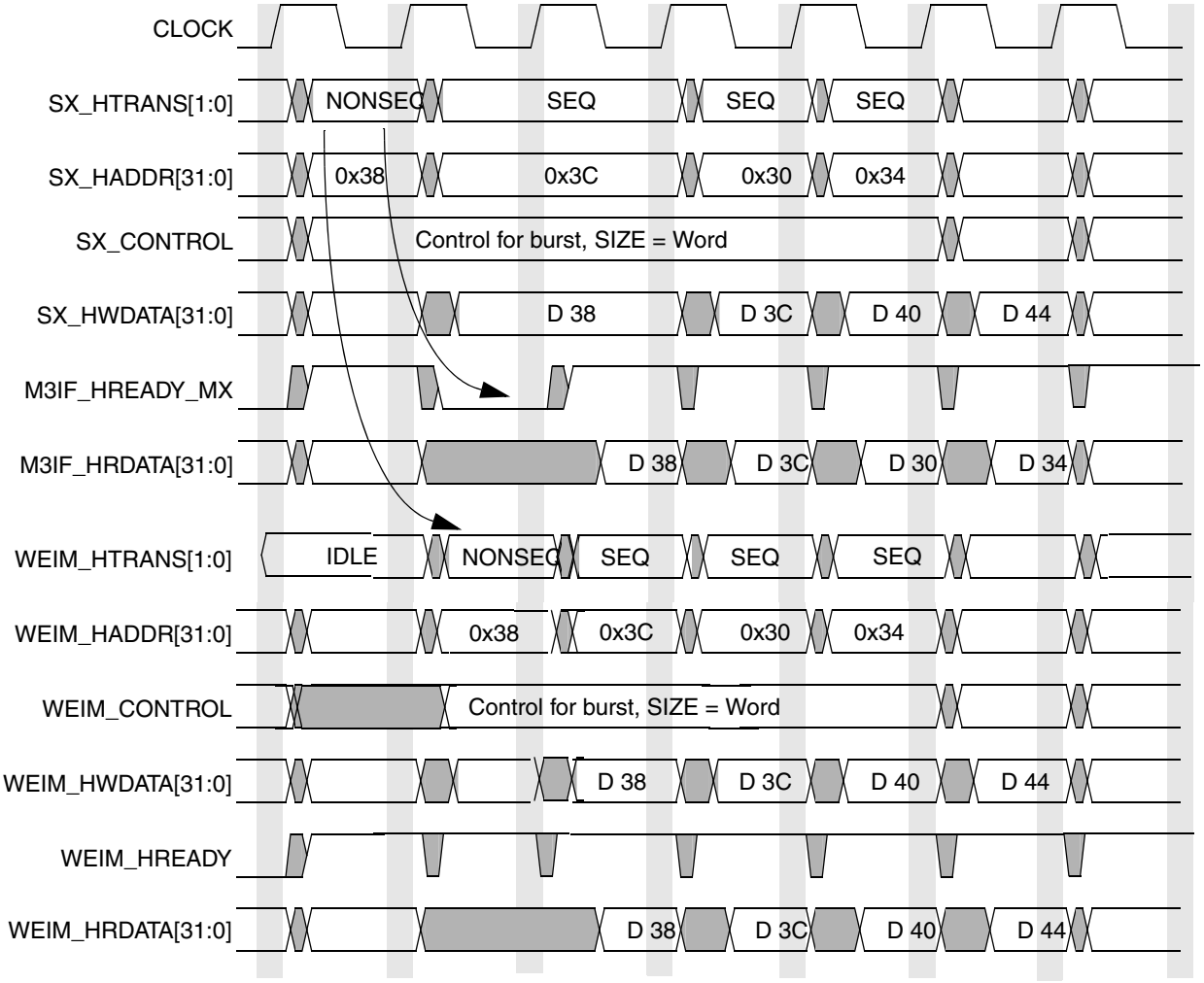


Figure 35-21. M3A Simple Transfer Timing Diagram

35.3.3.2 M3A—Find First 1 (FF1) Algorithm

Arbitration between the various requests is done by Find First 1 algorithm based on round robin algorithm (see Figure 35-22). If two or more masters request access to the M3IF the master with the token, or the master that is the first to receive the token (in case the master with the token does not request access) gains control over the AHB bus or enables his request signal to the MAB. For example, if masters 0, 1, and 6 requesting access at the same time and the token is at master 2, then master 6 gains control over the AHB

bus or his request signal to the MAB is enabled, since it is the first to receive the token (this is done to reduce arbitration time, in this example 4 clock cycles are saved). The round robin pointer increase its value in two cases, if the master with the token does not request access or if the master with the token requests access and gains the AHB bus/enable request - on the cycle that the master gains the AHB bus/enable request the round robin pointer increases its value. If a master requests an access and has the token but does not gain access because no new access can pass on, the round robin does not increase until the master gains the AHB bus/enable request.

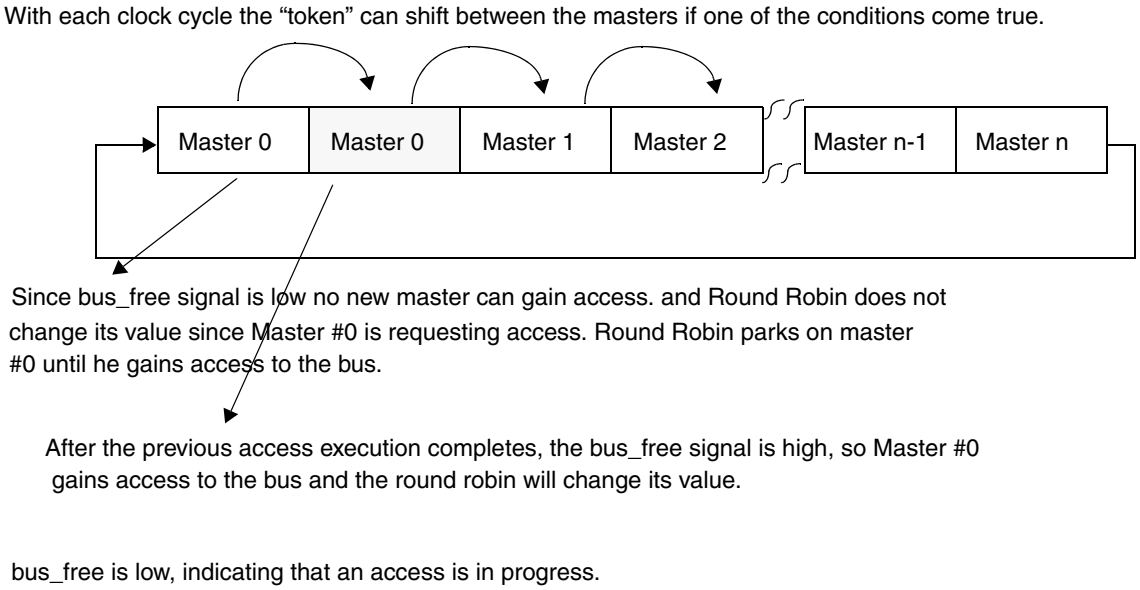


Figure 35-22. M3A—Round Robin Token Chain—Equal Priority

There is an option to program the round robin to work with different priority using MRRP field of the M3IF control register. When the MRRP equal to 0 no priority is given to any master so the probability to gain access is equal for each one of the masters.

If one or more bits of the MRRP are set to 1, the priority changes, and all master whose bits are set to 1 get together 50% priority of gaining access. For example, if both master 1 and master 4 bits set to 1 in the MRRP field, the priority to gain access of master 1 and 4 together is 50%. If only one bit is set the respective master (that his bit is set) alone has 50% priority to gain access.

NOTE

By default (after reset) all MRRP bits are cleared, means that all M3IF masters have the same priority.

The 50% priority refers only to the round robin mechanism. If a non priority master (MRRP respective bit is not set) with the token is already waiting to gain the bus the new coming request from the priority master (MRRP respective bit is set) does not gain access before the previous master request is completed. Additionally, the priority master cannot terminate an ongoing access of any other masters. Figure 35-23 shows the round robin chain in case that MRRP configured to 8’b00010010 - master 1 and 4 set to 1.

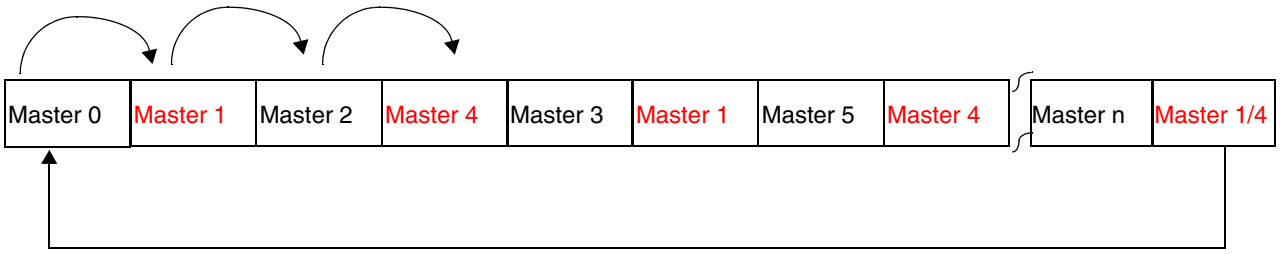


Figure 35-23. M3A—Round Robin Token Chain—Masters 1 and 4 has 50% Priority

35.3.3.3 bus_free Signal Algorithm

When the bus_free signal asserts high, it indicates the new master can gain access. The bus_free asserts high in the following cases:

- When the previous access is a non-ESDRAMC access, the HTRANS bus of the previous master that gained the access is equal to NON SEQUENTIAL or IDLE, and HREADY is asserted high.
- When the previous access is an ESDRAMC access, and the new master that gains access was also an ESDRAMC access.
- When the previous access or accesses are ESDRAMC accesses and all previous accesses are finished.

NOTE

To avoid contention between the memory controllers/memories, there is a special signal from the MPGs and from the MAB to the M3A that goes to the bus_free algorithm indicating which kind of slave is still using shared I/O pins, so no new access to a different memory begins.

35.3.4 Master Arbitration and Buffering (MAB)

35.3.4.1 Overview of MAB Operation

The MAB arbiter uses the same programmable arbiter as the M3A (Section 35.3.3.2, “M3A—Find First 1 (FF1) Algorithm”). All incoming requests from the different masters are put on hold until access is granted by the arbiter. The MAB communicates with the ESDRAMC and grants access at the earliest possible time, for example when the ESDRAMC is ready to handle a new memory request. Each time ESDRAMC can get a new access, the new access is sampled into the CONTROL and DATA buffers so ESDRAMC receives stable inputs during the time the access is in progress. The DATA buffer is sampled according to ESDRAMC write acknowledge response. Figure 35-24 shows MAB operation block diagram.

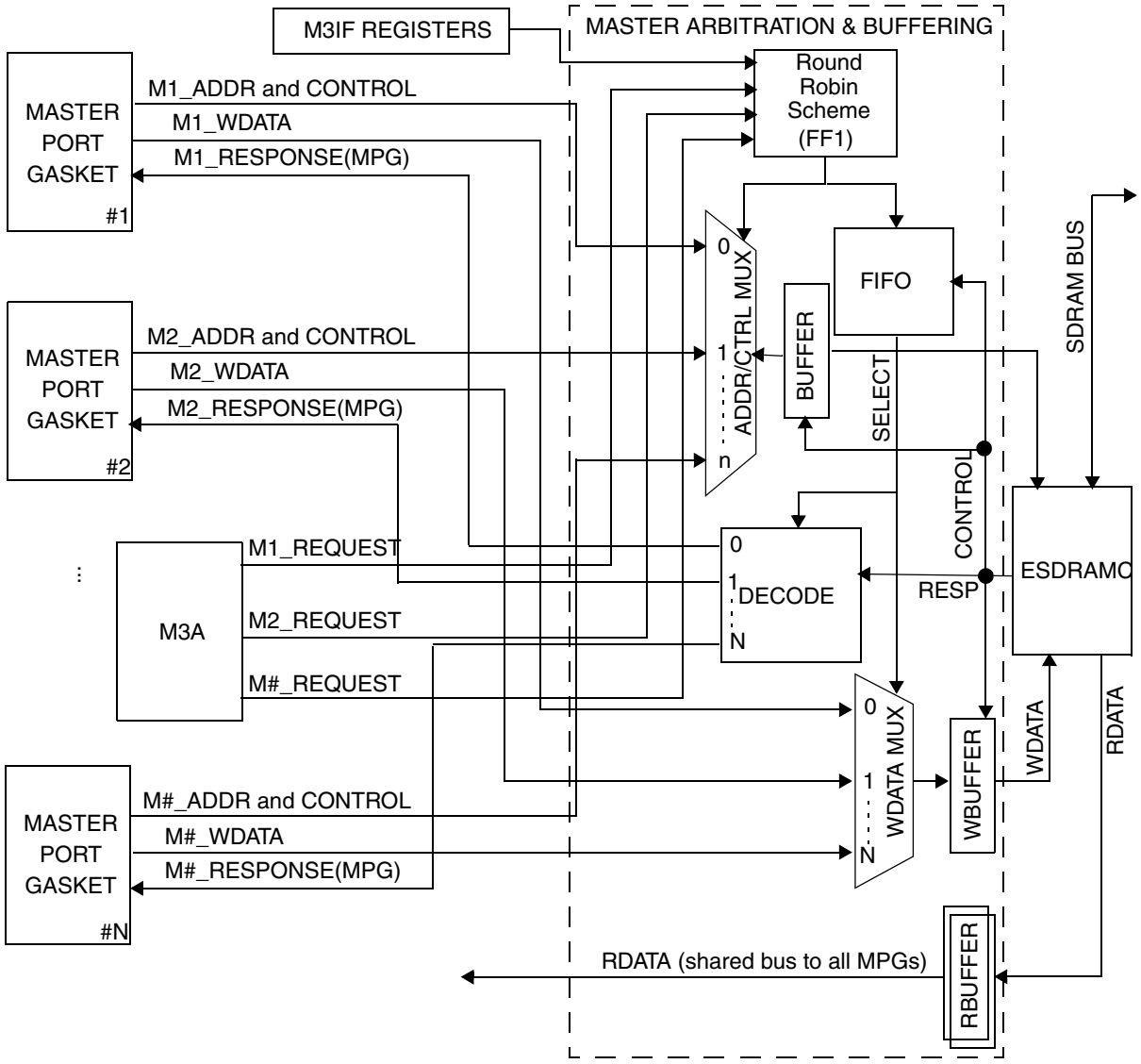


Figure 35-24. MAB Overview Block Diagram

35.3.4.2 M3B—Find First 1 (FF1) Algorithm

This operation of the arbiter algorithm is identical to the M3A arbiter algorithm described in Section 35.3.3.2, "M3A—Find First 1 (FF1) Algorithm."

Each time NEW_ACCESS goes HIGH it indicates that the EDRAMC is ready to handle a new memory request, meaning that the previous request is either completed or controlled by the SDRAM memory. During the same cycle the memory port is granted to the master with the token. Figure 35-25 shows the arbitration process for 4 master requests. At the first clock cycle masters M0, M1, and M2 simultaneously request the memory port. At the rising edge of the clock (while NEW_ACCESS is HIGH) master M0 has the token, so the memory port is controlled by M0 (see MASTER_CONTROL signals). During that time

all M3IF_HREADY_M# signals are low, besides M3IF_HREADY_M3 which was the previous served master.

The same arbitration process occurs for the following requests. Master M1 has the token and grant access to the ESDRAMC (see MASTER_CONTROL). HREADY of master M0 asserted high and accomplished the previous access. After several cycles M0 assert the REQUEST signal again due to a new access.

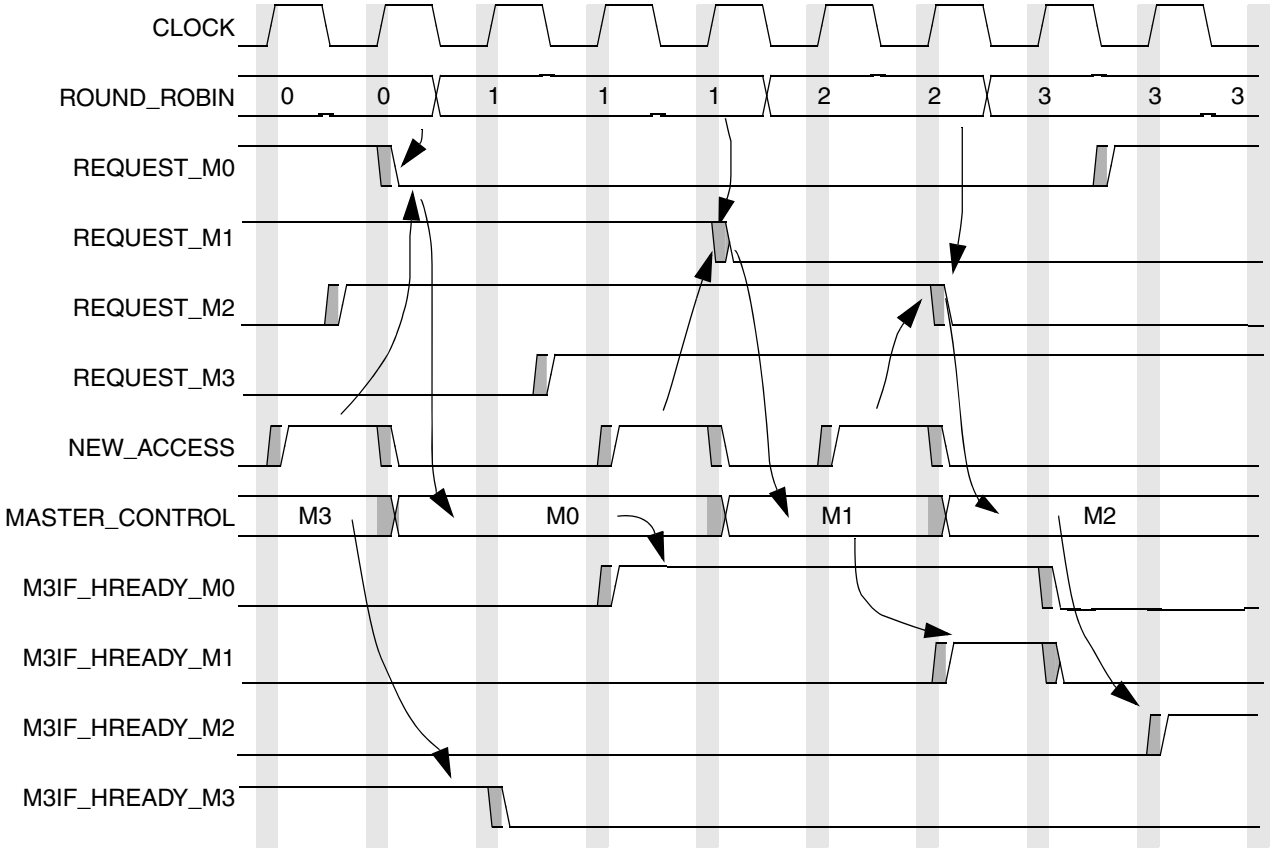


Figure 35-25. MAB Arbitration Process Time Diagram

Once a master has control over the port, the other requests remain on hold, meaning that their M3IF_HREADY_M# is LOW. The current master has control over the memory port until it completes the requested transfer. The new master gaining access to the memory port is the master with the new token.

The MAB ADDR/CTRL MUX and WDATA MUX connect (using the round robin algorithm) the selected master and when NEW_ACCESS arrives the MUX output is sampled by the MAB so that a stable bus is provided to ESDRAMC. After a new access is detected by the ESDRAMC, data transfer between the memory and the masters can start. In order for the MAB to serve more than one master at a time, a cyclic 4 entry FIFO with two pointers (read/write) is used. The FIFO read pointer is used (by the Decode block) as the selector for the memory RESPONSE signals and for the WDATA MUX, while the FIFO write pointer is used to add a new master to the FIFO entries. (since ESDRAMC hides latency a new access can start before previous access ended. This why this MUX control should work separately for information to the ESDRAMC and ESDRAMC response.

Figure 35-26 shows a detailed multi master memory request time diagram. The diagram shows two masters presenting memory request (AMBA AHB) signals and their conversion by the MPG and MAB to the ESDRAMC. The HRDATA timing is also shown with the respective M3IF_HREADY_M# signal. The HRDATA bus from the memory (during READ transfers) is shared with all present masters (except 64-bit masters that become a 64-bit bus after decoding by MPG64), while the arbitration is completed by the use of the M3IF_HREADY_M# signals at the master level.

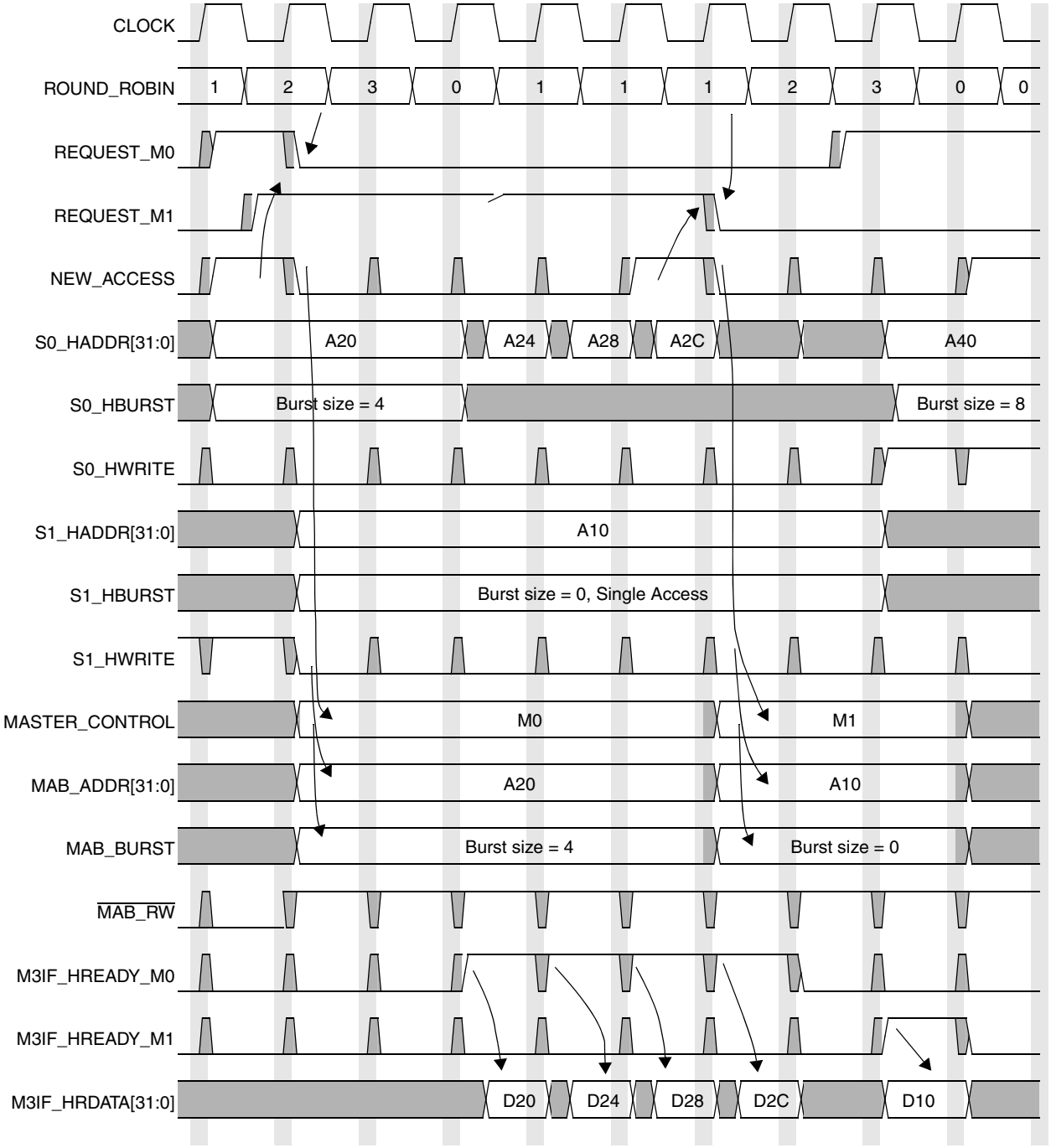


Figure 35-26. MAB Multi-Master Request Time Diagram

35.3.4.3 M3IF Operation During HMASTLOCK Accesses

If a HMASTLOCK access to any memory controller (memory space) is initiated by one of the M3IF masters, the request need to pass the arbitration like a regular access. After the access passes the arbitration it “locks” the arbitration and all other accesses (regardless to memory space destination) will pend until the HMASTLOCK signal negates (from the master that initiated the HMASTLOCK access).

While HMASTLOCK is asserted, all accesses initiated by the locking master are executed without arbitration, while all other masters accesses will pend.

NOTE

During HMASTLOCK high, the locking master is not allowed to change the memory space destination from SDR/DDR SDRAM space to non-SDR/DDR SDRAM, or vice versa.

35.3.4.4 DVFS Protocol

When CCM needs to perform a frequency change (due to the DVFS algorithm) the CCM asserts the dvfs_req signal. After dvfs_req assertion, the M3IF blocks all new accesses to the ESDRAMC (accesses to other memory controllers are not affected). More on, the M3IF (MAB) requests the ESDRAMC to place the external SDR/LPDD R SDRAM memory into self-refresh mode.

Once, all active accesses to the ESDRAMC (prior to dvfs_req assertion) are completed and the external SDR/LPDDR/DDR2 SDRAM memory is in self refresh mode the M3IF asserts the dvfs_grant signal, which indicates the CCM that the EMI is ready for the frequency change.

After the frequency change is completed, the CCM negates the dvfs_req. After dvfs_req negation the M3IF signals to the ESDRAMC to exit from self refresh mode, and all pending accesses to the ESDRAMC resume.

NOTE

Accesses to non- ESDRAMC memory controllers are not affected during the MAB DVFS protocol.

35.3.5 Watermark and Snooping Logic

35.3.5.1 Watermark in a System

M3IF has watermark capability. However, not all chips require watermarks. Therefore, in some chips the watermark pins of the M3IF are not connected and are assigned with a constant value to disable this capability. See the design specification to determine whether M3IF should have a watermark enabled or not.

35.3.5.2 Watermark Overview

The watermark is a security feature, that is used to protect a configurable memory region (watermark space) for up to 8 predefined different CS. The access to the watermark space is permitted only to one or more preselected M3IF ports, means that only the masters that are connected to those M3IF ports can

access the watermark space. The masters are system dependent and cannot be changed by the software. The watermark ports are using programming and constant in a given system (the watermark port is connected by the system architecture to VCC/GND). The preselected ports are MPG5 and MPG6, which means that all masters that access the M3IF using those ports can access the memory regions that are defined as watermark regions. Access to the watermark space by other masters is considered as a watermark violation, and a watermark interrupt is generated if enabled through the M3IFWCSR register (register details at Section 35.3.5, “Watermark and Snooping Logic”).

One watermark space can be defined for each predefined CS mapped by the M3IF. The watermark space is configurable through a set of 8 configuration registers that contains the watermark space base address for each chip select. The base address has a resolution of 1Kbyte, means:

- The lower base address can be set at the end of the first Kbyte at address 0x3FF.
- The base address can be set in steps of 1 Kbyte.

Figure 35-27 illustrates a watermark space of 48 Mbyte in SDRAM CSD0 memory region. The red shaded area in the figure is the watermark space defined in SDRAM CSD0 memory region. This memory region is accessible only by those masters that are connected to the predefined M3IF watermark ports. Accesses to this region by other masters generates an interrupt (if enabled through the watermark control and status register) and sets the respective watermark space (clear by one) status bit. The blue shaded area is the non protected SDRAM CSD0 memory region, means that it is accessible by all masters connected to the M3IF.

NOTE

In case that WEIM operates in collapse mode (CS0 and CS1 are combined as one space) the watermark region definition works as if CS0 and CS1 where separate spaces, for example, there are two watermark spaces. For example, if a watermark is defined in M3IFWCFG0 register as A3FF_FFFF, then the watermark region ends at A7FF_FFFF and accesses to A800_0000 and up are enabled for all masters (the region A800_0000 - AFFF_FFFF although belong to CS0 (due to collapse mode) is not seen as a watermark region). An additional watermark need to be defined in M3IFWCFG1 in order to define it as a watermark region as well.

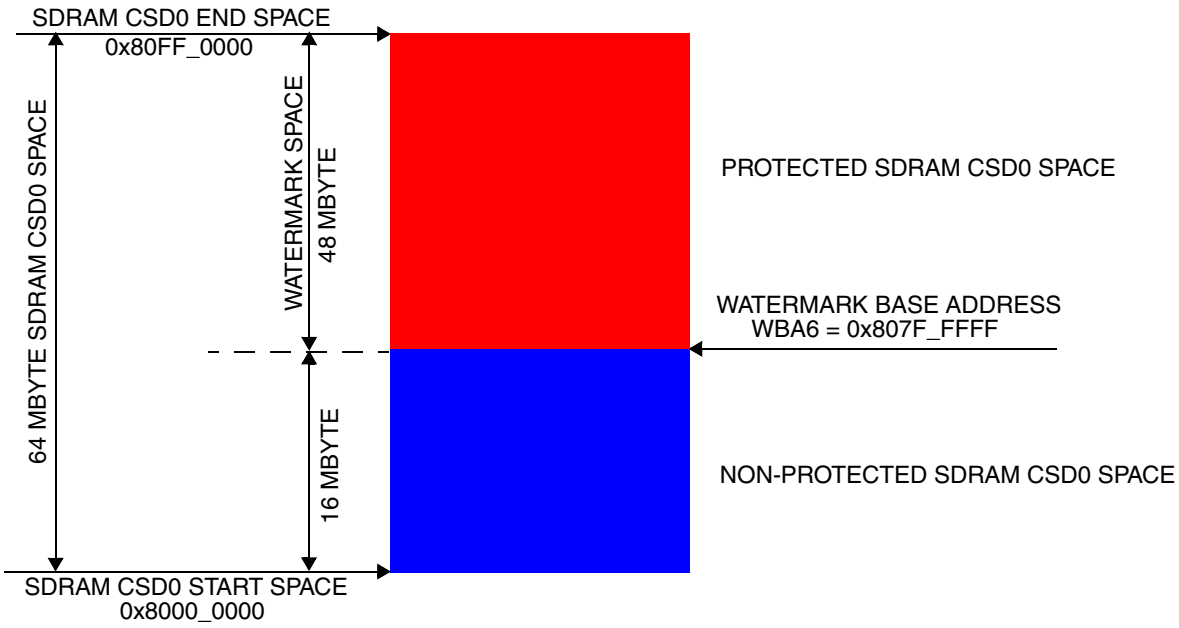


Figure 35-27. SDRAM CS0 Watermark Space Visualization

35.3.5.3 Snooping Overview

The M3IF snooping feature (used by the Image Processor Unit, IPU), monitors and detects write accesses to a configurable window (snooping window) in one of the memory regions mapped by the M3IF. The snooping window base address, memory region, and window size are configurable parameters through the M3IFSCFG0 register (register details at Section 35.2.3.4, “M3IF Snooping Configuration Register 0 (M3IFSCFG0)”). The snooping window is further divided into 64 equally sized segments.

A detected write access to the snooping window results in:

- The respective segment status bit in M3IFSSR0 and/or M3IFSSR1 registers is set.
- DMA_ACCESS strobe is asserted for 1 clock cycle if the snooping segment enable bit is set for the snooped segment. The snooping segment enable bit is configured using 2 snooping configuration registers, M3IFSCFG1 and M3IFSCFG2.

It is the software’s responsibility to clear the snooped segment status bits, but the snooped segment status bit is set for each snooping detection regardless of its value.

35.4 Initialization/Application Information

35.4.1 M3IF in a System

This section provides an example of M3IF initialization, integration and configuration in a given system.The system requirements are listed below:

- Several masters with external memories access capabilities.
  - — ARM I-Cache—32/64-bits data bus, according to chip demands.

- ARM D-Cache—32/64-bits data bus, according to chip demands.
- DSP Platform—32 bit data bus using watermark ports (according to SOC requirements)
  - Watermark space (0xA00E_FFFF) in WEIM CS0 and WEIM CS1 (0xA81F_FFFF)
  - Watermark Interrupt enable
- Layer 2 Cache—64 bit data bus
- Layer 2 Cache—32 bit data bus
- IPU—32-bit data bus with the following snooping requirements
  - 512-Kbyte snooping window in SDRAM CSD1 memory region
  - Snooping window base address 0x9000_F000
  - Enable snooping for segments, 3, 7, 12-27, 32, 36-56

- The system uses 2 SDRAM memory devices (32 bits), a 32-bit Flash (using WEIM CS0) and one SRAM (using WEIM CS1). The number of memory devices depends on the chip definitions.

35.4.1.1 Watermark Requirements Settings

The following M3IF settings are needed to implement the system watermark requirements:

- WATERMARK_ PORT[7:0] connected to VCC or GND according to system definition (hard connecting to VCC/GND in the system level).
- Write address 0xA00E_FC00 to M3IFWCFG0 register, in order to set the watermark base address for WEIM CS0 memory region.
- Write address 0xA81E_FC00 to M3IFWCFG1 register, in order to set the watermark base address for WEIM CS1 memory region.
- Write 0x8000_0000 to M3IFWCSR register in order to enable the watermark interrupt generation.

35.4.1.2 Snooping Window Settings

The following M3IF settings are needed to implement the system snooping requirements:

- Write 0x9000_F011 to M3IFSCFG0 register to set the snooping window base address and size.
- Write 0x0FFF_F088 to M3IFSCFG1 register to enable snooping for the specified lower segments.
- Write 0x01FF_FFF1 to M3IFSCFG2 register to enable snooping for the specified higher segments.

Chapter 36
NAND Flash Controller (NFC)

This chapter describes the NAND Flash controller (NFC) module implemented on this device, and covers the following topics:

36.1 Overview

The NFC provides the system’s interface to standard NAND Flash devices, and hides the complexities of accessing the NAND Flash. It provides a glueless interface to 8-bit and 16-bit NAND Flash parts, with page sizes of 512 bytes, 2 Kbytes or 4 Kbytes.

Figure 36-1 shows a block diagram of the NFC, which is composed of various control logic units and a 4.5-Kbyte internal RAM buffer.

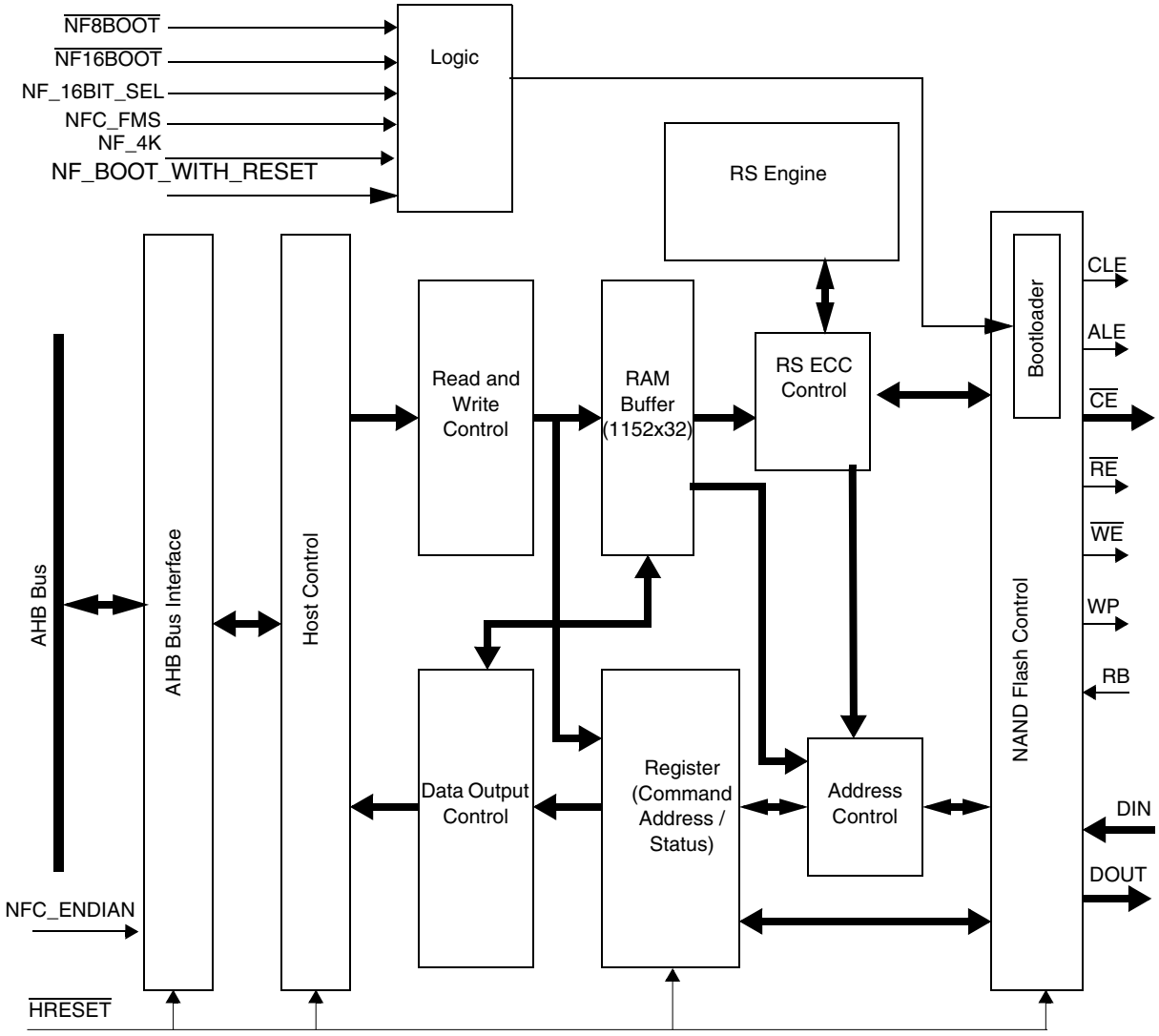


Figure 36-1. NAND Flash Controller Block Diagram

36.1.1 Features

The NAND Flash controller (NFC) includes the following features:

- x8/x16 (pin-configurable) NAND Flash interface
- Internal RAM buffer (4 Kbytes + 512 bytes)
  - Can be configured as Boot RAM
  - Operates as a buffer during normal operation
  - Registers and internal RAM buffer are memory-mapped to the same AHB region
- Manual interface with NAND Flash devices

  - Supports all NAND Flash products of up to 64-Kbyte blocks (for instance, NFC supports single-level cell (SLC) NAND Flash devices with 512 bytes per page, 16 Kbytes per block, and memory size up to 8 Gbits)
  - Supports SLC NAND Flash devices with 2 Kbytes per page, 128 Kbytes per block, and memory size up to 64 Gbits
  - Supports multi-level cell (MLC) NAND Flash devices with 4 Kbytes per page, 512 Kbytes per block, and memory size up to 256 Gbits
- Supports MLC NAND with two options for Reed-Solomon error correction (configurable):
  - Corrects four 9-bit symbol errors in 528 bytes (512 main bytes + 16 bytes spare)
  - Corrects eight 9-bit symbol errors in 538 bytes (512 main bytes + 26 bytes spare)
- Advanced high-performance bus (AHB) host interface:
  - Supports read and write bursts
  - Supports 16-bit or 32-bit bus transfers
- Supports DMA requests for page read, section read and other read operations.
- Supports error correction (ECC) mode and ECC mode bypass
- Supports multiple reset (reset of NFC and NAND Flash device)
- Internal boot code loader during power-up provides advanced data protection (can be enabled or disabled)
  - Data protection
  - RAM buffer write-protect mode provides write protection for the lower 2 Kbytes of RAM buffer) (see Section 36.4.3.6, “Controller Status and Result of Flash Operation Register 2 (ECC_STATUS_RESULT2)”)
  - Write-protect mode for NAND Flash devices provides block-based write protection of NAND Flash
- Automatic write protection for RAM buffer during power-up, in addition to run-time write protection modes for both the RAM buffer and the NAND Flash device.
- Handshaking feature: INT pin indicates ready/busy status of NFC
- Special arbitration logic enables sharing of I/O pins with other memory controllers

36.1.2 Modes of Operation

The NFC has several modes of operation that correspond to different boot, page size, and I/O bus width configurations. The mode of operation is determined by five input signals: NFC_FMS, NF_4K, NF8BOOT, NF16BOOT, NF_16BIT_SEL. See Section 36.5.2, “Modes of Operation,” for more details.

36.2 External Signal Description

This section describes the NFC external signals.

36.2.1 Overview of Signals

The following signals shown in Table 36-1 are used to configure and control the NFC and its attached Flash device.

Table 36-1. NFC Signal Properties

Name	Abbreviation	Function	I/O	Reset
HCLK_IN	—	133 MHz AHB clock.	I	enable
$\overline{\text{HRESET}}$	—	WARM reset (active low)	I	0
$\overline{\text{IPI\_INT\_NFC}}$	—	NAND Flash controller interrupt	O	1
IPP_FLASH_CLK	—	Clock for the Flash side	I	enable
IPP_NFC_ALE_OUT	ALE#	Flash address latch enable	O	0
IPP_NFC_CE*n*_OUT	CE#	Flash #*n* chip enable (*n* = 0,1,2,3)	O	1
IPP_NFC_CLE_OUT	CLE#	Flash command latch enable	O	0
IPP_NFC_RB_IN	RB	Flash ready/busy	I	1
IPP_NFC_RE_OUT	RE#	Flash read enable	O	1
IPP_NFC_READ_DATA_IN [15:0]	—	NFC data input from the NAND Flash	I	
IPP_NFC_WE_OUT	WE#	Flash write enable	O	1
IPP_NFC_WP_OUT	WP#	Flash write protect	O	1
IPP_NFC_WRITE_DATA_OUT [15:0]	—	NFC data output towards the NAND Flash	O	0000
$\overline{\text{IPP\_RESET}}$	—	Power on reset for booting (active low)	I	1
NF_16BIT_SEL	—	Use 8 or 16-bits NAND Flash	I	1
$\overline{\text{NF8BOOT}}$	—	Boot from 8-bit NAND Flash (active low)	I	1
NFC_FMS	—	Flash memory select (512-byte / 2-Kbyte page)	I	0
NF_4K	—	Flash device is a 4-Kbyte page device	I	0
$\overline{\text{NF16BOOT}}$	—	Boot from 16-bit NAND Flash (active low)	I	1
NF_BOOT_WITH_RESET	—	Perform reset command before boot sequence	I	—

36.2.2 Detailed Signal Descriptions

Table 36-2 gives detailed descriptions of the NFC external signals.

Table 36-2. NFC Detailed Signal Descriptions

Signal (Signal Abbreviation)	I/O	Description
HCLK_IN	I	AHB clock input. This is the clock signal for the NFC, which arrives from the AHB side. Its frequency can be up to 133 MHz.
$\overline{\text{HRESET}}$		Warm reset. This signal produces a warm reset, causing the NFC and the NAND Flash device to cease current operation, and set all internal registers to their default state. See Figure 36-33 for a timing diagram of the reset operation. The AHB bus interface is connected directly to this signal, and will cause a reset immediately when this line goes to low state. Warm reset has no effect on the contents of main/spare area buffers.
$\overline{\text{IPI\_INT\_NFC}}$	O	NFC interrupt. This output is the NFC interrupt, and is asserted to low when an NFC event takes place. In addition, it is asserted to low when any of the following occur: • NAND Flash command input • NAND Flash address input • NAND Flash data input • NAND Flash data output The signal returns to high when basic operation and boot loading is done, or when a warm or hot reset is released.
IPP_FLASH_CLK		This clock signal controls the NAND Flash controller's state machine when interfacing with a NAND Flash device. Maximum supported clock frequency is 50 MHz.
IPP_NFC_ALE_OUT (ALE#)	O	Flash address latch enable. The ALE# output controls the activating path for addresses sent to the address register of NAND Flash (NAND_FLASH_ADD). When active high, addresses are latched into the NAND FC address register of NAND Flash through the I/O ports on the rising edge of the WE# signal.
IPP_NFC_CE*n*_OUT(*n* = 0…3) (CE#)	O	Flash #*n* chip enable (*n* = 0…3). This signal indicates the NAND Flash selection. When the NAND Flash device is in the Busy state, or when the NAND Flash device is accessed, this signal is low.
IPP_NFC_CLE_OUT (CLE#)	O	Flash command latch enable. The CLE# output controls the activating path for commands sent to the command register of NAND Flash (NAND_Flash_CMD). When active high, commands are latched into the command register of NAND Flash through the I/O ports on the rising edge of the IPP_NFC_WE_OUT (WE#) signal.
IPP_NFC_RB_IN (RB)	I	Flash ready/busy. This signal indicates the status of the NAND Flash operation. When low, it indicates that a program, erase, or random read operation of NAND Flash is in process. Upon completion of the process, this signal returns to high state. This signal is connected to an open drain output, using a 100 KΩ pull-up resistor that is outside the external NAND Flash memory device.
IPP_NFC_READ_DATA_IN[15:0]	I	The AHB host uses this 16-bit signal to read data from registers or from internal memory.
IPP_NFC_RE_OUT (RE#)	O	Flash read enable. This output is the NAND Flash device serial data output control. When active, this signal drives the data from the NAND Flash device onto the NAND Flash I/O bus, allowing the NFC to read the data. When reading a burst from the NAND Flash device, this signal increments the NAND Flash internal column address counter by one.

Table 36-2. NFC Detailed Signal Descriptions (continued)

Signal (Signal Abbreviation)	I/O	Description
IPP_NFC_WE_OUT (WE#)	O	Flash write enable.This output controls writes to the NAND Flash I/O port, thus allowing the NAND Flash device to read data. Commands, address and data are latched on the rising edge of this signal.
IPP_NFC_WP_OUT (WP#)	O	Flash write protect. This signal provides inadvertent program/erase protection during power transitions and is automatically controlled by NFC. This pin status is only active (held low) during power-up.
IPP_NFC_WRITE_DATA_OUT [15:0]	O	The AHB host uses this 16-bit signal to write data to registers or to internal memory.
IPP_RESET_B	I	This input is the power on reset (POR) signal in the NFC. When asserted, a POR takes place.
$\overline{\text{NF8BOOT}}$, $\overline{\text{NF16BOOT}}$, NF_16BIT_SEL		The $\overline{\text{NF8BOOT}}$ and $\overline{\text{NF16BOOT}}$ are boot signals that determine whether the chip will boot from the NAND Flash device. They are also used to control the external bus width of the NAND Flash (8 bit or 16 bit). If either of the boot inputs is asserted (that is, either $\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ is low) at System Power-On reset (IPP_RESET_B rising edge), a 4-Kbyte boot code is copied from the NAND Flash device to the RAM buffer. If neither of the two boot signals are asserted, then the input signal NF_16BIT_SEL is read to help determine the operating mode of the NFC. See Section 36.5.2, "Modes of Operation," for more details. **Note:** The boot signals should remain at the same value before and after the boot process.
NF_BOOT_WITH_RESET	I	Perform a reset command (0xFF), before automatic boot load.
NFC_FMS, NF_4K	I	Flash memory select. The NFC_FMS and NF_4K signals indicates the size of the NAND Flash page (512 bytes, 2 Kbytes, or 4Kbytes). 00 NAND Flash page size is 512 bytes. 10 NAND Flash page size is 2 Kbytes. 01 NAND Flash page size is 4 Kbytes. 11 NAND Flash page size is 4 Kbytes.

36.3 NFC Buffer Memory Space

Table 36-3 shows the organization of the buffer memory space in the NFC.

Table 36-3. Data (Buffer) Organization in Memory

Address	Use	Access
0x0000– 0x01FE	Main area buffer 0	R/W
0x0200– 0x03FE	Main area buffer 1	R/W
0x0400– 0x05FE	Main area buffer 2	R/W
0x0600– 0x07FE	Main area buffer 3	R/W
0x0800– 0x09FE	Main area buffer 4	R/W
0x0A00– 0x0BFE	Main area buffer 5	R/W
0x0C00– 0x0DFE	Main area buffer 6	R/W

Table 36-3. Data (Buffer) Organization in Memory (continued)

Address	Use	Access
0x0E00– 0x0FFE	Main area buffer 7	R/W
0x1000– 0x103E	Spare area buffer 0	R/W
0x1040– 0x107E	Spare area buffer 1	R/W
0x1080– 0x10BE	Spare area buffer 2	R/W
0x10C0– 0x10FE	Spare area buffer 3	R/W
0x1100– 0x113E	Spare area buffer 4	R/W
0x1140– 0x117E	Spare area buffer 5	R/W
0x1180– 0x11BE	Spare area buffer 6	R/W
0x11C0– 0x11FE	Spare area buffer 7	R/W
0x1200– 0x1BFE	Reserved	—
0x1E00– 0x1E1C	Registers	R/W

36.3.1 Main and Spare Area Buffers

The main area buffer is a general data block. The spare area buffer is used for a variety of functions, including error correction. All spare area buffers 0–7 have the same organization, which is shown in Table 36-4. The host can use all of the spare area except for the area set aside for the ECC code: so for example, the AHB host can write data to a reserved area of the spare area buffer during a program operation.

Table 36-4. Spare Area Buffer Organization

Spare Buffer Base Address Offset	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0x1000	2nd logical sector number (LSN)								1st logical sector number (LSN)							
0x1002	1st wrap count (WC)[1]								3rd logical sector number (LSN)							
0x1004	Bad block information (BI)								2nd wrap count (WC)[1]							
0x1006	RS ECC code for main and spare area data (1st)								Reserved							
0x1008	RS ECC code for main and spare area data (3rd)								RS ECC code for main and spare area data (2nd)							
0x100A	RS ECC code for main and spare area data (5th)								RS ECC code for main and spare area data (4th)							
0x100C	RS ECC code for main and spare area data (7th)								RS ECC code for main and spare area data (6th)							
0x100E	RS ECC code for main and spare area data (9th)								RS ECC code for main and spare area data (8th)							
0x1010	RS ECC code for main and spare area data (11th)								RS ECC code for main and spare area data (10th)							
0x1012	RS ECC code for main and spare area data (13th)								RS ECC code for main and spare area data (12th)							
0x1014	RS ECC code for main and spare area data (15th)								RS ECC code for main and spare area data (14th)							
0x1016	RS ECC code for main and spare area data (17th)								RS ECC code for main and spare area data (16th)							

Table 36-4. Spare Area Buffer Organization (continued)

Spare Buffer Base Address Offset	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0x1018	Reserved								RS ECC code for main and spare area data (18th)							
0x101A–0x103E	Reserved															

[1] Wrap count and other bytes have the same wrap count information, and are used as error correction for wrap count itself.

36.4 Memory Map and Register Definitions

This section includes the module memory map and detailed descriptions of all registers. For the base address of a particular module instantiation, see the system memory map.

36.4.1 Memory Map

Table 36-5 shows the NFC memory map.

Table 36-5. NFC Module Register Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x1E00	Reserved	—	—	—
0x1E02	Reserved	—	—	—
0x1E04 (RAM_BUFFER_ADDRESS)	RAM buffer address register	R/W	0x0000_0000	36.4.3.1/36-11
0x1E06 (NAND_FLASH_ADDR)	NAND Flash address register	R/W	0x0000_0000	36.4.3.2/36-12
0x1E08 (NAND_FLASH_CMD)	NAND Flash command register	R/W	0x0000_0000	36.4.3.3/36-13
0x1E0A (NFC_CONFIGURATION)	NFC internal buffer lock control register	R/W	0x0000_0001	36.4.3.4/36-13
0x1E0C (ECC_STATUS_RESULT1)	RS ECC status register 1	Read-only	0x0000_0000	36.4.3.5/36-14
0x1E0E (ECC_STATUS_RESULT2)	RS ECC status register 2	Read-only	0x0000_0000	36.4.3.6/36-14
0x1E10 (SPAS)	Spare only size register	R/W	0x0000_0000	36.4.3.7/36-15
0x1E12 (NF_WR_PROT)	NAND Flash write protection register	R/W	0x0000_0002	36.4.3.8/36-15
0x1E14	Reserved	—	—	—
0x1E16	Reserved	—	—	—
0x1E18 (NAND_FLASH_WR_PR_ST)	NAND Flash write protection status register	R/W	0x0000_0492	36.4.3.9/36-16
0x1E1A (NAND_FLASH_CONFIG1)	NAND Flash operation configuration register 1	R/W	0x0000_0c0a	36.4.3.10/36-18
0x1E1C (NAND_FLASH_CONFIG2)	NAND Flash operation configuration register 2	R/W	0x0000_0000	36.4.3.11/36-20

Table 36-5. NFC Module Register Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x1E20 (UNLOCK_START_BLK_ADD0)	Address to unlock in write protection mode—start register 0	R/W	0x0000_0000	36.4.3.12/36-21
0x1E22 (UNLOCK_END_BLK_ADD0)	Address to unlock in write protection mode—end register 0	R/W	0x0000_0000	36.4.3.13/36-22
0x1E24 (UNLOCK_START_BLK_ADD1)	Address to unlock in write protection mode—start register 1	R/W	0x0000_0000	36.4.3.12/36-21
0x1E26 (UNLOCK_END_BLK_ADD1)	Address to unlock in write protection mode—end register 1	R/W	0x0000_0000	36.4.3.13/36-22
0x1E28 (UNLOCK_START_BLK_ADD2)	Address to unlock in write protection mode—start register 2	R/W	0x0000_0000	36.4.3.12/36-21
0x1E2A (UNLOCK_END_BLK_ADD2)	Address to unlock in write protection mode—end register 2	R/W	0x0000_0000	36.4.3.13/36-22
0x1E2C (UNLOCK_START_BLK_ADD3)	Address to unlock in write protection mode—start register 3	R/W	0x0000_0000	36.4.3.12/36-21
0x1E2E (UNLOCK_END_BLK_ADD3)	Address to unlock in write protection mode—end register 3	R/W	0x0000_0000	36.4.3.13/36-22

36.4.2 Register Summary

Figure 36-2 shows the key to the register fields and Table 36-6 shows the register figure conventions.

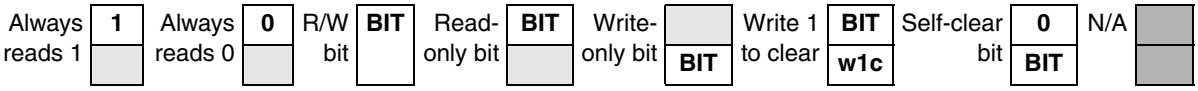


Figure 36-2. Key to Register Fields

Table 36-6. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	

Table 36-6. Register Figure Conventions (continued)

Convention	Description
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 36-7 shows the NFC register summary.

Table 36-7. NFC Register Summary

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E04 (RAM_BUFFER_ADDRESS)	R	0	0	0	0	0	0	0	0	0	0	ACTIVE_CS		0	RBA		
	W																
0x1E06 (NAND_FLASH_ADDR)	R	ADDR															
	W																
0x1E08 (NAND_FLASH_CMD)	R	CMD															
	W																
0x1E0A (NFC_CONFIGURATION)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BLS	
	W																
0x1E0C (ECC_STATUS_RESULT1)	R	NOSER4				NOSER3				NOSER2				NOSER1			
	W																
0x1E0E (ECC_STATUS_RESULT2)	R	NOSER8				NOSER7				NOSER6				NOSER5			
	W																
0x1E10 (SPAS)	R	0	0	0	0	0	0	0	0	SPAS							
	W																
0x1E12 (NF_WR_PROT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WPC		
	W																
0xBASE_1E14 (Reserved)	R	Reserved															
	W																
0xBASE_1E16 (Reserved)	R	Reserved															
	W																
0x1E18 (NAND_FLASH_WR_PR_ST)	R	0	0	0	0	US3	LS3	LTS3	US2	LS2	LTS2	US1	LS1	LTS1	US0	LS0	LTS0
	W																

Table 36-7. NFC Register Summary (continued)

Name		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E1A (NAND_FLASH_CONFIG1)	R	0	0	0	0	FP_INT	PPB		SYM	NF_CE	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	dma_mode	ECC_MODE
	W																
0x1E1C (NAND_FLASH_CONFIG2)	R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
	W																
0x1E20 (UNLOCK_START_BLK_ADD0)	R	USBA0															
	W																
0x1E22 (UNLOCK_END_BLK_ADD0)	R	UEBA0															
	W																
0x1E24 (UNLOCK_START_BLK_ADD1)	R	**USBA1**															
	W																
0x1E26 (UNLOCK_END_BLK_ADD1)	R	**UEBA1**															
	W																
0x1E28 (UNLOCK_START_BLK_ADD2)	R	**USBA2**															
	W																
0x1E2A (UNLOCK_END_BLK_ADD2)	R	**UEBA2**															
	W																
0x1E2C (UNLOCK_START_BLK_ADD3)	R	**USBA3**															
	W																
0x1E2E (UNLOCK_END_BLK_ADD3)	R	**UEBA3**															
	W																

36.4.3 Register Descriptions

36.4.3.1 RAM Buffer Address Register (RAM_BUFFER_ADDRESS)

RBA specifies which part of the RAM buffer is transferred to/from Flash memory. The bit assignments for the register are shown in Figure 36-3 and the field descriptions are shown in Table 36-8.

Offset 0x1E04 (RAM_BUFFER_ADDRESS) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	ACTIVE_CS		0	RBA		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-3. RAM Buffer Address Register

Table 36-8. RAM Buffer Address Field Descriptions

Field	Description
15–6	Reserved.
5–4 ACTIVE_CS	Active chip select. Defines the chip-select line to be asserted during any NAND operation. For example, if ACTIVE_CS is set to 0b01, then all NFC operations are executed to chip select 1 (ipp_nfc_ce1_out) 00 - chip select 0 is active 01 - chip select 1 is active 10 - chip select 2 is active 11- chip select 3 is active
3	Reserved.
2–0 RBA	RAM buffer address. Specifies the RAM buffer number to use for data transfers to or from the NAND Flash device. 0000 1st internal RAM buffer 0001 2nd internal RAM buffer 0010 3rd internal RAM buffer 0011 4th internal RAM buffer 0100 5th internal RAM buffer 0101 6th internal RAM buffer 0110 7th internal RAM buffer 0111 8th internal RAM buffer

36.4.3.2 NAND Flash Address Register (NAND_FLASH_ADDR)

The NAND Flash address (NAND_FLASH_ADDR) register is a read-write register containing the address of the NAND Flash device that will be read, programmed or erased. The address in the NAND_FLASH_ADD register is written to the Flash device. The bit assignments for the register are shown in Figure 36-4 and the field descriptions are shown in Table 36-9.

Offset 0x1E06 (NAND_FLASH_ADDR) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-4. NAND Flash Address Register

Table 36-9. NAND Flash Address Register Field Description

Field	Description
15–0 ADDR	NAND Flash address. NAND Flash address which will be read, programmed or erased. This address is written to the NAND Flash device.

36.4.3.3 NAND Flash Command Register (NAND_FLASH_CMD)

The bit assignments for the register are shown in Figure 36-5 and the field descriptions are shown in Table 36-10.

Offset 0x1E08 (NAND_FLASH_CMD) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-5. NAND_FLASH_CMD Register

Table 36-10. NAND_FLASH_CMD REGISTER Field Description

Field	Description
15–0 CMD	NAND Flash command. This field contains the CMD that is written to the NAND Flash device.

36.4.3.4 NFC Internal Buffer Lock Control Register (NFC_CONFIGURATION)

The bit assignments for the register are shown in Figure 36-6 and the field descriptions are shown in Table 36-11.

Offset 0x1E0A (NFC_CONFIGURATION) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BLS	
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 36-6. NFC_CONFIGURATION Register

Table 36-11. NFC_CONFIGURATION Register Field Descriptions

Field	Description
15–2	Reserved
1–0 BLS	Buffer lock set. This field specifies the buffer lock status of first four sections of the internal buffer. The other four sections are always unlocked.(for more details see section Section 36.6.4, “Write Protection Operation.” 00 Locked 01 Locked (default) 10 Unlocked 11 Locked

36.4.3.5 Controller Status and Result of Flash Operation Register 1 (ECC_STATUS_RESULT1)

The fields in this register shows the number of symbol errors for the first four sections of 528/538 bytes of data (including 512 bytes of main data plus 16/26 bytes of spare data, depending on the setting of the ECC_MODE bit in the NAND_FLASH_CONFIG1 register) as a result of the RS ECC check. The bit assignments for the register are shown in Figure 36-7 and the field descriptions are shown in Table 36-12.

Offset 0x1E0C (ECC_STATUS_RESULT1) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOSER4				NOSER3				NOSER2				NOSER1			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-7. ECC_STATUS_RESULT1 Register

Table 36-12. ECC_STATUS_RESULT1 Register Field Descriptions

Field	Description
15–12 11–8 7–4 3–0 NOSER*n* (*n* = 4,3,2,1)	Number of symbol errors for *n*th section of the internal RAM (528 or 538 bytes, depending on the setting of the ECC_MODE bit in the NAND_FLASH_CONFIG1 register). These fields shows the number of symbols with errors in 512 bytes main data plus 16/26 bytes spare data (528/538 bytes total) as a result of the RS(511,503) or RS(511,495) ECC check upon read. For the entire 528/538 bytes the number of correctable symbols is 4/8, depending on ECC_MODE. 0000 No error 0001–1000Indicates the number of symbol errors (correctable error) 1111 Unknown number of errors (uncorrectable error) Others Reserved

36.4.3.6 Controller Status and Result of Flash Operation Register 2 (ECC_STATUS_RESULT2)

The fields in this register shows the number of symbol errors for the last four data sections of 528/538 bytes (including 512 bytes of main data plus 16/26 bytes of spare data, depending on the setting of the ECC_MODE bit in the NAND_FLASH_CONFIG1 register) as a result of the RS ECC check. Figure 36-8 shows the bit assignments for the register, and Table 36-13 shows the field descriptions.

Offset 0x1E0E (ECC_STATUS_RESULT2) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NOSER8				NOSER7				NOSER6				NOSER5			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-8. ECC_STATUS_RESULT2 Register

Table 36-13. ECC_STATUS_RESULT2 Field Descriptions

Field	Description
15–12 11–8 7–4 3–0 NOSER*n* (*n* = 8,7,6,5)	Number of symbol errors for *n*th section of the internal RAM (528 or 538 bytes, depending on the setting of the ECC_MODE bit in the NAND_FLASH_CONFIG1 register). These fields shows the number of symbols with errors in 512 bytes main data plus 16/26 bytes spare data (528/538 bytes total) as a result of the RS ECC check upon read. For the entire 528/538 bytes the number of correctable symbols is 4/8, depending on ECC_MODE. 0000 No error 0001–1000Indicates the number of symbol errors (if errors are correctable) 1111 Unknown number of errors (uncorrectable error) Others Reserved

36.4.3.7 Spare Area Size Register (SPAS)

This register specifies the total size (in 16-bit half-words) for the spare area of the NAND device. The bit assignments for the register are shown in Figure 36-9, and the field descriptions are shown in Table 36-14.

Offset 0x1E10 (SPAS) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	SPAS							
W																
RESET	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1

Figure 36-9. Spare Area Size Register

Table 36-14. Spare Area Size Register Description

Field	Description
15–8	Reserved
7–0 SPAS	Spare area size. This field specifies the total size (in 16-bit half-words) of the spare area for the NAND device. The size refers to a full page: for example, SPAS should be set to 64 for a 2 Kbyte SLC device.

36.4.3.8 NAND Flash Write Protection Register (NF_WR_PROT)

This register specifies the protection command (LOCK, UNLOCK, or LOCK TIGHT) performed by the NFC. The bit assignments for the register are shown in Figure 36-10, and the field descriptions are shown in Table 36-15.

Offset 0x1E12 (NF_WR_PROT) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	WPC		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 36-10. NAND Flash Write Protection Register

Table 36-15. NAND Flash Write Protection Register Field Descriptions

Field	Description
15–3	Reserved
2–0 WPC	Write protection command. This command field specifies the write-protect operation that the NFC performs. The command is performed on the protection mechanism of the chip-select that is configured in the ACTIVE_CS field. For example, if ACTIVE_CS = 0b01, then writing to WPC will change the state of LTS1, US1 and LS1. 100 Unlock NAND Flash blocks according to the block address range specified in the UNLOCK_START_BLK_ADD*n* and UNLOCK_END_BLK_ADD*n* registers, where *n* refers to the value of ACTIVE_CS. 010 Lock all NAND Flash blocks 001 Lock-tight locked blocks (see also Section 36.6.4, “Write Protection Operation”)

36.4.3.9 NAND Flash Write Protection Status Register (NAND_FLASH_WR_PR_ST)

This register is a status register which reads the NAND Flash write protection status (lock, unlock or lock tight) for each of the four chip selects. The write protection mechanism is the same for each chip select. The bit assignments for the register are shown in Figure 36-11 and the field descriptions are shown in Table 36-16.

Offset 0x1E18 (NAND_FLASH_WR_PR_ST) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	US3	LS3	LTS3	US2	LS2	LTS2	US1	LS1	LTS1	US0	LS0	LTS0
W																
RESET	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0

Figure 36-11. NAND_FLASH_WR_PR_ST Register

Table 36-16. NAND_FLASH_WR_PR_ST Register Field Descriptions

Field	Description
15–12	Reserved
11 US3	Unlocked status. This bit indicates whether there are any unlocked blocks in the NAND Flash that is connected to chip select 3. This bit is updated every time a command is issued to the NAND Flash. 0 There are no unlocked blocks in NAND Flash 1 There are unlocked block(s) in NAND Flash
10 LS3	Locked status. This bit indicates whether there are any locked blocks in the NAND Flash that is connected to chip select 3. This bit is updated every time a command is issued to the NAND Flash. 0 There are no locked blocks in NAND Flash 1 There are locked block(s) in NAND Flash

Table 36-16. NAND_FLASH_WR_PR_ST Register Field Descriptions (continued)

Field	Description
9 LTS3	Lock-tight status: Indicates if any of the locked blocks of the NAND connected to chip select 3 have lock-tight status. This bit is updated every time a command is issued to the NAND Flash. 0 No locked block are lock-tight 1 At least one locked block is lock-tight
8 US2	Unlocked status. This bit indicates whether there are any unlocked blocks in the NAND Flash that is connected to chip select 2. This bit is updated every time a command is issued to the NAND Flash. 0 There are no unlocked blocks in NAND Flash 1 There are unlocked block(s) in NAND Flash
7 LS2	Locked status. This bit indicates whether there are any locked blocks in the NAND Flash that is connected to chip select 2. This bit is updated every time a command is issued to the NAND Flash. 0 There are no locked blocks in NAND Flash 1 There are locked block(s) in NAND Flash
6 LTS2	Lock-tight status: Indicates if any of the locked blocks of the NAND connected to chip select 2 have lock-tight status. This bit is updated every time a command is issued to the NAND Flash. 0 No locked blocks are lock-tight 1 At least one locked block is lock-tight
5 US1	Unlocked status. This bit indicates whether there are any unlocked blocks in the NAND Flash that is connected to chip select 1. This bit is updated every time a command is issued to the NAND Flash. 0 There are no unlocked blocks in NAND Flash 1 There are unlocked block(s) in NAND Flash
4 LS1	Locked status. This bit indicates whether there are any locked blocks in the NAND Flash that is connected to chip select 1. This bit is updated every time a command is issued to the NAND Flash. 0 There are no locked blocks in NAND Flash 1 There are locked block(s) in NAND Flash
3 LTS1	Lock-tight status: Indicates if any of the locked blocks of the NAND connected to chip select 1 have lock-tight status. This bit is updated every time a command is issued to the NAND Flash. 0 No locked blocks are lock-tight 1 At least one locked block is lock-tight
2 US0	Unlocked status. This bit indicates whether there are any unlocked blocks in the NAND Flash that is connected to chip select 0. This bit is updated every time a command is issued to the NAND Flash. 0 There are no unlocked blocks in NAND Flash 1 There are unlocked block(s) in NAND Flash

Table 36-16. NAND_FLASH_WR_PR_ST Register Field Descriptions (continued)

Field	Description
1 LS0	Locked status. This bit indicates whether there are any locked blocks in the NAND Flash that is connected to chip select 0. This bit is updated every time a command is issued to the NAND Flash. 0 There are no locked blocks in NAND Flash 1 There are locked block(s) in NAND Flash
0 LTS0	Lock-tight status: Indicates if any of the locked blocks of the NAND connected to chip select 0 have lock-tight status. This bit is updated every time a command is issued to the NAND Flash. 0 No locked blocks are lock-tight 1 At least one locked block is lock-tight

Table 36-17 shows the write protect modes associated with different US*n*, LS*n*, and LTS*n* settings.

Table 36-17. Write Protect Modes

State	Status Bit Settings (US, LS, LTS)
Lock—All blocks are locked.	010
Unlock-lock—There are both locked and unlocked blocks.	110
Unlock-lockt—There are unlocked blocks. Cannot change to another state	101
Lockt—All blocks are locked, and cannot change to another state	011

36.4.3.10 NAND Flash Operation Configuration Register (NAND_FLASH_CONFIG1)

This register is a configuration register for the NAND Flash device to control the ECC enable or disable, mask interrupt. The bit assignments for the register are shown in Figure 36-12 and the field descriptions are shown in Table 36-18

Offset 0x1E1A (NAND_FLASH_CONFIG1) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	FP_INT	PPB		SYM	NF_CE	NFC_RST	NF_BIG	INT_MSK	ECC_EN	SP_EN	DMA_MODE	ECC_MODE
W																
RESET	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0

Figure 36-12. NAND_FLASH_CONFIG1 Register

Table 36-18. NAND_FLASH_CONFIG1 Register Field Descriptions

Field	Description
15–12	Reserved
11 FP_INT	Full page interrupt. By default NFC generates an interrupt (during program,/read) after each section of 512 bytes (plus accompanied spare bytes). If this bit is set, then the interrupt will be generated only after the whole page was read/programmed. This bit affects the interrupt only during read or program of a full page. 0 - NFC generates interrupt after each section of 512 bytes plus 16/26 spare bytes (default) 1 - NFC generates interrupt only after the entire read/program operation is complete.
10–9 PPB	Pages per block. Indicates how many pages are in each block of the NAND Flash. 00 - 32 pages per block 01 - 64 pages per block 10 - 128 pages per block 11 - 256 pages per block
8 SYM	1 Enable one Flash clock cycle per access of RE# and WE#, (symmetric mode). 0 Enable two Flash clock cycles per access of RE# and WE#, (asymmetric mode). See timing diagrams in Section 36.6.3, “Symmetric Mode—One Flash Clock Cycle per Input or Output Data Cycle.”
7 NF_CE	NAND Flash force CE. Setting this bit to 1 forces the CE# signal to the NAND Flash device to 0. This bit allows a greater range of support for new NAND Flash devices. 0 CE# signal operates normally 1 CE# signal is forced to 0
6 NFC_RST	NFC reset. This bit resets the NFC state machine. This bit should be used when issuing reset command to the NAND Flash device during operation. This bit is self-cleared, and resets also NF_BIG, INT_MSK and SP_EN. 0 Do not reset the NFC state machine 1 Reset the NFC state machine
5 NF_BIG	NAND Flash big-endian mode. This bit enables big-endian mode when writing from internal RAM to the NAND Flash device or reading from NAND Flash device to internal RAM. 0 Little-endian mode 1 Big-endian mode
4 INT_MSK	Mask interrupt bit. This bit enables the interrupt by masking or not masking the interrupt bit. 0 Mask interrupt is disabled (interrupt enabled) 1 Mask interrupt is enabled (interrupt disabled)
3 ECC_EN	ECC operation enable. This bit determines whether ECC operation is executed or bypassed 0 ECC operation is bypassed 1 ECC operation is executed
2 SP_EN	NAND Flash spare enable. This bit determines whether host reads/writes are to NAND Flash spare data only or NAND Flash main and spare data. This feature is supported only with memories with 1/2K page size. Note: There is no ECC in this mode of operation. 0 NAND Flash main and spare data is enabled 1 NAND Flash spare only data is enabled

Table 36-18. NAND_FLASH_CONFIG1 Register Field Descriptions (continued)

Field	Description
1 DMA_MODE	This bit defines the DMA_REQ signal mode of operation during a page read. DMA_REQ can be asserted after every section of the page is read (main + relevant part of the spare), or only at the end of the page read. Other read operations that assert dma_req are not affected by this bit, 0 — DMA_REQ is asserted after each section is read out. 1 — DMA_REQ is asserted only at the end of a page read.
0 ECC_MODE	This bit selects the ECC capabilities. Error correction mechanism can fix 4-symbol errors or 8-symbol errors. In 4-bit ECC mode, the NFC uses 16 bytes of spare area for every 512 bytes section of the NAND device (7 bytes for user-specific application and 9 bytes for the ECC). In 8-bit ECC mode, the NFC uses 26 bytes of spare area for every 512 bytes section of the nand device (7 bytes for user-specific application, 18 bytes for the ECC and 1 reserved byte). To use 8-symbol ECC mode, you must have a NAND device that has at least 26 bytes of spare area for every 512 B main section. 0 — 8-symbol error correction (reset value). 1 — 4-symbol error correction

36.4.3.11 NAND Flash Operation Configuration 2 (NAND_FLASH_CONFIG2)

This register controls the NAND Flash signals (CLE, ALE, WE, RE, CE) and sets the interrupt after command completion. The bit assignments for the register are shown in Figure 36-13 and the field descriptions are shown in Table 36-19.

Offset 0x1E1C (NAND_FLASH_CONFIG2) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT	0	0	0	0	0	0	0	0	0	FDO			FDI	FADD	FCMD
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-13. NAND_FLASH_CONFIG2 Register

Table 36-19. NAND_FLASH_CONFIG2 Register Field Descriptions

Field	Description
15 INT	Interrupt. This field determines the state of the interrupt output of the NAND Flash controller. It is set by the controller when a basic operation is done. It can also be written to by the host. 0 Basic operation or boot loading is still running 1 Basic operation or boot loading is done. **Note:** This bit resets to 0, but soon after power-up it will change to 1. When booting from NAND Flash, the INT bit will change from 0 to 1 after the boot code has been transferred. For more information, see Section 36.5.2, “Modes of Operation.”
14–6	Reserved
5–3 FDO	NAND Flash data output. This bit enables NAND Flash data output. The bit is automatically cleared when the operation is completed. 001 One page data out[1] 010 NAND Flash ID data out 100 NAND Flash status register data out **Note:** Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.

Table 36-19. NAND_FLASH_CONFIG2 Register Field Descriptions (continued)

Field	Description
2 FDI	NAND Flash data input. This field enables NAND Flash data input.The bit is automatically cleared when the operation is completed. 0 No NAND Flash data input operation 1 Enable NAND Flash data input operation **Note:** Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.
1 FADD	NAND Flash address input. This field enables NAND Flash address input. The bit is automatically cleared when the operation is completed. 0 No NAND Flash address input operation 1 Enable NAND Flash address input operation **Note:** Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.
0 FCMD	NAND Flash command input. This field enables the NAND Flash command input. The bit is automatically cleared when the operation is completed. 0 No NAND Flash command input operation 1 Allow NAND Flash command input operation **Note:** Only one of the bit fields (FCMD/FADD/FDI/FDO) can be set at any given time.

[1] Page size is determined by SP_EN register bit (main + spare or spare only).

36.4.3.12 Address to Unlock in Write Protection Mode—Start for Chip Select *n* (UNLOCK_START_BLK_ADD*n*, *n* = 0,1,2,3)

These registers contains the starting address of block memory in the NAND Flash that is unlocked in write protection mode for chip select *n* (*n* = 0,1,2,3). The bit assignments for the registers are shown in Figure 36-14 and the field descriptions are shown in Table 36-20.

Offset 0x1E20 (UNLOCK_START_BLK_ADD0)
0x1E24 (UNLOCK_START_BLK_ADD1)
0x1E28 (UNLOCK_START_BLK_ADD2)
0x1E2C (UNLOCK_START_BLK_ADD3)

Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USBA0															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-14. UNLOCK_START_BLK_ADD*n* Register

Table 36-20. UNLOCK_START_BLK_ADD*n* Register Field Description

Field	Description
15–0 USBA*n*	Unlock start block address for chip select *n* (*n* = 0,1,2,3). Starting address of block memory in the NAND Flash that is unlocked in write protection mode. For more details on this, see Section 36.6.4.4, “Write Protection Status.”

36.4.3.13 Address to Unlock in Write Protection Mode—End for Chip Select *n* (UNLOCK_END_BLK_ADD*n*, *n* = 0,1,2,3)

Ending address of block memory in the NAND Flash that is unlocked in write protection mode for chip select *n* (*n* = 0,1,2,3). The bit assignments for the registers are shown in Figure 36-15 and the field descriptions are shown in Table 36-21.

Offset 0x1E22 (UNLOCK_END_BLK_ADD0)
0x1E26 (UNLOCK_END_BLK_ADD1)
0x1E2A (UNLOCK_END_BLK_ADD2)
0x1E2E (UNLOCK_END_BLK_ADD3)

Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UEBA*n*															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 36-15. UNLOCK_END_BLK_ADD*n* Register

Table 36-21. UNLOCK_END_BLK_ADD*n* Register Field Description

Field	Description
15–0 UEBA*n*	Unlock end block address for chip select *n* (*n* = 0,1,2,3). Ending address of block memory in the NAND Flash that is unlocked in write protection mode for chip select *n*. For more details, see Section 36.6.4.4, "Write Protection Status."

36.5 Functional Description

This section provides the functional description for the NAND Flash controller.

36.5.1 Overview

The operation of the NFC begins by the AHB host initiating a read from the NAND Flash device by configuring the controller and then waiting for an interrupt from the NFC. When it receives the interrupt, the NFC inputs a page from the NAND Flash device, and upon completion, generates an interrupt to the AHB host.

When the AHB host receives this interrupt, it reads the content from the internal RAM buffer of the NFC. To complete the operation the AHB host checks the status of the operation by reading the NFC status registers.

The 4-Kbyte RAM buffer is used as the boot RAM during a cold reset (if the system is configured for a boot to be carried out from the NAND Flash device). After the boot procedure completes, the RAM is available as buffer RAM.

In addition, the NAND Flash controller provides an x16 bit and x32 bit interface to the AHB bus on the chip side, and an x8/x16 interface to the NAND Flash device on the external data bus.

36.5.2 Modes of Operation

The NAND FLASH Controller operating modes are described in this section.

The operating mode is determined by five input lines: NFC_FMS, NF_4K, $\overline{\text{NF8BOOT}}$, $\overline{\text{NF16BOOT}}$, NF_16BIT_SEL, as shown in Table 36-22.

The IC can boot from a NAND Flash device if exactly one of the two signals $\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ must be low. If both signals are high, a boot from the NAND Flash device does not occur. If both signals are low, the situation is undefined.

The page size of the NAND Flash is determined by the values of NFC_FMS and NF_4K. The NAND Flash is 512 bytes if both NFC_FMS and NF_4K are low, 2 Kbyte if NFC_FMS is high and NF_4K is low, and 4 Kbyte if NFC_FMS is low and NF_4K is high. See the signal description for more information.

When booting from a NAND Flash device, the bus width used corresponds to which of the signals $\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ is asserted (low). When not booting from the NAND Flash device, the bus width is determined by the value of the NF_16BIT_SEL signal (0 = 8-bit bus, 1 = 16-bit bus).

Table 36-22. NAND FLASH Controller Operating Modes

NFC_FMS	NF_4K	$\overline{\text{NF8BOOT}}$	$\overline{\text{NF16BOOT}}$	NF_16BIT_SEL	Function
0	0	1	1	0	Do not boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size is 512 bytes
0	0	1	1	1	Do not boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size is 512 bytes.
1	0	1	1	0	Do not boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size 2KB
1	0	1	1	1	Do not boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size 2KB
0	1	1	1	0	Do not boot from NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size 4KB
0	1	1	1	1	Do not boot from NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size 4KB
X	1	1	0	X	Boot from x16 NAND Flash. NAND Flash is configured to 16-bits I/O bus width and page size 4 Kbytes + 218 bytes spare.
X	1	0	1	X	Boot from x8 NAND Flash. NAND Flash is configured to 8-bits I/O bus width and page size 4 Kbyte + 218 bytes spare.
X	X	0	0	X	Not defined (do not use this setting)

36.5.3 Booting From a NAND Flash Device

A Boot from the NAND Flash device only occurs if one of the boot inputs is asserted ($\overline{\text{NF8BOOT}}$ or $\overline{\text{NF16BOOT}}$ is low) at system power-on reset ($\overline{\text{IPP\_RESET}}$ rising edge).

Boot from a NAND Flash device proceeds as follows:

1. The boot loader copies 1 page of 4 Kbytes main data + 218 bytes spare data from the NAND Flash to the NFC internal RAM buffer.
2. There are 5 address latch cycles during the bootloader. This is hard-coded and cannot be modified.
3. There is a read confirm command after the address cycles (0x30)
   - ECC is always calculated for each 512-byte section separately, and not for the whole 4-Kbyte page. Therefore, each 512 bytes of data in the NAND Flash is follow by 26 bytes of spare data (repeated 8 times for the entire 4-Kbyte page).
4. After exiting from reset state, the host reads the code from the NFC's internal RAM buffer.

Figure 36-16 shows boot mode operation from a NAND Flash device.

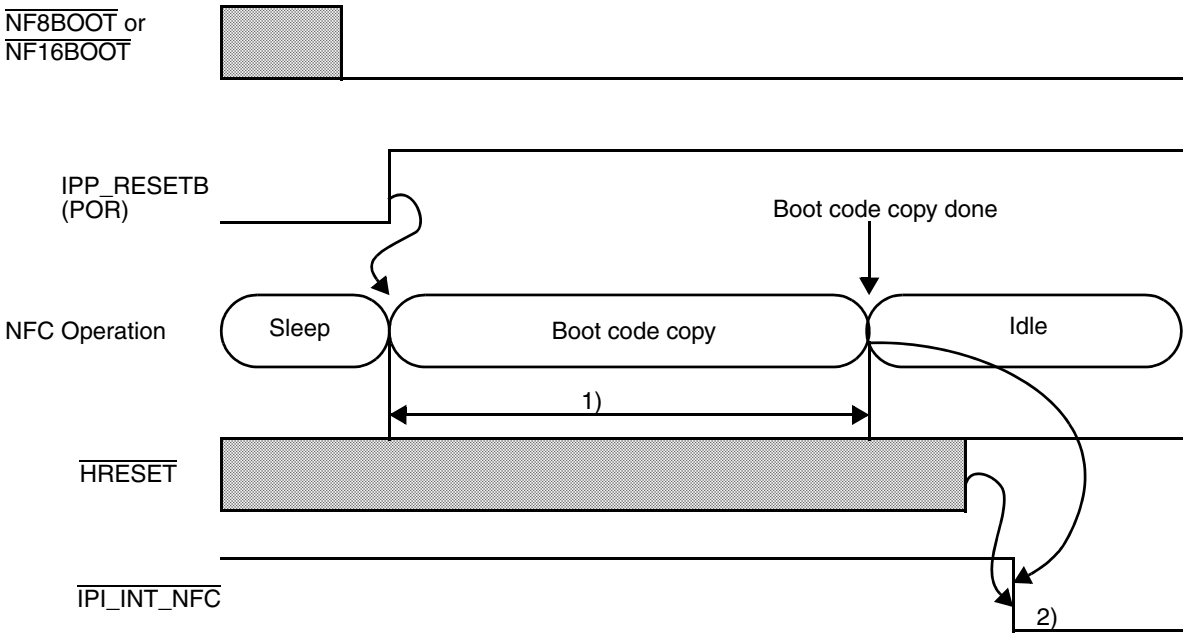


Figure 36-16. Boot Mode Operation

NOTE

The time it takes the boot copy to load 4 Kbytes is dependent on the NAND device and the frequency of Flash clock. The host must not read the boot code in the RAM buffer (4 Kbytes) until after the boot code copy is completed.

The interrupt signal ($\overline{\text{IPI\_INT\_NFC}}$) goes high to low when the boot code copy is completed, and upon the $\overline{\text{HRESET}}$ rising edge. If $\overline{\text{HRESET}}$ goes low to high before the boot code copy is done, the interrupt signal ($\overline{\text{IPI\_INT\_NFC}}$) goes from high to low as soon as the boot code copy is completed.

The interrupt can be relevant for cases of secured boot (booting from ROM and then enabling the NFC boot).

36.5.4 NAND Flash Control Submodule

The NAND Flash control submodule generates the following control signals:

- CE# (Flash chip enable)
- RE# (Read enable for read operations)
- WE# (Flash write enable)
- CLE# (Flash command latch enable)
- ALE# (Flash address latch enable).

The submodule also monitors the RB (Flash ready/busy indication) signal to check if the NAND Flash is in the middle of an operation.

The ratio of IPP_FLASH_CLK to HCLK frequency is required to be an integer greater than or equal to 2. IPP_FLASH_CLK and HCLK must be synchronous.

The boot loader is part of the NAND Flash control submodule.

For NAND Flash programming or read operations, the RAM buffer address (RBA) field in the RAM_BUFFER_ADDRESS register is set according to page size as follows:

- 512-byte page:

  Every page is written to 512 bytes in the NFC internal RAM buffer. The RBA is chosen to indicate which 512 bytes the page is written to/read from. After the program/read operation is done, RBA remains unchanged.
- 2-Kbyte page:

  In this case, the NFC programs/reads to/from 4 sections of the internal RAM. Legal values of RBA are 0b0000 or 0b0100. After the program/read operation is done, the RBA automatically increments to the next legal value (0b0000 changes to 0b0100, and vice versa)
- 4-Kbyte page:

  In this case, the NFC programs/reads to/from the entire internal RAM. The only legal value of RBA is 0b0000.

The error correcting code (ECC) is always calculated separately for each 512-byte section. See Section 36.5.6, “Reed-Solomon Error Correcting Code Engine (ECC Engine)” for more details.

Figure 36-17, Figure 36-18, and Figure 36-19 show timing diagrams for NAND Flash read, program, and erase operations.

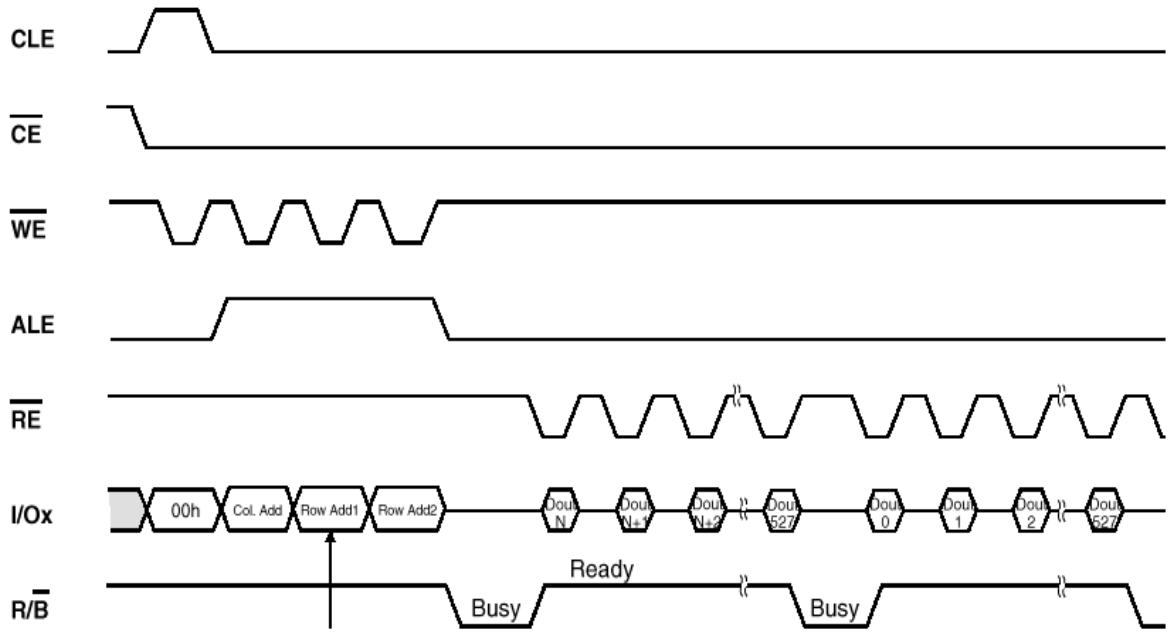


Figure 36-17. Read Operation

Note: ALE can be asserted and negated separately for each address phase.

PAGE PROGRAM OPERATION

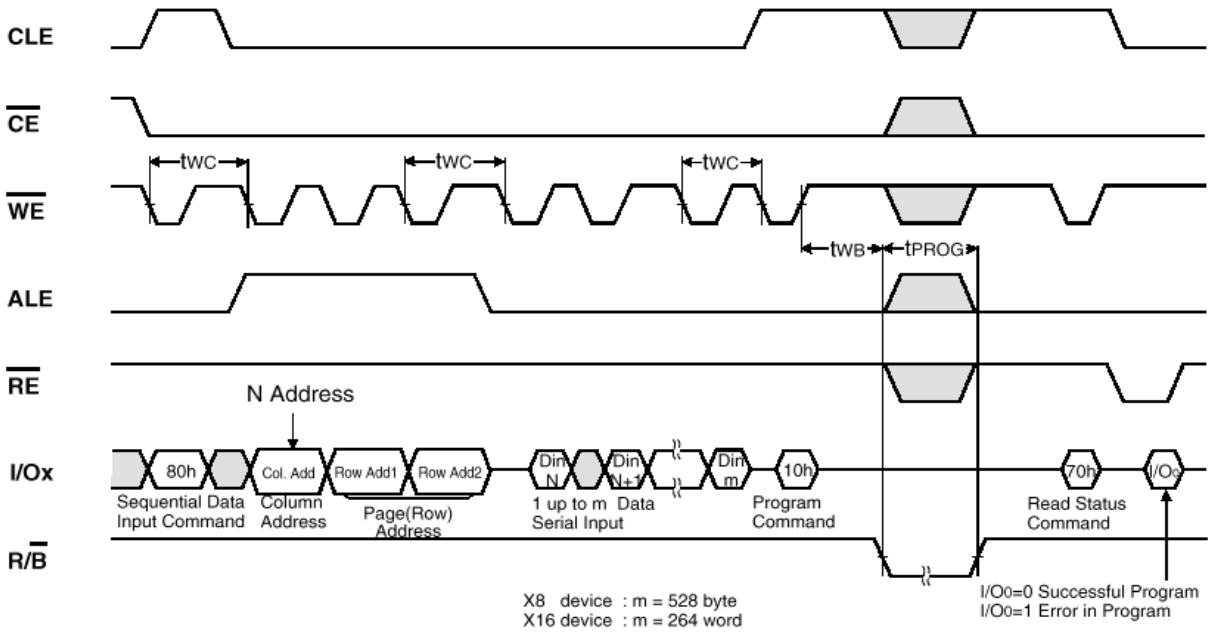


Figure 36-18. Program Operation

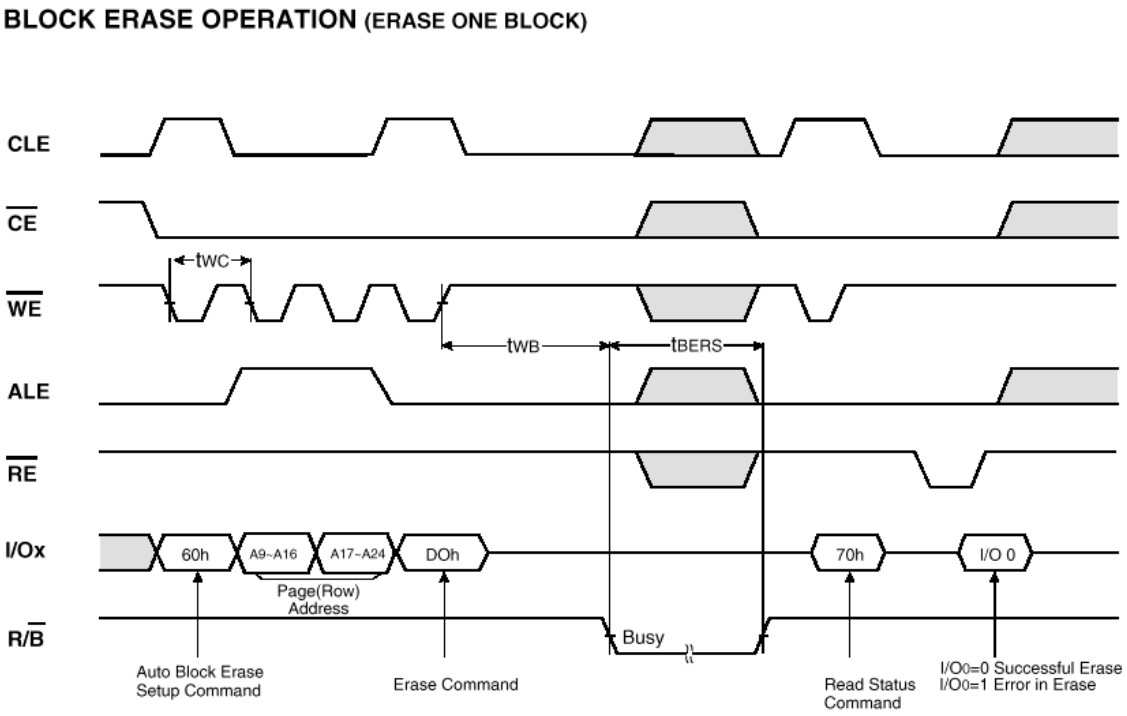


Figure 36-19. Erase Operation

36.5.5 DMA Request Operation

DMA request signal triggers in the following cases:

- After reading a page from the NAND Flash (based on DMA_MODE-bit configuration).
- After read ID operation.
- After reading spare only area.

When asserted, the NFC asserts the DMA_REQ signal for 8 Flash clock cycles, and then it is negated automatically.

36.5.6 Reed-Solomon Error Correcting Code Engine (ECC Engine)

The error correcting code (ECC) engine inside NFC can correct 4 or 8 erroneous 9-bit symbols, depending on the ECC_MODE configuration bit:

- When ECC_MODE = 0, NFC detects and correct up to 8 9-bit symbols per 538 bytes of data (512 bytes main data + 26 bytes spare data). This mode can only be used with devices with sufficient spare area (at least 26 bytes per section).
- When ECC_MODE = 1, NFC detects and correct up to 4 9-bit symbols per 528 bytes of data (512 bytes main data + 16 bytes spare data).

When the NFC accesses the NAND Flash device for a program operation, it generates ECC codes of 9/18 bytes (these bytes will override the corresponding spare area data that is written in the internal RAM).

When the NFC accesses the NAND Flash device for a read operation, it performs a RS detection algorithm, which indicates how many symbol errors were detected and corrected.

The ECC code is updated by the NFC automatically. After a read operation, the host can determine the number of errors per 528/538 bytes by reading the status registers ECC_STATUS_RESULT*n*. During a program operation the RS ECC bytes are written to the NAND Flash device, but not to the internal RAM. The host can obtain the ECC from the NAND Flash device spare area.

The ECC is always calculated per 512 bytes. If memory with 2-Kbyte (or 4-Kbyte) page size is used, the ECC is calculated 4 (or 8) times, respectively. NAND Flash with larger page sizes is filled with repeated writes of 512 bytes data followed by spare data:

512 bytes main + spare bytes + 512 bytes main + spare bytes + 512 bytes main + spare bytes + ...

The ECC operation can be bypassed using ECC_EN bit in CONFIG1 register.

The NAND Flash device programming operation proceeds as follows:

1. Write a page of data to the internal RAM.
2. Set RBA (RAM_BUFFER_ADDRESS register) to point to the section in which the data is in.
3. Configure the command and address sequence for program operation.
4. Program one page of data by writing 1 to the FDI bit in NAND_FLASH_CONFIG2 register (the RS ECC code will override the corresponding spare area bytes).

Read sequence:

1. Set RBA to point to the section in which you want the data to be written in the internal RAM.
2. Configure the command and address sequence for read operation.
3. Read 1 page by writing 1 to the FDO field in the NAND_FLASH_CONFIG2 register (the RS algorithm will automatically fix correctable symbol errors).

36.5.7 Address Control Module

The address control module is responsible for address control and generation. The module has the following functions:

- Defines the RAM buffer address generation (RAM buffer address for data in/ data out).
- Generates and takes into account the lock state sequence (see Section 36.6.4, “Write Protection Operation”) and therefore contains the Flash memory lock address comparator, and RAM buffer lock address comparator which are used to determine if this area is protected or not.
- Generates the RAM buffer address for boot load.

36.5.8 RAM Buffer (SRAM)

The internal RAM buffer is a 4608 byte (4.5 Kbyte) single-port RAM buffer which is a synchronous high-performance design. This memory has 1152 32-bit words, from which 1024 words are used for the main buffer and the remaining 128 words are allotted to a spare area that is used for error correction and other applications.

The NFC logically divides the RAM into 8 sections of 512 bytes for main data and 64 bytes for spare data. When reading (or programming) the NAND device, the NFC writes the main data from the NAND device into the main section, and the spare data from the NAND device into the spare section. If the NAND device spare-area is less then 64 bytes per 512 bytes of main data, then, the NFC's spare-section in the RAM will not be fully utilized, and some data in the spare section will not be valid.

For example, a NAND device with 2 Kbytes of main area and 64 bytes of spare area has 16 bytes of spare area per 512 bytes of main area. In this case, the first 16 bytes of spare area will be located in spare section 1, the next 16 bytes in spare section 2, and so on.)

If a NAND device is used in which the spare area size is not evenly divisible by the number of main sections, then this remainder is located at the last spare section. For example, if a NAND device is used with a 4-Kbyte page and 218 bytes of spare area (27.25 bytes of spare area for each 512 bytes of main area), then the NFC has 26 bytes per section (the largest even number less than 27.25). The remaining 10 bytes of spare area will be added the last spare section, so that the last spare section has 36 bytes.

This memory is used as a boot RAM memory during boot from the NAND Flash device, and as a buffer during normal operation.

36.5.9 Read and Write Control

The read and write control block contains a connection to the internal bus (which is connected to the Internal RAM buffer and the registers).

It is also responsible for RAM buffer Control and Register Control, RAM buffer Lock Control and Address and Data latches.

36.5.10 Data Output Control

This module defines the data output of 16-bits to the internal bus which is driven to the AHB interface. It includes RAM buffer data output, register data output and RAM buffer synchronization for the read mode pipeline.

36.5.11 Host Control

This module defines host control which is connected to the AHB interface though the internal bus. It detects the chip enable signals, controls the reset and output enable signals, and generates the $\overline{\text{SRAM\_WE}}$ signal.

36.5.12 AHB Bus Interface

The AHB bus interface is an adapter between ABMA AHB bus and the internal bus. 16-bit and 32-bit AHB bus widths are supported for both burst and non-burst operation. The internal bus width is 32 bits.

36.5.12.1 Big/Little-Endian Operation

The AHB bus interface supports both big- and little-endian data types. The NFC_ENDIAN pin controls the endian mode. Only the AHB side is controlled by the NFC_ENDIAN pin.

The endianness between the internal RAM and external NAND Flash devices is controlled by the NFC_BIG bit in CONFIG1 register.

36.5.12.2 Burst Access Support

When a data transaction from the AHB is a burst it creates a synchronous burst on the internal bus. Table 36-23 lists NFC supported access burst types.

Table 36-23. NAND Flash Burst Access Support

HBURST	BURST TYPE	SUPPORTED	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	No	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	No	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	No	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

NOTE

NFC supports bursts of 16/32-bit words only. Bursts of byte words (8-bits) are not supported.

36.5.13 I/O Pin Sharing

The NAND Flash controller has logic that allows it to share I/O pins with signals of another memory controller. For example, the 16 I/O signals of the NAND Flash controller share I/O pins with the data signals of the WEIM when interfacing to the PSRAM.

The arbitration between the NFC and the other memory has hard priority favoring the other memory. When another memory requests the bus, the NFC halts its operation as soon as possible, and the other memory is granted access to the pins. The only NFC operations that do not halt in the middle are short operations such as command, address phase or spare-area access. This priority-based arbitration mechanism can cause long delays in NFC accesses when the I/O bus is shared with another memory that is accessed frequently.

36.6 Initialization/Application Information

This section describes how to operate the NFC using its registers and its interrupts, and covers the following topics:

- Section 36.6.1, “Normal Operation,” provides instructions on to operate a NAND Flash device using the NFC.
- Section 36.6.2, “ECC Operation” describes the NFC’s error-correction operations.

- Section 36.6.3, "Symmetric Mode—One Flash Clock Cycle per Input or Output Data Cycle" describes symmetric mode, which accommodates lower NAND Flash frequencies.
- Section 36.6.4, "Write Protection Operation" describes write protection, which is used when the programmer wishes to protect part of the NAND Flash device memory from being written except in certain cases. There are two levels of protection: software (for frequently-changed memory locations), and hardware (for memory locations whose contents are rarely changed).

36.6.1 Normal Operation

Normal operations are composed of fundamental building block operations, in addition to specific operations, as shown in Figure 36-20 through Figure 36-25.

36.6.1.1 Fundamental Building Block Operations

36.6.1.1.1 Preset Operation

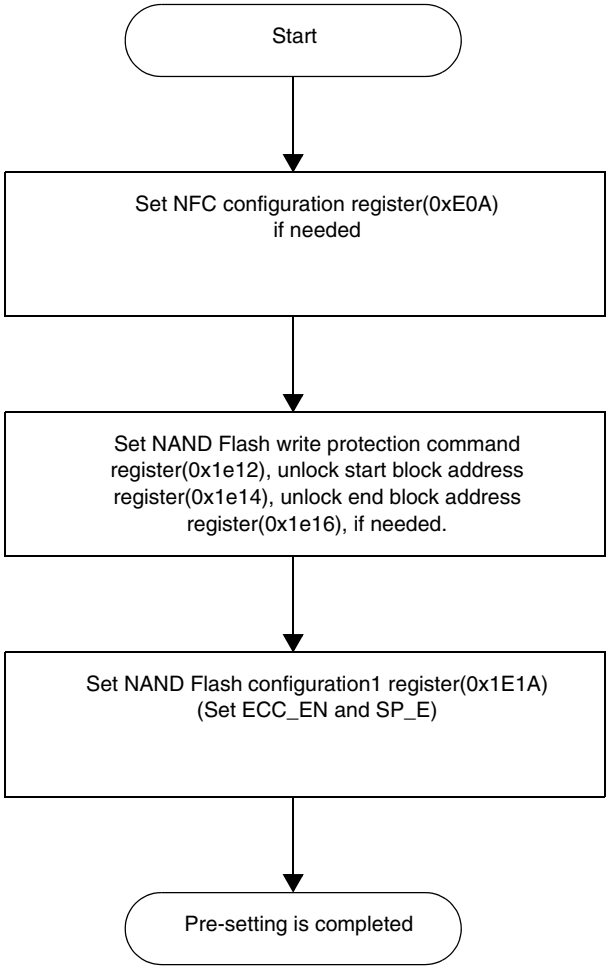


Figure 36-20. Flow Chart of Preset Operation

36.6.1.1.2 NAND Flash Command Input Operation

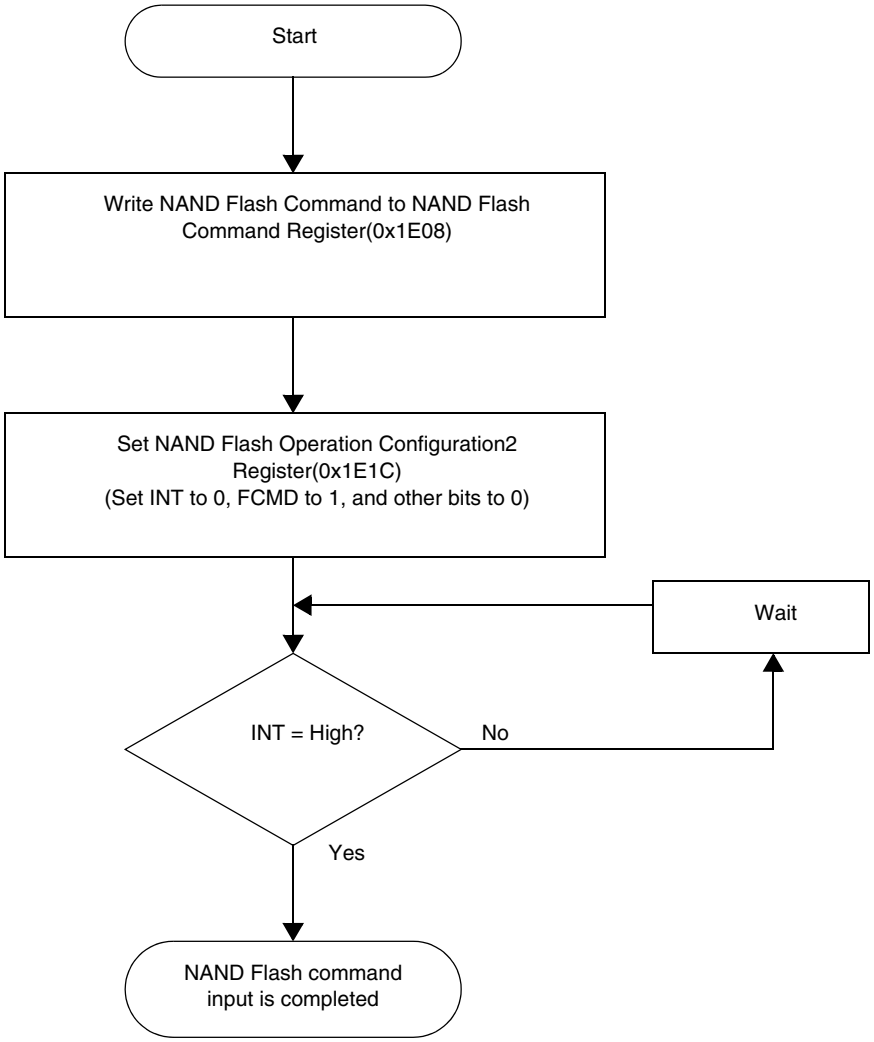


Figure 36-21. Flow Chart of NAND Flash Command Input Operation

36.6.1.1.3 NAND Flash Address Input Operation

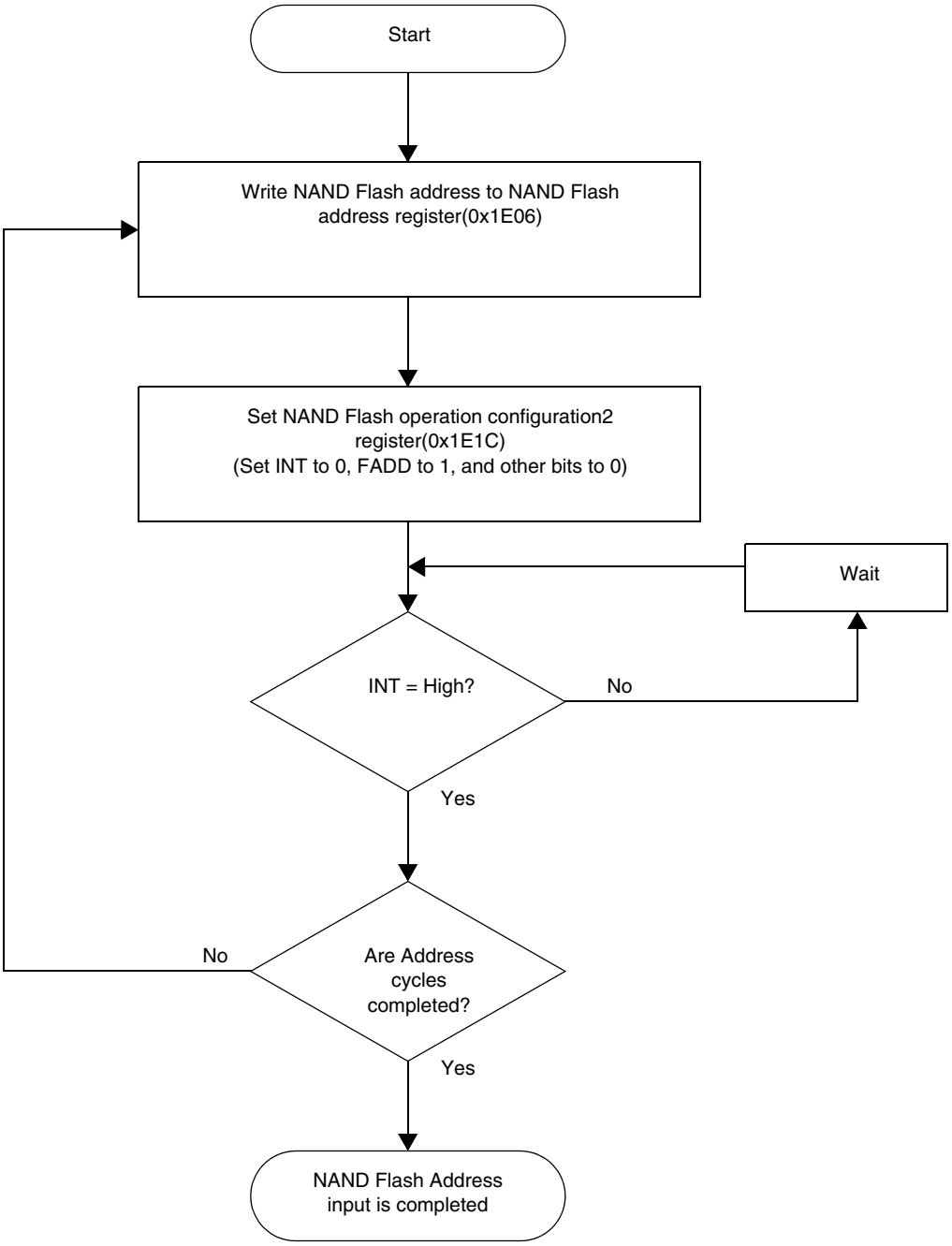


Figure 36-22. Flow Chart of NAND Flash Address Input Operation

36.6.1.1.4 NAND Flash Data Input (Program) Operation

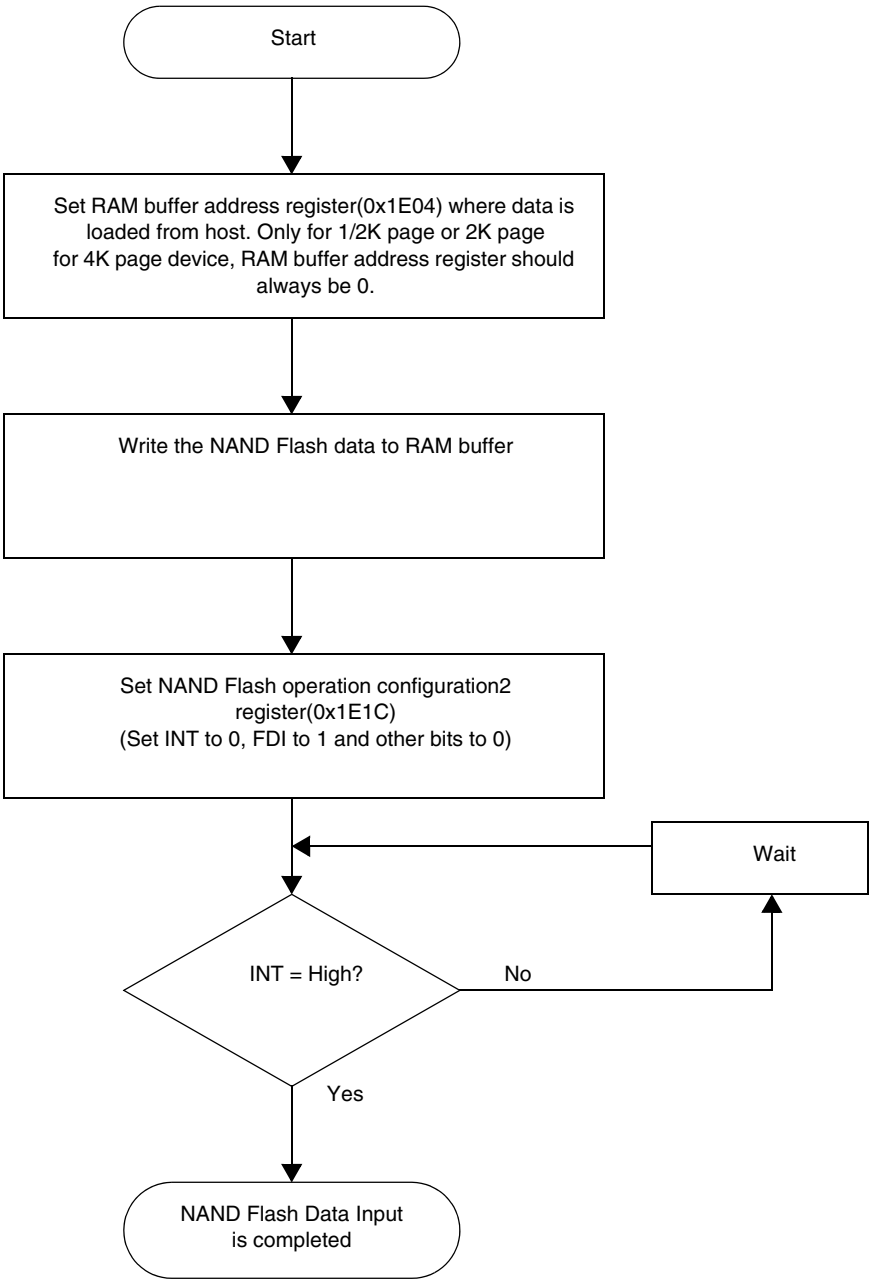


Figure 36-23. Flow Chart of NAND Flash Data Input Operation

36.6.1.1.5 NAND Flash Data Output Operation (Read)

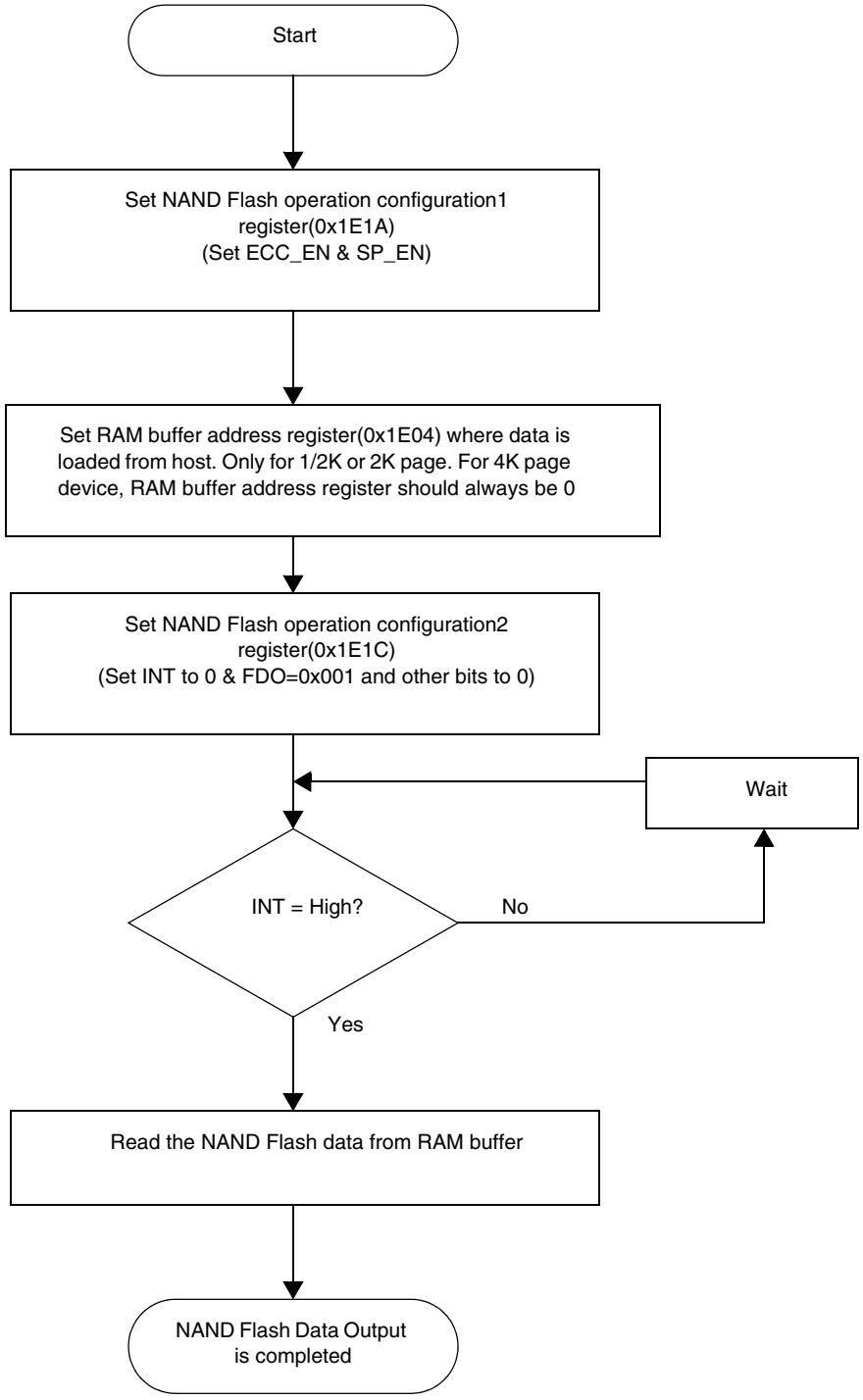


Figure 36-24. Flow Chart of NAND Flash Data Output Operation

36.6.1.2 Read NAND Flash ID Read Operation

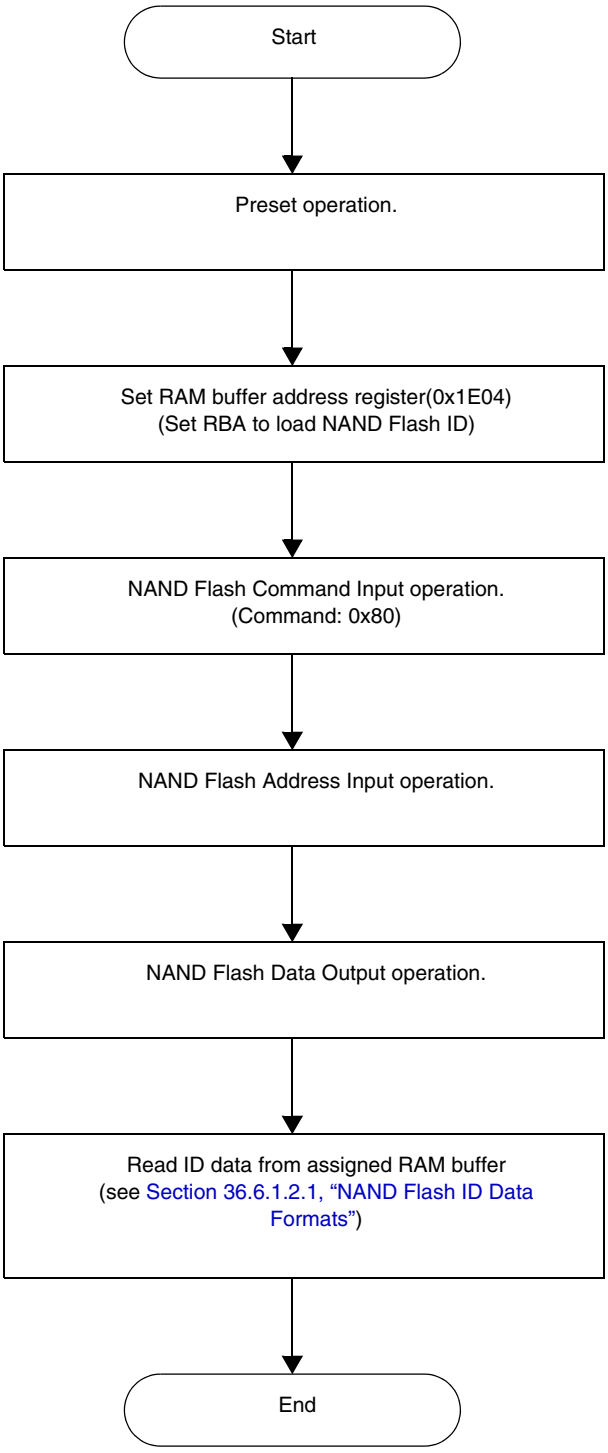


Figure 36-25. Flow Chart of Read NAND Flash ID Operation

36.6.1.2.1 NAND Flash ID Data Formats

The format of NAND Flash ID data stored in the RAM buffer (for x8 NAND Flash) is shown in Figure 36-26

RAM buffer of RBA address	Half-word 1				Half-word 2				Half-word 3				
	ID byte 1		ID byte 2		ID byte 3		ID byte 4		ID byte 5		ID byte 6		
	LSB	…	…	MSB	…	…	…	…	…	…	…	…	

Figure 36-26. NAND Flash ID Data Format (x8)

The format of NAND Flash ID data stored in the RAM buffer (for x16 NAND Flash) is shown in Figure 36-27.

RAM buffer of RBA address	Half-word 1				Half-word 2				Half-word 3				
	ID byte 1		0x*nn*		ID byte 2		0x*nn*		ID byte 3		0x*nn*		
	LSB	…	…	MSB	…	…	…	…	…	…	…	…	

RAM buffer of RBA address	Half-word 4				Half-word 5				Half-word 6				
	ID byte 4		0x*nn*		ID byte 5		0x*nn*		ID byte 6		0x*nn*		
	LSB	…	…	MSB	…	…	…	…	…	…	…	…	

Figure 36-27. NAND Flash ID Data Format (x16)

36.6.1.3 NAND Flash Status Read Operation

Figure 36-28 shows the steps in the read NAND Flash status operation.

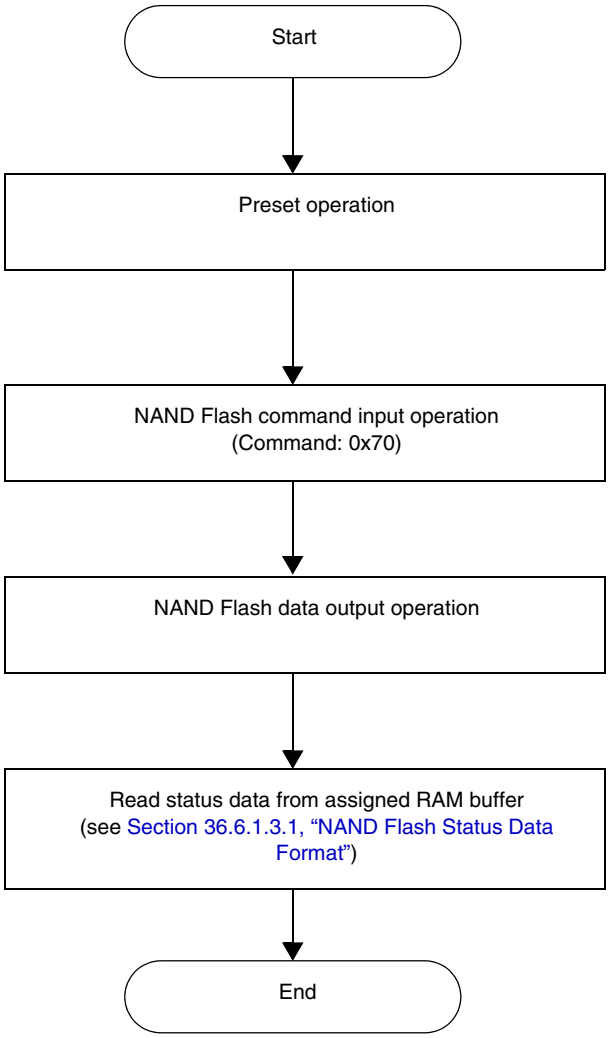


Figure 36-28. Flow Chart of Read NAND Flash Status Operation

36.6.1.3.1 NAND Flash Status Data Format

The assignment of NAND Flash status data stored in the RAM buffer (for both x8 and x16 NAND Flash) is shown in Figure 36-29.

RAM buffer of RBA address	first half-word				- - - - - - - -
	Status byte 1		0x*nn*		
	LSB			MSB	

Figure 36-29. NAND Flash Status Data Format

36.6.1.4 Read NAND Flash Data Operation

Figure 36-30 shows read NAND Flash data operation.

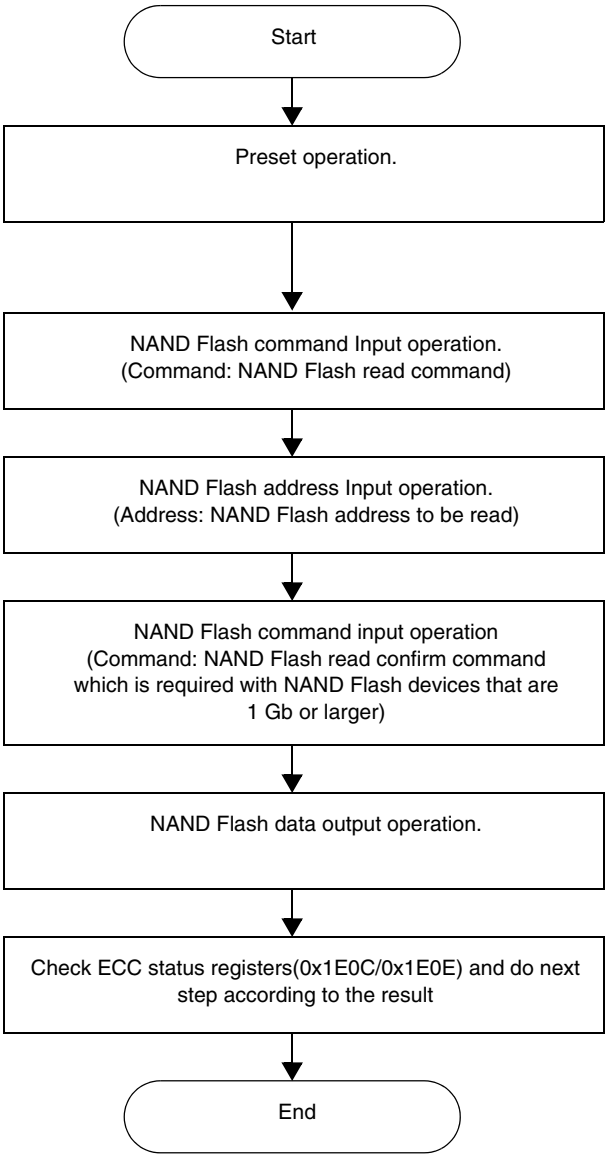


Figure 36-30. Flow Chart of Read NAND Flash Data Operation

36.6.1.5 Program NAND Flash Data Operation

Figure 36-31 shows program NAND Flash data operation.

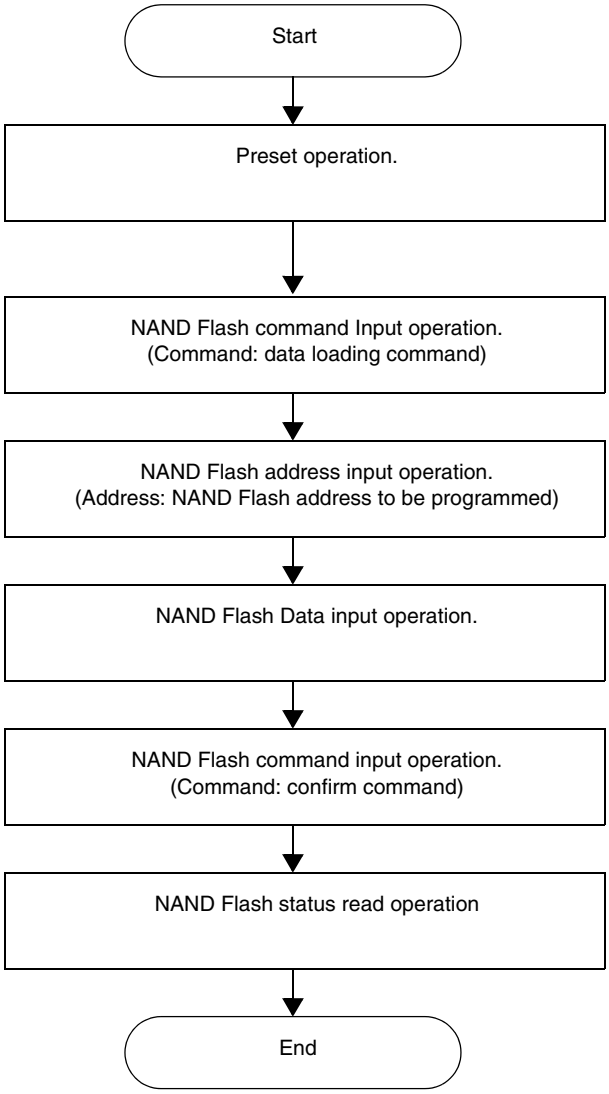


Figure 36-31. Flow Chart of Program NAND Flash Data Operation

36.6.1.6 Erase NAND Flash Data Operation

Figure 36-32 shows erase NAND Flash data operation.

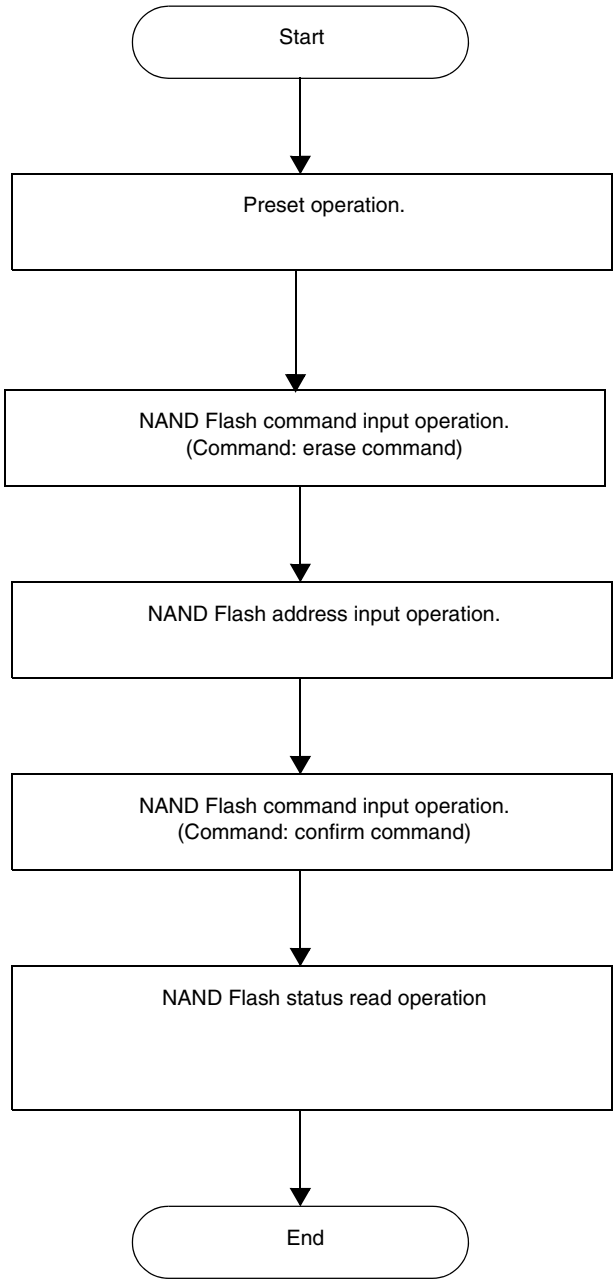


Figure 36-32. Flow Chart of Erase NAND Flash Operation

36.6.1.7 HOT Reset (Controller and NAND Flash Reset)

A reset causes the NFC and the NAND Flash device to cease their current operation, and causes the internal registers to revert to their default state.

Figure 36-33 shows the reset operation timing.

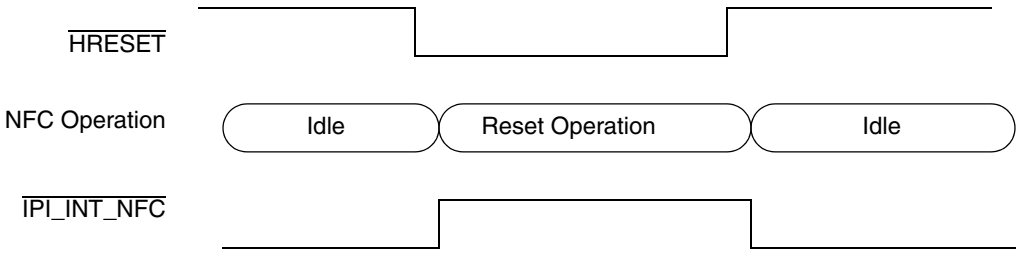


Figure 36-33. Reset Operation Timing

Figure 36-34 shows the reset operation sequence.

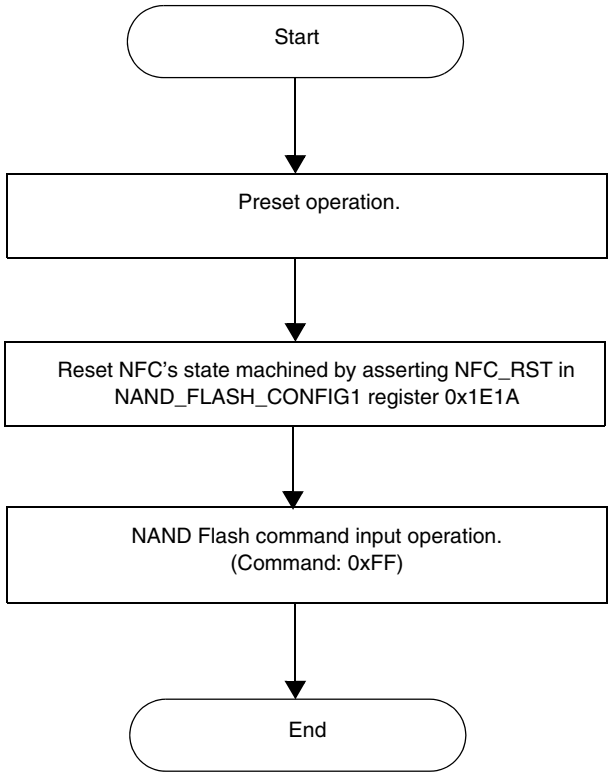


Figure 36-34. Flow Chart of Reset Operation

36.6.2 ECC Operation

36.6.2.1 ECC Normal Operation

When the NFC accesses the NAND Flash device for program operation, it generates ECC code (9/18 bytes for each 528/538 bytes). Since the generated ECC code is not updated to the internal buffer RAM, but is updated to the NAND Flash spare area immediately upon program operation, the host can read the generated ECC code only from the NAND Flash spare area.

When the NFC accesses the NAND Flash device for a read operation, it reads the ECC code, detects the number of errors and their position and corrects up to four/eight symbols (9-bits each) if applicable. Table 36-24 shows the ECC code assignment of the NAND Flash spare area. This ECC code is updated by NFC automatically. After the read operation, the host can determine whether there are errors or not by reading the ECC_STATUS_RESULT*n* registers.

In order to generate ECC and carry out the correction by the NFC,

- Program with ECC operation / Read with ECC operation

In order to generate ECC by the NFC and carry out the correction by the host,

- Program with ECC operation / Read without ECC operation

36.6.2.2 ECC Bypass

In ECC bypass operation, the spare area is copied from NFC internal RAM buffer to the NAND Flash device during program operation. During read operations the ECC detect-fix mechanism does not work, and the status register is not updated.

Table 36-24. ECC Code/Result Readability

Operation	Read Operation		Program Operation		
	ECC Code from Spare Area Buffer	ECC Status Register	ECC Code from Spare Area Buffer	ECC Status Register	ECC Content in NAND Flash Device
ECC operation	ECC code copied from NAND Flash device spare area	Valid	Invalid (old data[1])	—	ECC code generated by the NFC
ECC bypass	User data copied from NAND Flash device spare area	not Valid	Invalid (old data)	—	Data from spare area of NFC's internal RAM

Note:

1 The ECC code in the spare buffer is not updated during program operations, so the ECC code in the spare area buffer is obsolete.

36.6.3 Symmetric Mode—One Flash Clock Cycle per Input or Output Data Cycle

In the NFC's default state, two Flash clock cycles are used for each access of RE# or WE#. By setting the SYM bit in CONFIG1 register, the WE# and RE# periods during read or program change to one Flash clock cycle instead of two. In this way, lower Flash clock frequencies can be accommodated. The SYM bit also changes the RE# duty cycle to be about 50% during read operation.

In symmetric mode (SYM = 1) the data is latched into NFC on the falling edge of RE#. In default mode (SYM = 0), the data is latched on rising edge of RE#.

Figure 36-36, Figure 36-38, and Figure 36-37, and Figure 36-35 show timing diagrams for input/output operations with different SYM bit settings.

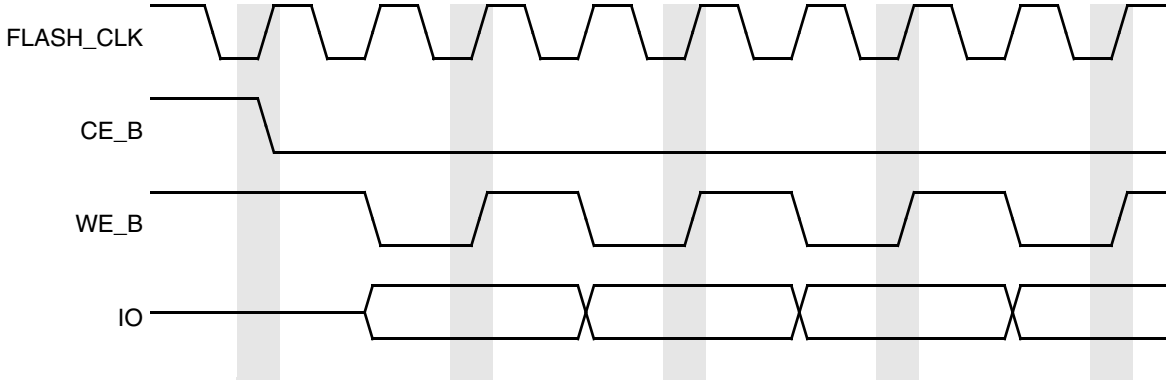


Figure 36-35. Two Flash Clock Cycles per Data Input (SYM bit =0)

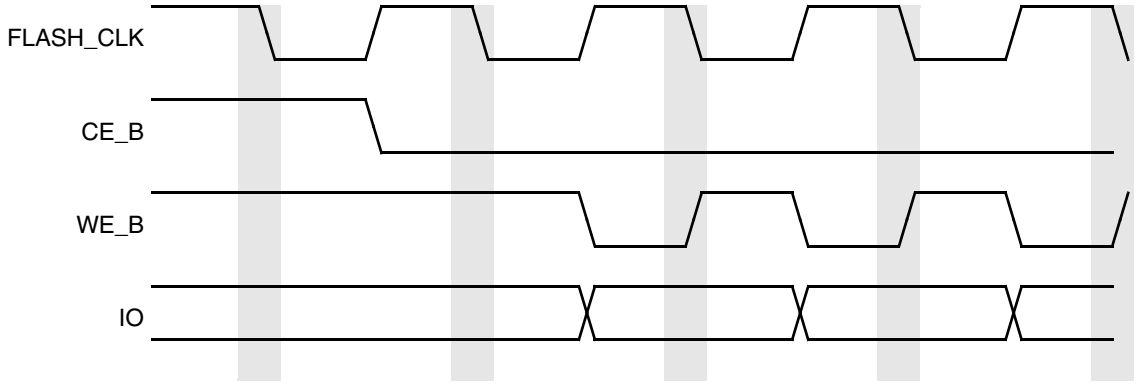


Figure 36-36. One Flash Clock Cycle per Data Input (SYM bit =1)

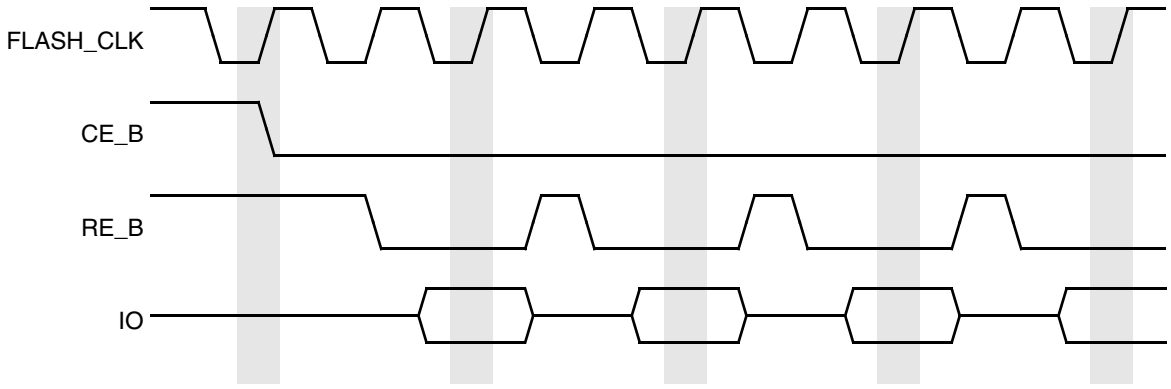


Figure 36-37. Two Flash Clock Cycles per Data Output (SYM bit =0)

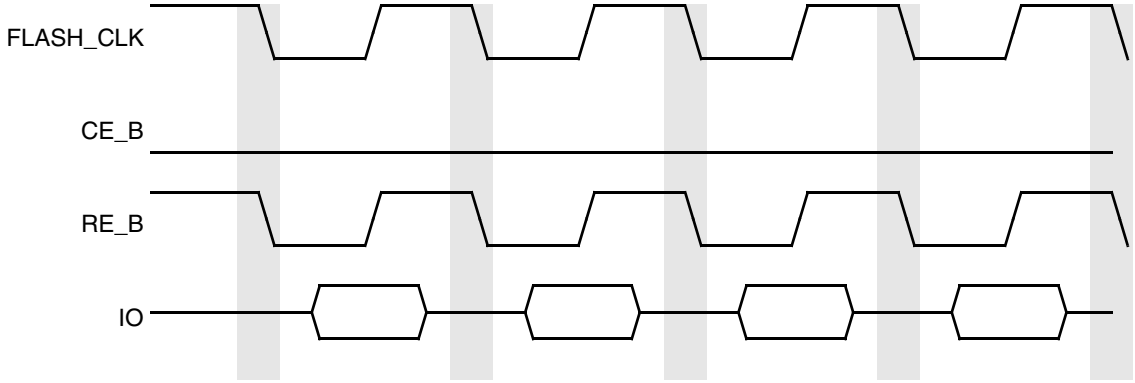


Figure 36-38. One Flash Clock Cycle per Data Output (SYM bit =1)

36.6.4 Write Protection Operation

The NFC offers software and hardware write-protection features. Both are described in this section.

36.6.4.1 Write Protection for RAM Buffer (LSB 2 Kbytes)

The NFC offers a software write protection feature for the first 2 Kbytes (+ accompanied spare area data) of the RAM buffer, which protects RAM buffer data. This write protection is carried out by setting the WPC bit of the NF_WR_PROT register.

The default state is locked state, and the first 2 Kbytes go to this state after a cold or warm reset.

Write protection availability for main/spare memory regions in the RAM buffer is described on Table 36-25. Figure 36-39 shows a state diagram of RAM buffer write protection.

Table 36-25. Write Protection for Main/Spare RAM Buffer

Main Area	Spare Area	
1st section of RAM buffer	1st section of RAM buffer	Write Protection Available
2nd section of RAM buffer	2nd section of RAM buffer	
3rd section of RAM buffer	3rd section of RAM buffer	
4th section of RAM buffer	4th section of RAM buffer	
5th section of RAM buffer	5th section of RAM buffer	Write Protection not available
6th section of RAM buffer	6th section of RAM buffer	
7th section of RAM buffer	7th section of RAM buffer	
8th section of RAM buffer	8th section of RAM buffer	

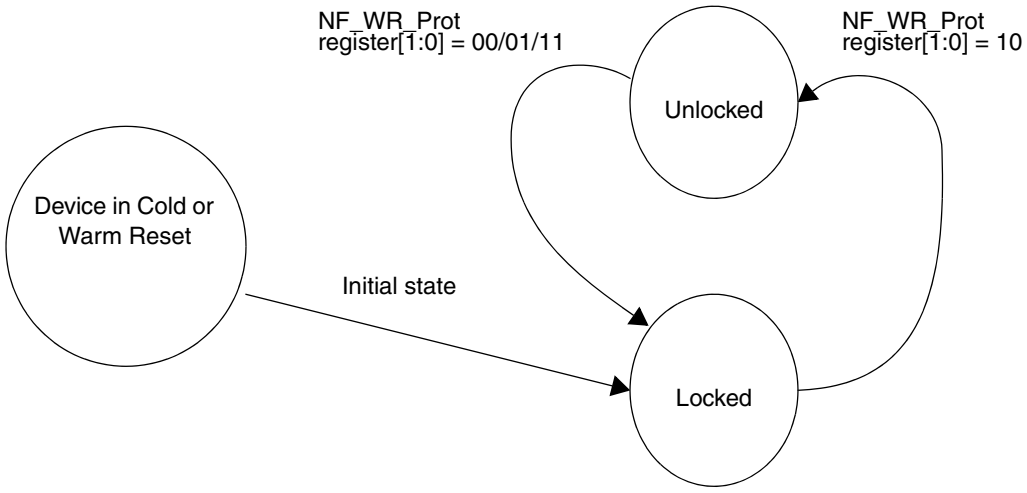


Figure 36-39. State Diagram of RAM Buffer Write Protection

36.6.4.2 Write Protection Modes

The NFC offers both hardware and software write protection options for the NAND Flash device. The software write protection feature is used by executing the LOCK BLOCK command or LOCK-TIGHT BLOCK command. The hardware write protection feature is used by executing a cold or warm reset. The WP# signal is asserted only upon POR.

36.6.4.3 Write Protection Commands

There are two write protection states: locked and lock-tight.

- Locked state means that memory block in question is write-protected (cannot be written to), but the UNLOCK command can unlock it. This is useful for frequently-changed memory blocks.

- Lock-tight state is a higher level of protection, and means that the memory block in question is write protected, but the UNLOCK command cannot unlock it. This is useful for memory blocks whose contents are rarely changed.

The followings summarizes the locking functionality.

— - All blocks power up in a locked state except block zero. The UNLOCK command can unlock these blocks.

— - The LOCK-TIGHT BLOCK command locks blocks so that they cannot be unlocked.

– Lock-tight state can be changed to locked state only when cold or warm reset is executed.

— - Writing to the unlock start/end address registers (UNLOCK_START_BLK_ADD and UNLOCK_END_BLK_ADD) while the NFC is in the lock-tight state does not affect the unlock address.

36.6.4.4 Write Protection Status

The current write protection status of the NFC can be read in NAND Flash write protection status register (NAND_FLASH_WR_PR_ST). The unlock status, lock status, and lock-tight status bits (US*n*, LS*n*, and LTS*n*) are not cleared by hot reset. These write protection status bits are updated only when a command is issued to the NAND device.

Figure 36-40 shows a state diagram for the write protection of the NFC.

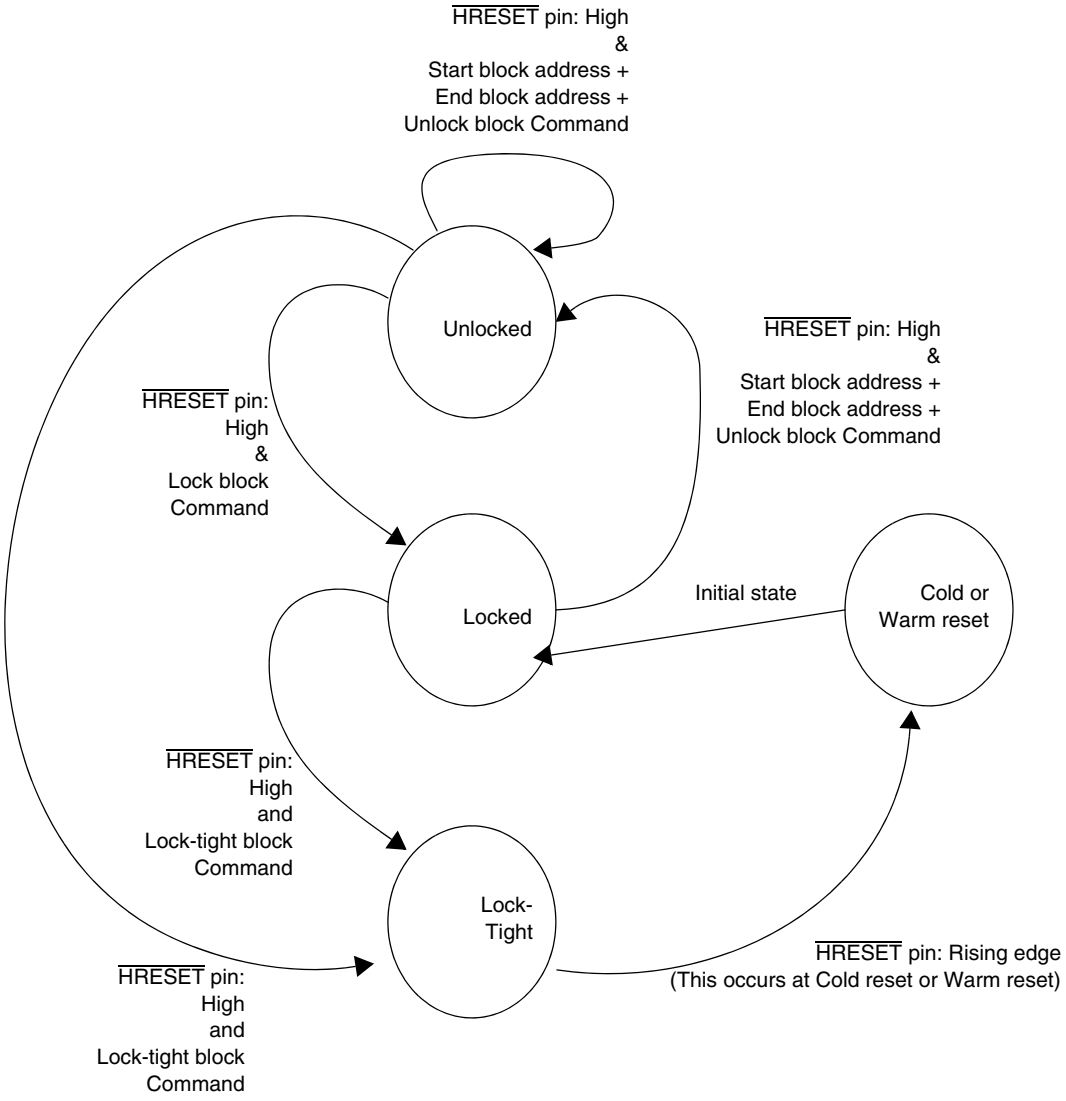


Figure 36-40. State diagram of NAND Flash Write Protection

36.6.4.4.1 Lock Sequence

The following describes the lock sequence:

1. Command sequence: LOCK BLOCK command (0x02)
2. All blocks default to locked after initial cold reset or warm reset.
3. Locking some of the blocks is not available; all memory blocks are locked upon reset except block zero.
4. Unlocked memory blocks can be locked by using the LOCK BLOCK command. The status of a locked memory block can be changed to unlocked or lock-tight using the appropriate software commands.

36.6.4.4.2 Unlock Sequence

The following describes the unlock sequence:

1. Command sequence: start block address + end block address + UNLOCK BLOCK command (0x04)
2. Unlocked blocks can be programmed or erased
3. The status of an unlocked block can be changed to locked or lock-tight using the appropriate software commands
4. Only one consecutive area can be released to unlocked state from locked state; Unlocking of nonconsecutive blocks is not available.

36.6.4.4.3 Lock-tight Sequence

The following describes the “lock-tight” sequence:

1. Only locked blocks can be “locked-tight” by the LOCK-TIGHT BLOCK command.
2. Command sequence: LOCK-TIGHT BLOCK command (0x01)
3. Lock-tight blocks revert to the locked state at cold/warm reset.

36.6.5 Memory Configuration Examples

The following figures show memory connections for various bit widths. An 8-bit configuration example is shown in Figure 36-41 and a 16-bit configuration example is shown in Figure 36-42.

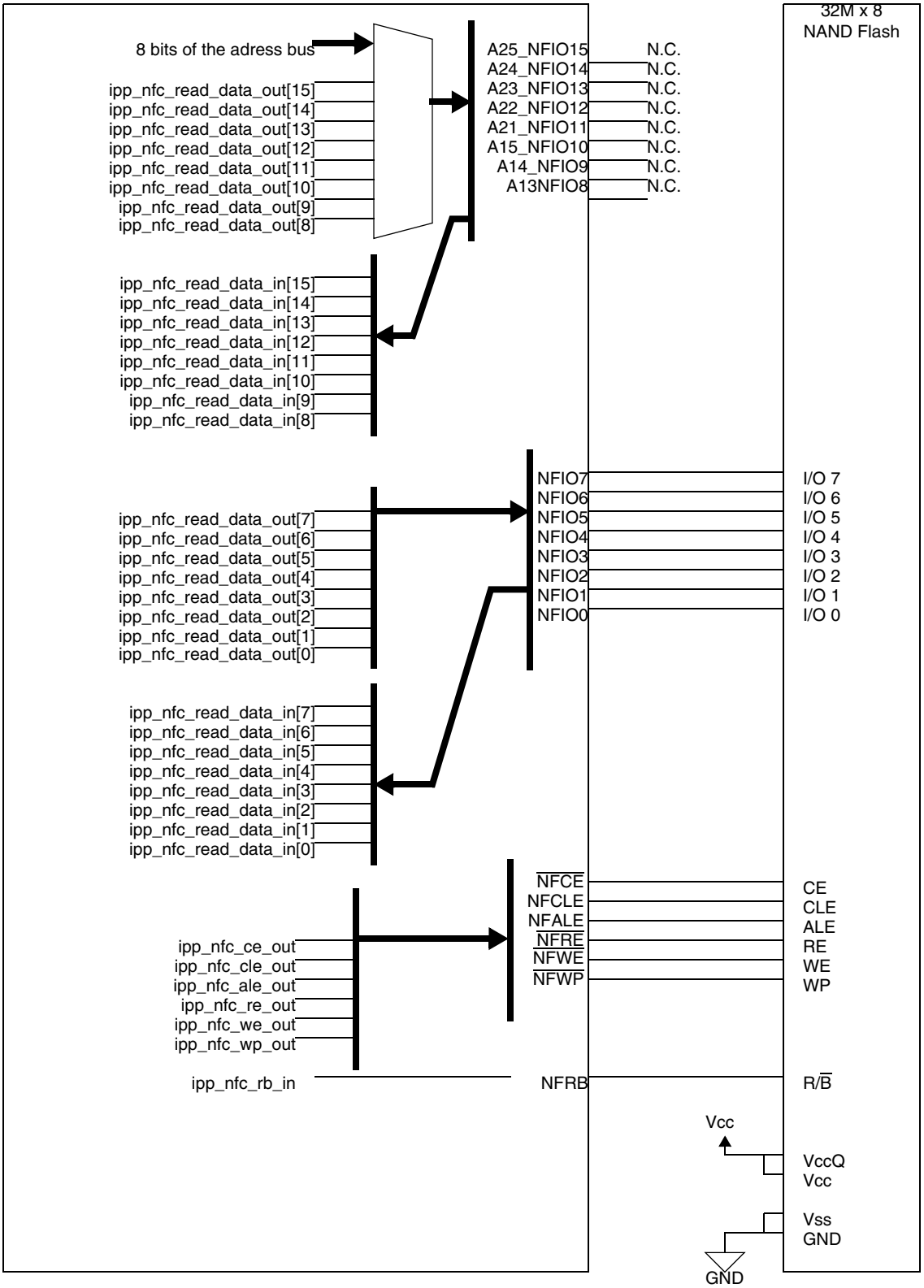


Figure 36-41. 256 Mbit (32 M x 8-bit) NAND Flash Connection Diagram

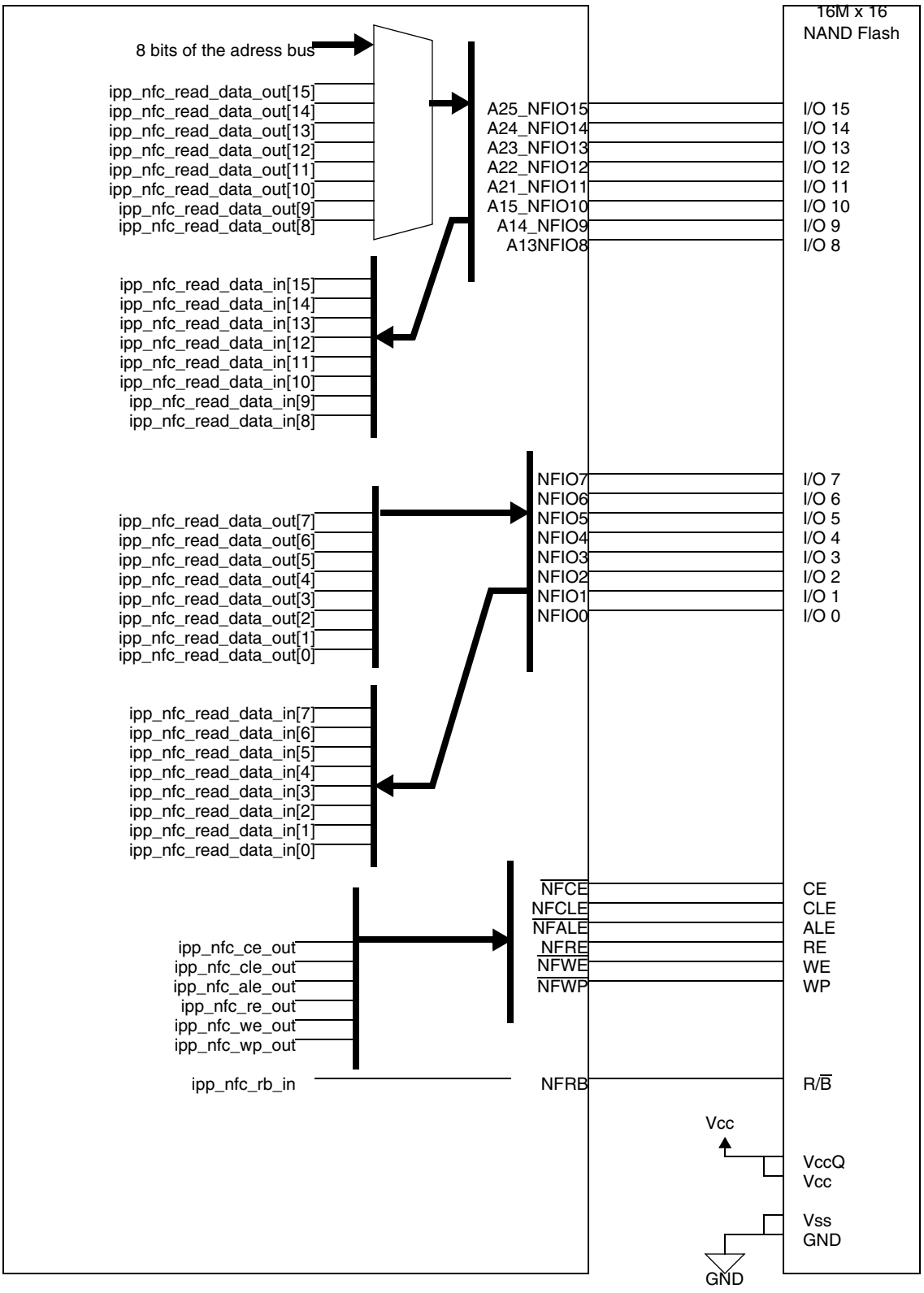


Figure 36-42. 256 Mbit (16 M x 16-bit) NAND Flash Connection Diagram

36.6.6 Verified NAND models.

The following part number models have been verified and are guaranteed to be supported by the NFC:

- k9f1g08u0m
- k9f1g16u0m
- k9f5608q0c
- k9f5616q0c
- tc58nyg0d9bxgj5
- tc58nvg4d1dtg00

NOTE

The NFC controller can support high-speed NAND Flash by supplying higher frequencies to the Flash clock input.

Chapter 37
Pulse-Width Modulator (PWM)

The pulse-width modulator (PWM) has a 16-bit counter, and is optimized to generate sound from stored sample audio images; it can also generate tones. It uses 16-bit resolution and a 4×16 data FIFO to generate sound.

37.1 Overview

This section presents an overview of the PWM. Figure 37-1 gives the PWM block diagram.

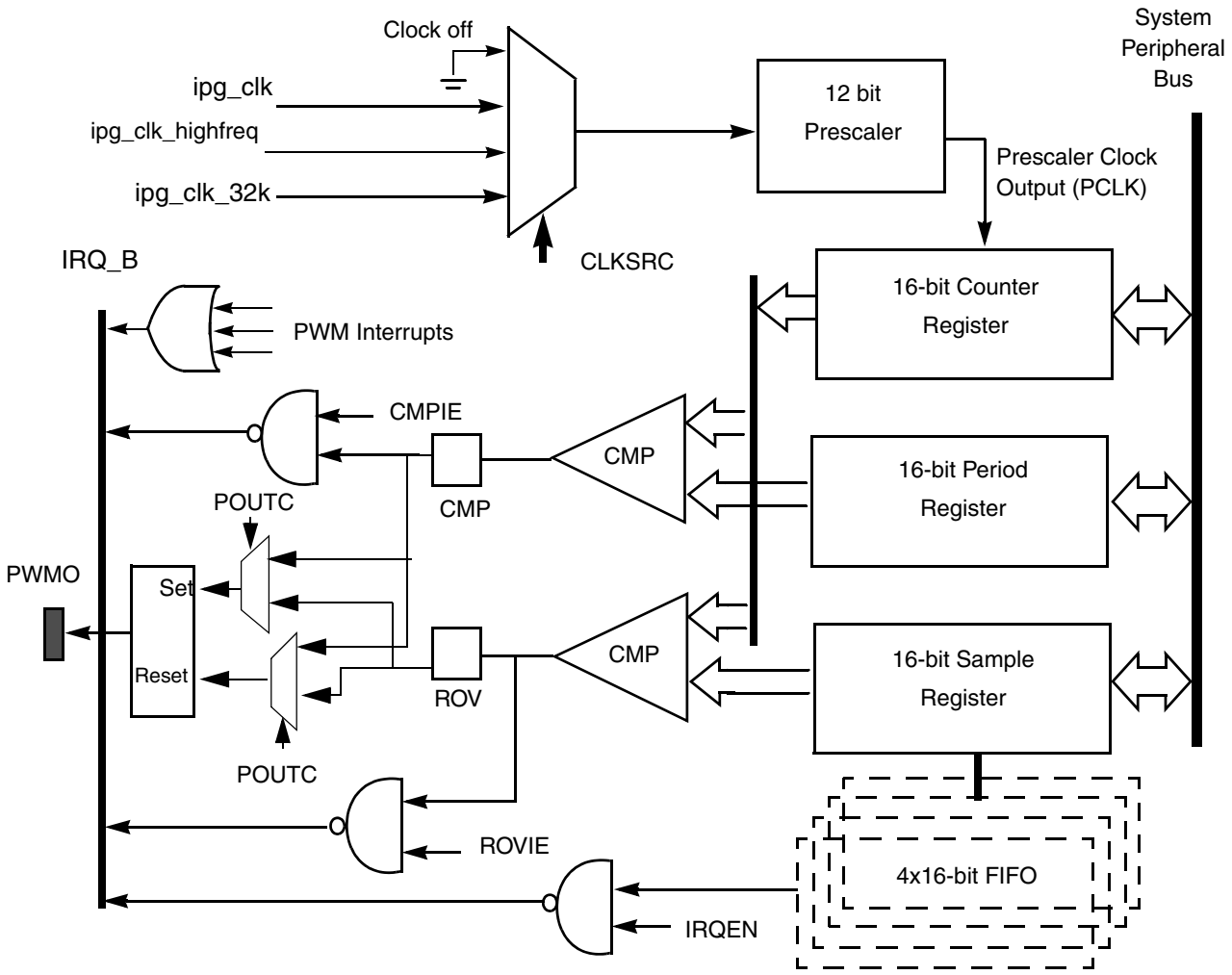


Figure 37-1. Pulse-Width Modulator Block Diagram

The following features characterize the PWM:

- 16-bit up-counter with clock source selection
- 4×16 FIFO to minimize interrupt overhead
- 12-bit prescaler for division of clock
- Sound and melody generation
- Active high or active low configured output
- Can be programmed to be active in low-power and debug modes
- Interrupts at compare and rollover

37.2 Signal Description

The PWM follows the IP Bus protocol for interfacing with the processor core. It does not have any interface signals with any other module inside the chip except for clock and reset inputs from the clock module and interrupt signals to the processor interrupt handler. There is a single output signal going outside the chip boundary.

37.2.1 External Signals

PWM has a single output signal to the chip boundary named ipp_do_pwmo.

Table 37-1 describes the external signals.

Table 37-1. External Signals

Name	Direction	Function	Reset State	Pull Up
ipp_do_pwmo	Output	This is the functional output of the PWM. The modulated signal of the module is observed at this pin. It can be viewed as a clock signal whose period and duty cycle can be varied with different settings of the PWM. The smallest period can be two ipg_clk clock periods with duty cycle of 50%	0	—

37.3 Memory Map and Register Definition

The PWM module includes six user-accessible registers. Section 37.3.2, “Register Descriptions” on page 37-4 provides the detailed descriptions for all of the PWM registers.

The PWM memory map is shown in Table 37-2.

Table 37-2. PWM Memory Map

Offset	Register	Access	Reset Value	Section/Page
0xBASE_00 (PWMCR)	PWM Control Register (PWMCR)	R/W	0x0000_0000	37.3.2.1/37-5
0xBASE_04 (PWMSR)	PWM Status Register (PWMSR)	R/W	0x0000_0008	37.3.2.2/37-7
0xBASE_08 (PWMIR)	PWM Interrupt Register (PWMIR)	R/W	0x0000_0000	37.3.2.3/37-8
0xBASE_0C (PWMSAR)	PWM Sample Register (PWMSAR)	R/W	0x0000_0000	37.3.2.4/37-8

Table 37-2. PWM Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0xBASE_10 (PWMPR)	PWM Period Register (PWMPR)	R/W	0x0000_FFFE	37.3.2.5/37-9
0xBASE_14 (PWMCNR)	PWM Counter Register (PWMCNR)	R	0x0000_0000	37.3.2.6/37-10

37.3.1 Register Summary

Figure 37-2 shows the key to the register fields, and Table 37-3 shows the register figure conventions.

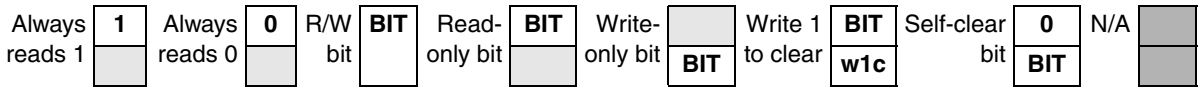


Figure 37-2. Key to Register Fields

Table 37-3. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 37-4 shows the PWM register summary.

Table 37-4. PWM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_00 (PWMCR)	R	0	0	0	0	FWM		STOPEN	DOZEN	WAITEN	DBGEN	BCTR	HCTR	POUTC		CLKSRC	
	W																
	R	PRESCALER												SWR	REPEAT		EN
	W																
0xBASE_04 (PWMSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
	W										w1c	w1c	w1c	w1c			
0xBASE_08 (PWMIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
	W																
0xBASE_0C (PWMSAR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SAMPLE[15:0]															
	W																
0xBASE_10 (PWMPR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PERIOD[15:0]															
	W																
0xBASE_14 (PWMCNR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	COUNT[15:0]															
	W																

37.3.2 Register Descriptions

This section contains the detailed register descriptions for the PWM registers.

37.3.2.1 PWM Control Register (PWMCR)

The PWM control register (PWMCR) is used to configure the operating settings of the PWM. It contains the prescaler for the clock division.

0xBASE_00 (PWMCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	FWM		STOP EN	DOZ EN	WAIT EN	DBG EN	BCT R	HCT R	POUTC		CLKSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER												SWR	REPEAT		EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-3. PWM Control Register (PWMCR)

Table 37-5. PWMCR Field Descriptions

Field	Description
31–28	Reserved.
27–26	FIFO Water Mark. These bits are used to set the data level at which the FIFO empty flag will be set and the corresponding interrupt generated 00 FIFO empty flag is set when there are more than or equal to 1 empty slots in FIFO 01 FIFO empty flag is set when there are more than or equal to 2 empty slots in FIFO 10 FIFO empty flag is set when there are more than or equal to 3 empty slots in FIFO 11 FIFO empty flag is set when there are more than or equal to 4 empty slots in FIFO
25 STOPEN	Stop Mode Enable. This bit keeps the PWM functional while in stop mode. When this bit is cleared, the input clock is gated off in stop mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in stop mode 1 Active in stop mode
24 DOZEN	Doze Mode Enable. This bit keeps the PWM functional in doze mode. When this bit is cleared, the input clock is gated off in doze mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in doze mode 1 Active in doze mode
23 WAITEN	Wait Mode Enable. This bit keeps the PWM functional in wait mode. When this bit is cleared, the input clock is gated off in wait mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in wait mode 1 Active in wait mode
22 DBGEN	Debug Mode Enable. This bit keeps the PWM functional in debug mode. When this bit is cleared, the input clock is gated off in debug mode. This bit is not affected by software reset. It is cleared by hardware reset. 0 Inactive in debug mode 1 Active in debug mode

Table 37-5. PWMCR Field Descriptions (continued)

Field	Description
21 BCTR	Byte Data Swap Control. This bit determines the byte ordering of the 16-bit data when it goes into the FIFO from the sample register. 0 byte ordering remains the same 1 byte ordering is reversed
20 HCTR	Half-word Data Swap Control. This bit determines which half word data from the 32 bit IP-Bus interface is written into the lower 16 bits of the sample register. 0 Half word swapping does not take place 1 Half words from write data bus are swapped
19–18 POUTC	PWM Output Configuration. This bit field determines the mode of PWM output on the output pin. 00 Output pin is set at rollover and cleared at comparison 01 Output pin is cleared at rollover and set at comparison 10 PWM output is disconnected 11 PWM output is disconnected
17–16 CLKSRC	Select Clock Source. These bits determine which clock input will be selected for running the counter. After reset the system functional clock is selected. The input clock can also be turned off if these bits are set to 00. This field value should only be changed when the PWM is disabled 00 Clock is off 01 ipg_clk 10 ipg_clk_highfreq 11 ipg_clk_32k
15 –4 PRESCALER	Counter Clock Prescaler Value. This bit field determines the value by which the clock will be divided before it goes to the counter. 0x000 Divide by 1 0x001 Divide by 2 ... 0xFFF Divide by 4096
3 SWR	Software Reset. PWM is reset when this bit is set to 1. It is a self clearing bit. A write 1 to this bit is a single wait state write cycle. When the module is in reset state this bit is set and is cleared when the reset procedure is over. Setting this bit resets all the registers to their reset values except for the STOPEN, DOZEN, WAITEN, and DBGEN bits in this control register. 0 PWM is out of reset 1 PWM is undergoing reset
2–1 REPEAT	Sample Repeat. This bit field determines the number of times each sample from the FIFO is to be used. 00 Use each sample once 01 Use each sample twice 10 Use each sample four times 11 Use each sample eight times
0 EN	PWM Enable. This bit enables the PWM. If this bit is not enabled, the clock prescaler and the counter is reset. When the PWM is enabled, it begins a new period, the output pin is set to start a new period while the prescaler and counter are released and counting begins. 0 PWM disabled 1 PWM enabled

37.3.2.2 PWM Status Register (PWMSR)

The PWM status register (PWMSR) contains seven bits which display the state of the FIFO and the occurrence of rollover and compare events. The FIFOAV bit is read-only but the other four bits can be cleared by writing 1 to them. FE, ROV, and CMP bits are associated to FIFO-Empty, Roll-over, and Compare interrupts, respectively.

0xBASE_04 (PWMSR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	FWE	CMP	ROV	FE	FIFOAV		
W										w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 37-4. PWM Status Register (PWMSR)

Table 37-6. PWMSR Field Descriptions

Field	Description
31–7	Reserved.
6 FWE	FIFO Write Error Status. This bit shows that an attempt has been made to write FIFO when it is full. 0 FIFO write error not occurred 1 FIFO write error occurred
5 CMP	Compare Status. This bit shows that a compare event has occurred. 0 Compare event not occurred 1 Compare event occurred
4 ROV	Roll-over Status. This bit shows that a roll-over event has occurred. 0 Roll-over event not occurred 1 Roll-over event occurred
3 FE	FIFO Empty Status Bit. This bit indicates the FIFO data level in comparison to the water level set by FWM field in the control register. 0 Data level is above water mark 1 When the data level falls below the mark set by FWM field
2–0 FIFOAV	FIFO Available. These read-only bits indicate the data level remaining in the FIFO. An attempted write to these bits will not affect their value and no transfer error is generated. 000 No data available 001 1 word of data in FIFO 010 2 words of data in FIFO 011 3 words of data in FIFO 100 4 words of data in FIFO 101 unused 110 unused 111 unused

37.3.2.3 PWM Interrupt Register (PWMIR)

The PWM Interrupt register (PWMIR) contains three bits which control the generation of the compare, rollover and FIFO empty interrupts.

0xBASE_08 (PWMIR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CIE	RIE	FIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-5. PWM Interrupt Register (PWMIR)

Table 37-7. PWMIR Field Descriptions

Field	Description
31–3	Reserved.
2 CIE	Compare Interrupt Enable. This bit controls the generation of the Compare interrupt. 0 Compare Interrupt not enabled 1 Compare Interrupt enabled
1 RIE	Roll-over Interrupt Enable. This bit controls the generation of the Rollover interrupt. 0 Roll-over interrupt not enabled 1 Roll-over Interrupt enabled
0 FIE	FIFO Empty Interrupt Enable. This bit controls the generation of the FIFO Empty interrupt. 0 FIFO Empty interrupt disabled 1 FIFO Empty interrupt enabled

37.3.2.4 PWM Sample Register (PWMSAR)

The PWM sample register (PWMSAR) is the input to the FIFO. 16-bit words are loaded into the FIFO. The FIFO can be written and read when the PWM is disabled. The PWM will run at the last set duty-cycle setting if all the values of the FIFO has been utilized, until the FIFO is reloaded or the PWM is disabled. When a new value is written, the duty cycle changes after the current period is over.

A value of zero in the sample register will result in the ipp_pwm_pwmo output signal being always low/high (POUTC =00 it will be low and POUTC = 01 it will be high), and hence no output waveform will be produced. If the value in this register is higher than the PERIOD + 1, the output will never be reset/set depending on POUTC value.

0xBASE_0C (PWMSAR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAMPLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-6. PWM Sample Register (PWMSAR)

Table 37-8. PWMSAR Field Descriptions

Field	Description
31–16	Reserved.
15–0 SAMPLE	Sample Value. This is the input to the 4x16 FIFO. The value in this register denotes the value of the sample being currently used.

37.3.2.5 PWM Period Register (PWMPR)

The PWM period register (PWMPR) determines the period of the PWM output signal (PWMO) in terms of the prescaler clock output (PCLK). After the counter value matches PERIOD + 1, the counter is reset to start another period. The frequencies of PWMO and PCLK are related by the following equation:

$$f_{PWMO} = f_{PCLK} / (PERIOD + 2) \qquad \textit{Eqn. 37-1}$$

A value of zero in the PWMPR will result in a period of two clock cycles for the output signal. Writing 0xFFFF to this register will achieve the same result as writing 0xFFFE.

A change in the period value due to a write in PWMPR results in the counter being reset to zero and the start of a new count period.

0xBASE_10 (PWMPR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 37-7. PWM Period Register (PWMPR)

Table 37-9. PWMPR Field Descriptions

Field	Description
31–16	Reserved.
15–0 PERIOD	Period Value. These bits determine the Period of the count cycle. The counter counts up to [Period Value] +1 and is then reset to 0x0000.

37.3.2.6 PWM Counter Register (PWMCNR)

The read-only pulse-width modulator counter register (PWMCNR) contains the current count value and can be read at any time without disturbing the counter.

0xBASE_14 (PWMCNR) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-8. PWM Counter Register (PWMCNR)

Table 37-10. PWMCNR Field Descriptions

Field	Description
31–16	Reserved
15–0 COUNT	Counter Value. These bits are the counter register value and denotes the current count state the counter register is in.

37.4 Functional Description

The following sections detail the PWM operation and function.

37.4.1 Operation

The output of the PWM is a toggling signal whose frequency and duty cycle can be modulated by programming the appropriate registers. It has a 16-bit up counter which counts from 0x0000 until the counter value equals the [Value in Period register] + 1. After this match occurs the Counter is reset to 0x0000.

At the beginning of a count period cycle, the PWMO pin is set to one (default) and the counter begins counting up from 0x0000. The sample value in the sample FIFO is compared on each count of prescaler clock. When the sample and count values match, the PWMO signal is cleared to zero (default). The counter continues counting until the period match occurs and subsequently another period cycle begins.

When the PWM is enabled the counter starts running and generates an output with the reset values in the period and sample registers. It is recommended that the programming of these registers be done before PWM is enabled.

A hardware reset results in all the PWM count and sample registers begin cleared and the FIFO being flushed. The control register shows that FIFO is empty and it can be written into, and the PWM is disabled. A software reset has the same results, however the state of the STOPEN, DOZEN, WAITEN, and DBGEN bits in the control register are not affected. Software reset can be asserted even when the PWM is in disabled state.

37.4.1.1 Clocks

The clock that feeds the prescaler can be selected from:

- **High-frequency Clock** (ipg_clk_highfreq) pat_ref or ckih

  This is a high frequency clock provided by the clock module (CM). This clock is supposed to be on in the low-power mode when the ipg_clk is turned off. Thus the PWM can be run on this clock in the low-power mode. The CCM is expected to provide this clock after synchronizing it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low-power mode.
- **Low-frequency Reference Clock** (ipg_clk_32k) CKIL

  This is the 32 KHz low-frequency reference clock which is provided by the CM. This clock is supposed to be on in the low-power mode when ipg_clk is turned off. Thus PWM can be run on this clock in the low-power mode. The CCM is expected to provide this clock after synchronizing

it to ahb_clk in the normal functional mode and switch to the unsynchronized version in the low-power mode.

- **Global Functional Clock** (ipg_clk)
  This clock is supposed to be on in normal operations. In low-power modes it can be switched off.

The clock input source is determined by the CLKSRC field of the PWM control register. **The CLKSRC value should only be changed when the PWM is disabled.**

A change in the value of the PRESCALER field of the control register is immediately reflected in the output clock frequency.

37.4.1.2 FIFO

Digital sample values can be loaded into the pulse-width modulator as 16-bit words. The byte order can be changed using the BCTR and HCTR bits of the control register. A 4-word (16-bit) FIFO minimizes interrupt overhead. A maskable interrupt is generated when the number of data words fall below the water level set by the FWM field in the control register.

A write in the sample register results in the value being stored into the FIFO if it is not full. A write when the FIFO is full sets FWE (FIFO write error) bit in the status register and the FIFO contents remain unchanged. The FIFO can be written into when the PWM is disabled. The FIFOAV field shows how many data words are currently contained in the FIFO and if it can be written into.

A read on the sample register yields the current FIFO value being used or will be used by the PWM for generation on the output signal. Therefore a write and a subsequent read on the sample register may result in different values being obtained.

37.4.1.3 Rollover and Compare Event

The counter is reset to 0x0000 after its value equals the PERIOD + 1 and resumes counting thereafter. This event is referred to as a rollover. When PERIOD = 0x0000, the counter is reset after count reaches 0x0001. Therefore PERIOD = 0xFFFF or 0xFFFE results in the counter value being reset after count till 0xFFFF. During a rollover event the output is either set (default), reset or has no effect according to the programming of the POUTC field in the control register. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

When the counter value reaches the sample value the output of the PWM is reset (default), set or has no effect according to the programming of the POUTC field of control register. This event is referred to as a compare event. This event can also generate an interrupt if the respective interrupt enable bit is set in the control register.

If the rollover event sets the PWM output signal the compare event will reset it and vice versa for a particular programming configuration of POUTC field.

37.4.1.4 Low-Power Mode Behavior

In low-power modes if the clock from the selected clock source is available, the PWM counter continues to run and an output is produced depending on whether the control bit for that mode is set. In the absence

of the clock itself or if the corresponding low-power bit in the control register is 0, the counter is reset and resumes counting when it exits the low-power mode.

37.4.1.5 Debug Mode Behavior

In debug mode, PWM has the option of continuing to run or be halted. If the DBGEN bit is not set in the PWMCR, the PWM is halted. If the DBGEN bit is set, then the PWM will continue to run in the debug mode.

Chapter 38
Smart Direct Memory Access (SDMA) Controller

38.1 Introduction

The smart direct memory access (SDMA) controller offers highly-competitive DMA features combined with software-based virtual-DMA flexibility. It enables data transfers between peripheral I/O devices and internal/external memories.

The SDMA controller helps maximize system performance by offloading the CPU in dynamic data routing.

NOTE

In this document, “AP” refers to interfaces to the ARM (Applications Processor) core.

38.1.1 Overview

Figure 38-1 shows a block diagram of the SDMA controller. It includes the custom RISC core along with its RAM, ROM, DMA units, the CRC unit, and the scheduler.

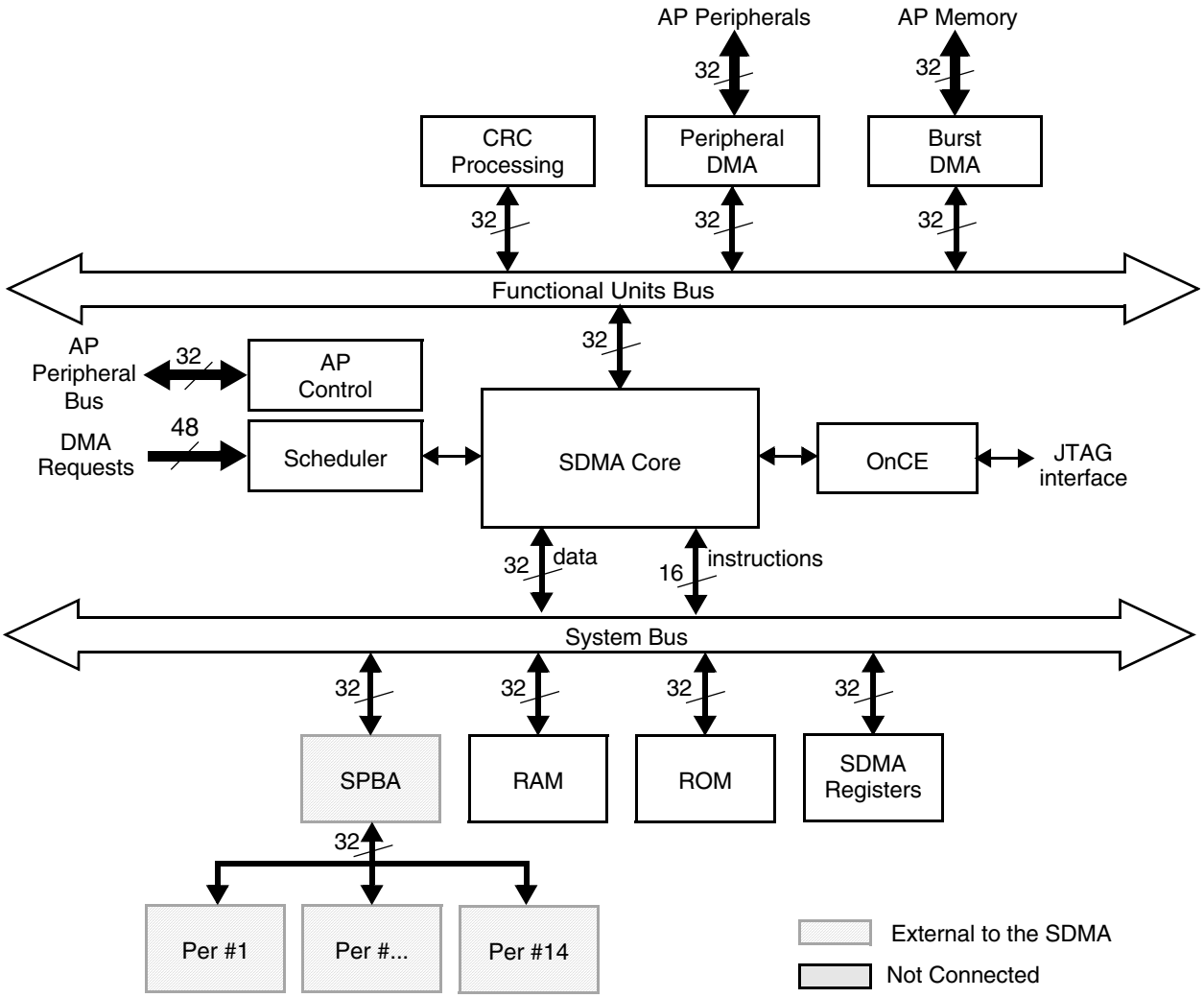


Figure 38-1. SDMA Block Diagram

The SDMA core executes short routines that perform DMA transfers; these routines are called *scripts*. The SDMA core interfaces to its own memory using the SDMA system bus. The SDMA system bus supports a 32-bit data path and a 16-bit address bus. The system bus datapath is used for both 16-bit instruction (program) memory access and 32-bit data access. DMA units interface to the core using the Functional Unit Bus and use dedicated registers to perform DMA transfers.

The SDMA memory contains a ROM and a RAM. The ROM contains startup scripts (for example, boot code) and other common utilities, which are referenced by the scripts that reside in the RAM. The internal RAM is divided into a context area and a script area (more details about this mapping are available in Section 38.12.5.1, “Instruction Memory Map,” and Section 38.12.5.2, “Data Memory Map”).

Every transfer channel requires one context area to keep the contents of all the core and unit registers while inactive. Channel scripts are downloaded into the internal RAM by the SDMA using a dedicated channel that is started during the boot sequence. Downloads are invoked using commands and pointers provided by the AP. Every channel contains a corresponding channel script located in RAM and/or ROM that can be reconfigured independently as-needed. Channel scripts can be stored in an external memory and downloaded when needed. The SDMA can be configured with any mixture of scripts to enable an endless combination of supported services.

The scheduler monitors and detects DMA requests, mapping them to channels, and mapping individual channels to a pre-configured priority. At any given point, the scheduler presents the highest priority channel that requires service to the SDMA core. A special SDMA core instruction is used to “conditionally yield” the current channel being executed to an eligible channel that requires service. If (and only if) there is an eligible channel pending, will the current channel execution be preempted.

There are two `yield` instructions that differently determine the eligible channels: In the first version, eligible channels are pending channels with a strictly higher priority than the current channel priority. In the second version (`yieldge`), eligible channels are pending channels with a priority that is greater or equal to the current channel priority. The scheduler detects devices that need service through its 48 DMA request inputs. After a request is detected, the scheduler determines the channel(s) that is (are) triggered by this request and marks it (them) as pending in the “Channel Pending (EP)” register. The priorities of all the pending channels are continuously evaluated in order to update the highest pending priority. The channel pending flag is cleared by the channel script when the transfer has completed.

The AP Control module contains the control registers used to configure the 32 individual channels. There are 48 Channel Enable registers, and every register maps one DMA request to any desired combination of channels. The 32 Priority registers are used to assign a programmable 1-of-7 level priority to every possible channel. This module also contains all other control registers that the AP can access.

The 48 DMA requests that are connected to the scheduler come from a variety of sources. The “receive register full” and “transmit register empty” signals found in the UART and USB ports are typical examples of DMA requests that can be connected to the SDMA. These requests can be used to trigger a specific SDMA channel, or several channels.

There is an OnCE compatible debug port for product development. The OnCE includes support for setting breakpoints, single-step and trace, and register dump capability. In addition, all memory locations are accessible from the debug port.

38.1.2 Features

The following are the SDMA features:

- Multi-channel DMA supporting up to 32 time-division multiplexed DMA channels
- Hardware or software driven triggers for each channel
- 48 hardware driven triggers that can be mapped to any channel.
- Memory accesses including linear addressing, FIFO addressing and 2D addressing
- Fast context-switching with priority-based preemptive multi-tasking
- 16-bit instruction-set micro-RISC engine (the SDMA core)

- Two DMA units with some or all the following features:
  - Auto-flush and prefetch capability
  - Flexible address management (increment, decrement, and no address changes on source and destination address)
  - Misaligned data-transfer support
  - Unidirectional and bidirectional flows (copy mode)
  - Up to eight-word buffers for configurable burst transfers
- Support of byte-swapping and CRC calculations
- An available API and library of scripts
- Little-endian and big-endian modes
- Hardware handshakes for low-power entry sequence
- Security support to lock contents of the SDMA script RAM.
- 4-Kbyte ROM containing startup scripts (for example, boot code) and other common utilities that can be referenced by RAM-located scripts
- 8-Kbyte RAM area is divided into a processor context area and a code space area used to store channel scripts that are downloaded from the system memory
- Debug support, including a OnCE port, real-time monitors, and embedded cross-trigger events
- Supported clock frequencies in process:
  - Configurable clock options for the SDMA core and the AP DMA units
    - 1:2 ratio with maximum of SDMA core running at AP Peripheral Bus speed and DMA running at max DMA frequency.
    - 1:1 ratio when both SDMA core and AP DMA clocks are set to the AP Peripheral Bus speed.
- Peripheral bus interface for configuration register programming by the AP.
- The SDMA RISC engine (arithmetic and logic operations, plus CRC acceleration), which is referred to as the “SDMA core.”
- An internal peripheral bus connected to the Shared Peripherals Bus Interface (SPBA) that enables access to up to 14 shared peripherals. SDMA supports 32-bit accesses to word peripherals and 16-bit accesses to half-word peripherals.
- The peripheral DMA unit that is hooked-up to the AP Crossbar Switch to service AP peripherals
- The burst DMA unit is able to perform burst accesses to the external memory
- All the DMA units are 32-bit AHB masters. They are connected to different buses, thus allowing concurrent accesses.

38.2 Functional Description

Figure 38-2 shows the SDMA topology, and is composed of the following components:

- SDMA Core (Section 38.3, “SDMA Core”)
- SDMA Scheduler (Section 38.4, “Scheduler”)
- Functional Units:

  - CRC (Section 38.5.1, “CRC Calculation Unit.”)
  - Burst DMA (Section 38.5.2, “Burst DMA Unit.”)
  - Peripheral DMA (Section 38.5.3, “Peripheral DMA Unit.”)
- AP Control for AP control register access.
- Internal RAM and ROM Memory (Section 38.12, “SDMA Programming Model)
- OnCE debug Port (Section 38.18, “The OnCE Controller.”)

The functional unit bus provides access by the SDMA core to the CRC hardware accelerator and the DMA units. The system bus provides access to SDMA internal memory and also supports up to 14 peripherals.

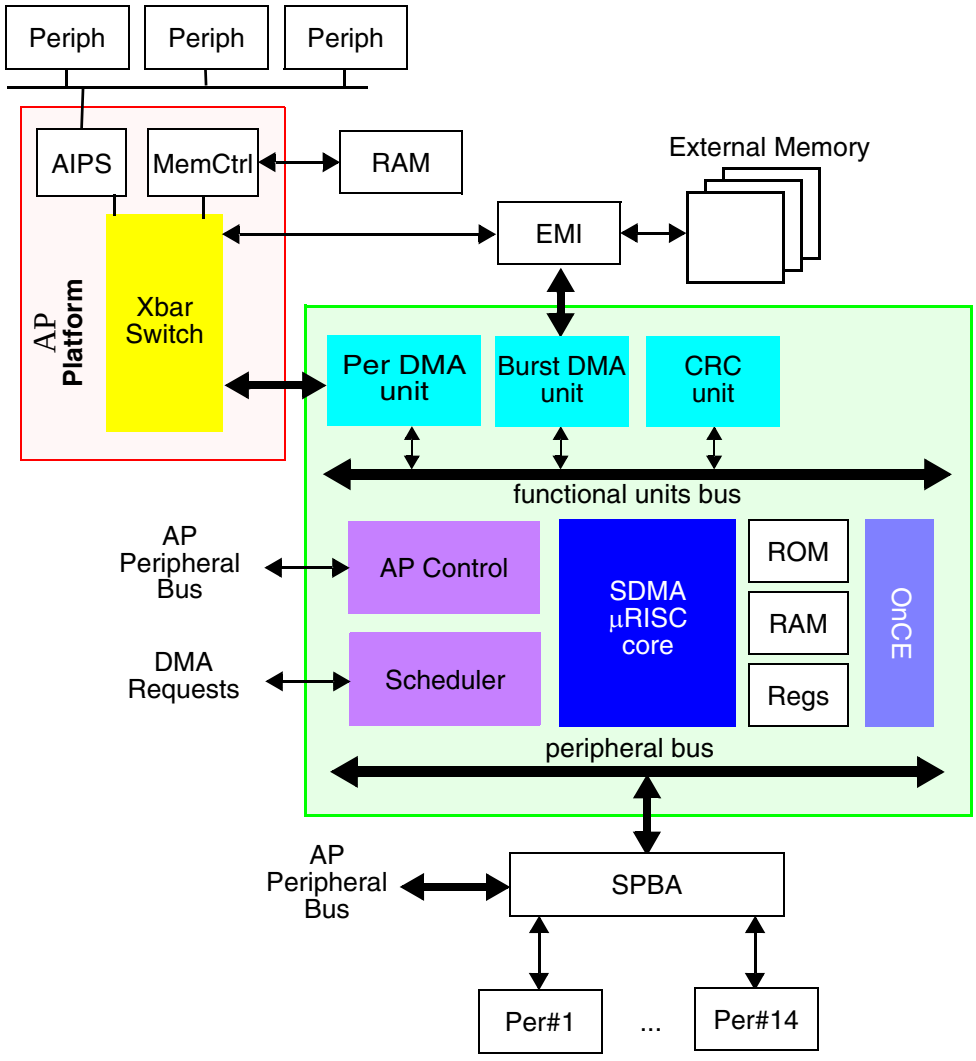


Figure 38-2. SDMA Connections

38.3 SDMA Core

The SDMA core is a customized RISC-like processor that is specifically developed to control DMA units and perform L1 tasks like byte-stuffing or framing. The SDMA core incorporates on-chip debug capability using the OnCE.

The SDMA core is based on a 32-bit register architecture with 16-bit instructions. There are eight general purpose 32-bit registers, four flags (T, LM, SF, and DF), and four PCU registers (PC, RPC, SPC, and EPC) that can address 16,384 16-bit instructions.

38.3.1 SDMA Core Structure

Figure 38-3 shows the structure of the SDMA core. It also shows the different registers, calculation resources, and possible data movements.

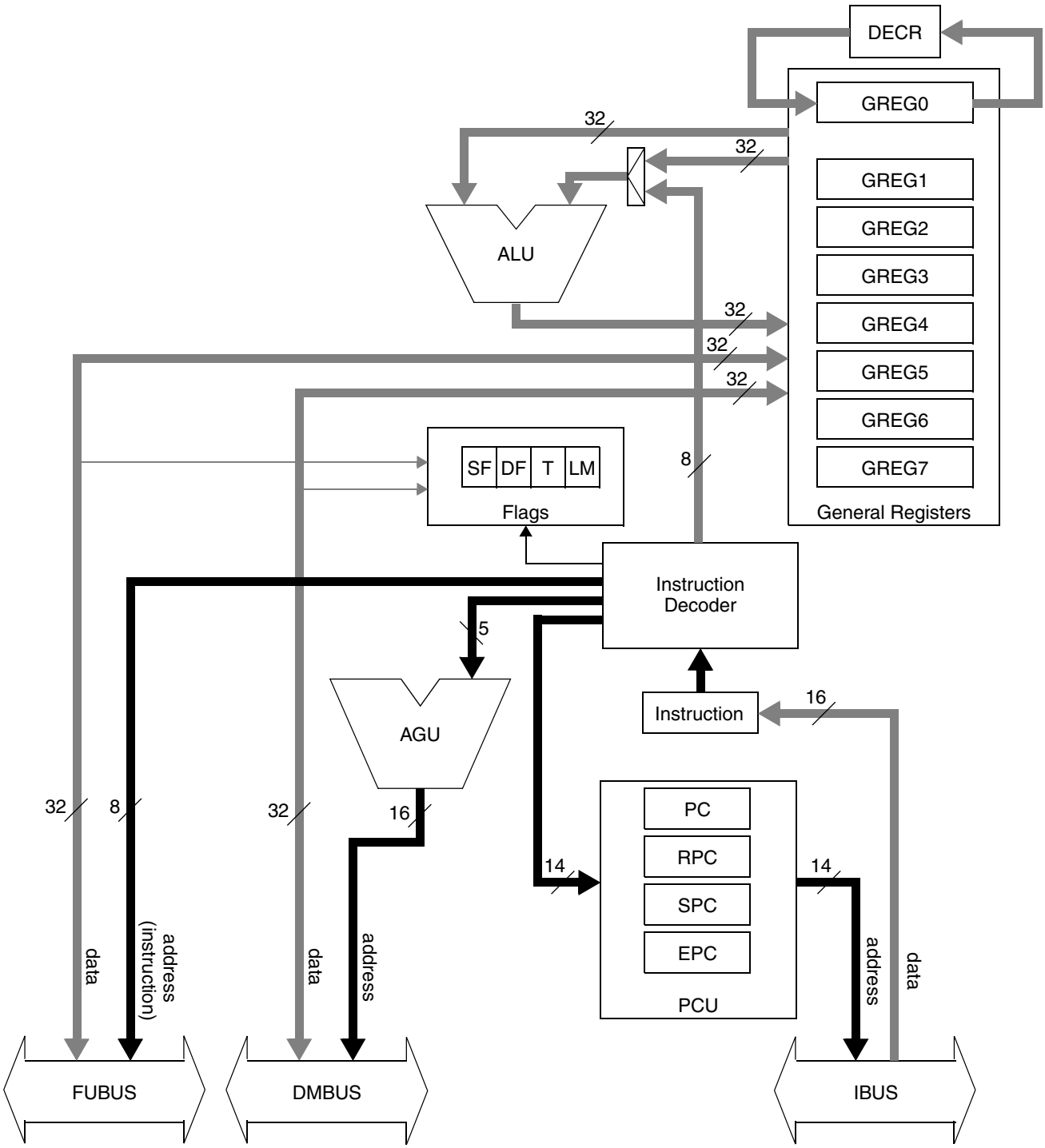


Figure 38-3. SDMA Core

- The Program Control Unit (PCU) is described in Section 38.3.2, “Program Control Unit (PCU).” It handles the state of the core and generates the instruction fetch addresses. Instructions are retrieved from the Instruction Bus (IBUS) and stored in the SDMA core instruction register prior to their decoding. The PCU contains the following registers:
  - The Program Counter (PC) contains the address of the current instruction.
  - The Return Program Counter (RPC) contains the address of the instruction that follows a jump to the subroutine.
  - The Start Program Counter (SPC) contains the address of the first instruction of the current hardware loop.
  - End Program Counter (EPC) contains the address of the last instruction of the current hardware loop.
- The other core registers are the general purpose registers (GREGn) and the flags.
  - The general purpose registers can be used to hold data and addresses. They can be loaded with immediate values (for example, 8-bit data that are encoded in the instruction), results of calculations that were performed with the ALU, 32-bit data that comes from the memory or peripherals using the Data Memory Bus (DMBUS), 32-bit data that comes from the DMAs or CRC using the Functional Units Bus (FUBUS) or another general purpose register. Their content can be the operands of the ALU, the data to send on either bus (DMBUS or FUBUS), or a pointer to memory (DMBUS address).
  - The general register 0 (GREG0) is also the hardware loop counter. In hardware loops, it cannot be used for any other purpose. This register uses a dedicated decrement unit (DECR) shown in Figure 38-3.
  - The flags reflect the status of operations:
    - SF and DF are set when the last load or store on either bus (FUBUS or DMBUS) received an error response.
    - LM is set when the core is executing instructions inside a hardware loop.
    - T is set when the ALU operation result was 0 or the loop counter reaches 0 (the latter is preponderant when an ALU operation is the last instruction of a hardware loop).
- The ALU has two operands: any general register and either a second general register or an immediate value. The result is always stored into the first general register. A NOP function can be utilized by moving a register’s contents into itself (For example, the instruction: `mov R0,R0`).
- The 16-bit instructions are fetched using the instruction bus (IBUS) whose address is driven by the PC. The SDMA RAM and ROM are visible to the core as 16-bit devices through this interface.
- The memory (RAM and ROM), memory mapped registers, and external peripherals are accessed using the DMBUS. The address is always taken from a general register whose content is added to a 5-bit immediate value. This is the only available addressing mode. The DMBUS is a 32-bit data bus. Except for the peripherals that are external to the SDMA, the address accuracy is the 32-bit word (for example, adding 1 to an address points to the next word, not the next byte).
- The functional units are accessed using the FUBUS connection. The data is exchanged with any general register, but the address (which in fact is the instruction and the selector of the functional unit) comes from an 8-bit field of the corresponding load or store.

38.3.2 Program Control Unit (PCU)

This part of the SDMA core is dedicated to the control of the RISC engine as implied by the instructions that are executed. Its behavior is determined by the instruction type and the inputs of the SDMA. It contains the PC, RPC, SPC, and EPC registers that are described in Section 38.3.1, "SDMA Core Structure."

38.3.2.1 Instruction Types

The state sequence and the delay of execution vary according to the type of the instruction. There are six possible categories of instructions, as follows:

1. `standard`—Most of the instructions belong to this category, and always last 1 cycle.
2. `ldf`/`stf`—These are respectively the load and store instructions that access the functional units. They last 1+n cycles where n is the number of wait-states of the targeted functional unit.
3. `ld`/`st`—These are the load and store instructions that access the memory and peripherals. They last 1+n cycles where n is the number of wait-states of the targeted device (1 for the ROM, RAM, and memory mapped registers, 1 + the external peripheral wait-states). These instructions always last at least two cycles, but the core is able to handle them in one cycle. The first wait-state is inserted outside the core.
4. `branch`—These are all the instructions that cause the Program Counter to point to another instruction other than the following one (for example, one that breaks the sequential flow). There are the absolute jumps, the conditional branches, the jump to the sub-routines, and the return from the sub-routine.
5. `loop`-modified load or store—The hardware `loop` instruction modifies the potential behavior of any load or store inside the `loop` (for example, when the LM flag is set). A jump may be implied after any such load or store if it received an error. The error causes an early exit of the loop, which means a jump to the instruction that follows the one that is pointed to by EPC. An additional cycle is required by the PCU to perform the jump (+1 to the `ld`/`st`/`ldf`/`stf` original execution delay). Although there is usually an implicit jump after the last instruction of the loop when the PC goes back to SPC, this is performed at no cycle cost.
6. `done`—The `done`, `yield`, or `yieldge` instructions are used to control channel switching. When no channel switching is performed, these instructions last a single cycle. When there is a change of channel or context switch, the delay is variable and depends on many factors (as detailed in Section 38.4.4, "Context Switching").

38.3.2.2 PCU States

The PCU state is visible through outputs of the SDMA (see Section 38.19.8.4, "Real-Time Debug Outputs") or the OnCE status register (see Section 38.18.4.9, "OnCE Status Register (OSTAT)").

The PCU state is a four-bit field that can take the values shown in Table 38-1. Figure 38-4 shows the possible state transitions and the corresponding conditions.

Table 38-1. PCU States

Value	State	Description
0	Program	The is the usual instruction cycle.
1	Data	This state is inserted when there are wait-states during a load or a store on the data bus (ld/st type).
2	Change of Flow	This is the second cycle of any instruction that breaks the sequence of instructions (branch and done types). This state lasts only a single cycle; it is always followed by the Program state.
3	Error in Loop	This state is used when an error causes a hardware loop exit (loop-modified load or store type). This state only lasts a single cycle; it is always followed by the Program state.
4	Debug	he SDMA is stopped in debug mode.
5	Functional Unit	This state is inserted when there are wait-states during a load or a store on the functional units bus (ldf/stf type).
6	Sleep	No script is running: The core is idle after saving the last channel context.
7	Save	The context switch FSM is saving the current channel.
8	Program in Sleep	Same as Program except there is no associated channel, this state is used when instructions are executed after entering debug mode, whereas the core was in either Sleep mode.
9	Data in Sleep	This is the same as Data except there is no associated channel.
10	Change of Flow in Sleep	This is the same as Change of Flow except there is no associated channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
11	Error in Loop in Sleep	This is the same as Error in Loop except there is no associated. channel. This state only lasts a single cycle, and is always followed by the Program in Sleep state.
12	Debug in Sleep	This is the same as Debug except the core was put in debug mode when no channel was active.
13	Functional Unit in Sleep	This is the same as Functional Unit except there is no associated channel.
14	Sleep after Reset	This shows that no script is running, and the core is idle after a reset. When a channel becomes active, no context is restored but the core starts its boot program located at address 0 (or the address available in register in Section 38.11.3.22, “Channel 0 Boot Address (CHN0ADDR)”).
15	Restore	The context switch FSM is restoring the next channel context.

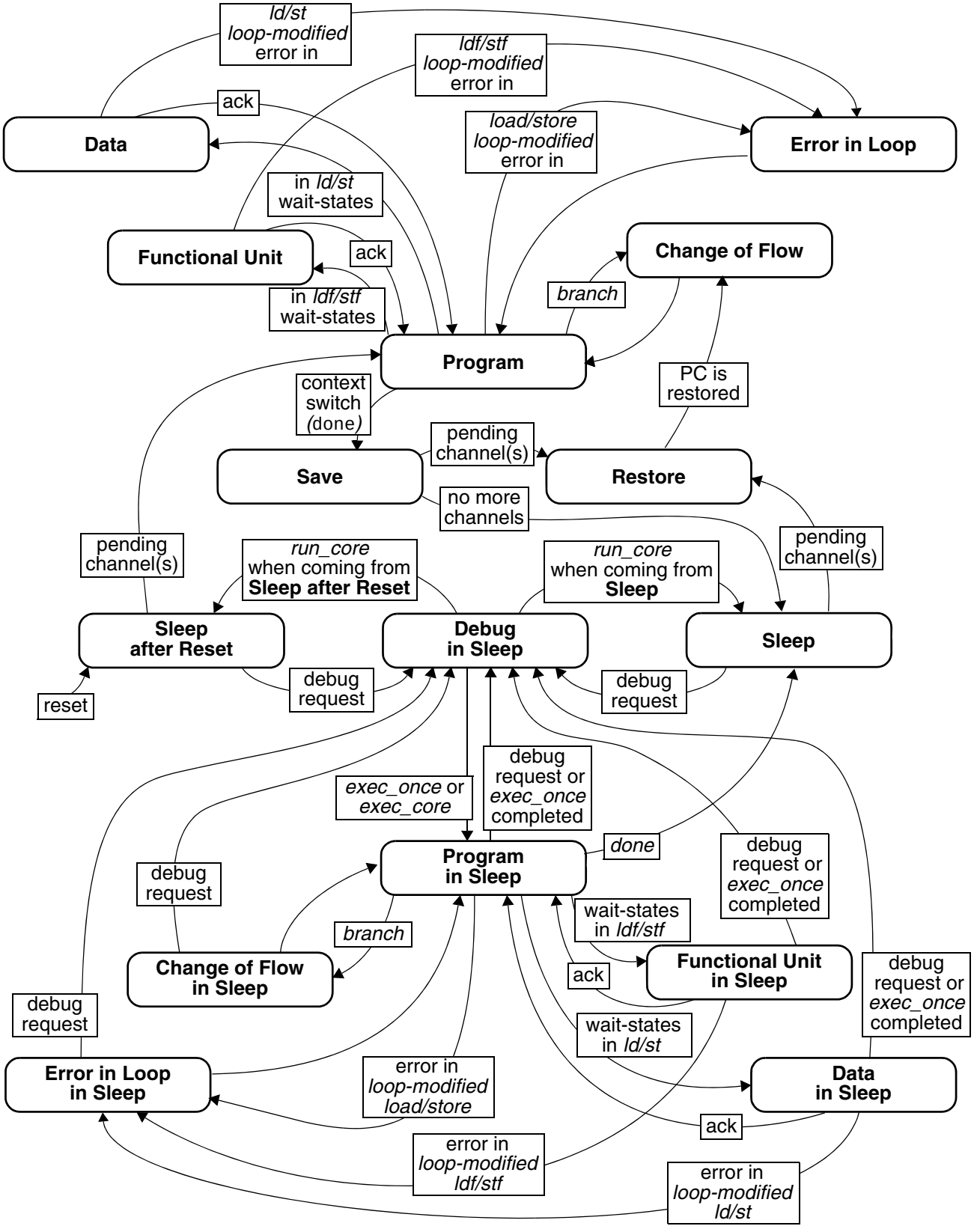


Figure 38-4. PCU State Diagram

38.3.3 SDMA Core Memory

The SDMA has two memory spaces: one for the instructions and one for the data. As both spaces share the same resources (ROM and RAM devices), the system bus manages possible conflicts when the core accesses the same resource for both an instruction read and a data read or write. Program and data memory is further described in Section 38.12.5, “Address Space.”

Instructions of 16-bit width are stored in 32-bit wide devices and can be accessed as data. The mapping is big endian: an even instruction address (terminated by 0) accesses the most significant part of the 32-bit data (bits [31:16]), and an odd instruction address (terminated by 1) accesses the least significant part of the 32-bit data (bits [15:0]). Instructions can be fetched out of internal ROM or RAM.

Data can be read from ROM, RAM, memory mapped registers, and external peripherals, and written to the same devices (except the ROM).

The ROM contains bootload scripts, channel scripts, and common subroutines which may be referenced by channel scripts elsewhere in the ROM or RAM.

The RAM is divided into a context area and a code space area which may be used to store channel scripts. The RAM contains undefined values after a hardware reset. Channel scripts and initial context values are downloaded into RAM using channel 0 which is reserved for bootload functions.

38.4 Scheduler

All channel scheduling hardware is included in the scheduler.

38.4.1 Primary Functions

The scheduler is a hardware-based design used to coordinate the timely execution of 32 virtual DMA channels by the SDMA core on the basis of channel status and priority. The scheduler performs the following functions:

- Monitors, detects, and registers the occurrence of any one of the 48 DMA requests
- Links a specific request to a channel or group of channels (channel mapping)
- Ignores requests that are not mapped to a previously configured channel
- Maintains a list of all the channels that are requesting service
- Assigns a pre-programmed priority level (1 of 7) to every channel requesting service
- Detects and flags overrun/underrun conditions

38.4.2 Channels and DMA Requests

38.4.2.1 Channels

A Virtual Channel (hereafter simply called a channel) manages a flow of data through the SDMA. Flows are typically unidirectional. The SDMA can have up to 32 simultaneously operating channels, numbered from 0 to 31. Channel 0 is usually dedicated to control the SDMA script downloading. All the channels can be assigned by the AP software.

38.4.2.2 DMA Requests

A DMA request is caused by externally (for example, external to the SDMA) controlled conditions (for example, UART receive FIFO reaches a threshold). The SDMA currently supports up to 48 DMA requests.

38.4.2.3 Mapping from DMA Requests to Channels and Priorities

A channel can stall waiting on a single DMA request. A single DMA request can awake more than one channel (in fact, any request can awake any combination of channels). The mapping between DMA requests and channels is program-controlled. There is a storage element assigned for each of the 48 requests that contains a bitmap table of the channels that are awakened by the event. Every channel also has a three-bit register that indicates its priority.

38.4.3 Scheduler Functional Description

Section 38.4.3.1, "Scheduler Overview," describes the behavior of the SDMA scheduler—from the channel enabling conditions to the highest priority pending channel selection.

38.4.3.1 Scheduler Overview

The scheduler algorithm is built in hardware. It is provided with possibilities for the AP to control its behavior. Figure 38-5 shows a functional overview. The scheduler processes incoming DMA requests, maps detected requests to 0, one, or several channels, maintains a list of channels that are requesting

service (pending channels), identifies the top priority and its associated channel, and selects the next active channel when the current channel yields.

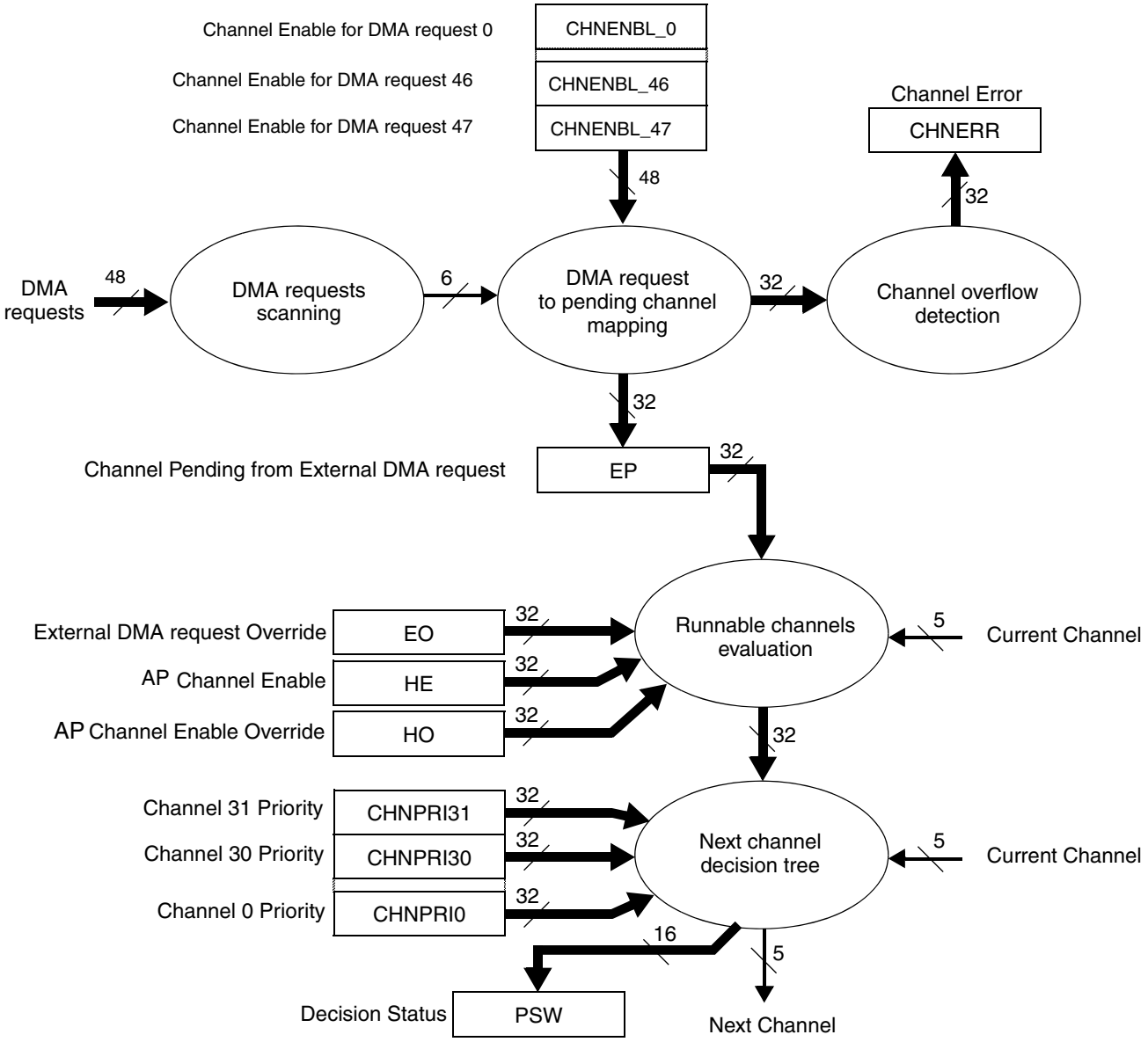


Figure 38-5. SDMA Hardware Scheduler

38.4.3.2 DMA Requests Scanning

The scheduler contains a 48-bit edge detection device that detects the rising edge of every DMA request and transmits the request number to the next stage. The DMA requests are assumed to be generated on the same reference clock as the SDMA core clock; they are detected as soon as the signal goes from a 1-to-n-cycles low state to a 1-to-m-cycles high state.

This system is able to detect single-cycle pulses as well as level-based DMA requests such as a FIFO threshold crossing. In this case, the SDMA provides a memory mapped register that can be used by the channel script to monitor the DMA requests lines, and thus determines whether the data transfer is done or not done, and then continues with the transfer or closes the channel.

When several DMA requests are detected at the same time, they are forwarded to the next scheduler stage at the rate of one request per cycle. No request is lost.

Figure 38-6 shows examples of valid DMA requests.

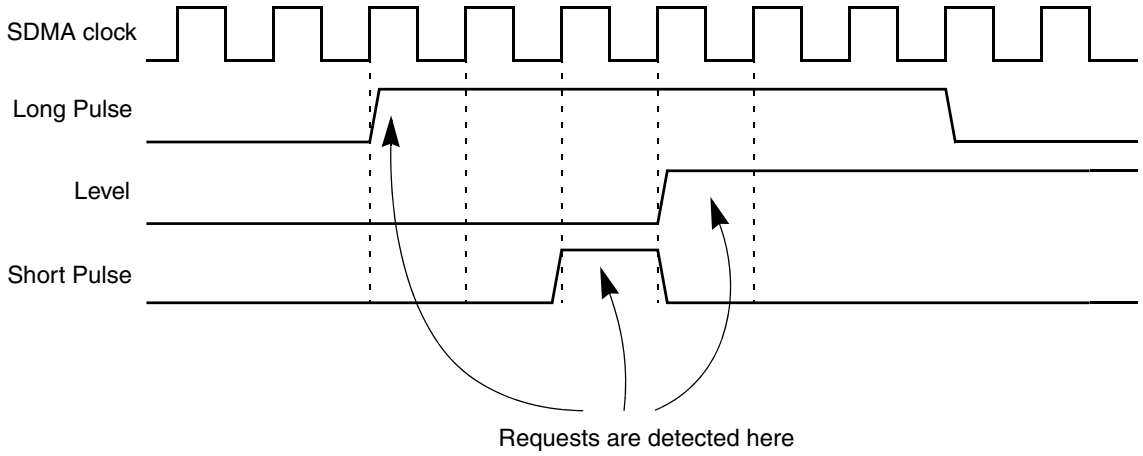


Figure 38-6. Examples of Valid DMA Requests

The DMA request inputs are connected to various sources that depend on the SoC. The exact list of DMA request inputs and their associated number is available in each respective project-specific chapter.

38.4.3.3 Mapping DMA Requests to Pending Channels

Whenever a DMA request is detected by the first stage, its number is used in the second stage to determine the channels that have to be activated. This is performed with an array of 48 registers that are 32 bits wide: There are 48 Channel Enable Registers (CHNENBLn), one register per DMA request. The DMA request number selects the Channel Enable Registers, and every bit of this 32-bit register indicates that the corresponding channel must be activated when it is a 1.

This information is passed on the EP register. For every bit of the Channel Enable Register that is set, the corresponding bit of the EP register is also set, and the remaining bits of EP are left unchanged. The transformation of EP is summarized by Equation 38-1:

$$EP = EP \text{ or } CHNENBLn \qquad \textit{Eqn. 38-1}$$

The EP register is used to know which channels require service because they received a DMA request.

Typical contents of the CHNENBLn registers are all 0s, except for a single bit set. For example, a DMA request triggers one channel, but all 0s or several 1s are possible. One DMA request could activate several channels, and the channel execution sequence can be controlled by the channel priorities and numbers, as explained in the next sections. Table 38-2 illustrates an example configuration.

NOTE

From the table, the DMA request 0 is programmed to simultaneously trigger channels 0, 1, and 31. Also, DMA requests 30-47 are not used in this example. The remaining channels 2 to 30, are configured to be triggered by DMA requests 29 to 1, respectively.

Table 38-2. Channel Enable RAM Programming Example

DMA Request Number	Channel (31 … 0)
0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
1	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
18	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
19	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
20	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
21	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
22	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
23	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
24	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
25	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
26	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

Table 38-2. Channel Enable RAM Programming Example (continued)

DMA Request Number	Channel																															
	31																															0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.4.3.4 Channel Overflow

A channel overflow occurs when a DMA request requires service from channel *n* by setting bit *n* of the register EP, but this bit is already set, meaning channel *n* is already pending. This can come from an overrun/underrun condition. This detection is possible only when the DMA requests are pulses, because a level-based DMA request stays high until it is serviced, even though an underrun or overrun condition occurs, thus preventing another edge detection of the DMA request.

The channel overflow information is saved in the 32-bit CHNERR register (1 bit per channel). You can configure the SDMA to trigger an interrupt to the AP when there are 1s in CHNERR. Every bit of CHNERR is masked with the corresponding bit of INTRMASK and if it gives a 1, the corresponding bit of INTR is set, triggering the interrupt.

38.4.3.5 Runnable Channels Evaluation

The EP register is used in conjunction with several other 32-bit registers to determine the channels that are runnable. Registers EO, DO, HO and HE, are controlled by the AP. EP is controlled by the DMA requests and their mapping to channels.

Several channels may be runnable at any given time. The i^{th} channel is runnable if (and only if) the condition below is true:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i]) ***Eqn. 38-2***

After reset, the HE[i], HO[i], EP[i], and EO[i] bits are all cleared whereas the DO[i] bits are all set. The functions associated with DO are not available for this device. When DO[i] is set, the scheduler condition becomes:

(HE[i] or HO[i]) and (EP[i] or EO[i]) ***Eqn. 38-3***

The registers in these equations are controlled as follows:

- AP (host) channel enable flag HE[i] may be set or cleared by the AP with the HSTART and STOP_STAT registers. It can also be cleared by the i^{th} channel script.

  Typical usage is for the AP to set this flag to activate the channel. The flag is cleared by the SDMA core when the transfer is done.
- Externally triggered channel pending flag EP[i] is set by the scheduler when the channel was activated by a DMA request. It can be cleared by the i^{th} channel script.
- The AP channel override flag HO[i] may be set or cleared by the AP. When set, it enables the i^{th} channel to run without the involvement of the AP.

  Typical usage is for the AP to set this flag for channels that do not need AP supervision such as channels that are controlled by DMA request events (EP).
- DO should always be set to 1 so that the runnable channel evaluation considers only HO, HE, EP, and EO.
- Externally triggered channel override flag EO[i] may be set or cleared by the AP. When set, it prevents the i^{th} channel from stopping and stalling on incoming peripheral DMA requests. This is the case when the channel is not handling data transfers with peripherals (for example, a memory to memory transfer).

The SDMA can clear the HE[i], and EP[i] bits by means of a `done` or `notify` instruction. The `done` instruction causes a reschedule; thus, enabling another channel to preempt the current one, while the `notify` instruction does not. The `done` and `notify` instructions can clear either HE[i] or EP[i] (never more than one at a time).

Table 38-3. Runnable Channel Selection Control

Register	Set by	Cleared By
HO	Write to HOSTOVR register	Write to HOSTOVR register
HE	Write to HSTART register	Write to STOP_STAT register or by the channel script with the `done` or `notify` instructions.

Table 38-3. Runnable Channel Selection Control (continued)

Register	Set by	Cleared By
DO	Write to DSPOVR register	Write to DSPOVR register
EO	Write to EVTOVER register	Write to EVTOVER register
EP	Set by external DMA request event input.	By the channel script with the `done` or `notify` instructions

38.4.3.6 Next Channel Decision Tree

The next channel number is computed from the runnable channels list, the current channel number, and their respective priorities. It is re-evaluated every cycle, but is only used when the current channel yields or terminates by executing a `yield`, `yieldge`, or `done` instruction.

The decision tree is based on the selection of the runnable channel that has the highest priority. The highest priority channel is selected according to the following rules:

- Runnable channels are sorted by priority.
- If one of the channels with the highest priority had been preempted by a channel with a higher priority, but did not want to yield to a channel of the same priority (for example, it executed a `yield`, not a `yieldge`), it is elected as the next channel.
- The channels that belong to the highest priority group are sorted by their number and the channel that has the highest number in this group becomes the next channel. For example, if priorities are the same, channel 31 will be selected before channel 30.

When the current channel requires a reschedule with a `yield`(`ge`) or a `done` instruction, the context switch decision is based on the instruction parameter, the current channel number and priority, and the next channel number and priority. The possible cases are all listed in the Table 38-4. The grayed cells in Table 38-4 correspond to unusual cases that should not occur with a typical usage of the SDMA.

Table 38-4. Channel Switching Decision with a yield, yield(ge), or done

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
`yield` (`done` 0)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Current
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the AP)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the AP)

Table 38-4. Channel Switching Decision with a yield, yield(ge), or done (continued)

Instruction	Current Channel	Next Channel	Priorities Comparison	New Running Channel/Comments
`yieldge` (done 1)	Runnable	Not runnable	none	Current
	Runnable	Runnable	Current > Next	Current
			Current = Next	Next[1]
			Current < Next	Next[1]
	Not runnable	Not runnable	none	none[2] (occurs when the channel was disabled by the AP)
	Not runnable	Runnable	none	Next[1] (occurs when the channel was disabled by the AP)
`done` (done >1)	Not runnable	Not runnable	none	none[2]
	Runnable	Not runnable	none	Current[3] (occurs when the `done` instruction does not disable the channel runnable condition)
	Not runnable	Runnable	none	Next[1]
	Runnable	Runnable	none	Current[3] (occurs when the `done` instruction does not disable the channel runnable condition)

1 Current channel script execution is stopped, its context is saved; the next channel context is restored and its script execution resumes

2 Current channel context is saved and SDMA enters IDLE mode

3 Current channel context is saved, then restored, and the current channel script resumes execution

Finally, when the SDMA is in IDLE mode and a runnable channel is elected as the next channel, its context is immediately restored and the script execution resumes.

The *combinatorial-decision* tree supports dynamic modifications of the EP, EO, HE, HO, and DO flags as well as dynamic modifications of the channel priorities. The propagation times are detailed in Section 38.4.3.8, “Scheduler Pipeline Timing Diagram.”

The decision tree status is available in the PSW register, which is continuously updated. It contains the next channel priority, the next channel number, the current channel priority, and the current channel number. When a priority is read as 0, it means the channel is not runnable.

A few examples of decisions are presented below:

- Channel 31 is running with priority 5, channels 13 and 24 are pending with the same priority 5; channel 24 is eligible as the next channel since 24 > 13.
- Channel 31 is running with priority 7, channels 13 and 24 are pending with priority 5; channel 31 is the next channel because its priority is greater than the other pending channels.
- Channels 7, 23, and 29 are pending with the same priority. Channel 7 is active and runs a `yieldge`; it is preempted by channel 29. After a period, channel 29 runs a `yieldge`; it is then preempted by channel 23 which is the selected channel, because channel 29 is the current channel. Later, channel 23 runs a `yieldge` and is preempted by channel 29. Channels 23 and 29 will go on switching after

every `yieldge` until one of them terminates. It is only at that point that channel 7 becomes eligible again.

- Channel 11 is running with priority 3, and channel 15 is pending with priority 4. When the channel 31 script executes a `yield` instruction, it gets preempted by channel 15; then channels 6 and 18 with priority 3 become pending. Because channel 11 was preempted after executing a `yield` and there is no pending channel with a strictly greater priority, it is eligible as the next channel (although its number 11 < 18).

38.4.3.7 Scheduler State Diagram

The Figure 38-7 summarizes the behavior of the SDMA scheduler with details about the exact mechanism of the priority decision tree. It is important to understand the scheduler is a hardwired pipeline, which means all the stages are performed simultaneously every cycle, but a change on any given stage is reflected on the next stage after the delays presented in Section 38.4.3.8, “Scheduler Pipeline Timing Diagram.”

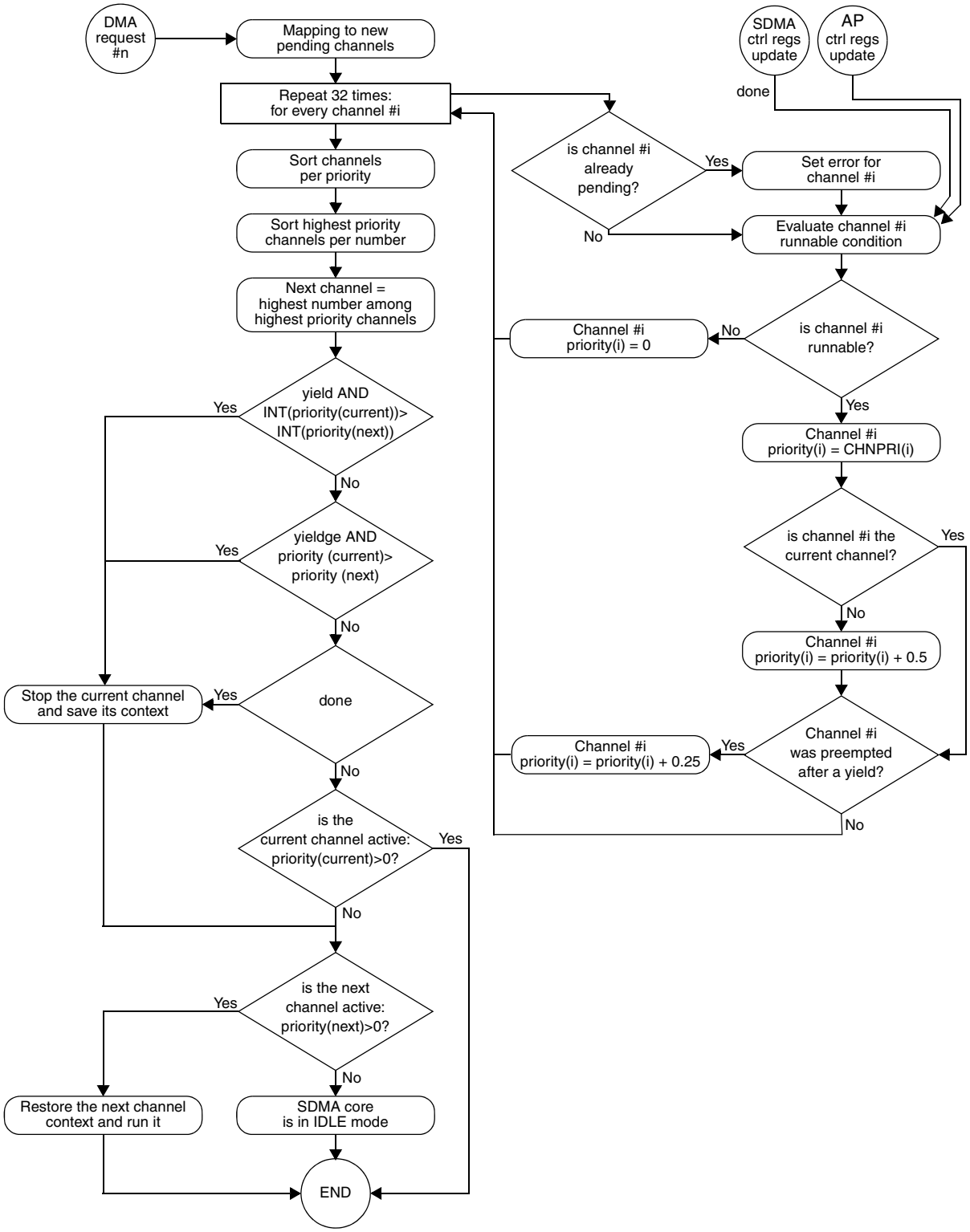


Figure 38-7. Scheduler State Diagram

38.4.3.8 Scheduler Pipeline Timing Diagram

The SDMA scheduler process of DMA-request and control-register modifications is not immediate. Figure 38-8 shows the exact delays of all the tasks. The reference clock is the SDMA core clock.

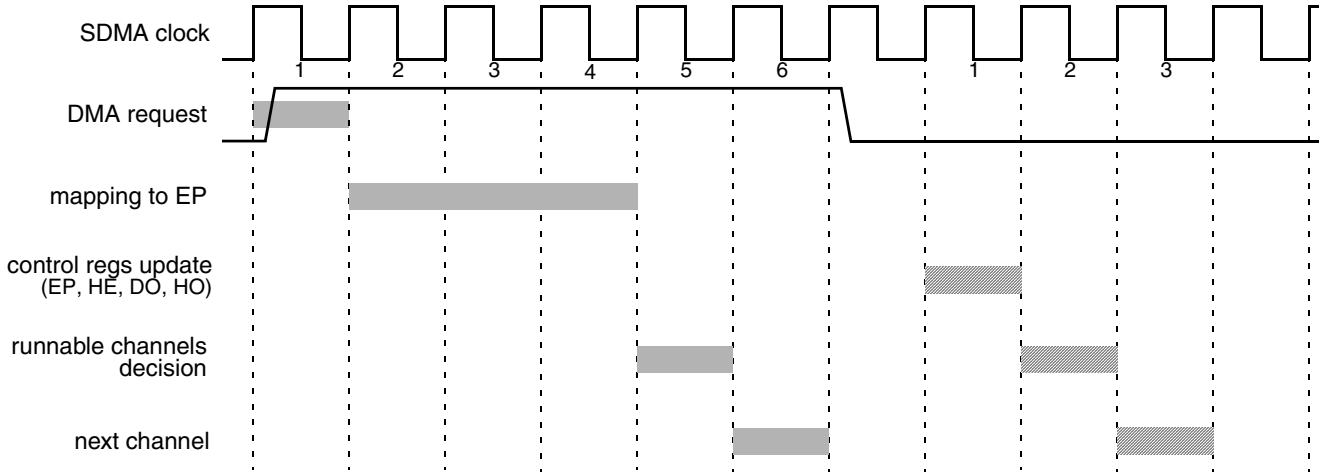


Figure 38-8. Scheduler Timing Diagram

Two numbers can be inferred from this timing diagram. First, it takes six SDMA core clock cycles to update the next channel from a DMA request. Second, it takes three SDMA core clock cycles to update the next channel from a direct modification of the condition registers (EP, DO, HE, or HO) by any processor. The processors that can modify these bits include SDMA with a done instruction or the AP with a write access through the corresponding control port on their respective peripheral bus).

38.4.3.9 Channel-DMA Request Mapping

The 48 DMA request inputs to the SDMA scheduler are listed in project-specific chapters. See the respective chapters for this information.

38.4.3.10 Examples: How to Start a Channel

A channel can be started when Equation 38-2 is true for channel *i*:

(HE[i] or HO[i]) and (DO[i]) and (EP[i] or EO[i])

Once this equation is true, the scheduler can start this channel according to the priority of all pending channels. Several examples of configuration are listed below:

1. To start a channel triggered by AP software:
   - Initially, configure HO[i]=0, DO[i]=1, and EO[i]=1 using registers indicated in Table 38-3.
   - AP software triggers the channel by writing to the HSTART register to set HE[i]=1, thereby setting Equation 38-2 true.
2. To start a channel triggered by DMA request event.
   - Initially, configure HO[i]=1, DO[i]=1, and EO[i]=0 using registers indicated in Table 38-3.

— The DMA request is asserted to trigger the channel by setting EP[i]=1, which makes Equation 38-2 true.

38.4.4 Context Switching

On execution of a `done` or `yield`(`ge`) instruction, the current channel may be changed either because it has finished (which necessarily happens when the `done` instruction is executed), or it was preempted by a higher priority channel (which is possible but not systematic when the `yield`(`ge`) is executed).

Upon a channel change the SDMA goes through a context switch procedure. When the current channel yields or ends, the context for that channel is saved into the context RAM locations for that channel. When the next channel starts running, its context is first restored from RAM.

Since context RAM is not yet initialized by reset, there will be no context restore at the beginning of the first channel (bootload channel) run after reset. It is expected that the bootload channel will be used to initialize the context for all other channels. When the bootload channel finishes running or yields, SDMA will enter its SAVE state and save that channel's context into RAM. Then, if the bootload channel is called again later, the context will be restored from RAM when the channel starts again.

The context structure for each channel is defined in Section 38.12.4, "Context Switching," and Table 38-41. There will be one context area reserved for each channel. When a channel ends or yields, the SDMA core registers are automatically saved into the context RAM and later restored from the context RAM when the channel is next run. The total RAM space reserved for 32-channel contexts is either 3K or 4K depending on whether the SMSZ bit is set in the CHN0ADDR register, which enables an additional 8 words of scratch RAM for each context.

38.4.4.1 Context Switch Modes

The exact procedure to save the context of the old channel, and to restore the context of the new channel depends on the context switch mode selected by the AP in the CONFIG control register. The following are the context switch modes:

- By default, the "dynamic" context switch is set. This mode provides the most efficient context switch for an average of eight cycles to stop the current channel, save its context, restore the next channel context, and resume its execution. It consists of saving modified registers of the current channel in the background (for example, during the channel execution)—which leaves very few registers to save when the switch is decided—resuming execution of the next channel as soon as possible (for example, when the minimal set of registers is restored), and continuing the restore phase during this execution.
- In "dynamic with no loop" mode, the same principle is followed except the modified registers are only saved in the background when the loop flag is not set. This mode offers almost the same effectiveness as the previous one, but it prevents the system from accessing the RAM during loops to save power. This is the recommended mode for an efficient context-switch when the loop bodies are short.
- In "dynamic power" mode, no background saving is performed, which reduces power consumption to the minimum. The modified registers are only saved when the context switch starts. The restore

phase is the same as before. This is the mode that achieves the optimal power consumption at the cost of a slower context-switch.

- In a “static” context switch, all the registers are saved when a context switch is decided, and all the registers are restored before starting the execution of the new channel. This mode enables a predictable behavior of the context switch since all the registers are restored prior to the channel start and all registers are saved after the channel termination.

NOTE

Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized during the context SAVE phase when the channel is done or yields. Subsequent calls to the same channel or different channels may use any of the dynamic context modes. This will ensure that all context locations for the bootload channel are initialized, and prevent undefined values in context RAM from being loaded during the context restore if the channel is re-started later.

38.4.4.2 Context Switch Procedure

The Program Control Unit goes into the *save* state, the current context is spilled into memory, and the next channel context is restored according to the context-switch mode that was selected by the AP.

The context switch procedure is as follows:

1. Load the current context’s spill base address.
2. Spill the modified registers of the current channel to memory according to the selected context switch mode while the channel is running.

   On a `done` or `yield(ge)` that causes the channel preemption, the PCU goes into the *save* state. In *static* mode, all the registers are saved; whereas, in either *dynamic* mode, the registers that were modified but not yet saved are then saved, and the PCU registers and flags are finally saved.
3. Put the SDMA core into *sleep* and wait for new channels to be serviced. This step is skipped if there are pending channels when the current channel is saved.

   As soon as there is at least one pending channel, the PCU goes into its *restore* state to restore the context of the channel that was elected by the scheduler.

   Once a channel is elected, it remains the current channel until its script requests a rescheduling operation with a `done` or `yield(ge)` instruction. That means the current channel cannot be modified by the AP, even if it is no more runnable or if its priority is modified.

   The AP can however force a reschedule by writing the corresponding bit in the CONFIG register, which has the same effect as if the script had executed a `done` instruction. That feature should only be used to stop the SDMA in emergency cases.
4. Load the context base-address of the new channel.

   In “static” mode, all the registers are restored. In either “dynamic” modes, only the PCU registers are restored.

   The new channel is running. In “static” mode, no more activity regarding context restoring or saving is performed. In either “dynamic” modes, the registers are restored in the background every time an access to the context RAM is possible, and priority is given to restoring the registers that

are required by the next instruction to be executed. When a register has not been restored and the next instruction needs it, this instruction gets stalled until the register was restored.

In “dynamic” and “dynamic with no loop” modes, background saving of dirty registers is performed every time an access to the context RAM is possible and allowed by the context switch mode.

NOTE

The contents of a channel context space in the context RAM depends on the selected context switch mode. In “dynamic” and “dynamic with no loop” modes, the contents of the context RAM tend to match the contents of the SDMA registers (except for the PCU registers and flags that are never saved in the background). In “dynamic power” and “static” modes, the contents of the context RAM remain unchanged until the channel terminates with a `done` or gets preempted.

38.4.4.3 Context Map in Memory

See Section 38.12.4, “Context Switching.”

38.5 Functional Units

The functional units are small systems that are used by the SDMA core to perform complex calculations like the CRC unit, or to handle data transfers between the core and a bus domain external to the SDMA.

The SDMA core is able to control and exchange data with these systems by sending instructions and reading or writing data from/to the functional units’ registers using the FUBUS. This is done with the `ldf` and `stf` instructions.

The following sections provide introductions to the available functional units. Section 38.17, “Functional Units Programming Model,” provides descriptions the functional units’ behaviors.

38.5.1 CRC Calculation Unit

The Cyclic Redundancy Check (CRC) unit can perform CRC calculation. A single byte of data can be processed every cycle, but up to four bytes can be simultaneously loaded.

The CRC unit supports the following set of polynomials:

```
CRC32:   X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X+1
CRC16:   X^16+X^15+X^2+1
CCITT16: X^16+X^12+X^5+1
IS136:   X^12+X^10+X^8+X^5+X^4+X^3+1
CRC10:   X^10+X^9+X^5+X^4+X+1
CRC8:    X^8+X^2+X+1
parity:  X^8+1
```

38.5.1.1 CRC Structure

Figure 38-9 describes the overall structure of the CRC unit and introduces its registers that are accessible by the SDMA core using the FUBUS.

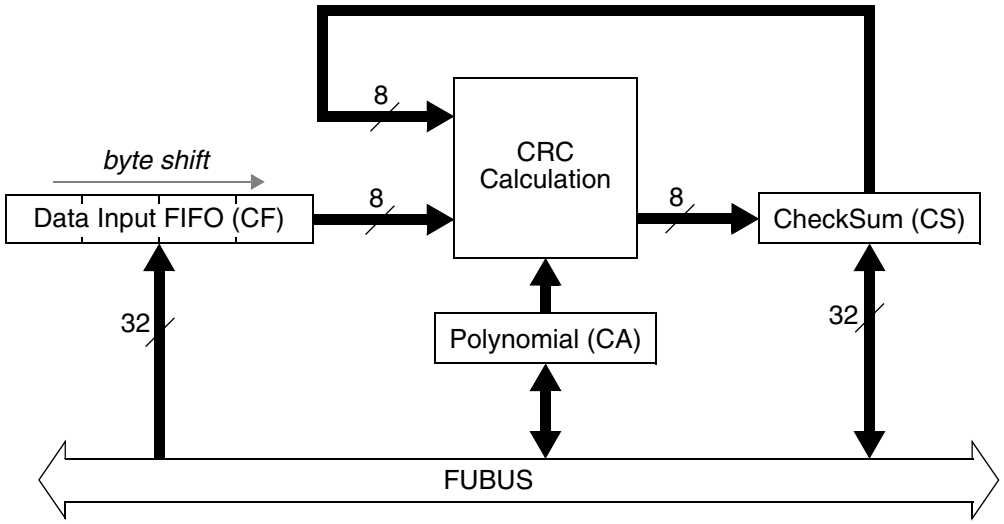


Figure 38-9. CRC Structure

38.5.1.2 CRC Data Processing

CRC processing requires the following three stages:

1. The preliminary initialization stage consists of selecting the desired polynomial, and storing the initialization pattern into the checksum register.
2. Data processing itself, which comes to feeding incoming bytes to the CRC input FIFO — bytes can be written one at a time (8-bit write access from the SDMA core), by pairs (16-bit write), or groups of four (32-bit write). One byte is processed every cycle: Any subsequent access may stall the SDMA core until completion of the previous calculation.
3. The CRC result (or checksum) can be retrieved at any time, but all data must be processed before it is made available.

38.5.1.3 CRC Registers

The CRC enables reading and writing to three register addresses, which trigger the operations described in Section 38.5.1.2, “CRC Data Processing.” The following are the three registers:

- *CA (CRC algorithm)*—The CA selects the desired polynomial. Reading and writing this register correspond to retrieving and changing the polynomial.
- *CS (CRC checksum)*—Writing to this register initializes the checksum accumulator, and reading this register yields the result from the CRC calculation (the result width depends on the polynomial size).
- *CF (CRC FIFO)*—Writing to this register loads the data into the input FIFO and triggers the CRC calculation process. It is not possible to read this register.

38.5.1.4 CRC Summary

Every operation mentioned in Section 38.5.1.3, "CRC Registers," takes time to be executed by the CRC unit. When the CRC receives a command, it immediately acknowledges the SDMA core provided it is not busy completing a previous operation. Therefore, wait-states are inserted by the CRC when a command succeeds another command that is not yet competed. Table 38-5 lists the number of cycles that the CRC takes to execute every possible command. When an operation lasts one cycle, it means the CRC is able to process another command in the next clock cycle (for example, no wait-state is inserted by the CRC because of the former operation that is processing).

Table 38-5. CRC Processing Summary

Operation	Command	Delay	Comments
Set the polynomial	Write CA	1	—
Retrieve the polynomial	Read CA	1	—
Initialize the checksum	Write CS	1	—
Retrieve the checksum	Read CS	1	—
Store 1 byte to process	Write CF	1	—
Store 2 bytes to process	Write CF	2	CRC is busy for 1 cycle
Store 4 bytes to process	Write CF	4	CRC is busy for 3 cycles

38.5.2 Burst DMA Unit

The burst DMA unit enables the SDMA core to perform data transfers to and from the AP memory. It is optimized for accessing SDRAM-like devices. It does not provide control to assign a privilege level to the DMA access. The burst DMA unit provides the SDMA with means to do the following:

- Perform up to 8-beat read and write bursts to the AP memory, which optimizes throughput when accessing SDRAM-type devices because of an internal, 36-byte FIFO
- Access the AP memory at once or twice the SDMA core frequency
- Copy data from one AP memory location to another AP memory location at the AP bus speed, which provides a very high throughput
- Control the method for addressing the AP memory (automatic increment of addresses or frozen addresses—the former aimed at accessing RAM-like memory and the latter aimed at accessing single-address FIFOs)
- Enable or disable automatic prefetch when reading data from the AP memory. When the prefetch mode is selected, the burst DMA automatically triggers external bursts to fill its FIFO without waiting for the SDMA core to request the corresponding data, greatly improving throughput.
- Rely on the DMA to automatically flush its FIFO content when there is enough data to generate an 8-beat burst to the AP memory. Or, it forces a flush when a data transfer must terminate.

  In the former case, the SDMA core may only be stalled when it tries writing data and there is not enough room left in the FIFO. In the latter case, the core is stalled until the data is effectively written to the AP memory.

In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the AP memory. This error status is retrieved by a later access to the burst DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the AP memory is caught.

- Handle address alignment issues between the AP memory map and the SDMA core data. This enables the core to read or write 32-bit data from the burst DMA, whereas the corresponding AP address is not 32-bit aligned. This drastically improves the SDMA scripts' efficiency since the same loop that transfers 32 bits at a time can be used regardless of the start and end addresses in the AP memory space.

This unit structure and registers are described in Section 38.5.2.1, "Burst DMA Structure," and Section 38.5.2.2, "Burst DMA Registers."

38.5.2.1 Burst DMA Structure

The burst DMA is depicted in Figure 38-10. It is essentially made up of a 36-byte FIFO, address registers, and a controlling state-machine. The 36-byte FIFO enables eight-word buffering with address alignment, and the state-machine manages clock adaptation when required.

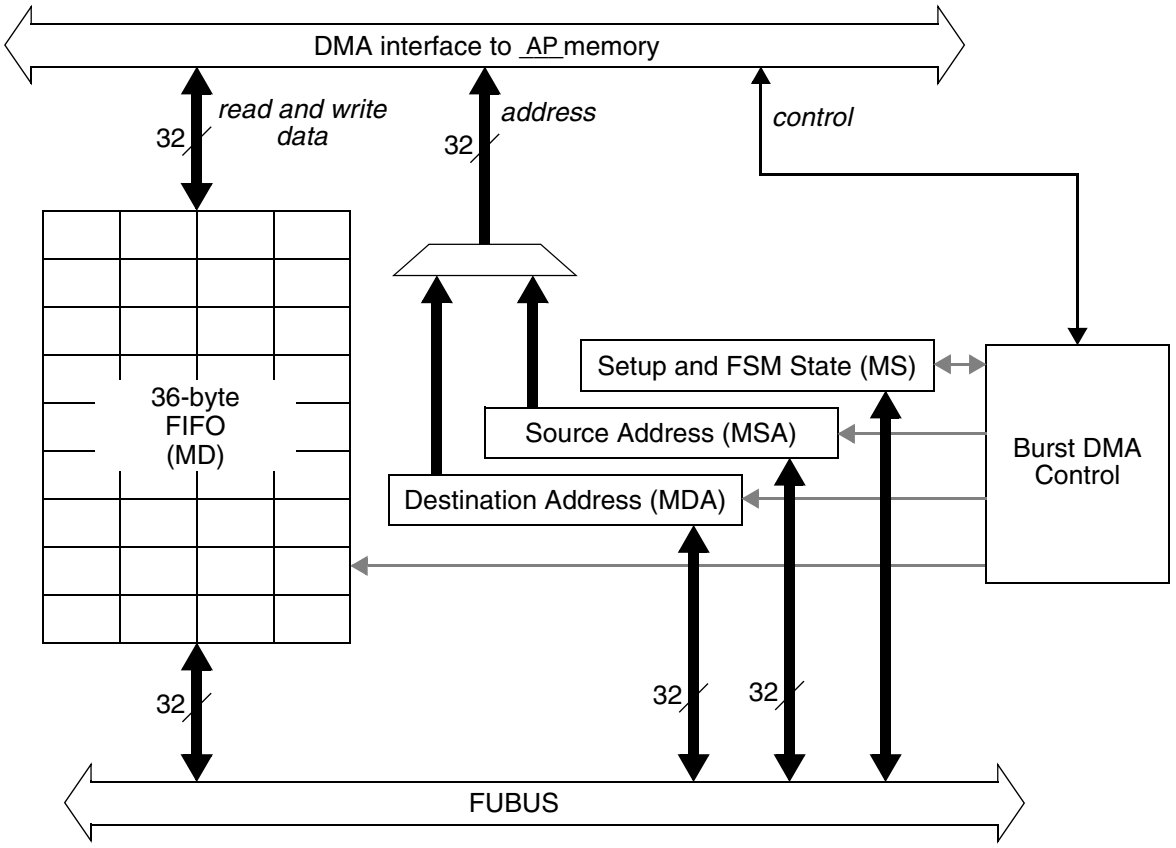


Figure 38-10. Burst DMA Structure

38.5.2.2 Burst DMA Registers

There are four registers, as follows, that may be accessed from the SDMA core:

- MSA (Memory Source Address) — Holds the source byte address in the AP memory map for reading data from this location. This register is automatically modified every time the core reads new data from the FIFO.
- MDA (Memory Destination Address) — Holds the destination byte address in the AP memory map for writing data to this location. This register is automatically modified every time the core writes new data into the FIFO.
- MD (Memory Data) — Labels the 36-byte FIFO access point: Reading a byte, halfword, or word from MD respectively retrieves the first 1, 2, or 4 bytes of the FIFO (for example, the bytes that were stored first by the DMA state-machine when transferring data from the AP memory).

  When the FIFO does not hold as many bytes as required by the SDMA core, the core is stalled until the missing bytes are read from the AP memory. In the case of prefetch mode, the DMA controller decides when it should start a burst to AP memory in order to reduce the risk to not have the required data for the future accesses of the core. When there is no prefetching, a burst is triggered when the required data is not available in the FIFO.

  Writing a byte, halfword, or word to MD stores 1, 2, or 4 bytes, respectively, at the end of the FIFO (for example, these bytes are transmitted to the AP memory after all the other bytes that were previously stored in the FIFO). When the FIFO does not have enough room left to hold the written data, the SDMA core is stalled until a sufficient amount of FIFO contents are flushed out to the AP memory. Flushing is decided by the DMA controller when there are enough bytes in the FIFO to perform the largest allowed burst to AP memory (the exact size depends on the burst start address and the AHB 1 Kbyte boundary rule). However, the SDMA core has the ability to force the flushing operation at any time, for example, when at the end of the data transfer, prior to channel closure.
- MS (Memory Setup) — Contains the state of the burst DMA control, the two flags that define whether each address register is incremented after every access to the external memory, and another flag that is set when a bus error occurred.

38.5.2.3 Data Transfers

Three typical usages have been identified that involve the burst DMA: the data transfer startpoint, the endpoint, or both. Every case requires a different procedure, as listed in the following sections:

38.5.2.3.1 Data Retrieval from the AP Memory

The following steps retrieve data from AP memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the source address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the source address register itself (MSA).
- Read data from the FIFO using the *ldf MD* instruction as many times as needed. If an error occurred during the fetch from AP memory, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from the FIFO.

38.5.2.3.2 Storing Data Into the AP Memory

The following steps store data from AP memory using the burst DMA unit:

- Set up the MS flags to reflect the mode for the destination address (incremented or frozen according to the type of accessed device: memory or peripheral FIFO), then initialize the destination address register itself (MDA).
- Store data into the FIFO using the *stf MD* instruction as many times as needed.
- When the transfer is finished and if the DMA worked in automatic flush mode, force the flush of the FIFO. This instruction is stalled until all the FIFO data is effectively sent to the AP memory and the error status of the transfer is available in the DF flag.

38.5.2.3.3 Transferring Data Between Two AP Memory Locations

The following steps copy data between two AP memory locations using the burst DMA unit:

- Set up the MS flags to reflect the modes for the source and destination addresses (all the combinations are possible), then initialize the source address register (MSA) and the destination address register (MDA). Both addresses must be word-aligned.
- Use as many *stf MD* instructions with the *COPY* flag as needed. Every instruction triggers a burst read of a given number of words from the source address (this number is provided to the burst DMA using the SDMA core general purpose register, which is referenced in the *stf* instruction). Once all the data is loaded into the FIFO, the DMA empties it with a write burst of the same count to the destination address. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.
- Once the transfer is done, there should be a final access to the burst DMA to check the error status.

38.5.3 Peripheral DMA Unit

The peripheral DMA unit is the second functional unit that connects the SDMA to the AP memory. Unlike the burst DMA, it does not support burst transfers and is optimized for accessing peripherals. It does not provide control to assign a privilege level to the DMA access. Its feature list comprises the following:

- Access to the AP peripherals or memory at once or twice the SDMA core frequency
- Data copy from one AP memory location to another AP memory location at memory bus speed, improving throughput
- Control of the method for addressing the AP memory (automatic increment or decrement of addresses or frozen addresses, the first ones aimed at accessing RAM-like memory and the last one aimed at accessing single-address FIFOs)
- Selectable automatic prefetch when reading data from the AP memory. In prefetch mode, the peripheral DMA automatically fetches another data—without waiting for the SDMA core to request it—when its data register is empty, which improves the throughput
- Selectable automatic flush. In this mode, the SDMA core may only be stalled when it tries writing data and the previous write operation is not finished yet; whereas, in forced flush mode, the core is stalled until the data is effectively written to the AP memory.

In automatic flush mode, the core receives an acknowledge that does not reflect the actual error status when the data is effectively written into the AP memory or the peripheral. This error status is retrieved by a later access to the peripheral DMA. Terminating a write data transfer with a forced flush command guarantees that any bus error to the AP memory has been caught.

This unit structure and registers are described in Section 38.5.3.1, "Peripheral DMA Structure," and Section 38.5.3.2, "Peripheral DMA Registers."

38.5.3.1 Peripheral DMA Structure

The peripheral DMA is shown in Figure 38-11. It is made up of a 32-bit data register, two address registers, and a controlling state-machine. The state-machine manages clock adaptation, when required.

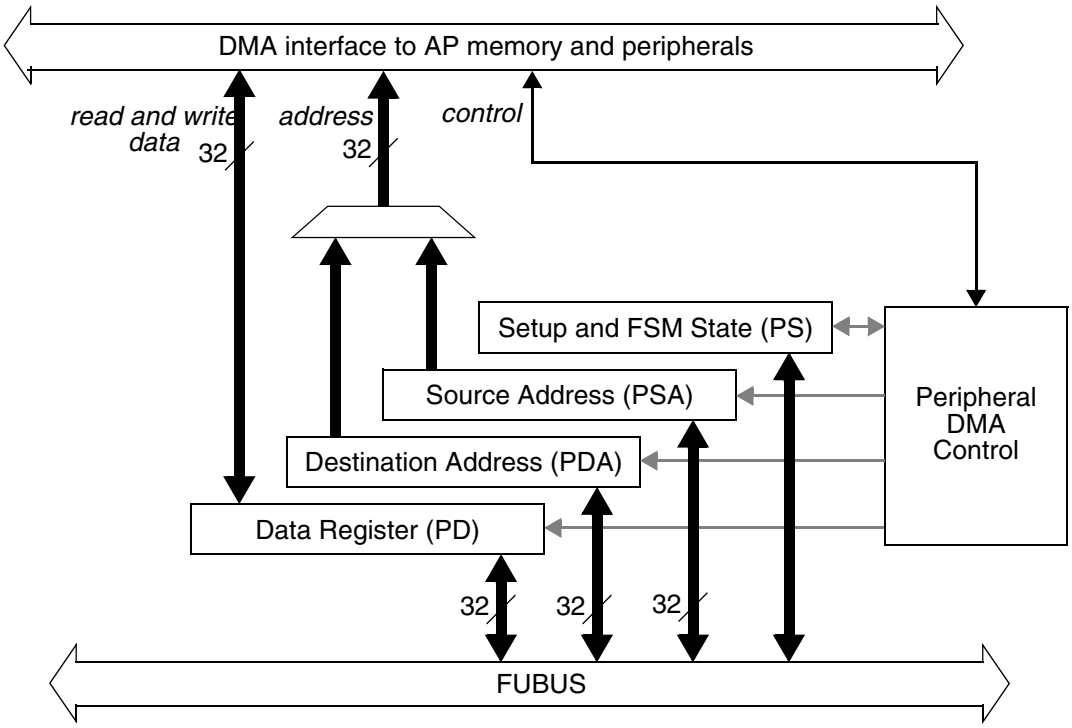


Figure 38-11. Peripheral DMA structure

38.5.3.2 Peripheral DMA Registers

According to Figure 38-11, the peripheral DMA has four registers that may be read or written by the SDMA core:

- *PD (Peripheral Data)* is the DMA 32-bit data register.
- *PSA (Peripheral Source Address)* holds the source byte address in the AP memory map for reading data from this location. This register is automatically modified every time the core reads a new data from PD.
- *PDA (Peripheral Destination Address)* holds the destination byte address in the AP memory map for writing data to this location. This register is automatically modified every time the core writes a new data into PD.

- *PS (Peripheral Setup)* contains the state of the peripheral DMA control, two configuration fields that define the way address registers are modified after every data access, two additional configuration fields that define the data size to access the source and destination devices, and another field that contains the latest transfer error status.

38.5.3.3 Peripheral DMA Data Transfers

There are three typical usages that involve the peripheral DMA, whether it is the data transfer start-point, endpoint, or both. Every case requires a different procedure, as described in Section 38.5.3.3.1, "Data Retrieval from the AP Memory or Peripheral," Section 38.5.3.3.2, "Storing Data into the AP Memory or Peripheral," and Section 38.5.3.3.3, "Transferring Data Between Two AP Memory Locations."

38.5.3.3.1 Data Retrieval from the AP Memory or Peripheral

The following steps retrieve data from AP memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the source (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the source address register itself (PSA) with an address that is aligned to the programmed data size.
- Read data from PD using the `ldf PD` instruction as many times as needed. If an error occurs during the fetch from the AP memory or peripheral, the DMA control tags the error status on the data and the SDMA core SF flag is set when reading this data from PD.

38.5.3.3.2 Storing Data into the AP Memory or Peripheral

The following steps store data to AP memory using the peripheral DMA unit:

- Set up the PS fields to reflect the mode and data size for the destination (incremented, decremented, or frozen address register; 8-bit, 16-bit, or 32-bit data transfers), then initialize the destination address register itself (PDA) with an address that is aligned to the programmed data size.
- Store data into PD using the *stf PD* instruction as many times as needed.
- When the transfer is finished and if the peripheral DMA worked in automatic flush mode, force the flush of PD. This instruction is stalled until PD contents are effectively sent to the AP memory or peripheral, and the error status of the transfer is available in the DF flag.

38.5.3.3.3 Transferring Data Between Two AP Memory Locations

The following steps copy data between two AP memory locations using the peripheral DMA unit:

- Set up the PS fields to reflect the modes and data size for the source and destination addresses (all the combinations of addressing modes are possible, but both data sizes must be identical), then initialize the source address register (PSA) and the destination address register (PDA). Both addresses must be aligned with the programmed data size.
- Use as many *stf PD* instructions with the *COPY* flag as needed. Every instruction triggers a single read from the source address; a single write of the received data immediately follows. The DMA acknowledges prior to instruction completion, which frees the SDMA core for other tasks at no delay cost.

- Once the transfer is done, there should be a final access to the peripheral DMA to check the error status.

38.6 SDMA Security Support

The SDMA provides support to SDMA software to block unauthorized updates to the scripts in RAM.

SDMA supports the following Security modes:

- Open Mode: has full control to load scripts and context into SDMA RAM. This is the default mode.
- Locked Mode: The AP loads scripts and channel contexts at startup when it is still executing known safe software. When finished, it locks the SDMA to prevent further updates to RAM and selected registers. More details described in Section 38.6.1, “Locked Mode.”

There is no SDMA Privilege signal associated with the Peripheral DMA interface.

38.6.1 Locked Mode

The LOCK bit in the SDMA_LOCK register provides support for SDMA scripts to freeze RAM contents after the initial bootload routine to prevent future unauthorized updates to SDMA RAM. After initial RAM contents are uploaded, AP software can set the LOCK bit to secure the RAM contents to prevent future updates by an unauthorized. After the LOCK bit is written with a ‘1’, the SDMA is “locked” until reset.

The LOCK bit can be read in the SDMA’s internal memory map in the LOCK register (see Section 38.13.3.20, “Lock Status Register (LOCK)”). SDMA scripts which load information into RAM can check the value of the LOCK bit to determine if an upload to RAM is allowed. If not allowed, the script can refuse to allow the request to copy data into the RAM to continue. While SDMA is locked, attempts to write to the SDMA_LOCK, CHN0ADR, ILLINSTADDR, and ONCE_ENB registers will be ignored. All registers remain readable. Writes to other registers are still allowed.

Once the SDMA is locked, the LOCK bit can only be cleared by a reset. A hardware reset will always clear the LOCK bit. A software reset initiated by writing to the RESET register will only clear the LOCK bit if the SRESET_LOCK_CLR bit in the SDMA_LOCK register is set. Since SDMA_LOCK register cannot be updated if SDMA is locked, the SRESET_LOCK_CLR bit must be configured before setting the LOCK bit. The SREST_LOCK_CLR bit will also be cleared by resets that clear the LOCK bit.

The SDMA RISC core uses the ILLINST and CHN0ADDR registers as pointers to determine where to jump to after an illegal instruction or upon boot after a reset. The LOCK bit prevents updates to these registers to protect against unauthorized changes to these pointers.

While SDMA is locked, the ONCE_ENB register cannot be written to prevent the OnCE under AP control from being used to gain access to SDMA internal memory. If AP control of the OnCE is enabled before setting the LOCK bit, the AP can use the ONCE for debug purpose after LOCK is set.

38.7 OnCE and PCU Debug States

The SDMA has two different debug modes in which the OnCE performs debug instructions. See Figure 38-4 for an example of the PCU states in debug. The following are the two debug states:

- When a channel is running (that is, when CCR and CCPRI are different from 0, which can be read in the PSW register), SDMA can execute a `SoftBkpt` instruction from the channel script or receive a debug request. When either happens, the SDMA enters its "Classical" *Debug* state, which is described in Section 38.17.4, "OnCE and Real-Time Debug."
- When a channel is not running, the SDMA can be in *Sleep* state or in *Sleep after Reset* state. If a debug request is sent to the core, it enters its *Debug in Sleep* state. This debug mode works similarly to the "Classical" *Debug* state, except it returns to the original state (*Sleep* or *Sleep after Reset*) when the debug mode is left using the `exec_core` instruction of the OnCE. From this *Debug in Sleep* state, the SDMA can execute a program whereas no channel is running. If a new debug request is sent to the core or if a `SoftBkpt` is executed, it comes back to this *Debug in Sleep* state.

The OnCE is provided with several instructions that can be executed when the core is in either debug state. Table 38-6 summarizes the behavior of these OnCE debug instructions. There exists other secondary OnCE instructions that are described in Section 38.17.4, "OnCE and Real-Time Debug."

Table 38-6. SDMA in Debug Mode

Instruction	Debug	Debug in Sleep
`exec_once`	`exec_once` <instruction> SDMA executes the <instruction> and returns to the *Debug* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.	`exec_once` <instruction> SDMA executes the <instruction> and returns to the *Debug in Sleep* state. The Program Counter (PC) is not incremented. This command must not be used with an instruction that modifies the PC value.
`run_core`	`run_core` <instruction> SDMA executes the <instruction>, leaves the *Debug* state and continues executing the channel script from the position where it stopped. This command must not be used with an instruction that modifies the PC value.	`run_core` <instruction> SDMA executes the <instruction> and returns to its *Sleep* or *Sleep after Reset* initial state. This command must not be used with an instruction that modifies the PC value.
`exec_core`	`exec_core` <instruction> It is similar to `run_core` except it requires an instruction that changes the PC value (jump, branch...): the SDMA jumps to the new PC value, leaves the *Debug* state and starts executing instructions from this new PC value.	`exec_core` <instruction> If the previous state was *Sleep after Reset*, the SDMA returns to this state, and Chn0Addr value overrides the PC value. Otherwise, the SDMA jumps to the new PC value and starts executing instructions from this new PC.

NOTE

The feature `exec_core` in *Debug in Sleep* after *Sleep after Reset* was added for the Channel boot (channel 0) to allow the debugger to return to *Sleep after Reset* state with a new PC value. The SDMA will be ready to boot at the Chn0Addr address.

38.8 SDMA Clocks and Low Power Modes

The SDMA receives several root clocks from the SoC clock controller module and performs adaptive clock gating to optimize its power consumption. From a user standpoint, clock gating and power mode selection are fully automatized inside the SDMA. Root clock control is available from the SoC clock controller module.

There are numerous clock sources that are used in the SDMA. They belong to one of two possible clock domains listed in Table 38-7, and have frequency constraints within each domain. Clocks are considered asynchronous between domains.

Within the AP/SDMA clock domain, all clocks must come from the same PLL. The AP DMA interfaces (peripheral DMA and burst DMA) receive their clock from the AP DMA clock source whose frequency can be once or twice the frequency of the SDMA core clock. The DMA interfaces are designed to work at the AP DMA frequency, but the SDMA core is physically limited to a maximum 104 MHz frequency. Since this is lower than the maximum AP DMA frequency, the SDMA core clock is tied to the AP peripheral clock frequency.

The AP Peripheral Bus Clock source must be an exact sub-frequency of the SDMA Core clock source (any integer value greater or equal to 1).

Table 38-7. Clocking Scheme

Clock Domain	Source Clock	Comments
AP	SDMA core (SDMA main core)	Source clock for the core and all its operations; this clock is thus used by most of the SDMA sub-modules.
	AP DMA	DMA interface for the peripheral DMA and the burst DMA. It is balanced with the main clock source, and its frequency is either once or twice the main clock frequency.
	AP peripheral	Connection to the AP peripheral bus. It is a sub-frequency of the main clock frequency.
JTAG	TCK	Clock for JTAG access, limited to maximum of 1/8 of the SDMA core clock frequency.

The JTAG clock is sampled by the SDMA main clock to determine its rising edge. This simplifies design and clock management, but it also adds a ratio constraint between those two clocks. It is guaranteed the JTAG interface works properly when the frequency of TCK is lower than $1/8^{th}$ of the frequency of the SDMA main clock (which is about 8 MHz when the SDMA core clock frequency is 66 MHz).

38.8.1 Clock Gating and Low Power Modes

The SDMA automatically performs power saving without requiring user involvement. It implements two levels of automatic clock gating.

38.8.1.1 Coarse Clock Gating

Every sub-module clock comes from one of the five available sources, and is gated with the sub-module specific enabling condition. Table 38-8 displays the sub-module clocks and their source. It also indicates the relationships that may exist between different sub-modules clock enables.

Table 38-8. Sub-Modules Clocks

Sub-Module	Source Clocks	Enabling Condition and Comments	Related Enabling Conditions
Core	SDMA Main Core	The core sub-module clock is running when the core is not in one of its sleep states (Sleep or Sleep after Reset) or there is a pending channel. Typically, the core sub-module clock is stopped once all the channels are processed and the core enters its sleep state. A new pending channel awakes the core sub-module clock.	None
Memories	SDMA Main Core	The clock activation only occurs during a core access.	Disabled when Core sub-module clock is disabled or no memory access in progress
Scheduler	SDMA Main Core	Its clock only runs when scheduling is needed: for example, when there are pending channels, upon reception of a DMA request, and anytime the AP modifies the channel running conditions.	None
AP Control	SDMA Main Core and AP peripheral	The AP peripheral clock is solely used to determine the frequency ratio with the SDMA main clock. The control registers' clock is based on SDMA main clock; it is active when the AP or the SDMA modifies the contents of one of these registers.	None
CRC	SDMA Main Core	The CRC clock is based on SDMA main clock and is only active during data processing.	Disabled when Core sub-module clock is disabled
Burst DMA	SDMA Main Core and AP DMA	The burst DMA has two clocks: The first clock is derived from the SDMA main core clock and drives registers that are connected to the FUBUS. The second clock is derived from the AP DMA clock and drives registers that are connected to the AP DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the burst DMA is not used by the running channel script).	Disabled when Core sub-module clock is disabled
Peripheral DMA	SDMA Main Core and AP DMA	The peripheral DMA has two clocks: The first clock is derived from SDMA main clock and drives registers that are connected to the FUBUS. The second clock is derived from the AP DMA clock and drives registers that are connected to the AP DMA bus outside the SDMA. Both clocks are enabled during active phases of data transfers (for example, these clocks are turned off when the peripheral DMA is not used by the running channel script).	Disabled when Core sub-module clock is disabled
OnCE	SDMA Main Core	The OnCE clock is derived from main source clock. It is disabled by default. In order to use the OnCE, its clock must be explicitly turned on, either by enabling the OnCE access from the AP peripheral bus (register ONCE_ENB), or by driving the clk_gating_off input pin high. This is a SDMA input whose driver depends on the SoC implementation (typically a JTAG controller). The OnCE also receives the TCK input, which is the JTAG clock. It does not use it as a functional clock; the TCK input is sampled instead. See Section 38.18.5.2, "Synchronization Implementation."	When enabled, all other clocks are systematically on (clock gating is off)

38.8.1.2 Refined Clock Gating

The SDMA implements a second level of clock gating on a register-per-register basis. Unlike the first level that covers all the SDMA flip-flops, except the synchronizers (only five flip-flops are always running), the second level is only available for eligible registers, which amounts to about 90% of the SDMA flip-flops.

These gated registers are only clocked when the hardware logic detects a new data loading. This additional gating further reduces dynamic power consumption.

38.8.1.3 Low Power Modes and User Control

Power savings are automatically managed by the SDMA hardware without any user involvement; however, one can distinguish three different power modes: SLEEP, RUN, and DEBUG. The following Table describes these modes, and shows how to switch from one mode to another.

Table 38-9. Power Modes

Power Mode	Sub-modules								Comments
	Core	Memories	Scheduler	AP Control	CRC	Burst DMA	Peripheral DMA	OnCE	
SLEEP	off[1]	off	wait[2]	wait	off	off	off	off	Set when the PCU state is either *Sleep* or *Sleep after Reset* and the SDMA is not in DEBUG mode. This is the default mode after reset.
RUN	on[3]	wait	wait	wait	wait	wait	wait	off	Set for the other PCU states that are reachable out of debug: *Program*, *Data*, *Change of Flow*, *Error in Loop*, *Debug*, *Functional Unit*, *Save,* or *Restore*.
DEBUG	on	on	on	on	on	on	on	on	Set regardless of the PCU state when clock gating is turned off to use the OnCE features (either *clk_gating_off* pin high or ONCE_ENB[0] set).

1 *off*: no clock

2 *wait*: only clocked when accessed or stimulated

3 *on*: clock is always running

It is possible to control the SDMA power mode. The procedures to force the SDMA into either mode are described in Section 38.8.1.3.1, “SLEEP Mode.”

38.8.1.3.1 SLEEP Mode

This is the default mode after reset; therefore, resetting the SDMA forces this mode. However, the common procedure is as follows:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Disable all channels (using the STOP_STAT control register, and the HO, DO, EO if necessary).
- Wait for the active channels to complete or force a reschedule using the reschedule bit in the RESET register.

- The SDMA is in SLEEP mode making it possible to completely shut off its clock from the chip level clock controller using the procedure described in Section 38.8.1.4, “Stop Mode Response.”

38.8.1.3.2 RUN Mode

This is the default mode when a channel is running:

- Ensure the *clk_gating_off* pin is low and ONCE_ENB[0] is cleared.
- Activate at least one channel (using the HSTART control registers, a DMA request, and/or the HO, DO, EO register bits).

38.8.1.3.3 DEBUG Mode

The DEBUG mode must be set when one needs to use the debugging facilities of the SDMA:

- Ensure the SDMA clocks are running from the CCM.
- Set the *clk_gating_off* pin high or use the SDMA to set ONCE_ENB[0].

38.8.1.4 Stop Mode Response

The SDMA receives a stop request from the chip level clock controller. This request may be asserted when the chip enters the stop low power mode. If the SDMA is running when the request is received, then the SDMA will complete all pending channels before returning to the SLEEP state. The SDMA sends an acknowledgement to the clock controller when the SLEEP state is entered indicating that the SDMA’s clocks can be turned off.

38.8.2 Reset

After reset (either received from the reset module or a software reset required by the AP), the SDMA is in IDLE mode. It will start its boot code located at address 0 once a channel is activated. Activating a channel can be done by the AP after programming a positive priority and setting the channel bit in the EVTPEND register.

There will not be a context RESTORE for the first channel (bootload channel) called after a reset because the context data in RAM has not been initialized. Static context mode should be used for the first channel called after reset to ensure that the all context RAM for that channel is initialized. Subsequent calls to the same channel or different channels may use any of the dynamic context modes

38.9 Software Interface

A separate document exists that fully describes the SDMA Application Programming Interface (API): See the latest revision of document MOT-SFS-IAPI-SAS-001 (version 0.04).

38.10 Initialization Information

38.10.1 Hardware Reset

After reset, the program RAM, context RAM, data RAM, and RAM containing the channel enable registers (CHNENBLn) have unpredictable contents. The active register set is assigned to channel 0 and the PC is initialized to all zeros. However, since the channel enable register is all zeros, there are no active channels and the SDMA is halted waiting for the boot channel to start.

The AP will have to setup the SDMA in order to boot it. The CONFIG register must be initialized to determine the DMA/core clock ratio (1 or 2). Channel Enable Registers must also be initialized.

To start up the SDMA, the AP first creates some channel control blocks (CCB) and buffer descriptors (BD) in AP memory for the boot channel (channel 0) and then initializes the channel 0 pointer register (MC0PTR) to the address of the first control block. The HSTART, HOSTOVR and EVTOVR registers are then configured according to Equation 38-2 to allow channel 0 to run.

Upon being enabled, the SDMA begins executing the script located at the address indicated by the Channel 0 Boot Address register (CHN0ADDR) in the program memory. The reset value of CHN0ADDR points to the default bootload script in ROM. This ROM script will read the channel 0 pointer register (MC0PTR) to determine the location of the Channel Control Block (CCB) in AP memory. The script will then begin fetching by DMA the first channel control block which contains a pointer to the location channel 0 Buffer Descriptor chain which is also fetched using DMA. If the buffer descriptor contains a valid command, the script interprets the command in each buffer descriptor and proceeds to implement the command and move on to the next buffer descriptor control block. The buffer descriptor commands for channel zero are typically set up to load SDMA's program RAM, Data RAM, and initial values for the channel contexts. Some channel scripts expect particular parameters to be passed

There are two ways to make the SDMA boot on a user-defined script. The OnCE (either using its JTAG interface or its AP Control interface) can be used to download any code in the SDMA RAM and force the SDMA to boot on that code. Also, the CHN0ADDR register in the AP programming model can be modified to point to user code in RAM which would need to either have been loaded using the ONCE or default bootload routine (ex before a S/W reset).

38.10.2 Channel Script Execution

The execution of an SDMA script depends on both the instructions that make up the script, the data context upon which it operates, and commands or parameters passed in the buffer descriptor or. All these items must be initialized before the script is allowed to execute. Each of the 32 channels has a separate context, but may share scripts and locations in data RAM.

Channels are initialized by the AP by using channel 0 to download any required scripts and data values and the channels initial context. The context contains all the initial values of the SDMA core registers. This includes the Program Counter (PC) which is set to the start of the desired script in SDMA program memory.

The AP selects which trigger conditions that must occur for the channel to start by configuring the CHNENBL, HOSTOVR and EVTOVR registers. The trigger events include AP setting HE (HSTART) or

a hardware DMA request asserts an event input to SDMA. The channel can becomes active according to its priority compared with other runnable channels when the selected trigger(s) cause Equation 38-2 to evaluate as true.

The specific parameters to be passed to each script in the buffer descriptor or context are documented in the software documentation for each script.

38.10.3 Initialization and Script Execution Setup Sequence

To summarize, the following steps are minimally required to setup SDMA and run channel scripts.

- Perform Hardware Reset. The program RAM, context RAM, data RAM and CHNENBLn registers have unpredictable contents after this reset.
- Initialize CHNENBLn registers to map DMA request events to desired channels.
- Configure CHNPRIn registers to select priority for runnable channels. A non-zero priority is required for the channel to run.
- Configure the CONFIG register to select DMA to SDMA core clock ratio.
- Set up channel control blocks and buffer descriptors in AP to specify the loading of SDMA program RAM and channel contexts for each SDMA channel to be used.
- Configure MC0PTR register with base address of AP Channel Control Block base address.
- Initialize CHNENBLn registers to map DMA request events to associated channel. Reference Section 38.4.3.3, “Mapping DMA Requests to Pending Channels.”
- Configure CHNPRIn registers to set priority for each channel to be run.
- For each channel to be run, configure HOSTOVR (HO) and EVTOVR (EO) registers to select which events (hardware and/or software trigger events) must occur for the channel to be runnable. Reference Section 38.4.3.5, “Runnable Channels Evaluation.”
- Set bit 0 of the HSTART register to set HE[0] and allow Channel 0 to run (assumes EO[0] andDO[0] were both set in previous step). This will cause SDMA to load the program RAM and channel contexts configured previously.
- Wait for Channel 0 to finish running. This is indicated by HI[0]=1 in the SDMA_INTR register, or by optional interrupt to the AP.
- Set the LOCK bit in the SDMA_LOCK register to prevent un-authorized uploads of data to SDMA RAM.
- Additional channel scrips can now be run by enabling the selected software or hardware trigger event according to Equation 38-2 on page 38-18.

38.11 AP Memory Map and Control Register Definitions

The AP controls the SDMA by means of several interface registers. Those registers are described in the current section.

38.11.1 AP Memory Map

The following table provides the memory map for the AP control SDMA registers.

Table 38-10. AP Memory Map

Offset	Register	Access	Reset Value	Section/Page
General Registers				
AP_BASE+000 (MC0PTR)	AP (MCU) Channel 0 Pointer	R/W	0x0000_0000	38.11.3.2/38-48
AP_BASE+004 (INTR)	Channel Interrupts	R/W	0x0000_0000	38.11.3.2/38-48
AP_BASE+008 (STOP_STAT)	Channel Stop/Channel Status	R	0x0000_0000	38.11.3.3/38-49
AP_BASE+00C (HSTART)	Channel Start	R/W	0x0000_0000	38.11.3.4/38-50
AP_BASE+010 (EVTOVR)	Channel Event Override	R/W	0x0000_0000	38.11.3.5/38-51
AP_BASE+014 (DSPOVR)	Channel BP Override	R/W	0xFFFF_FFFF	38.11.3.6/38-51
AP_BASE+018 (HOSTOVR)	Channel AP Override	R/W	0x0000_0000	38.11.3.7/38-52
AP_BASE+01C (EVTPEND)	Channel Event Pending	R	0x0000_0000	38.11.3.8/38-53
AP_BASE+024 (RESET)	Reset Register	R	0x0000_0000	38.11.3.9/38-54
AP_BASE+028 (EVTERR)	DMA Request Error Register	R	0x0000_0000	38.11.3.10/38-55
AP_BASE+02C (INTRMASK)	Channel AP Interrupt Mask	R/W	0x0000_0000	38.11.3.11/38-56
AP_BASE+030 (PSW)	Schedule Status	R	0x0000_0000	38.11.3.12/38-56
AP_BASE+034 (EVTERRDBG)	DMA Request Error Register	R	0x0000_0000	38.11.3.13/38-57
AP_BASE+038 (CONFIG)	Configuration Register	R/W	0x0000_0003	38.11.3.14/38-58
AP_BASE+03C (SDMA_LOCK)	SDMA LOCK	R/W	0x0000_0000	38.11.3.15/38-59
AP_BASE+040 (ONCE_ENB)	OnCE Enable	R/W	0x0000_0000	38.11.3.16/38-60
AP_BASE+044 (ONCE_DATA)	OnCE Data Register	R/W	0x0000_0000	38.11.3.17/38-61
AP_BASE+048 (ONCE_INSTR)	OnCE Instruction Register	R/W	0x0000_0000	38.11.3.18/38-62
AP_BASE+04C (ONCE_STAT)	OnCE Status Register	R	0x0000_E000	38.11.3.19/38-62
AP_BASE+050 (ONCE_CMD)	OnCE Command Register	R/W	0x0000_0000	38.11.3.20/38-64
AP_BASE+058 (ILLINSTADDR)	Illegal Instruction Trap Address	R/W	0x0000_0001	38.11.3.21/38-65
AP_BASE+05C (CHN0ADDR)	Channel 0 Boot Address	R/W	0x0000_0050	38.11.3.22/38-65
AP_BASE+060 (EVT_MIRROR)	DMA Requests	R	0x0000_0000	38.11.3.23/38-66
AP_BASE+064 (EVT_MIRROR2)	DMA Requests 2	R	0x0000_0000	38.11.3.24/38-67
AP_BASE+070 (XTRIG_CONF1)	Cross-Trigger Events Configuration Register 1	R/W	0x0000_0000	38.11.3.25/38-68
AP_BASE+074 (XTRIG_CONF2)	Cross-Trigger Events Configuration Register 2	R/W	0x0000_0000	38.11.3.25/38-68
AP_BASE+100+n*4 (CHNPRIn)[1]	Channel Priority Registers	R/W	0x0000_0000	38.11.3.26/38-70
AP_BASE+200+n*4 (CHNENBLn)[2]	Channel Enable RAM	R/W	Undefined	38.11.3.27/38-71
AP_BASE+2C0 -AP_BASE+2FC	Reserved	—	Undefined	—

[1] CHNPRIn: For n= 0 to 31

[2] CHNENBLn: For n= 0 to 47

38.11.2 Register Summary

The following definitions serve as a key for the AP control SDMA register summary.

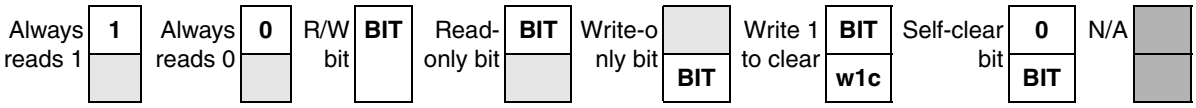


Figure 38-12. Key to Register Fields

Table 38-11 provides a key for register figures.

Table 38-11. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

All registers are clocked with the SDMA clock (which means the AP must ensure that the SDMA clock is running when it wants to access any register).

Table 38-12. AP SDMA Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_BASE+000 (MC0PTR)	R	MCOPTR[31:16]															
	W																
	R	MCOPTR[15:0]															
	W																
AP_BASE+004 (INTR)	R	HI[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	HI[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
AP_BASE+008 (STOP_STAT)	R	HE[31:16]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	HE[15:0]															
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
AP_BASE+00C (HSTART)	R	HSTART[31:16]/HE[31:16]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
	R	HSTART[15:0]/HE[15:0]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
AP_BASE+010 (EVTOVR)	R	EO[31:16]															
	W																
	R	EO[15:0]															
	W																
AP_BASE+014 (DSPOVR)	R	DO[31:16]															
	W																
	R	DO[15:0]															
	W																
AP_BASE+018 (HOSTOVR)	R	HO[31:16]															
	W																
	R	HO[15:0]															
	W																
AP_BASE+01C (EVTPEND)	R	EP[31:16]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
	R	EP[15:0]															
	W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr

Table 38-12. AP SDMA Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_BASE+024 (RESET)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RESCHED	RESET
	W																
AP_BASE+028 (EVTERR)	R	CHNERR[31:16]															
	W																
	R	CHNERR[15:0]															
	W																
AP_BASE+02C (INTRMASK)	R	HIMASK[31:16]															
	W																
	R	HIMASK[15:0]															
	W																
AP_BASE+030 (PSW)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCP[2:0]			NCR[4:0]					CCP[2:0]			CCR[4:0]				
	W																
AP_BASE+034 (EVTERRDBG)	R	CHNERR[31:16]															
	W																
	R	CHNERR[15:0]															
	W																
AP_BASE+038 (CONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	DSPDMA	RTDOBS	0	0	0	0	0	0	ACR	0	0	CSM[1:0]	
	W																
AP_BASE+03C (SDMA_LOCK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SRESETLOCKCLR	LOCK
	W																

Table 38-12. AP SDMA Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
AP_BASE+040 (ONCE_ENB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ENB
	W																
AP_BASE+044 (ONCE_DATA)	R	DATA[31:16]															
	W																
	R	DATA[15:0]															
	W																
AP_BASE+048 (ONCE_INSTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	INSTRUCTION[15:0]															
	W																
AP_BASE+04C (ONCE_STAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
	W																
AP_BASE+050 (ONCE_CMD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	CMD[3:0]			
	W																
AP_BASE+058 (ILLINSTADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	ILLINSTADDR[13:0]													
	W																
AP_BASE+05C (CHN0ADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	SMSZ	CHN0ADDR[13:0]													
	W																
AP_BASE+060 (EVT_MIRROR)	R	EVENTS[31:16]															
	W																
	R	EVENTS[15:0]															
	W																

Table 38-12. AP SDMA Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
AP_BASE+064 (EVT_MIRROR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EVENTS[47:32]															
	W																
AP_BASE+070 (XTRIG_CONF1)	R	0	CNF3	NUM3						0	CNF 2	NUM2					
	W																
	R	0	CNF1	NUM1						0	CNF 0	NUM0					
	W																
AP_BASE+074 (XTRIG_CONF2)	R	0	CNF7	NUM7						0	CNF 6	NUM6					
	W																
	R	0	CNF5	NUM5						0	CNF 4	NUM4					
	W																
AP_BASE+100+n*4 (CHNPRIn)[1]	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHNPRIn[2:0]		
	W																
AP_BASE+200+n*4 (CHNENBLn)[2]	R	ENBLn[31:16]															
	W																
	R	ENBLn[15:0]															
	W																
AP_BASE+2C0-AP_BASE+2FC RESERVED	R	RESERVED															
	W																
	R																
	W																

1 CHNPRIn: For n=0 to 31

2 CHNENBLn: For n=0 to 47

38.11.3 Register Descriptions

The following sections provide figures and detailed field descriptions of the SDMA registers.

38.11.3.1 AP Channel 0 Pointer (MC0PTR)

The following table presents the register; Table 38-13 provides its field descriptions.

AP_BASE+000 (MC0PTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MC0PTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC0PTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-13. AP Channel 0 Pointer (MC0PTR)

Table 38-13. MC0PTR Field Descriptions

Field	Description
31–0 MC0PTR	**Channel 0 Pointer** contains the 32-bit address, in AP memory, of channel 0 control block (the boot channel). See the API document MOT-SFS-IAPI-SAS-001 (version 0.04) for the use of this register. The AP has a read/write access and the SDMA has a read-only access.

38.11.3.2 Channel Interrupts (INTR)

Figure 38-14 presents the register; Table 38-14 provides its field descriptions.

AP_BASE+004 (INTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HI[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HI[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-14. Channel Interrupts (INTR) Register

Table 38-14. INTR Field Descriptions

Field	Description
31–0 HI[31:0]	The AP Interrupts register contains the 32 HI[i] bits. If any bit is set, it will cause an interrupt to the AP. This register is a “write-ones” register to the AP. When the AP sets a bit in this register the corresponding HI[i] bit is cleared. The interrupt service routine should clear individual channel bits when their interrupts are serviced, failure to do so will cause continuous interrupts. The SDMA is responsible for setting the HI[i] bit corresponding to the current channel when the corresponding done instruction is executed.

38.11.3.3 Channel Stop/Channel Status (STOP_STAT)

Figure 38-15 presents the register; Table 38-15 provides its field descriptions.

AP_BASE+008 (STOP_STAT)
Wait State: 0

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HE[31:16]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE[15:0]															
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-15. Channel Stop/Channel Status (STOP_STAT) Register

Table 38-15. STOP_STAT Field Descriptions

Field	Description
31–0 HE	This 32-bit register gives access to the AP Enable bits. There is one bit for every channel. This register is a “write-ones” register to the AP. When the AP writes 1 in bit i of this register, it clears the HE[i] and HSTART[i] bits. Reading this register yields the current state of the HE[i] bits.

38.11.3.4 Channel Start (HSTART)

Figure 38-16 presents the register; Table 38-16 provides its field descriptions.

AP_BASE+00C (HSTART) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HSTART[31:16]/HE[31:16]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HSTART[15:0]/HE[15:0]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-16. Channel Start (HSTART) Register

Table 38-16. HSTART Field Descriptions

Field	Description
31–0 HSTART/HE	The HSTART/HE registers are 32 bits wide with one bit for every channel. When a bit is written to 1, it enables the corresponding channel. Two physical registers are accessed with that address (HSTART and HE), which enables the AP to trigger a channel a second time before the first trigger is processed. • This register is a “write-ones” register to the AP. Neither HSTART[i] bit can be set while the corresponding HE[i] bit is cleared. • When the AP tries to set the HSTART[i] bit by writing a one (if the corresponding HE[i] bit is clear), the bit in the HSTART[i] register will remain cleared and the HE[i] bit will be set. • If the corresponding HE[i] bit was already set, the HSTART[i] bit will be set. The next time the SDMA channel *i* attempts to clear the HE[i] bit by means of a done instruction, the bit in the HSTART[i] register will be cleared and the HE[i] bit will take the old value of the HSTART[i] bit. • Reading this register yields the current state of the HSTART[i] bits. This mechanism enables the AP to pipeline two HSTART commands per channel.

38.11.3.5 Channel Event Override (EVTOVR)

Figure 38-17 presents the register; Table 38-17 provides its field descriptions.

AP_BASE+010 (EVTOVR) Access: User Read/Write
Wait State: 0

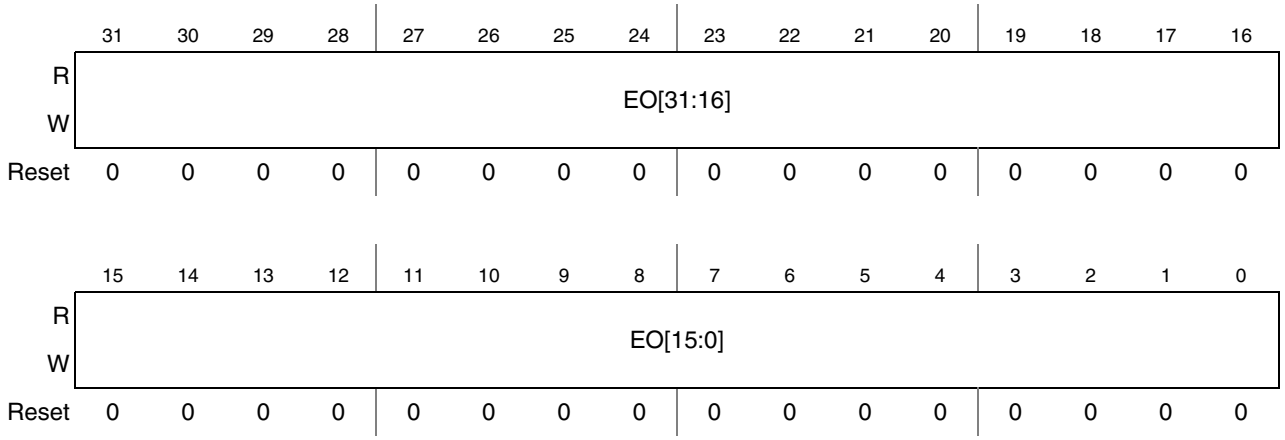


Figure 38-17. Channel Event Override (EVTOVR) Register

Table 38-17. EVTOVR Field Descriptions

Field	Description
31–0 EO	The Channel Event Override register contains the 32 EO[i] bits. A bit set in this register causes the SDMA to ignore DMA requests when scheduling the corresponding channel.

38.11.3.6 Channel BP Override (DSPOVR)

Figure 38-18 presents the register; Table 38-18 provides its field descriptions.

AP_BASE+014 (DSPOVR) Access: User Read/Write
Wait State: 0

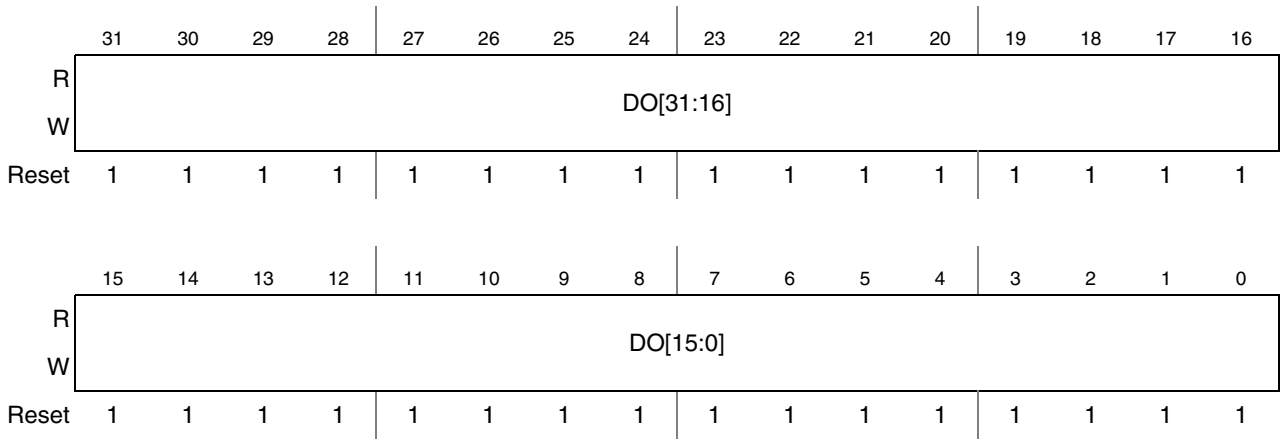


Figure 38-18. Channel DSP Override (DSPOVR) Register

Table 38-18. DSPOVR Field Descriptions

Field	Description
31–0 DO	This register is reserved. **All DO bits should be set to the reset value of 1.** A setting of 0 will prevent SDMA channels from starting according to Equation 38-2 on page 38-18. 0 - Reserved 1 - Reset value.

38.11.3.7 Channel AP Override (HOSTOVR)

Figure 38-19 presents the register; Table 38-19 provides its field descriptions.

AP_BASE+018 (HOSTOVR) Access: User Read/Write
Wait State: 0

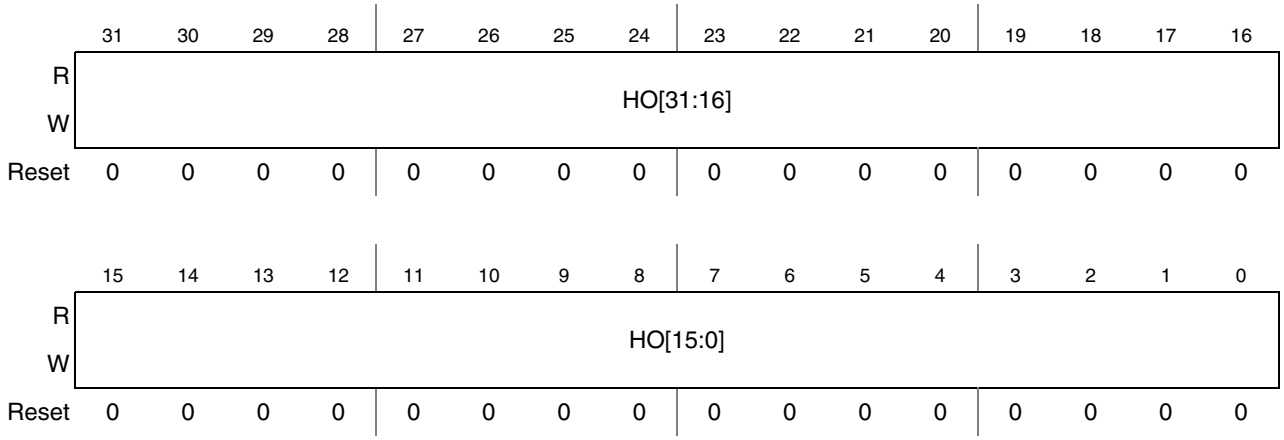


Figure 38-19. Channel AP Override (HOSTOVR) Register

Table 38-19. HOSTOVR Field Descriptions

Field	Description
31–0 HO	The Channel AP Override register contains the 32 HO[i] bits. A bit set in this register causes the SDMA to ignore the AP enable bit (HE) when scheduling the corresponding channel.

38.11.3.8 Channel Event Pending (EVTPEND)

Figure 38-20 presents the register; Table 38-20 provides its field descriptions.

AP_BASE+01C (EVTPEND) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EP[31:16]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EP[15:0]															
W	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-20. Channel Event Pending (EVTPEND) Register

Table 38-20. EVTPEND Field Descriptions

Field	Description
31–0 EP	The Channel Event Pending register contains the 32 EP[i] bits. Reading this register enables the AP to determine what channels are pending after the reception of a DMA request. • Setting a bit in this register causes the SDMA to reevaluate scheduling as if a DMA request mapped on this channel had occurred. This is useful for starting up channels, so that initialization is done before awaiting the first request. The scheduler can also set bits in the EVTPEND register according to the received DMA requests. • The EP[i] bit may be cleared by the done instruction when running the channel *i* script. This a “write-ones” mechanism: Writing a ‘0’ does not clear the corresponding bit.

38.11.3.9 Reset Register (RESET)

Figure 38-21 presents the register; Table 38-21 provides its field descriptions.

AP_BASE+024 (RESET) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RESCHED	RESET
W															slfclr	slfclr
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-21. Reset Register

Table 38-21. RESET Field Descriptions

Field	Description
31–2	Reserved
1 RESCHED	When set, this bit forces the SDMA to reschedule as if a script had executed a done instruction. This enables the AP to recover from a runaway script on a channel by clearing its HE[i] bit using the STOP register, and then forcing a reschedule using the RESCHED bit. The RESCHED bit is cleared when the context switch starts.
0 RESET	When set, this bit causes the SDMA to be held in a software reset. The internal reset signal is held low 16 cycles; the RESET bit is automatically cleared when the internal reset signal rises.

38.11.3.10 DMA Request Error Register (EVTERR)

Figure 38-22 presents the register; Table 38-22 provides its field descriptions.

AP_BASE+028 (EVTERR) Access: User Read-Only
Wait State: 0

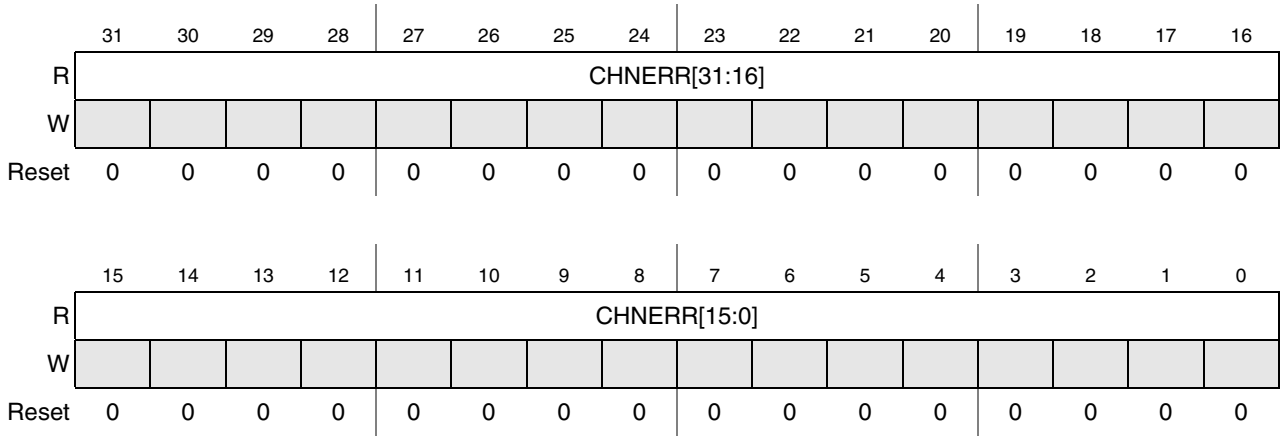


Figure 38-22. DMA Request Error (EVTERR) Register

Table 38-22. EVTERR Field Descriptions

Field	Description
31–0 CHNERR	This register is used by the SDMA to warn the AP when an incoming DMA request was detected and it triggers a channel that is already pending or being serviced. This probably means there is an overflow of data for that channel. • An interrupt is sent to the AP if the corresponding channel bit is set in the INTRMASK register. • This is a “write-ones” register for the scheduler. It is only able to set the flags. The flags are cleared when the register is read by the AP or during SDMA reset. • The CHNERR[i] bit is set when a DMA request that triggers channel *i* is received through the corresponding input pins and the EP[i] bit is already set; the EVTERR[i] bit is unaffected if the AP tries to set the EP[i] bit, whereas, that EP[i] bit is already set.

38.11.3.11 Channel AP Interrupt Mask Flags (INTRMASK)

Figure 38-23 presents the register; Table 38-23 provides its field descriptions.

AP_BASE+02C (INTRMASK) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HIMASK[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIMASK[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-23. Channel AP Interrupt Mask Flags (INTRMASK) Register

Table 38-23. INTRMASK Field Description

Field	Description
31–0 HIMASK	The Interrupt Mask Register contains 32 interrupt generation mask bits. If bit HIMASK[i] is set, the HI[i] bit is set and an interrupt is sent to the AP when a DMA request error is detected on channel *i* (for example, EVTERR[i] is set).

38.11.3.12 Schedule Status (PSW)

Figure 38-24 presents the register; Table 38-24 provides its field descriptions.

AP_BASE+030 (PSW) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NCP[2:0]			NCR[4:0]					CCP[2:0]			CCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-24. Schedule Status (PSW) Register

Table 38-24. PSW Field Descriptions

Field	Description
31–16	Reserved
15–13 NCP[2:0]	The Next Channel Priority gives the next pending channel priority. When the priority is 0, it means there is no pending channel and the NCR value has no meaning. 0 No running channel 1–7 Active channel priority
12–8 NCR[4:0]	The Next Channel Register indicates the number of the next scheduled pending channel with the highest priority.
7–4 CCP[2:0]	The Current Channel Priority indicates the priority of the current active channel. When the priority is 0, no channel is running: The SDMA is idle and the CCR value has no meaning. In the case that the SDMA has finished running the channel and has entered sleep state, CCP will indicate the priority of previous running channel. 0 No running channel 1–7 Active channel priority
3–0 CCR[4:0]	The Current Channel Register indicates the number of the channel that is being executed by the SDMA.In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel.

38.11.3.13 DMA Request Error Register for Debug (EVTERRDBG)

Figure 38-25 presents the register; Table 38-25 provides its field descriptions.

AP_BASE+034 (EVTERRDBG)
Wait State: 0

Access: User Read-Only

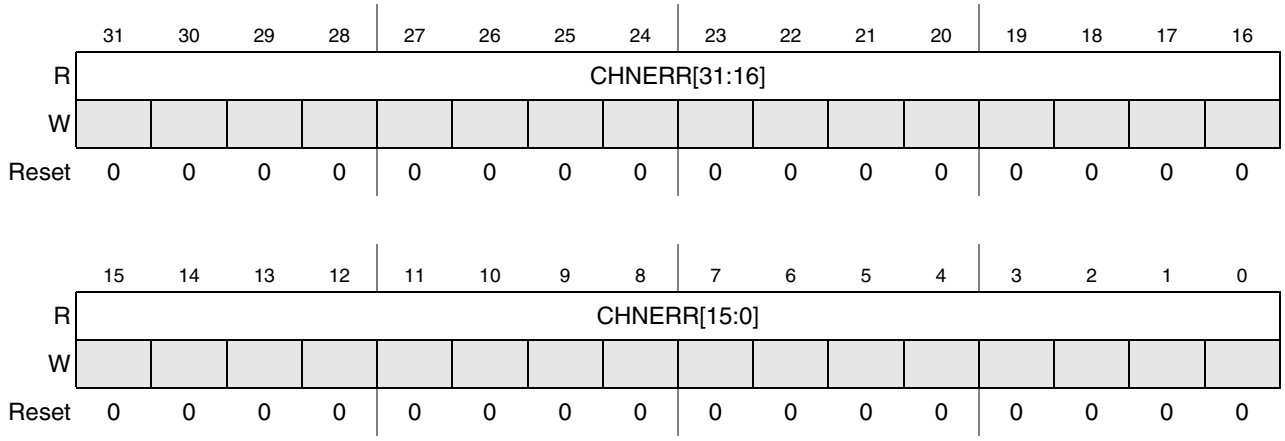


Figure 38-25. DMA Request Error for Debug (EVTERRDBG) Register

Table 38-25. EVTERRDBG Field Descriptions

Field	Description
31–0 CHNERR	This register is the same as EVTERR, except reading it does not clear its contents. This address is meant to be used in debug mode. The AP OnCE may check this register value without modifying it.

38.11.3.14 Configuration Register (CONFIG)

Figure 38-26 presents the register; Table 38-26 provides its field descriptions.

AP_BASE+038 (CONFIG) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	DSP DMA	RTD OBS	0	0	0	0	0	0	ACR	0	0	CSM[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 38-26. Configuration Register (CONFIG)

Table 38-26. CONFIG Register Field Descriptions

Field	Description
31–13	Reserved
12 DSPDMA	This bit's function is reserved and **should be configured as zero**. 0 - Reset Value 1 - Reserved
11 RTDOBS	Indicates if Real-Time Debug pins are used: They do not toggle by default in order to reduce power consumption. 0 RTD pins disabled 1 RTD pins enabled
10–5	Reserved
4 ACR	AP DMA / SDMA Core Clock Ratio. Selects the clock ratio between AP DMA interfaces (burst DMAand peripheral DMA) and the internal SDMA core clock. The frequency selection is determined separately by the chip clock controller. This bit has to match the configuration of the chip clock controller that generates the clocks used in the SDMA. 0 AP DMA interface frequency equals twice core frequency 1 AP DMA interface frequency equals core frequency

Table 38-26. CONFIG Register Field Descriptions (continued)

Field	Description
3–2	Reserved
1–0 CSM	Selects the Context Switch Mode. The AP has a read/write access. The SDMA cannot modify that register. The value at reset is 3, which selects the dynamic context switch by default. That register can be modified at anytime but the new context switch configuration will only be taken into account at the start of the next restore phase. NOTE: The first call to SDMA's channel 0 Bootload script after reset should use static context switch mode to ensure the context RAM for channel 0 is initialized in the channel SAVE Phase. After Channel 0 is run once, then any of the dynamic context modes can be used. 0 static 1 dynamic low power 2 dynamic with no loop 3 dynamic

38.11.3.15 SDMA Lock Register (SDMA_LOCK)

Figure 38-27 presents the register; Table 38-27 provides its field descriptions.

AP_BASE+03C (SDMA_LOCK)
Wait State: 0

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SRESET LOCK CLR	LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-27. SDMA LOCK(SDMA_LOCK) Register

Table 38-27. ONCE_ENB Field Descriptions

Field	Description
31–2	Reserved

Table 38-27. ONCE_ENB Field Descriptions (continued)

Field	Description
1 SRESET_LOCK_CLR	The SRESET_LOCK_CLR bit determine if the LOCK bit is cleared on a software reset triggered by writing to the RESET register. This bit cannot be changed if LOCK=1. SREST_LOCK_CLR is cleared by conditions that clear the LOCK bit. 0 Software Reset does not clear the LOCK bit. 1 Software Reset clears the LOCK bit.
0 LOCK	The LOCK bit is used to restrict access to update SDMA script memory through ROM channel zero scripts and through the OnCE interface under MCU control. The LOCK bit is set: • The SDMA_LOCK, ONCE_ENB,CH0ADDR, and ILLINSTADDR registers cannot be written. These registers can be read, but writes are ignored. • SDMA software executing out of ROM or RAM may check the LOCK bit in the LOCK register on page 38-99 to determine if certain operations are allowed, such as up-loading new scripts. Once the LOCK bit is set to 1, only a reset can clear it. The LOCK bit is cleared by a hardware reset. LOCK is cleared by a software reset only if SRESET_LOCK_CLR is set. 0 LOCK disengaged. 1 LOCK enabled.

38.11.3.16 OnCE Enable (ONCE_ENB)

Figure 38-28 presents the register; Table 38-28 provides its field descriptions.

AP_BASE+040 (ONCE_ENB) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-28. OnCE Enable (ONCE_ENB) Register

Table 38-28. ONCE_ENB Field Descriptions

Field	Description
31–1	Reserved
0 ENB	The OnCE Enable register selects the OnCE control source: When cleared (0), the OnCE registers are accessed through the JTAG interface; when set (1), the OnCE registers may be accessed by the AP through the addresses described, as follows. • After reset, the OnCE registers are accessed through the JTAG interface. • Writing a 1 to ENB enables the AP to access the ONCE_* as any other SDMA control register. • When cleared (0), all the ONCE_xxx registers cannot be written. The value of ENB cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

38.11.3.17 OnCE Data Register (ONCE_DATA)

Figure 38-29 presents the register; Table 38-29 provides its field descriptions.

AP_BASE+044 (ONCE_DATA)
Wait State: 0

Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATA[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-29. OnCE Data Register (ONCE_DATA)

Table 38-29. ONCE_DATA Field Descriptions

Field	Description
31–0 DATA	Data register of the OnCE JTAG controller. See Section 38.17.4, “OnCE and Real-Time Debug,” for information on this register.

38.11.3.18 OnCE Instruction Register (ONCE_INSTR)

Figure 38-30 presents the register; Table 38-30 provides its field descriptions.

AP_BASE+048 (ONCE_INSTR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INSTR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-30. OnCE Instruction Register (ONCE_INSTR)

Table 38-30. ONCE_INSTR Field Descriptions

Field	Description
31–16	Reserved
15–0 INSTR	Instruction register of the OnCE JTAG controller. See Section 38.17.4, “OnCE and Real-Time Debug,” for information on this register.

38.11.3.19 OnCE Status Register (ONCE_STAT)

Figure 38-31 presents the register; Table 38-31 provides its field descriptions.

AP_BASE+04C (ONCE_STAT) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-31. OnCE Status Register (ONCE_STAT)

Table 38-31. ONCE_STAT Field Descriptions

Field	Description
31–16	Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. Its states are as follows: • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the RISC engine idle state). The “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel: They are used when entering debug mode after reset. The reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the AP peripheral interface. 0 The JTAG interface controls the OnCE. 1 The AP peripheral interface controls the OnCE.

Table 38-31. ONCE_STAT Field Descriptions (continued)

Field	Description
6–3	Reserved
2–0 ECDR	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one of the EDR bits is set (the meaning of the encoding is given below). The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addra_cond, addrb_cond, and data_cond conditions. The value of those fields is given by the EDR bits. EDR[0] 1 matched addra_cond EDR[1] 1 matched addrb_cond EDR[2] 1 matched data_cond

38.11.3.20 OnCE Command Register (ONCE_CMD)

Figure 38-32 presents the register; Table 38-32 provides its field descriptions.

AP_BASE+050 (ONCE_CMD) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	CMD[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-32. OnCE Command Register (ONCE_CMD)

Table 38-32. ONCE_CMD Field Descriptions

Field	Description
31–4	Reserved
3–0 CMD	Writing to this register will cause the OnCE to execute the command that is written. When needed, the ONCE_DATA and ONCE_INSTR registers should be loaded with the correct value before writing the command to that register. For a list of the OnCE commands and their usage, see Section 38.17.4, “OnCE and Real-Time Debug.” 0 `rstatus` 1 `dmov` 2 `exec_once` 3 `run_core` 4 `exec_core` 5 `debug_rqst` 6 `rbuffer` 7–15 *reserved*

38.11.3.21 Illegal Instruction Trap Address (ILLINSTADDR)

Figure 38-33 presents the register; Table 38-33 provides its field descriptions.

AP_BASE+058 (ILLINSTADDR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	ILLINSTADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 38-33. Illegal Instruction Trap Address (ILLINSTADDR)

Table 38-33. ILLINSTADDR Field Descriptions

Field	Description
31–14	Reserved
13–0 ILLINSTADDR	The Illegal Instruction Trap Address is the address where the SDMA jumps when an illegal instruction is executed. It is 0x0001 after reset. The value of ILLINSTADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

38.11.3.22 Channel 0 Boot Address (CHN0ADDR)

Figure 38-34 presents the register; Table 38-34 provides its field descriptions.

AP_BASE+05C (CHN0ADDR) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SMSZ	CHN0ADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

Figure 38-34. Channel 0 Boot Address (CHN0ADDR) Register

Table 38-34. CHN0ADDR Register Field Descriptions

Field	Description
31–15	Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context The value of SMSZ cannot be changed if the LOCK bit in the SDMA_LOCK register is set.
13–0 CHN0ADDR	This 14-bit register is used by the boot code of the SDMA. After reset, it points to the standard boot routine in ROM (channel 0 routine). By changing this address, you can perform a boot sequence with your own routine. The very first instructions of the boot code fetch the contents of this register (it is also mapped in the SDMA memory space) and jump to the given address. The reset value is 0x0050 (decimal 80). The value of CHN0ADDR cannot be changed if the LOCK bit in the SDMA_LOCK register is set.

38.11.3.23 DMA Requests (EVT_MIRROR)

Figure 38-35 presents the register; Table 38-35 provides its field descriptions.

AP_BASE+060 (EVT_MIRROR) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EVENTS[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-35. DMA Requests (EVT_MIRROR)

Table 38-35. EVT_MIRROR Field Descriptions

Field	Description
31–0 EVENTS	This register reflects the DMA requests received by the SDMA for events 31-0. The AP and the SDMA have a read-only access. There is one bit associated with each of 32 DMA request events. This information may be useful during debug of the modules that generate the DMA requests. The EVT_MIRROR register is cleared following read access. 0 DMA request event not pending 1 DMA request event pending

38.11.3.24 DMA Requests 2 (EVT_MIRROR2)

Figure 38-36 presents the register; Table 38-36 provides its field descriptions.

AP_BASE+064 (EVT_MIRROR2) Access: User Read-Only
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[47:32]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-36. DMA Requests (EVT_MIRROR2)

Table 38-36. EVT_MIRROR2 Field Descriptions

Field	Description
31–15	Reserved
15–0 EVENTS[47:32]	This register reflects the DMA requests received by the SDMA for events 47-32. The AP and the SDMA have a read-only access. There is one bit associated with each of DMA request events. This information may be useful during debug of the modules that generate the DMA requests. The EVT_MIRROR2 register is cleared following read access. 0 - DMA request event not pending 1- DMA request event pending

38.11.3.25 Cross-Trigger Events Configuration Register (1) and (2) (XTRIG_CONF1 and XTRIG_CONF2)

Figure 38-37 presents the XTRIG_CONF1 register; Table 38-37 provides its field descriptions.

AP_BASE+070 (XTRIG_CONF1) Access: User Read/Write
Wait State: 0

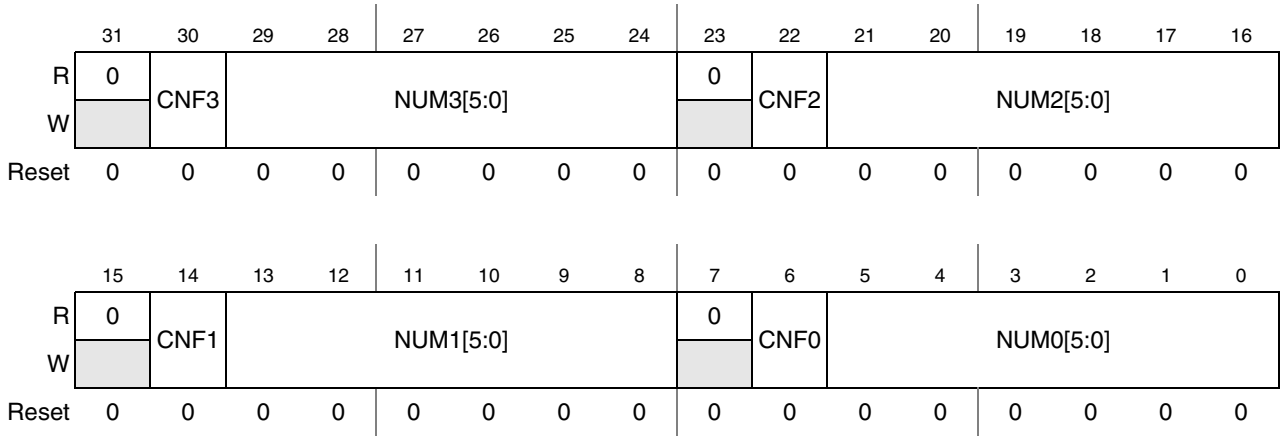


Figure 38-37. Cross-Trigger Events Configuration Register (1) (XTRIG_CONF1)

Table 38-37. XTRIG_CONF1 Field Descriptions

Field	Description
31	Reserved
30 CNF3	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by the reception of a DMA request or by the starting of a channel script execution. 0 channel 1 DMA request
29–24 NUM3[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23	Reserved
22 CNF2	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM2[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
15	Reserved
14 CNF1	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
13–8 NUM1[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

Table 38-37. XTRIG_CONF1 Field Descriptions (continued)

Field	Description
7	Reserved
6 CNF0	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
5–0 NUM0[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

Figure 38-38 presents the XTRIG_CONF2 register; Table 38-38 provides its field descriptions.

AP_BASE+074 (XTRIG_CONF2) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	CNF7	NUM7[5:0]						0	CNF6	NUM6[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CNF5	NUM5[5:0]						0	CNF4	NUM4[5:0]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-38. Cross-Trigger Events Configuration Register (2) (XTRIG_CONF2)

Table 38-38. XTRIG_CONF2 Field Descriptions

Field	Description
31	Reserved
30 CNF7	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
29–24 NUM7[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
23	Reserved
22 CNF6	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
21–16 NUM6[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

Table 38-38. XTRIG_CONF2 Field Descriptions (continued)

Field	Description
15	Reserved
14 CNF5	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
13–8 NUM5[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.
7	Reserved
6 CNF4	Configuration of the SDMA event line number *i* that is connected to the cross-trigger. It determines whether the event line pulse is generated by receiving a DMA request or by starting a channel script execution. 0 channel 1 DMA request
5–0 NUM4[5:0]	Contains the number of the DMA request or channel that triggers the pulse on the cross-trigger event line number *i*.

38.11.3.26 Channel Priority Registers (CHNPRIn)

Figure 38-39 presents the register; Table 38-39 provides its field descriptions.

AP_BASE+100+n*4 (CHNPRIn[1]) Access: User Read/Write
Wait State: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHNPRIn[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-39. Channel Priority Registers (CHNPRIn)

1 for n = 1 to 31

Table 38-39. CHNPRIn Field Descriptions

Field	Description
31–3	Reserved
2–0 CHNPRIn	This contains the priority of channel number *n*. Useful values are between 1 and 7; 0 is reserved by the SDMA hardware to determine when there is no pending channel. Reset value is 0, which prevents the channels from starting.

38.11.3.27 Channel Enable RAM (CHNENBLn)

Figure 38-40 presents the register; Table 38-40 provides its field descriptions.

AP_BASE+200+n*4 (CHNENBLn[1])
Wait State: 1

Access: User Read/Write

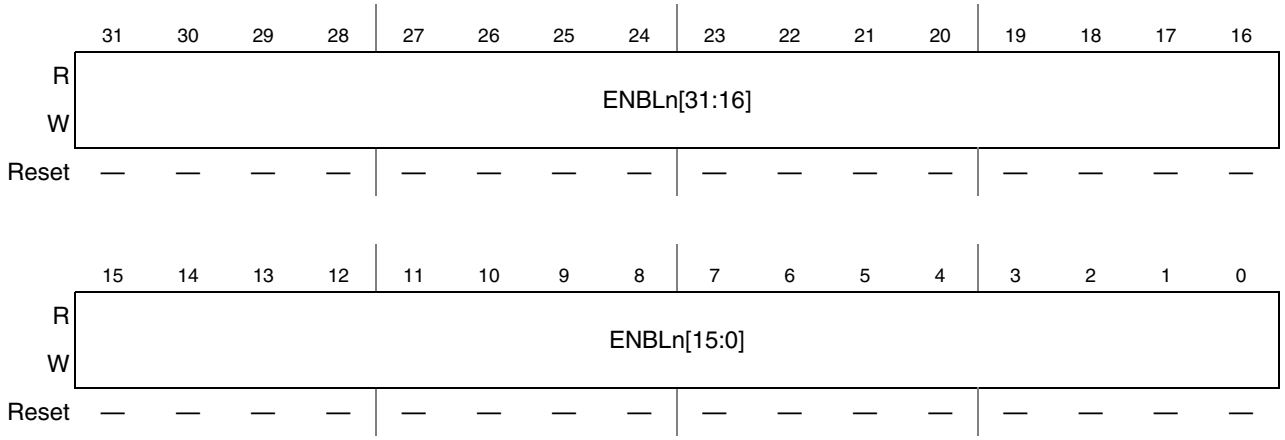


Figure 38-40. Channel Enable RAM (CHNENBLn) Register

[1] for n = 0 to 47

Table 38-40. CHNENBLn Field Descriptions

Field	Description
31–0 ENBLn	This 32-bit value selects the channels that are triggered by the DMA request number *n*. If ENBLn[i] is set to 1, bit EP[i] will be set when the DMA request *n* is received. These 48 32-bit registers are physically located in a RAM, with no known reset value. It is thus essential for the AP to program them before any DMA request is triggered to the SDMA, otherwise an unpredictable combination of channels may be started.

38.12 SDMA Programming Model

The following section describes the programming model for the SDMA RISC engine, including its processor, memory, and internal control registers.

All addresses are related to the internal SDMA memory map, which is completely different from the AP memory maps. The AP processor has no access to any hardware resource described, except when those resources are described in Section 38.11, “AP Memory Map and Control Register Definitions.”

38.12.1 State and Registers Per Channel

The SDMA can be seen as a set of 32 identical devices that are able to perform one data transfer channel each. Only one channel can work at a time, but every channel state is available at any time. This chapter lists the components of every channel state.

38.12.2 General Purpose Registers

Each channel has eight general purpose registers of 32 bits for use by scripts. General register 0 has a dedicated function for the `loop` instruction, but otherwise can be used for any purpose.

38.12.3 Functional Unit State

Each channel context has some state that is part of the functional units. The specific allocation of this state is part of the functional unit definition that is described in Section 38.17.1, “Burst DMA Unit,” Section 38.17.2, “Peripheral DMA Unit.” This state must be saved/restored on context switches.

38.12.3.1 Program Counter Register (PC)

The PC is 14 bits. Since instructions are 16 bits in width and all memory in the SDMA is 32 bits in width, the low order bit of the PC selects which half of the 32-bit word contains the current instruction. A low order bit of zero selects the most significant half of the word.[1]

38.12.3.2 Flags

Each channel has the following four flags:

- The T bit reflects the status of some arithmetic and test instructions. It is set when the result of an addition or a subtraction is zero and cleared otherwise. It is also the copy of the tested bits. Finally, it can also be set when the loop counter (GReg0) reaches zero. When the last instruction of the hardware loop is an operation that can modify the T flag, its effect on T is discarded and replaced by the GReg0 status.
- Two additional bits, SF and DF, are used to indicate error conditions resulting from loading data sources and storing to destinations, respectively. Access errors set these bits, and successful transactions clear them. They can also be cleared by specific instructions (`CLRF` and `loop`). The source fault (SF) is updated by the loads LD and LDF; the destination fault (DF) is updated by the stores ST and STF.

  Access errors are caused by several conditions including writing to the ROM, writing to a read-only memory mapped register, accessing an unmapped address, or any transfer error received by a peripheral when it is accessed.

  The SF and DF flags have a major impact on the behavior of the hardware loop: If SF or DF is set when starting a hardware loop and it is not masked by the `loop` instruction, the loop body will not be executed. Inside the loop body, if a load or store sets the corresponding SF or DF flag, the loop exits immediately. Testing the status of the T flag at the end of the loop (as well as testing both SF and DF) tells if the loop exited abnormally as any anticipated exit prevents GReg0 from reaching the zero value and thus setting the T flag. This is also valid if the fault occurs at the last instruction of the last loop.
- The last flag is the loop mode flag, LM, which is composed of two bits. The most significant bit indicates when the processor is currently operating in loop mode. It is set by the `loop` instruction and is cleared after execution of the last instruction of the last loop. The least significant bit is set when the program counter points to the last instruction of a loop on the last path. It is used for a channel that is restored with this configuration to know that the next program counter is EPC. As with the dynamic context switch Greg0, which indicates when the program must get out of the loop, it can be restored only on the last instruction of the loop. This, however, is too late to fetch the next instruction after the loop.

1. For example, big-endian.

38.12.3.3 Return Program Counter (RPC)

The RPC is 14 bits. It is set by the jump to the subroutine instructions and used by the return from the subroutine instructions. Instructions are available to transfer its contents to and from a general register.

38.12.3.4 Loop Mode Start Program Counter (SPC)

The SPC is 14 bits. It is set by the `loop` instruction to the location immediately following it.

38.12.3.5 Loop Mode End Program Counter (EPC)

The EPC is 14 bits. It is set by the `loop` instruction to the location of the next instruction after the loop.

38.12.4 Context Switching

Each channel has a separate context consisting of the eight general purpose registers and additional registers representing the state of the functional units. The active registers and functional units contain the context of the active channel. The context of inactive channels is stored in SDMA RAM, which is part of the SDMA address space.

In a function of the selected context switching mode (Section 38.4.4, “Context Switching”), modified registers by the program can be saved in the channel RAM space while the program is going on. In every cycle, a write access to the RAM is possible.

On a `done` or `yield(ge)` instruction, SDMA goes into “real” context switching. In one of the dynamic modes, modified registers not previously saved, as well as the PC-Loop registers, are stored into the context area of the channel that will be closed. The new PC-Loop registers are loaded from the context area of the new channel. All other registers are restored while the program is executed, giving priority to registers used by the decoded instruction. Therefore, in the best case, only the PC and Loop registers should be saved and restored during this context-switching phase, which only requires five SDMA cycles.

In static mode, the context switch stores all registers in the old channel RAM space, and restores all registers from the new channel RAM space. It requires 26 SDMA cycles.

The address of the context memory for channel *i* is *CONTEXT_BASE* + 24**i* or *CONTEXT_BASE* + *32***i* where CONTEXT_BASE equals 0x0800. Table 38-41 presents the layout of a channel context in memory:

Table 38-41. Layout of a Channel Context in Memory for SDMA

OFFSET	31	30	29–16	15	14	13–0
0	SF	—	RPC	T	—	PC
1	LM		EPC	DF	—	SPC
2	GR0					
3	GR1					
4	GR2					
5	GR3					
6	GR4					

Table 38-41. Layout of a Channel Context in Memory for SDMA (continued)

OFFSET	31	30	29–16	15	14	13–0
7	GR5					
8	GR6					
9	GR7					
10	MDA (burst DMA)					
11	MSA (burst DMA)					
12	MS (burst DMA)					
13	MD (burst DMA)					
14	PDA (peripheral DMA)					
15	PSA (peripheral DMA)					
16	PS (peripheral DMA)					
17	PD (peripheral DMA)					
18	CA (CRC)					
19	CS (CRC)					
20	Reserved[1]					
21	Reserved[1]					
22	Reserved[1]					
23	Reserved[1]					
24	Scratch RAM (optional)					
25	Scratch RAM (optional)					
26	Scratch RAM (optional)					
27	Scratch RAM (optional)					
28	Scratch RAM (optional)					
29	Scratch RAM (optional)					
30	Scratch RAM (optional)					
31	Scratch RAM (optional)					

38.12.5 Address Space

The SDMA has four internal buses, as follows:

- The Instruction bus reads instructions from the memory. Its address map is described in Section 38.12.5.1, “Instruction Memory Map.”
- The Data bus (DMBUS) accesses the same memories as those visible on the Instruction bus, some memory-mapped registers (scheduler status and OnCE registers), and up to 14 peripherals. Its address map is described in Section 38.12.5.2, “Data Memory Map.”

- The Functional Units bus (FUBUS) accesses the burst DMA, peripheral DMA, and CRC internal registers. The addressing mechanism is further detailed in Section 38.17, “Functional Units Programming Model.”
- The Context Switch bus reads/writes registers into context-switch RAM space. It is a 64-bit bus dedicated for accessing this RAM space for updating the context of the running channel. While the program is going on, this bus has the lowest priority compared to the Instruction and Data buses, except for restoring a register needed for the decoded instruction to be executed. On the save part of a context switch (when the PCU is in its slave state), this is the only one used. On the restore part, the Instruction bus has the priority to read the next instruction at the restored PC and otherwise the Context Switch bus is used. It is not possible to control the actual data transfers that occur on this bus.

38.12.5.1 Instruction Memory Map

The instruction memory map is based on a 14-bit address bus and a 16-bit data (instruction) bus. Each address corresponds to a 16-bit data location. Instructions are fetched from either program ROM or program RAM. An SDMA script is able to change the contents of the program RAM, which is also visible from the data bus.

The first two instruction locations (at 0 and 1) are special. Location 0 is where the PC is set on reset. Location 1 is where the PC is set upon the execution of an illegal instruction. It is expected that both of these locations will contain a `jmp` to handle routines.

Table 38-42. SDMA Instruction Memory Space

Device	SDMA Address (Hex)	Base Address Label	Module Name	WS	Description
ROM	0x0000 ↓ 0x07FF	SDMA_IBUS_ROM_ADDR	—	0	4-Kbyte internal ROM with boot code and standard routines.
RAM	0x1000 ↓ 0x1FFF	SDMA_IBUS_RAM_ADDR	—	0	8-Kbyte internal RAM with channels context and user data/routines.

38.12.5.2 Data Memory Map

All of the data accessible to SDMA scripts make up the data memory space of the SDMA. This address space has several components:

- ROM (also visible on the Instruction bus)
- RAM (also visible on the Instruction bus)
- Shared Peripherals Registers
- SDMA Internal Registers (scheduler, OnCE, and registers that are also accessible by the AP)

SDMA scripts can read and write to the context RAM, data RAM, shared peripheral registers, and internal registers.

The address range is 16 bits and the data width is 32 bits. Each address corresponds to a 32-bit data word. When accessing peripheral registers (USB and so on), the data width may be different. The exact address map for the peripherals depends on the project (as presented in each respective chapter).

The SDMA can perform only 32-bit access to shared peripheral registers. For this device, shared peripherals registers are aliased as if byte addressed. This is a consequence of connections through the SPBA shared peripheral bus outside of the SDMA. The result is that although address space 4 Kwords (16KB) is allocated for each peripheral, only the first 4 Kbytes of the peripheral's register space can accessed. For example, the shared peripheral register at address 0x3000 is mapped also to addresses 0x3001, 0x3002, 0x3003. A read or write access to any of any of these 4 addresses will respond as if the access was to address 0x3000.

Data access is performed with *ld* and *st* instructions that take the address from a general purpose register in the core (GReg*n*). The mapping between the general purpose register contents and the address bus is given in Table 38-43.

Table 38-43. GReg*n* to DMBUS Address Mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
address															

Grayed bits are simply discarded but they must be cleared to ensure forward-script compatibility.

- sz (bit 31) indicates the peripheral data width: 0 is used for a 32-bit peripheral and 1 is used for a 16-bit peripheral.
- address (bits 15 down to 0) is the address of the accessed resource (internal memory, internal register, or shared peripheral).

Table 38-44. SDMA Data Memory Space

Device	SDMA Address (Hex)	Size	Description
ROM	0x0000 → 0x03FF	4 Kbytes	4-Kbyte internal ROM with boot code and standard routines
Reserved	0x0400 → 0x07FF	4 Kbytes	4 Kbytes Reserved
RAM	0x0800 → 0x0FFF	8 Kbytes	8-Kbyte internal RAM with channels contexts and user data/routines
per1	0x1000 → 0x1FFF	16 Kbytes	*peripheral 1* memory space (4 Kbytes peripheral's address space)
per2	0x2000 → 0x2FFF	16 Kbytes	*peripheral 2* memory space (4 Kbytes peripheral's address space)
per3	0x3000 → 0x3FFF	16 Kbytes	*peripheral 3* memory space (4 Kbytes peripheral's address space)
per4	0x4000 → 0x4FFF	16 Kbytes	*peripheral 4* memory space (4 Kbytes peripheral's address space)
per5	0x5000 → 0x5FFF	16 Kbytes	*peripheral 5* memory space (4 Kbytes peripheral's address space)
per6	0x6000 → 0x6FFF	16 Kbytes	*peripheral 6* memory space (4 Kbytes peripheral's address space)
Registers	0x7000 → 0x7FFF	16 Kbytes	Memory mapped registers
per7	0x8000 → 0x8FFF	16 Kbytes	*peripheral 7* memory space (4 Kbytes peripheral's address space)

Table 38-44. SDMA Data Memory Space (continued)

Device	SDMA Address (Hex)	Size	Description
per8	0x9000 → 0x9FFF	16 Kbytes	*peripheral 8* memory space (4 Kbytes peripheral's address space)
per9	0xA000 → 0xAFFF	16 Kbytes	*peripheral 9* memory space (4 Kbytes peripheral's address space)
per10	0xB000 → 0xBFFF	16 Kbytes	*peripheral 10* memory space (4 Kbytes peripheral's address space)
per11	0xC000 → 0xCFFF	16 Kbytes	*peripheral 11* memory space (4 Kbytes peripheral's address space)
per12	0xD000 → 0xDFFF	16 Kbytes	*peripheral 12* memory space (4 Kbytes peripheral's address space)
per13	0xE000 → 0xEFFF	16 Kbytes	*peripheral 13* memory space (4 Kbytes peripheral's address space)
per14	0xF000 → 0xFFFF	16 Kbytes	*peripheral 14* memory space (4 Kbytes peripheral's address space)

38.13 SDMA Internal (Core) Memory Map and Internal Register Definitions

The actual SDMA memory mapped registers are summarized in the following sections; for peripherals' memory maps, see the respective chapters.

38.13.1 SDMA Internal (Core) Registers Memory Map

Table 38-45. SDMA Internal Registers Memory Map

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x7000 (MC0PTR)	AP (MCU) Channel 0 Pointer	R	0x0000_0000	38.13.3.1/38-83
0x7002 (CCPTR)	Current Channel Pointer	R	0x0000_0000	38.13.3.2/38-83
0x7003 (CCR)	Current Channel Register	R	0x0000_0000	38.13.3.3/38-84
0x7004 (NCR)	Highest Pending Channel Register	R	0x0000_0000	38.13.3.4/38-85
0x7005 (EVENTS)	External DMA Requests Mirror	R	0x0000_0000	38.13.3.5/38-85
0x7006 (CCPRI)	Current Channel Priority	R	0x0000_0000	38.13.3.6/38-87
0x7007 (NCPRI)	Next Channel Priority	R	0x0000_0000	38.13.3.7/38-87
0x7009 (ECOUNT)	OnCE Event Cell Counter	R/W	0x0000_0000	38.13.3.8/38-88
0x700A (ECTL)	OnCE Event Cell Control Register	R/W	0x0000_0000	38.13.3.9/38-89
0x700B (EAA)	OnCE Event Address Register A	R/W	0x0000_0000	38.13.3.10/38-91
0x700C (EAB)	OnCE Event Cell Address Register B	R/W	0x0000_0000	38.13.3.11/38-91
0x700D (EAM)	OnCE Event Cell Address Mask	R/W	0x0000_0000	38.13.3.12/38-92
0x700E (ED)	OnCE Event Cell Data Register	R/W	0x0000_0000	38.13.3.13/38-93
0x700F (EDM)	OnCE Event Cell Data Mask	R/W	0x0000_0000	38.13.3.14/38-93
0x7018 (RTB)	OnCE Real-Time Buffer	R/W	0x0000_0000	38.13.3.15/38-94

Table 38-45. SDMA Internal Registers Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x7019 (TB)	OnCE Trace Buffer	R	0x0000_0000	38.13.3.16/38-95
0x701A (OSTAT)	OnCE Status	R	0x0000_0000	38.13.3.17/38-96
0x701C (MCHN0ADDR)	Channel 0 Boot Address	R	0x0000_0000	38.13.3.18/38-98
0x701D (ENDIANESS)	ENDIAN Mode Status Register	R	0x0000_0000	38.13.3.19/38-99
0x701E (LOCK)	Lock Status Register	R	0x0000_0000	38.13.3.20/38-99
0x701F (EVENTS2)	External DMA Requests Mirror #2	R	0x0000_0000	38.13.3.21/38-100
0x7020 (HE)	AP Enable Register	R	0x0000_0000	38.13.3.22/38-100

38.13.2 Register Summary

The following definitions serve as a key for the SDMA internal register summary.

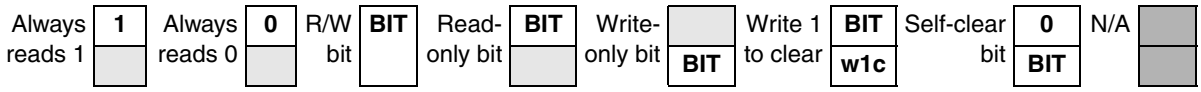


Figure 38-41. Key to Register Fields

Table 38-46 provides a key for register figures.

Table 38-46. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Note: for n = 0 to 31

Table 38-47 presents a summary of the SDMA internal (core) registers.

Table 38-47. SDMA Internal Registers Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7000 (MC0PTR)	R	MC0PTR[31:16]															
	W																
	R	MC0PTR[15:0]															
	W																
0x7001 RESERVED	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
0x7002 (CCPTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CCPTR[15:0]															
	W																
0x7003 (CCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	CCR[4:0]				
	W																
0x7004 (NCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NCR[4:0]				
	W																
0x7005 (EVENTS)	R	EVENTS[31:16]															
	W																
	R	EVENTS[15:0]															
	W																
0x7006 (CCPRI)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CCPRI[2:0]		
	W																

Table 38-47. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7007 (NCPRI)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	NCPRI[2:0]		
	W																
0x7009 (ECOUNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ECOUNT[15:0]															
	W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
0x700A (ECTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	EN	CNT	ECTC[1:0]		DTC[1:0]		ATC[1:0]		ABTC[1:0]		AATC[1:0]		ATS[1:0]	
	W																
0x700B (EAA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAA[15:0]															
	W																
0x700C (EAB)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAB[15:0]															
	W																
0x700D (EAM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EAM[15:0]															
	W																
0x700E (ED)	R	ED[31:16]															
	W																
	R	ED[15:0]															
	W																
0x700F (EDM)	R	EDM[31:16]															
	W																
	R	EDM[15:0]															
	W																

Table 38-47. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7018 (RTB)	R	RTB[31:16]															
	W																
	R	RTB[15:0]															
	W																
0x7019 (TB)	R	0	0	0	TBF	TADDR[13:2]											
	W																
	R	TADDR[1:0]		CHFADDR[13:0]													
	W																
0x701A (OSTAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
	W																

Table 38-47. SDMA Internal Registers Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x701C (MCHN0ADDR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	SMSZ	CHN0ADDR[13:0]													
	W																
0x701D (ENDIANESS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AP-END
	W																
0x701E (LOCK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LOCK
	W																
0x701F (EVENTS2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EVENTS[47:32]															
	W																
0x7020 (HE)	R	HE[31:16]															
	W																
	R	HE[15:0]															
	W																

38.13.3 SDMA Core Register Descriptions

The SDMA core has access to several memory mapped registers through its internal data bus. They are described in the following sections.

38.13.3.1 AP (MCU) Channel 0 Pointer (MC0PTR)

The following table shows the register; Table 38-48 provides its field descriptions.

0x7000 (MC0PTR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MC0PTR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MC0PTR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-42. AP (MCU) Channel 0 Pointer (MC0PTR) Register

Table 38-48. MC0PTR Field Descriptions

Field	Description
31–0 MC0PTR	Contains the address—in the AP memory space—of the initial SDMA context and scripts that are loaded by the SDMA boot script running on channel 0.

38.13.3.2 Current Channel Pointer (CCPTR)

Figure 38-43 shows the register; Table 38-49 provides its field descriptions.

0x7002 (CCPTR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CCPTR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-43. Current Channel Pointer (CCPTR) Register

Table 38-49. CCPTR Field Descriptions

Field	Description
31–16	Reserved
15–0 CCPTR	Contains the start address of the context data for the current channel: Its value is *CONTEXT_BASE* + 24**CCR* or *CONTEXT_BASE* + 32**CCR* where CONTEXT_BASE = 0x0800. The value 24 or 32 is selected according to the programmed channel scratch RAM size in the register shown in Section 38.11.3.22, “Channel 0 Boot Address (CHN0ADDR).”

38.13.3.3 Current Channel Register (CCR)

Figure 38-44 shows the register; Table 38-50 provides its field descriptions.

0x7003 (CCR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	CCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-44. Current Channel Register (CCR)

Table 38-50. CCR Field Descriptions

Field	Description
31–5	Reserved
4–0 CCR	Contains the number of the current running channel whose context is installed. In the case that the SDMA has finished running the channel and has entered sleep state, CCR will indicate the previous running channel. The PST bits in the OSTAT register indicate when the SDMA is in sleep state.

38.13.3.4 Highest Pending Channel Register (NCR)

Figure 38-45 shows the register; Table 38-51 provides its field descriptions.

0x7004 (NCR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	NCR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-45. Highest Pending Channel Register (NCR)

Table 38-51. NCR Field Descriptions

Field	Description
31–5	Reserved
4–0 NCR	Contains the number of the pending channel that the scheduler has selected to run next.

38.13.3.5 External DMA Requests Mirror (EVENTS)

Figure 38-46 shows the register; Table 38-52 provides its field descriptions.

0x7005 (EVENTS) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EVENTS[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-46. External DMA Requests Mirror (EVENTS)

Table 38-52. EVENTS Field Descriptions

Field	Description
31–0 EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 0-31. The EVENTS2 register displays events 32-47.

NOTE

This register is very useful in the case of DMA requests that are active when a peripheral FIFO level is above the programmed watermark. The activation of the DMA request (rising edge) is detected by the SDMA logic and it can enable one or several channels. One of the channels accesses the peripheral and reads or writes a number of data that matches the watermark level (for example, if the watermark is four words, the channel reads or writes four words).

If the channel is effectively executed long after the DMA request was received, reading or writing the watermark number of data may not be sufficient to reset the DMA request (for example, if the FIFO watermark is four and at the channel execution it already contains nine pieces of data). This means no new rising edge may be detected by the SDMA, although there still remains transfers to perform. Therefore, if the channel were terminated at that time, it would not be restarted, causing potential overrun or underrun of the peripheral.

The proposed mechanism is for the channel to check this register after it has performed the "watermark" number of accesses to the peripheral. If the bit for the DMA request that triggers this channel is set, it means there is still another watermark number of data to transfer. This goes on until the bit is cleared. The same script can be used for multiple channels that require this behavior. The script can determine its channel number from the CCR register and infer the corresponding DMA request bit to check. It needs a reference table that is coherent with the request-channel matrix that the AP programmed.

38.13.3.6 Current Channel Priority (CCPRI)

Figure 38-47 shows the register; Table 38-53 provides its field descriptions.

0x7006 (CCPRI) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CCPRI[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-47. Current Channel Priority (CCPRI) Register

Table 38-53. CCPRI Field Descriptions

Field	Description
31–3	Reserved
2–0 CCPRI	Contains the 3-bit priority of the channel whose context is installed. It is 0 when no channel is running. 0 no running channel 1–7 current channel priority

38.13.3.7 Next Channel Priority (NCPRI)

Figure 38-48 shows the register; Table 38-54 provides its field descriptions.

0x7007 (NCPRI) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	NCPRI[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-48. Next Channel Priority (NCPRI) Register

Table 38-54. NCPRI Field Descriptions

Field	Description
31–3	Reserved
2–0 NCPRI	Contains the 3-bit priority of the channel the scheduler has selected to run next. It is 0 when no other channel is pending.

38.13.3.8 OnCE Event Cell Counter (ECOUNT)

Figure 38-49 shows the register; Table 38-55 provides its field descriptions.

0x7009 (ECOUNT) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECOUNT[15:0]															
W	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm	rwm
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-49. OnCE Event Cell Counter (ECOUNT) Register

Table 38-55. ECOUNT Field Descriptions

Field	Description
31–16	Reserved
15–0 ECOUNT	The event cell counter contains the number of times minus one that an event detection must occur before generating a debug request. • This register should be written before any attempt to use the event detection counter during an event detection process. • The counter is cleared on a JTAG reset.

38.13.3.9 OnCE Event Cell Control Register (ECTL)

Figure 38-50 shows the register; Table 38-56 provides its field descriptions.

0x700A (ECTL) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	EN	CNT	ECTC[1:0]		DTC[1:0]		ATC[1:0]		ABTC[1:0]		AATC[1:0]		ATS[1:0]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-50. OnCE Event Cell Control Register (ECTL)

Table 38-56. ECTL Field Descriptions

Field	Description
31–14	Reserved
13 EN	Event Cell Enable. If the EN bit is set, the event cell is allowed to generate debug requests (the cell is awakened). If it is cleared, the event detection unit is disabled and no hardware breakpoint is generated, but matching conditions are still reflected on the emulation pin. 0 Cell is disabled. 1 Cell is enabled.
12 CNT	Event Counter Enable. The event counter enable bit determines if the cell counter is used during the event detection. In order to use the event counter during an event detection process, the event cell counter register should be loaded with a value equal to the number of times minus one that an event occurs before a debug request is sent. After every event detection, the counter is decreased. When the counter reaches the value 0, the event detection cell sends a debug request to the core. The event counter register should be written and the EN bit should be set before each new event detection process uses the event counter. 0 Counter is disabled. 1 Counter is enabled.
11–10 ECTC[1:0]	The event cell trigger condition bits select the combination of address and data matching conditions that generate the final address/data condition. During program execution, if this event cell trigger condition goes to 1, a debug request is sent to the SDMA. The EN bit must be set to enable the debug request generation. 00 address ONLY 01 data ONLY 10 address AND data 11 address OR data

Table 38-56. ECTL Field Descriptions (continued)

Field	Description
9–8 DTC[1:0]	The data trigger condition bits define when data is considered matching after comparison with the data register of the event detection unit. The operations are performed on unsigned values. 00 equal 01 not equal 10 greater than 11 less than
7–6 ATC[1:0]	The address trigger condition bits select how the two address conditions (addressA and addressB) are combined to define the global address matching condition. The supported combinations are described, as follows. 00 addressA ONLY 01 addrA AND addrB 10 addrA OR addrB 11 reserved
5–4 ABTC[1:0]	The Address B Trigger Condition (ABTC) controls the operations performed by address comparator B. All operations are performed on unsigned values. This comparator B outputs the addressB condition. 00 equal 01 not equal 10 greater than 11 less than
3–2 AATC[1:0]	The Address A Trigger Condition (AATC) controls the operations performed by address comparator A. All operations are performed on unsigned values. This comparator A outputs the addressA condition. 00 equal 01 not equal 10 greater than 11 less than
1–0 ATS[1:0]	The access type select bits define the memory access type required on the SDMA memory bus. 00 read ONLY 01 write ONLY 10 read or write 11 —

38.13.3.10 OnCE Event Address Register A (EAA)

Figure 38-51 shows the register; Table 38-57 provides its field descriptions.

0x700B (EAA) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EAA[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-51. OnCE Event Address Register A (EAA)

Table 38-57. EAA Field Descriptions

Field	Description
31–16	Reserved
15–0 EAA	Event Cell Address Register A computes an address A condition. It is cleared on a JTAG reset.

38.13.3.11 OnCE Event Cell Address Register B (EAB)

Figure 38-52 shows the register; Table 38-58 provides its field descriptions.

0x700C (EAB) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EAB[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-52. OnCE Event Cell Address Register B

Table 38-58. EAB Field Descriptions

Field	Description
31–16	Reserved
15–0 EAB	Event Cell Address Register B computes an address B condition. It is cleared on a JTAG reset.

38.13.3.12 OnCE Event Cell Address Mask (EAM)

Figure 38-53 shows the register; Table 38-59 provides its field descriptions.

0x700D (EAM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EAM[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-53. OnCE Event Cell Address Mask (EAM)

Table 38-59. EAM Field Descriptions

Field	Description
31–16	Reserved
15–0 EAM	The Event Cell Address Mask contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before performing the address comparison. **Note:** There is a common address mask value for both address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The register is cleared on a JTAG reset.

38.13.3.13 OnCE Event Cell Data Register (ED)

Figure 38-54 shows the register; Table 38-60 provides its field descriptions.

0x700E (ED) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ED[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ED[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-54. OnCE Event Cell Data (ED) Register

Table 38-60. ED Field Descriptions

Field	Description
31–0 ED	The event cell data register contains a user defined data value. This data value is an input for the data comparator which generates the data condition. It is cleared on a JTAG reset.

38.13.3.14 OnCE Event Cell Data Mask (EDM)

Figure 38-55 shows the register; Table 38-61 provides its field descriptions.

0x700F (EDM) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDM[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDM[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-55. OnCE Event Cell Data Mask (EDM) Register

Table 38-61. EDM Field Descriptions

Field	Description
31–0 EDM	The event cell data mask register contains the user-defined data mask value. • This mask is applied to the data value latched from the memory bus before performing the data comparison. • Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. • The data mask is cleared on a JTAG reset.

38.13.3.15 OnCE Real-Time Buffer (RTB)

Figure 38-56 shows the register; Table 38-62 provides its field descriptions.

0x7018 (RTB) Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RTB[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTB[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-56. OnCE Real-Time Buffer (RTB) Register

Table 38-62. RTB Field Descriptions

Field	Description
31–0 RTB	The Real Time Buffer register stores and retrieves run time information without putting the SDMA in debug mode. Writing to that register triggers a pulse on a specific real-time debug pin whose connection depends on the chip implementation. The RTB value can be accessed by the OnCE under AP or JTAG control using the rbuffer command.

38.13.3.16 OnCE Trace Buffer (TB)

Figure 38-57 shows the register; Table 38-63 provides its field descriptions.

0x7019 (TB) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	TBF	TADDR[13:2]											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TADDR[1:0]		CHFADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-57. OnCE Trace Buffer (TB) Register

Table 38-63. TB Field Descriptions

Field	Description
31–29	Reserved
28 TBF	The Trace Buffer Flag is set when the buffer contains the addresses of a valid change of flow. The contents of the buffer should be ignored otherwise. 0 Invalid information 1 Valid information
27–14 TADDR	The target address is the address taken after the execution of the change of flow instruction.
13–0 CHFADDR	The change of flow address is the address where the change of flow is taken when executing a change of flow instruction.

38.13.3.17 OnCE Status (OSTAT)

Figure 38-58 shows the register; Table 38-64 provides its field descriptions.

0x701A (OSTAT) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PST[3:0]				RCV	EDR	ODR	SWB	MST	0	0	0	0	ECDR[2:0]		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-58. OnCE Status (OSTAT) Register

Table 38-64. OSTAT Field Descriptions

Field	Description
31–16	Reserved
15–12 PST[3:0]	The Processor Status bits reflect the state of the SDMA RISC engine. • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel-switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the RISC engine idle state). The “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 will be executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel. They are used when entering debug mode after reset; the reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Save 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change Flow Loop Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Restore
11 RCV	After each write access to the real time buffer (RTB), the RCV bit is set. This bit is cleared after execution of an `rbuffer` command and on a JTAG reset.
10 EDR	This flag is raised when the SDMA has entered debug mode after an external debug request.
9 ODR	This flag is raised when the SDMA has entered debug mode after a OnCE debug request.
8 SWB	This flag is raised when the SDMA has entered debug mode after a software breakpoint.
7 MST	This flag is raised when the OnCE is controlled from the AP peripheral interface. 0 JTAG interface controls the OnCE. 1 AP peripheral interface controls the OnCE.

Table 38-64. OSTAT Field Descriptions (continued)

Field	Description
6–3	Reserved.
2–0 ECDR[2:0]	Event Cell Debug Request. If the debug request comes from the event cell, the reason for entering debug mode is given by the EDR bits. If all three bits of the EDR are reset, then it did not generate any debug request. If the cell did generate a debug request, then at least one EDR bit is set; the meaning of the encoding is as follows: EDR[0] 1 matched addressA condition EDR[1] 1 matched addressB condition EDR[2] 1 matched data condition The encoding of the EDR bits is useful to find out more precisely why the debug request was generated. A debug request from an event cell is generated for a specific combination of the addressA, addressB, and data conditions; the value of those fields is given by the EDR bits.

38.13.3.18 Channel 0 Boot Address (MCHN0ADDR)

Figure 38-59 shows the register; Table 38-65 provides its field descriptions.

0x701C (MCHN0ADDR) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SMSZ	CHN0ADDR[13:0]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-59. Channel 0 Boot Address (MCHN0ADDR) Register

Table 38-65. MCHN0ADDR Field Descriptions

Field	Description
31–15	Reserved
14 SMSZ	The bit 14 (Scratch Memory Size) determines if scratch memory must be available after every channel context. After reset, it is equal to 0, which defines a RAM space of 24 words for each channel. All of this area stores the channel context. By setting this bit, 32 words are reserved for every channel context, which gives eight additional words that can be used by the channel script to store any type of data. Those words are never erased by the context switching mechanism. 0 24 words per context 1 32 words per context
13–0 CHN0ADDR[13:0]	Contains the address of the channel 0 routine programmed by the AP; it is loaded into a general register at the very start of the boot and the SDMA jumps to the address it contains. By default, it points to the standard boot routine in ROM.

38.13.3.19 ENDIAN Mode Status Register (ENDIANESS)

Figure 38-60 shows the register; Table 38-66 provides its field descriptions.

0x701D (ENDIANESS) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AP END
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-60. Endian Mode Status Register (ENDIANESS) Register

Table 38-66. ENDIANESS MODE Field Descriptions

Field	Description
31–1	Reserved
0 AP END	AP END indicates the endian mode of the Peripheral and Burst DMA interfaces. 0 AP is in big-endian mode 1 AP is in little-endian mode

38.13.3.20 Lock Status Register (LOCK)

Figure 38-61 shows the register; Table 38-67 provides its field descriptions.

0x701E (LOCK) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-61. Lock Status (LOCK) Register

Table 38-67. LOCK Field Descriptions

Field	Description
31–1	Reserved
0 LOCK	The LOCK bit reports the value of the LOCK bit in the SDMA_LOCK status register. SDMA software may use this value to determine if certain operations such as loading of new scripts is allowed. 0 - LOCK bit clear 1 - LOCK bit set

38.13.3.21 External DMA Requests Mirror #2 (EVENTS2)

Figure 38-62 shows the register; Table 38-68 provides its field descriptions.

0x701F (EVENTS2) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EVENTS[47:32]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-62. External DMA Requests #2 (EVENTS2) Register

Table 38-68. EVENTS2 Field Descriptions

Field	Description
31–16	Reserved
15-0 EVENTS	Reflects the status of the SDMA's external DMA requests. It is meant to allow any channel to monitor the states of these SDMA inputs. This register displays EVENTS 32-47. The separate EVENTS register displays events 0-31.

38.13.3.22 Host Enable Register (HE)

Figure 38-63 shows the register; Table 38-69 provides its field descriptions.

0x7020 (HE) Access: User Read-Only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HE[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 38-63. Host Enable (HE) Register

Table 38-69. HE Field Descriptions

Field	Description
31-0 HE	Reflects the status of the SDMA's host enable bits which are configured in the AP memory map by the HSTART or STOP_STAT registers.

38.14 SDMA Peripheral Registers

See the respective peripherals' chapters more information.

38.15 SDMA Initialization

The API document MOT-SFS-IAPI-SAS-001 (version 0.04) describes the setup of the SDMA. This section provides a quick description of several initialization procedures.

NOTE

There may be differences with the actual implementation in the API.

38.15.1 Hardware Reset

After reset, the RAM that holds contexts, data, scripts, and the DMA request-channels matrix has unpredictable content. The core registers are all reset to 0, including the PC; the PCU state is *Sleep after Reset*. No channel can be activated because all of the priorities are also reset to 0.

38.15.2 Standard Boot Sequence

The following is the standard boot sequence:

1. Initialize the CONFIG register—detailed in Section 38.11.3.14, "Configuration Register (CONFIG)"—to determine the AP DMA/core clock ratio (1 or 2)
2. Initialize the DMA request-channels matrix (see Section 38.11.3.27, "Channel Enable RAM (CHNENBLn)").

3. Program the channel control registers—Channel Event Override (EVTOVR), Channel BP Override (DSPOVR), Channel BP Override (HOSTOVR), and Channel Event Pending (EVTPEND)—according to the channel allocation.
4. Perform any necessary setup as required by the standard boot script in ROM (this is described in MOT-SFS-IAPI-SAS-001 (version 0.04)).
5. Trigger channel 0 with the Channel Start (HSTART) register, which starts the execution of the ROM script starting at address 0. This boot downloads channel scripts and contexts in RAM.

38.15.3 User-Defined Boot Sequence

The following is a user-defined boot sequence:

1. Initialize the Configuration Register (CONFIG), Channel Enable RAM (CHNENBLn), Channel Event Override (EVTOVR), Channel BP Override (DSPOVR), Channel AP Override (HOSTOVR), and Channel Event Pending (EVTPEND).
2. Use the OnCE (either using its JTAG interface or its AP control registers) to download any code in the SDMA RAM. Section 38.19.5.4, “Accessing the Memory” describes how to write data to the RAM using the OnCE.
3. Use the OnCE instructions to make the PC default value point to the new boot script start address, or rely on the ROM startup script, which first jumps to the address in Channel 0 Boot Address (CHN0ADDR). (This register default address points to the standard boot script.)

38.15.4 Script Loading and Context Initialization

The execution of an SDMA script depends on both the instructions that make up the script and the data context upon which it operates. Both must be initialized before the script is allowed to execute. Each of the 32 channels has a separate data context, but may share scripts and locations in the data RAM.

The AP manages the space in program RAM and data RAM. It also manages the assignment of SDMA channels to the device drivers that need them. Channels are initialized by the AP using the channel 0 boot script. The boot channel downloads any required scripts with their data and the channels’ initial contexts. Every context contains all the initial values of the registers, including the PC. Then the can enable any channel that becomes active and begins fetching and executing instructions from its script.

38.16 Instruction Description

The following sections introduce the instruction of the SDMA.

38.16.1 Scheduling Instructions

The following are scheduling instructions:

- done—The instruction causes certain scheduling or interrupt bits to be set or cleared, which may cause a change in the schedule-ability of the running channel. Then the instruction causes the SDMA to evaluate the current scheduling priorities and to choose the highest priority ready channel. If this channel is not the current channel, a context switch will take place. If there are no

runnable channels, the SDMA will enter the stopped mode. The done 5 has a special usage reserved for debug, as explained in Section 38.16.17, “Debug Instructions.”

- yield—These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater than the current channel priority.
- yieldge—These instructions are special cases of the done instruction. They do not modify the scheduling bits, but allow the highest pending channel (if it exists) to preempt the current channel if the pending channel priority is strictly greater or equal to the current channel priority.
- notify—The notify instruction affects the scheduling bits, but does not cause rescheduling.

38.16.2 Conditional Branch Instructions

The conditional branch instructions of an 8-bit displacement, which is sign-extended and added to the current PC (which points to the next instruction) if the condition is satisfied. Otherwise, control passes to the next sequential instruction.

- BF—Branch if False. The branch is taken if the T bit in the processor status is zero (false).
- BT—Branch if True. The branch is taken if the T bit in the processor status is one (true).
- BSF—Branch if Source Fault. The branch is taken if the SF bit in the processor status is one.
- BDF—Branch if Destination Fault. The branch is taken if the DF bit in the processor status is one.

38.16.3 Unconditional Jump Instructions

There are two varieties of unconditional control transfers: an absolute transfer and a through-register transfer. Absolute transfers have a 14-bit address field that replaces the current PC.

- JMP—Jump. Causes the processor to jump to an absolute address encoded in the instruction itself.
- JSR—Jump to Subroutine. Causes the processor to jump to a subroutine, the address of which is encoded in the instruction itself.
- JMPR—Jump through Register. Causes the processor to jump to an absolute address contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.
- JSRR—Jump to Subroutine through Register. Causes the processor to jump to a subroutine, the address of which is contained in a General register. This instruction is meant to be used when more than one level of subroutines are required.

38.16.4 Subroutine Return Instructions

The following are subroutine return instructions:

- RET—Return from Subroutine. The RET restores the contents of RPC to PC.
- LDRPC—Load from RPC to Register. THe LDRPC instruction is meant to be used when more than one level of subroutines are required. It stores the contents of RPC in any General register.

38.16.5 Loop Instruction

The following is a `loop` instruction:

`LOOP`—Enters Loop Mode. Before entering loop mode, the `loop` instruction can optionally clear the fault flags (SF and/or DF) based on a 2-bit field in the instruction. This feature is linked to the fact that setting SF or DF in loop mode will cause an immediate exit of the loop.

38.16.6 Miscellaneous Instructions

The following are miscellaneous instructions:

- `CLRF`—Clear Fault Flags. This instruction clears any combination of SF and DF.
- `MOV r,s`—This moves data from `GReg[s]` to `GReg[r]`.
- `LDI r,immediate`—This loads `GReg[r]` with a zero-extended immediate value.

38.16.7 Logic Instructions

The following are logic instructions:

- `XOR r,s`—This performs an `exclusive or` between `GReg[r]` and `GReg[s]`, and stores the result in `GReg[r]`.
- `XORI r,immediate`—This performs an `exclusive or` between `GReg[r]` and a zero-extended immediate value, and stores the result in `GReg[r]`.
- `OR r,s`—This performs an `or` between `GReg[r]` and `GReg[s]`, and stores the result in `GReg[r]`.
- `ORI r,immediate`—This performs an `or` between `GReg[r]` and a zero-extended immediate value and, stores the result in `GReg[r]`.
- `ANDN r,s`—This performs an `and` between `GReg[r]` and the `negated GReg[s]`, and stores the result in `GReg[r]`.
- `ANDNI r,immediate`—This performs an `and` between `GReg[r]` and the negated zero-extended immediate value, and stores the result in `GReg[r]`.
- `AND r,s`—This performs an `and` between `GReg[r]` and `GReg[s]`, and stores the result in `GReg[r]`.
- `ANDI r,immediate`—This performs an `and` between `GReg[r]` and a zero-extended immediate value, and stores the result in `GReg[r]`.

38.16.8 Arithmetic Instructions

Arithmetic instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the result is zero, otherwise it is cleared.

- `ADD r,s`—This performs the addition of `GReg[r]` and `GReg[s]`, and stores the result in `GReg[r]`.
- `ADDI r,immediate`—This performs the addition of `GReg[r]` and a zero-extended immediate value, and stores the result in `GReg[r]`.

- SUB r,s—This performs the subtraction of GReg[s] from GReg[r], and stores the result in GReg[r].
- SUBI r,immediate—This performs the subtraction of a zero-extended immediate value from GReg[r], and stores the result in GReg[r].

38.16.9 Compare Instructions

Compare instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if the comparison is true, otherwise it is cleared.

NOTE

Only one version of the immediate form is implemented. Non-equality comparisons to immediate values will require two instructions.

- CMPEQ r,s—This sets T when registers GReg[r] and GReg[s] are equal.
- CMPEQI r,immediate—This sets T when register GReg[r] and the zero-extended immediate value are equal.
- CMPLT r,s—This sets T when register GReg[r] is less than and not equal to GReg[s]. The comparison is signed.
- CMPHS r,s—This sets T when register GReg[r] is greater than or equal to GReg[s]. The comparison is signed.

38.16.10 Test Instructions

Test instructions modify the T bit in the processor status according to the result of the operation. The T bit is set if any bit in the result is one, otherwise it is cleared.

- TST r,s—This performs an and between GReg[r] and GReg[s], and sets T if the result is not zero.
- TSTI r,immediate—This performs an and between GReg[r] and a zero-extended immediate value, and sets T if the result is not zero.

38.16.11 Byte Permutation Instructions

These instructions shuffle the bytes in a register. For the purpose of describing these instructions, have the bytes in a register be numbered from the most significant as b_3, b_2, b_1, b_0.

- RORB r—The rotate right byte. The result is b_0, b_3, b_2, b_1.
- REVB r—The reverse bytes in word. The result is b_0, b_1, b_2, b_3.
- REVBLO r—The reverse, two low-order bytes. The result is b_3, b_2, b_0, b_1.

38.16.12 Bit Shift Instructions

The following are bit shift instructions:

- ROR1 r—The rotate right 1 bit. This instruction does a circular right shift of 1 bit.

- `LSR1 r`—The logical shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by a 0.
- `ASR1 r`—The arithmetic shift right 1 bit. This instruction shifts all bits to the right by 1. The high order bit is replaced by itself.
- `LSL1 r`—The logical shift left 1 bit. This instruction shifts all bits to the left by 1. The low order bit is replaced by zero.

38.16.13 Bit Manipulation Instructions

- `BCLRI r,n`—The bit clear is immediate; clears bit number *i* in register *r*.
- `BSETI r,n`—The bit set is immediate; sets bit number *i* in register *r*.
- `BTSTI r,n`—The bit test is immediate; tests bit number *i* in register *r* (T becomes equal to the selected register bit).

38.16.14 SDMA Memory Access Instructions

All memory accesses are 32 bits. Any memory location that is implemented with less than 32 bits (for example, peripheral registers) causes unimplemented bits to be read as 0s. All memory accesses will cause either the SF or DF flags in the processor status to be set if they cause a fault. What constitutes a fault, especially when accessing peripheral registers, is a property of the memory location.

- `LD r,(b,d)`—The load instruction creates an address by adding the displacement field (d) to the contents of the base register (b). The SDMA location at the resulting address is read and placed in the destination register (r).
- `ST r,(b,d)`—The store instruction creates an address in the same manner as the load instruction. The register (r) is stored in the SDMA location at the resulting address.

38.16.15 Functional Unit Instructions

The functional unit instructions have an 8-bit field that is placed on the functional unit bus. Some of these bits are used to select which functional unit should be involved in the transfer. The remaining bits are decoded by the selected functional unit so their specific use depends on the functional unit. See Section 38.17, “Functional Units Programming Model.”

There are two functional unit instructions, as follows:

- `LDF r,fub`—The 8-bit field is placed on the functional unit bus and a read is issued to the selected functional unit. As a result of this instruction, the SF may be set in the processor status.
- `STF r,fub`—The 8-bit field is placed on the functional unit bus and a write is issued to the selected functional unit. As a result of this instruction, the DF may be set in the processor status.

38.16.16 Illegal Instructions

All instruction encodings that are illegal cause the following actions:

- The current PC (which points to one beyond the offending instruction) is put in the EPC register.
- The loop mode bit is cleared.

- The PC is set to the value stored in the Illegal Instruction Trap Address (ILLINSTADDR) register (the default value is 0x0001).

ILLEGAL—Although any instruction other than those indicated in the SDMA specification will trigger the illegal instruction mechanism, the ILLEGAL instruction code is preferred as it will always be kept as *illegal* in the possible future versions of the SDMA core.

38.16.17 Debug Instructions

The following are debug instructions:

- `SOFTBKPT`—The software breakpoint instruction causes the core to stop and enter debug mode. The core can then be accessed and started by the OnCE debug module only.
- `done 5`—This instruction is used for debugging, as it copies the contents of the PCU registers and flags to the context memory. Information on this instruction is described in Section 38.19.5.2, "Saving the Context."
- `CpShReg`—This instruction copies the context memory into the PCU registers and flags. Modifying the corresponding memory location before executing this instruction enables you to have the channel continue from a new instruction address. This instruction is described in Section 38.19.5.3, "Restoring the Context."

38.17 Functional Units Programming Model

The functional unit instructions cause an 8-bit code, found in the low eight bits of the instruction, to be asserted on the functional unit control bus. Some of these bits are used to select one of several functional units. Functional units which can be selected include SDMA registers such as MSA and MSD which are not mapped in the SDMA memory map, and are accessible only through the functional unit bus. These Functional Unit Registers are listed in Table 38-70. In order to establish a programming convention, assume the selection bits are some number of the most significant bits of the 8-bit code. Furthermore, some number of the least significant bits is decoded by a given functional unit to establish the type of operation to perform.

Table 38-70. Functional Unit Registers

Functional Unit	Register	Register Name	Section/Page
Burst DMA Unit	MSA	Memory Source Address Register	38.17.1.1/38-108
	MDA	Memory Source Address Register	38.17.1.2/38-109
	MD	Memory Data Buffer Register	38.17.1.3/38-109 (Write) 38.17.1.5/38-111 (Read) 38.17.1.6/38-113
	MS	Memory State Register	38.17.1.4/38-109

Table 38-70. Functional Unit Registers (continued)

Functional Unit	Register	Register Name	Section/Page
Peripheral DMA Unit	PSA	Peripheral Source Address Register	38.17.2.1/38-122
	PDA	Peripheral Source Address Register	38.17.2.2/38-122
	PD	Peripheral Data Buffer Register	38.17.2.3/38-123 (Write) 38.17.2.5/38-125 (Read) 38.17.2.6/38-127
	PS	Peripheral State Register	38.17.2.4/38-123
CRC Unit	CA	Polynomial Register	38.17.3.1/38-132
	CS	Accumulator Register	38.17.3.2/38-133

More information regarding the functional units can be found in Section 38.5.1, “CRC Calculation Unit,”, Section 38.5.3, “Peripheral DMA Unit,” and Section 38.5.2, “Burst DMA Unit”.

38.17.1 Burst DMA Unit

The DMA instructions control the DMA state machine and may cause a DMA cycle on the associated memory bus. There are four registers associated with the burst DMA unit, a Memory Source Address register (MSA), a Memory Destination Address register (MDA), a Memory Data buffer (MD), and a state register (MS).
The burst DMA has two different uses:

- A data transfer between External Memory Interface and SDMA general register
- A data transfer in copy mode where blocks of data are transferred from the source address to the destination address

38.17.1.1 Memory Source Address Register (MSA)

The source address register contains the pointer into EMI memory associated with the next read data transfer. It has byte granularity.

Reading the register with the `ldf` instruction has no side effects, and gives the address value in the EMI memory of the next data that is read by the SDMA during an `ldf MD` instruction.

Writing the source address register has two side effects: If the prefetch bit is set, a DMA read cycle (8-word read access) is issued with the new address. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MSA to guarantee all the data is effectively written to memory.

The MSA register has two modes of programming:

- Frozen—In frozen mode, the MSA register is not modified after DMA accesses.
- Incremented (default mode)—In incremental mode, MSA is incremented by the number of bytes transferred during read cycles.

38.17.1.2 Memory Destination Address Register (MDA)

The destination address register contains the pointer into EMI memory associated with the next write data transfer. It has byte granularity.

Reading the MDA register with the `ldf` instruction has no side effects. It gives the address value in the EMI memory where the next SDMA data (`stf r,MD` instruction) is stored when MD FIFO is flushed.

Writing the destination address register has one side effect. Any data still located in the buffer is lost. If there is valid write data in the buffer, it is necessary to force the DMA to completely flush it out before modifying MDA to guarantee all the data is effectively written to memory.

The MDA register has two modes of programming:

- Frozen—In frozen mode, the MDA register is not modified after DMA accesses.
- Incremented (default mode)—The MDA register is incremented by the number of bytes transferred during write cycles.

38.17.1.3 Memory Data Buffer Register (MD)

The data buffer register consists of a bank of 36 bytes that behave like FIFO. This FIFO stores the eight words received when a read burst is triggered by the DMA (DMA is in read mode). The MD register is in write mode after a writing in MDA or after an `stf MD` instruction. In that case, a burst write access is automatically triggered when there are more than eight words in MD. For bandwidth optimization, any transfers between DMA and the EMI controller are based on burst accesses.

An `ldf r,MD|SIZE` instruction that reads the data buffer may cause a DMA cycle, as follows:

- If there are less bytes in the FIFO than the size parameter of the instruction. For instance, if only two bytes are available in MD and a 4-byte read is requested, a burst read access is executed to complete the two bytes.
- If the prefetch bit is set, and after reading there is enough space in the FIFO to store a full burst, a burst read access is triggered.

An `stf r,MD|SIZE` instruction that writes to the data buffer may cause a DMA cycle if the number of written bytes in MD is higher than 32 (eight words) or if the flush bit is set.

When DMA is used for data transfer between SDMA and EMI (reading or writing), no immediate error is possible because the module manages a data misalignment issue; therefore, it is allowed to read/write a word to/from a half-word address. However, the addresses (source or destination) must belong to the EMI memory mapping. The only potential error, in this mode, would be the error sent back by the EMI controller when an access to a super-user page is detected. The whole transfer on the DMA associated bus will be considered successful when there are no errors seen on the bus during the transfer. In copy mode, an immediate error could be returned to SDMA as described in Section 38.17.1.10, “Error Management.”

38.17.1.4 State Register (MS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. MS is also accessed to set-up the conditional yielding feature.

The initialization value of this register is 0 and it consists of the following:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	spriv	stype	0	0	dpriv	dtype
W																
R	0	0	0	0	y	d	e		0	0	n					
W																

Figure 38-64. MS Structure

Table 38-71. MS Field Descriptions

Field	Description
31–22	Reserved
21 spriv	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
20 stype	Source Mode. Indicates if MSA has to be incremented (or not) during accesses. 0 Frozen—MSA is not modified. 1 Incremented—MSA is incremented by the number of transferred bytes during read access.
19–18	Reserved
17 dpriv	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved
16 dtype	Destination Mode. Indicates if MDA has to be incremented (or not) during accesses. 0 Frozen—MDA is not modified. 1 Incremented—MDA is incremented by the number of transferred bytes during write access.
15–12	Reserved
11 y	Conditional Yielding selector. When selected, the `yield`/`yieldge` instructions will not switch channels if the Burst DMA is in Write Mode, and it has less than four bytes in its FIFO. This is aimed at reducing the number of inefficient FIFO flushes due to context switches. 0 Always yields 1 Yields conditionally (when there are less than four bytes in the FIFO in write mode)
10 d	Access Direction or DMA Mode. DMA is in write mode when data was written into MD by `stf MD` instructions, or if a previous DMA cycle on the external bus was a write access. Writing MDA or MSA changes the DMA mode to the respective value.DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by SDMA with an `ldf MD` instruction. Reading MDA or MSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9–8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 error read burst

Table 38-71. MS Field Descriptions (continued)

Field	Description
7–6	Reserved
5–0 n	Number of bytes in the MD FIFO.

38.17.1.5 Burst DMA Write (stf)

When received from a `stf` instruction, the function code bits are interpreted as follows, depending on the addressed register:

Register	7	6	5	4	3	2	1	0
MSA	s		p	freeze	r			spriv
MDA								dpriv
MD			f	cpy			sz	
MS								

Figure 38-65. STF Code Bits

Table 38-72. STF Code Bit Field Descriptions

Field	Description
7–6 s	Functional Unit selector 00 for Burst DMA
5 p (MSA)	Prefetch Flag 0 No prefetch 1 Prefetch required from new MSA
5 f (MD)	Forced Flush Flag 0 Automatic flush 1 FIFO contents are flushed (including the new written data).
4 freeze (MSA/MDA)	Address Freeze Mode 0 Address is normally incremented. 1 Address is frozen.
4 cpy (MD)	Copy Mode selection 0 Write Mode 1 Copy Mode
3–2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1–0 sz (MD/MS)	Transfer Size 00 size 0 (no data stored in the FIFO) 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

Table 38-72. STF Code Bit Field Descriptions (continued)

Field	Description
0 spriv (MSA)	The spriv value is ignored for this device. 0 = valid value 1 = Reserved
0 dpriv (MDA)	The dpriv value is ignored for this device. 0 = valid value 1 = Reserved

The possible write instructions are listed in Table 38-73 (unused bits should always be cleared).

Table 38-73. Burst DMA STF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	stf r,MSA	Writes content of the SDMA general register (r) to the source address register. MSA is in incremented mode.
00_0_1_00_00	stf r,MSA|FR	Writes content of the SDMA general register (r) to the source address register. MSA is in frozen mode.
00_1_0_00_00	stf r,MSA|PF	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access. MSA is in incremented mode.
00_1_1_00_00	stf r,MSA|PF|FR	Writes content of the SDMA general register (r) to the source address register, and starts a read burst access.
00_0_0_01_00	stf r,MDA	Writes content of the SDMA general register (r) to the destination address register. MDA is in incremented mode.
00_0_1_01_00	stf r,MDA|FR	Writes content of the SDMA general register (r) to the destination address register. MDA is in frozen mode.
00_1_0_10_00	stf r,MD|SZ0|FL	No data transfers between the SDMA and MD, but all valid written data of the MD is flushed to the memory. An acknowledge or error is sent back to the SDMA core on transfer completion.
00_0_0_10_01	stf r,MD|SZ8	8-bit (byte) transfer to write buffer MD
00_1_0_10_01	stf r,MD|SZ8|FL	8-bit (byte) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_10	stf r,MD|SZ16	16-bit (half-word) transfer to write buffer MD
00_1_0_10_10	stf r,MD|SZ16|FL	16-bit (half-word) transfer to write buffer MD and flush after transfer. All valid written data of the MD is flushed to memory.
00_0_0_10_11	stf r,MD|SZ32	32-bit (word) transfer to write buffer MD
00_1_0_10_11	stf r,MD|SZ32|FL	32-bit (word) transfer to write buffer MD and flush after transfer. All valid written data of MD is flushed to memory.
00_0_1_10_00	stf r,MD|CPY	No data transfer between SDMA and MD but starts a copy transfer whose length is given by the 4 LSB of *r* register. (Maximum burst length is eight words.)
00_0_0_11_11	stf r,MS	32-bit (word) transfer to status register MS
00_0_0_11_00	stf r,MS|SZ0	Clears the error flag (if set). Other MS bits are unchanged; this instruction is also known as `clref` MS.

NOTE

When a flush bit is set, the SDMA flushes the FIFO including the newly written data. An acknowledge is sent to the core before the flush completes (except if size 0 is used). The goal of this flush bit is to force a flush, but it is recommended to use it only when needed (for example, when finishing a row of pixels during 2D data transfers). Indeed, if this bit is omitted and if there are more than 32 bytes in the FIFO, a burst write access is automatically triggered.

Since all the `stf r,MD` instructions (including the copy mode) acknowledge the SDMA core before the store is effective (except if size 0 is used), it is recommended to perform an `ldf` from MS before terminating a channel in order to check the final error status. (The `ldf` from MS will stall the core until all the data was flushed out and the transfer status is known.)

After every `stf MD` instruction, the MDA is incremented by the number of bytes that are written in MD, except when it is programmed in frozen mode.

38.17.1.6 Burst DMA Read (ldf)

When received from an `ldf` instruction, the function code bits are interpreted as follows, depending on the addressed register:

<table>
<tr><th>Register</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>MSA</td><td colspan="2" rowspan="4">s</td><td></td><td></td><td colspan="2" rowspan="4">r</td><td></td><td></td></tr>
<tr><td>MDA</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>MD</td><td>p</td><td></td><td colspan="2">sz</td></tr>
<tr><td>MS</td><td></td><td></td><td></td><td></td></tr>
</table>

Figure 38-66. LDF Code Bits

Table 38-74. LDF Code Bit Field Descriptions

Field	Description
7–6 s	Functional Unit selector 00 for Burst DMA
5 p (MD)	Prefetch Flag 0 no prefetch 1 automatic prefetch

Table 38-74. LDF Code Bit Field Descriptions (continued)

Field	Description
3–2 r	Register selection 00 MSA 01 MDA 10 MD 11 MS
1–0 sz (MD)	Transfer Size 00 reserved 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)

Table 38-75 lists the possible write instructions (unused bits should always be cleared).

Table 38-75. Burst DMA LDF Instruction List

Binary	Assembly	Comments
00_0_0_00_00	ldf r,MSA	Copies the source address register value into an SDMA general register. It gives the memory address of the next data that will be read with an `ldf` MD instruction.
00_0_0_01_00	ldf r,MDA	Copies the destination address register value into an SDMA general register. It gives the memory address where the next incoming data will be flushed.
00_0_0_10_01	ldf r,MD|SZ8	8-bit (byte) read
00_1_0_10_01	ldf r,MD|SZ8|PF	8-bit (byte) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_10	ldf r,MD|SZ16	16-bit (half-word) read
00_1_0_10_10	ldf r,MD|SZ16|PF	16-bit (half-word) read. If after this reading, and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_10_11	ldf r,MD|SZ32	32-bit (word) read
00_1_0_10_11	ldf r,MD|SZ32|PF	32-bit (word) read. If after this reading and the MD FIFO is empty, a burst read access at the MSA address is triggered.
00_0_0_11_00	ldf r,MS	Copy the status register value into an SDMA general register.

NOTE

Read data is 0-extended before writing in the SDMA general registers. When reading the MD register, the DMA takes data from the FIFO if it is available. If part or whole data is not in the FIFO, an external burst read access is performed to provide the missing data. The SDMA is stalled as long as the required read data is not complete.

After every reading, MSA is incremented by the number of read bytes from MD FIFO, except when MSA is programmed in frozen mode.

38.17.1.7 Prefetch/Flush and Auto-Flush Management

The prefetch and auto-flush management enables the SDMA RISC machine to go on while a DMA access is performed. When the RISC core requires a prefetch (p = 1) to the Burst DMA, it will receive an immediate transfer acknowledge before the DMA has finished the external access. This enables the RISC core to do other things like accessing another DMA machine.

The basic principle in prefetch mode is for the DMA to anticipate data reads from the SDMA RISC engine by fetching external bursts of data as soon as there is enough space in the DMA FIFO to store it. If ever the RISC engine required data that is not available in the FIFO, the read acknowledge is delayed until the data is available, but it does not have to wait until the burst completes.

The auto-flush basic principle is similar: An automatic flush is triggered every time there are eight words to be written in the FIFO. If the FIFO is full and the RISC engine requires another write, it is stalled until the burst has started and enough space was freed in the FIFO to store that new data. This means the SDMA RISC engine does not have to wait for the completion of a burst to receive its acknowledge and continue its processing.

In particular, an auto-flush is executed when DMA is in write mode and if the following is true:

- If the FIFO is empty and the first write is to a word-aligned address of any size (ex: the 2 LSB of MDA[1:0]= 0x0), the auto-flush is triggered immediately after the write of the 32’nd byte.
- If the FIFO is empty, and if MDA is an odd byte address (1, 3, 5, 7,...) and an `stf MD|SZ8` is executed, the byte is flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is a half-word address (2, 6, 0xA,...) and an `stf MD|SZ16` is executed, the two bytes of the incoming data are flushed to memory. Once MDA increments to a word aligned address, the auto-flush will be triggered every 32 bytes.
- If the FIFO is empty, and if MDA is not a word-aligned address (ex 1, 2, 3, 5, 6, 7, 9,...), and an `stf MD|SZ32` is executed, the first 1 to 3 bytes will be flushed up to the next word aligned address. Afterwards, an auto-flush will be triggered each time the FIFO receives 32-bytes.

  Therefore, if an `stf MD|SZ32` is executed with MDA equal to 0x1 and with an empty MD FIFO, the bytes located at addresses 1, 2, and 3 are flushed, and the byte located at address 4 remains in MD FIFO. This solves the misalignment issue. Additionally, the next write instructions (`stf`) complete the FIFO until it contains eight words; then a burst write is executed by the DMA to empty the FIFO. Protocol on the external bus does not support bursts of different data types (byte, half-word, or word).

For example, consider the case where data is written using a byte access, `stf MD|SZ8`. The value of MDA during the very first byte write determines when the auto-flush will occur as follows:

- If MDA=0x0, the flush occurs following the write of byte 32
- If MDA=0x1, the flush occurs following the write of byte 1, byte 3 and byte 35.
- If MDA=0x2, the flush occurs following the write of byte 2 and byte 34.
- If MDA=0x3, the flush occurs following the write of byte 1 and byte 33.
- If MDA=0x4, the flush occurs following the write of byte 32

The flush command forces the DMA to flush all MD valid bytes to the EMI controller. An acknowledge is sent immediately to the SDMA, and any potential error is reported on a future access. It is thus essential to conclude a transfer with a last read from MS, which will stall the core until all data was flushed out and returned to the transfer status (acknowledge or error).

NOTE

During this kind of auto-flush—which occurs only at the beginning of a misaligned write transfer—no acknowledge is sent back to the SDMA, which is stalled until a flush is completed.

38.17.1.8 Data Alignment and Endianness

38.17.1.8.1 Burst DMA in Read Mode

For every read access to MD, the data returned to the SDMA core and the new FIFO state depends on the MSA status and the access size. The FIFO is considered as a stack of 36 bytes: Data is fetched externally on a 32-bit bus, but the valid bytes only are stored in the FIFO and left-aligned (for a transfer of consecutive words, it is only the first word that may be truncated). Table 38-76 shows the FIFO byte alignment strategy and the corresponding MSA, the returned data, and the new FIFO state for any access size of an internal read from MD.

Table 38-76. FIFO Read Configuration

Before Read		Internal Read Access Size	Read data	After Read	
MSA[1:0]	FIFO State			MSA[1:0]	FIFO State
00	x0 x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 and so on...	sz8	00 00 00 x0	01	x1 x2 x3 y0 y1 y2 y3 z0
		sz16	00 00 x0 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz32	x0 x1 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
01	x1 x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 and so on...	sz8	00 00 00 x1	10	x2 x3 y0 y1 y2 y3 z0 z1
		sz16	00 00 x1 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz32	x1 x2 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
10	x2 x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 and so on...	sz8	00 00 00 x2	11	x3 y0 y1 y2 y3 z0 z1 z2
		sz16	00 00 x2 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz32	x2 x3 y0 y1	10	y2 y3 z0 z1 z2 z3 t0 t1

Table 38-76. FIFO Read Configuration (continued)

Before Read		Internal Read Access Size	Read data	After Read	
MSA[1:0]	FIFO State			MSA[1:0]	FIFO State
11	x3 y0 y1 y2 y3 z0 z1 z2 z3 t0 t1 t2 and so on...	sz8	00 00 00 x3	00	y0 y1 y2 y3 z0 z1 z2 z3
		sz16	00 00 x3 y0	01	y1 y2 y3 z0 z1 z2 z3 t0
		sz32	x3 y0 y1 y2	11	y3 z0 z1 z2 z3 t0 t1 t2

38.17.1.8.2 Burst DMA in Write Mode

For every write access to the MD, the new FIFO state depends on the MDA status and the access size. The FIFO is considered as a stack of 36 bytes: Data is stored in the FIFO according to the internal access size and the former MDA value. Table 38-77 shows the FIFO byte alignment strategy corresponding to MDA, as well as the new FIFO state for any access size of an internal write to MD.

Table 38-77. FIFO Write Configuration

Before Write		Internal Write Access Size	Written Data	After Write	
MDA[1:0]	FIFO State			MDA[1:0]	FIFO State
00	tt uu vv ww ?? ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	01	tt uu vv ww x0 ?? ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	10	tt uu vv ww x0 x1 ?? ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	00	tt uu vv ww x0 x1 x2 x3 ?? ?? ?? ??
01	tt uu vv ww xx ?? ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	10	tt uu vv ww xx x0 ?? ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	11	tt uu vv ww xx x0 x1 ?? ?? ?? ?? ??
		sz32	x0 x1 x2 x3	01	tt uu vv ww xx x0 x1 x2 x3 ?? ?? ??

Table 38-77. FIFO Write Configuration (continued)

Before Write		Internal Write Access Size	Written Data	After Write	
MDA[1:0]	FIFO State			MDA[1:0]	FIFO State
10	tt uu vv ww xx yy ?? ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	11	tt uu vv ww xx yy x0 ?? ?? ?? ?? ??
		sz16	?? ?? x0 x1	00	tt uu vv ww xx yy x0 x1 ?? ?? ?? ??
		sz32	x0 x1 x2 x3	10	tt uu vv ww xx yy x0 x1 x2 x3 ?? ??
11	tt uu vv ww xx yy zz ?? ?? ?? ?? ?? and so on...	sz8	?? ?? ?? x0	00	tt uu vv ww xx yy zz x0 ?? ?? ?? ??
		sz16	?? ?? x0 x1	01	tt uu vv ww xx yy zz x0 x1 ?? ?? ??
		sz32	x0 x1 x2 x3	11	tt uu vv ww xx yy zz x0 x1 x2 x3 ??

NOTE

If the FIFO mode changes from a write to a read mode, all remaining written bytes in MD are lost but no error is returned. Typically, this happens if an `ldf` MD is executed after `stf` MD instructions. Before a mode change, it is recommended to force the flush of a potential remaining byte by a `stf MD|SZ0|FL` instruction. In the same way, if a FIFO mode changes from a read to a write mode, all prefetched data present in the FIFO is lost and no error is returned.

38.17.1.8.3 Endianness

Big and Little Endian are supported by the Burst DMA, but data is always stored in MD in Big Endian. Byte manipulation is performed when data is exchanged with an Burst controller (for example, during read or write burst accesses).

38.17.1.9 Copy Mode

A mechanism is available to perform fast AP-to-AP transfers. Data does not flow through the SDMA core: It is kept in the DMA FIFO. This mechanism is selected when writing MD with a special option in the instruction code (copy flag). It is possible to transfer up to eight words in one SDMA instruction (this does not mean in one cycle). In this mode, every time an `stf MD|CPY` is executed, a read burst is executed and directly followed by a write burst transfer. Burst transfers are limited to eight words. The size of the transfer (in words)—given by the SDMA general register (4 LSB)—is also limited to eight. The following

SDMA code shows how 100 bytes could be copied from the MSA address to the MDA address. This is sample code only.

Example 38-1. Burst DMA copy mode example

```
        ldi r0,@src
        stf r0,MSA                  // Source address setup
        ldi r1,@dst
        stf r1,MSA                  // Destination address setup
        ldi r0,0x64                 // data transfer counter
        ldi r1,0x8
MAIN_XFER:
        cmphs r0,r1                 // Is r0 >= 0x8
        bf LAST_XFER                // If not, jump to last transfer label
        stf r1,MD|CPY               // Copy 8 words from MSA to MDA address.
        subi r0,0x8                 // Decrement counter
        jmp MAIN_XFER               // return to main transfer loop
LAST_XFER:
        stf r0,MD|CPY
```

The main transfer loop is executed 12 times; then `r0` equals 4 and the last transfer loop is run.

In this mode, an acknowledge is transmitted to the core as soon as the read burst can start; thus, a first copy instruction returns an immediate acknowledge and subsequent copy instructions will be acknowledged as soon as the previous copy has finished.

38.17.1.10 Error Management

Another point to consider is the management of errors. Because the DMA immediately sends an acknowledge to the RISC core (except for the `stf MS|SZ0|FLS` instruction), it assumes no error will occur. If an error occurs, it is flagged (transfer error acknowledge) for the following DMA access. This should not be a problem if the DMA is used properly. The MD accesses are meant to stall the SDMA as little as possible to optimize throughput and hide calculation time. Therefore, final access to MS should be performed before closing a channel. This access waits until any pending operation is finished in the burst DMA and gather any remaining error.

In copy mode, an error could be immediately returned to the SDMA on execution of the `ldf` copy or `stf` copy instruction. It happens when MSA or MDA are not word addresses (for example, 0[4]). This is because copy mode must only be used for transferring a large packet of aligned data.

When an error is received during a *read* transfer to the external bus, which may occur during the burst accesses, the MD FIFO contains the valid beats of the burst, and the error flag of MS is set to 2’b11 (error read burst). It is possible to read MS (“n” field) to know how much valid data remains in MD and when MD is empty (after `ldf` instructions). The next read MD instruction sets the MS error flag to 2’b10 (error mode), and an error is sent back to the SDMA core. In error mode, it is possible to read MSA, which gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, gives rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

In “error read burst” mode, writing MDA, MSA, or MD, or starting a copy transfer by a `stf MD|COPY` instruction will cancel the error mode. Table 38-78 shows when an immediate error is sent back according to the executed instruction.

Table 38-78. Possibilities in ERROR READ BURST Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA (IU IPF) stf rn, MDA stf rn,MD|COPY	NO	Error mode is reset. MSA, MDA, or MD are updated and a DMA cycle may start. For the `stf MD|COPY`, a copy loop is executed.
stf rn, MS	NO	MS is updated.

ldf rn, MS
ldf rn, MSA
ldf rn, MDA
	NO	MS, MSA, and MDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, MD	YES/NO	Immediate error if there is no more data available for read in the FIFO.

When an error is received during a *write* transfer, the error is reported to the next DMA access. In this case, an error is sent to the SDMA core and the DMA goes to its error mode. Reading MS gives the number of bytes that remain in MD; reading MDA gives the address of the error data. Any attempt to read or write MD, or to modify MDA or MSA in error mode, give rise to an error; therefore, an error flag must be reset by clearing MS at the end of the SDMA code section responsible for error management.

Table 38-79. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, MD stf rn, MSA stf rn, MDA	Yes	Any attempt to modify MD, MSA, MDA will raise an immediate error and burst DMA remains in error mode. When address registers are write-accessed, an error is returned.
stf rn, MS	No	This is the only way to exit error mode. MS[9:8] must be reset by an `stf MS|SZ0` instruction.

ldf rn, MS
ldf rn, MSA
ldf rn, MDA
	No	MS, MSA, and MDA could be read in error mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, MD	Yes	Whatever the DMA direction (read or write), an `ldf rn` triggers an immediate error.

38.17.1.11 Conditional Yielding

The standard SDMA transfer is based upon a hardware loop that has the following structure:

Example 38-2. Hardware Loop

```
loop
load Rn,source  // can be ldf or ld
<computation>   // can be done through functional units
store Rn,dest   // can be st or stf
done 0          // yield
```

This structure needs to be kept independent of the functional units' particularities regarding the context switch. However, there can be variations in the context switch's efficiency, which can depend on the number of data received up to that point, and on the data itself.

The DMA, with its 8-word burst capability, has a preferable context switch period when its address register is 8-word aligned: It is the only moment that occurs once every eight loops when the succession of bursts is not broken by the context switch. When this is not the case, a context switch requires the storing (or loading) of less than eight words, which requires separate accesses and is far less efficient. The rest of the 8-word packet is stored (or loaded) after the context restore, and this is done as separate accesses.

The proposed solution is a conditional yielding, which occurs only when the DMA is in an optimum state. It does not require any modification to the scripts. The condition is decided at the DMA level.

The DMA can be programmed in two modes—conditional or always-true—for every channel, which provides complete flexibility. By default, the DMA is not in conditional mode.

The DMA condition is computed from the FIFO fill level and the various modes, as follows:

- When copy mode is selected, regardless of the transfer direction ('read' or 'write'), the condition is always true.
- In read mode, the condition is always true.
- In write mode, the condition is true when there are four bytes or less in the FIFO; it is false when there are more than four bytes. The 4-byte limit comes from the possibility of saving those bytes as MD with absolutely no impact on the bus accesses.

The aim at conditional yielding is to avoid splitting bus accesses (especially bursts).

38.17.2 Peripheral DMA Unit

The peripheral DMA unit is connected to the Multi-Layer DMA Crossbar Switch of the AP platform. Its goal is to perform data transfers between any modules connected to the DMA bus of this platform. These modules are either peripherals or memories. The peripheral DMA could be seen as the AP DMA controller.

The DMA performs data transfers in three modes:

- Read mode, where data is read from peripherals or from memory connected to the AP and copied in a SDMA general register.
- Write mode, where data of a general register has to be written in a peripheral or a memory.
- Copy mode, where data is read from a peripheral (or memory) at a source address (PSA) and automatically written to a peripheral (or memory) at a destination address (PDA).

In copy mode, no SDMA general register is involved as transferred data only goes through the data register of the DMA.

The peripheral DMA has three addressing modes: frozen, incremented, and decremented, as follows:

- Frozen mode—When source or destination addresses are frozen, their value is not modified after a transfer. This mode is typically used for addressing peripheral FIFOs located at a fixed address.
- Incremented mode—When source or destination addresses are in incremented mode, after every transfer they are incremented by the number of bytes transferred.

- Decremented mode—In decremented mode, addresses are decremented by the number of bytes transferred.

The peripheral DMA registers are as follows:

- Two, 32-bit address registers (PSA and PDA) that respectively contain the source address for a read access and the destination address for a write access
- A 32-bit status register (PS) that contains information on the peripheral DMA configuration, such as the number of valid bytes in the data register, the error flag, the source and destination address mode, and so on.
- A 32-bit data register (PD) that stores data involved in a data transfer

38.17.2.1 Peripheral Source Address Register (PSA)

The source address register contains a pointer to a source peripheral or a memory associated with the next read data transfer. It has byte granularity. It is based on the following:

- A 32-bit register (PSA) to store the address value
- A 2-bit register (stype) to store the source address mode (frozen, incremented, or decremented)
- A 2-bit register (ssize) to store the source target data path size (byte, half-word, or word)

Reading the register with the `ldf` instruction has no side effects and gives the address value of the next data that will be read by the SDMA during an `ldf MD` instruction. Writing the source address register may have side effects. If there is valid write data in the data register and the source address is changed, the write data is discarded. If the prefetch bit is set, a DMA read cycle is issued with the new address.

When PSA is to be written, you must specify the source target address mode, providing its size (byte, half-word, or word). This enables omission of the size field in all `ldf MD` instructions. When DMA performs a read cycle, its size is given by the value of the PSA source size register (ssize). If source is a memory in incremented mode, first programmed in word mode (`stf PSA|SZ32|I`), and if an SDMA script needs to read bytes from this memory, the size of the source target must be updated before executing new accesses. The source address mode and its size are given by labels added to the `stf PSA` instruction as described in the write section. The ssize and stype registers are part of the DMA status register (PS).

Writing to PSA may issue an immediate error if the source size is not compatible with the value to be written into the PSA register. For instance, writing a 2 in PSA and specifying that it is memory-accessed in word mode creates an immediate error.

38.17.2.2 Peripheral Destination Address Register (PDA)

The destination address register contains a pointer to a source peripheral or a memory associated with the next write data transfer. It has byte granularity. It is based on the following:

- A 32-bit register (PDA) to store the address value
- A 2-bit register (dtype) to store the destination address mode (frozen, incremented, or decremented)
- A 2-bit register (dsize) to store the destination target data path size (byte, half-word, or word)

Reading the register with the `ldf` instruction has no side effects, and gives the address value of the next data that will be written by SDMA during an `stf MD` instruction. Writing the destination register has no side effect. Similar to the PSA register, the destination address mode and source are specified in the `stf PDA` instruction and may also generate an error in case of incorrect programming.

38.17.2.3 Peripheral Data Register (PD)

The data register of the peripheral DMA is a 32-bit register. When the destination address is correctly set up, any writing to PD will automatically flush the new input data. The number of SDMA bytes that will be transferred is given by the PDA size register. Unlike other SDMA DMAs, PD is not a FIFO: It is not used to accumulate bytes that from the SDMA and must be packed before being sent to external memories. In read mode, and if the source address is correctly set up, an `ldf` instruction will empty PD. If a prefetch is required along with the instruction, the DMA will initiate a new read transfer.

Reading PD in prefetch mode only stalls the SDMA when the prefetched data is not yet available. Writing PD only stalls the SDMA if the previous write operation was not completed. As soon as the previous operation is over, the acknowledge is sent back to the SDMA RISC engine.

An error flag—part of PS—is set when an external access fails. The error is thus reported to the next SDMA instruction that involves the peripheral DMA.

38.17.2.4 Peripheral State Register (PS)

The state register contains the DMA state-machine value. It can be accessed in case of an error received during a transfer. Although all PS fields can be written by an `stf` instruction, it is recommended to access only the error bit (to reset it). Modifying other PS fields will provide an un-guaranteed DMA behavior.

The initialization value of PS is 0, and it consists of the following structure:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ssize		stype		dsize		dtype	
W																
R	0	0	0	0	0	d	e		0	0	0	0	0	n		
W																

Figure 38-67. PS Structure

Table 38-80. PS Field Descriptions

Field	Description
31–24	Reserved
23–22 ssize	Source Target Size. Determines the size of the read transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
21–20 stype	Source address Mode. Determines whether PSA is incremented, decremented, or kept unmodified after every read from the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
19–18 dsize	Destination Target Size. Determines the size of the write transfers on the external bus. It should match the accessed device characteristics. 00 *reserved* 01 Byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
17–16 dtype	Destination address Mode. Determines whether PDA is incremented, decremented, or kept unmodified after every write on the external bus. 00 Frozen Mode 01 Incremented Mode 10 Decremented Mode 11 *reserved*
15–11	Reserved
10 d	Direction Flag or DMA Mode. DMA is in write mode when data was written into PD by `stf PD` instructions, or if a previous DMA cycle on the external bus was a write access. Writing PDA or PSA does not change the DMA mode. DMA is in read mode when a previous DMA cycle was a read access, and DMA stays in read mode when data is read by the SDMA with an `ldf PD` instruction. Reading PDA or PSA does not change the DMA mode. 0 Read Mode 1 Write Mode
9–8 e	Error. Indicates if the previous access was acknowledged with a bus error. 00 No error was received. 01 *reserved* 10 Error mode 11 Error read
7–3	Reserved
2–0 n	number of bytes in PD

NOTE

dtype, dsize, stype, and ssize are updated when PSA and PDA are written.

38.17.2.5 Peripheral DMA Write (stf)—Write Mode

When written by an `stf` instruction, the function code bits are interpreted as follows:

Register	7	6	5	4	3	2	1	0
PSA	s		p	ar	am		sz	
PDA								
PD			pdsel					
PS			pssel					

Figure 38-68. STF Code Bits

Table 38-81. STF Code Bits Field Descriptions

Field	Description
7–6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PSA)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
3–2 am (PSA/PDA)	Address Mode. Determines how PSA or PDA is modified after every read or write access to the PD. 00 Frozen—Address registers are not modified after the transfer. 01 Incremented—Address registers are incremented by the number of transferred bytes. 10 Decremented—Address registers are decremented by the number of transferred bytes. 11 Updated—PSA and PDA are not modified. Either address mode is not modified, but the width of the data path is updated by the sz field.
1–0 sz	Transfer Size 00 *reserved* 01 byte (8 bits) 10 half-word (16 bits) 11 word (32 bits)
5–0 pdsel	PD access selector 001000 is the only valid option
5–0 pssel	PS access selector 111111 writes to PS 001100 only clears the error flag in PS

Due to the large number of possible `stf` instructions, Table 38-82 provides only a short list of all the possible write instructions:

Table 38-82. Peripheral DMA STF Instruction List

Binary	Assembly	Comments
11_00_00_01 11_00_00_10 11_00_00_11	stf Rn, PSA|SZ8 |F stf Rn, PSA|SZ16|F stf Rn, PSA|SZ32|F	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is frozen.
11_10_00_01 11_10_00_10 11_10_00_11	stf Rn,PSA|SZ8 |F|PF stf Rn,PSA |SZ16|F|PF stf Rn,PSA |SZ32|F|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • 1, 2, or 4 bytes are *fetched* from the peripheral source. • Source address is frozen.
11_00_01_01 11_00_01_10 11_00_01_11	stf Rn, PSA|SZ8 |I stf Rn, PSA|SZ16|I stf Rn, PSA|SZ32|I	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1,2 or 4 after read PD.
11_10_01_01 11_10_01_10 11_10_01_11	stf Rn, PSA|SZ8 |I|PF stf Rn, PSA|SZ16|I|PF stf Rn, PSA|SZ32|I|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA + 1, 2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_10_01 11_00_10_10 11_00_10_11	stf Rn, PSA|SZ8 |D stf Rn, PSA|SZ16|D stf Rn, PSA|SZ32|D	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA—1,2, or 4 after read PD.
11_10_10_01 11_10_10_10 11_10_10_11	stf Rn, PSA|SZ8 |D|PF stf Rn, PSA|SZ16|D|PF stf Rn, PSA|SZ32|D|PF	• Source is a byte, half-word, or word target at the Rn address. Any further PD read instructions will trigger a byte, half-word, or word access to the source. • Source address is in incremented mode: PSA = PSA—1,2, or 4 after read PD. • 1, 2, or 4 bytes are *fetched* from the peripheral source.
11_00_11_01 11_00_11_10 11_00_11_11	stf Rn, PSA|SZ8 |U stf Rn, PSA|SZ16 |U stf Rn, PSA|SZ32 |U	• *Update* source pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost.
11_10_11_01 11_10_11_10 11_10_11_11	stf Rn, PSA|SZ8 |PF|U stf Rn, PSA|SZ16 |PF|U stf Rn, PSA|SZ32 |PF|U	• *Update* source pointer, which becomes a pointer to a target accessed in byte, half-word, or word. • PSA value is not modified by Rn. • Bytes present in PD are lost. • 1, 2, or 4 bytes are *fetched* from the memory source.
11_01_00_01 11_01_00_10 11_01_00_11	stf Rn, PDA|SZ8 |F stf Rn, PDA|SZ16|F stf Rn, PDA|SZ32|F	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is frozen.

Table 38-82. Peripheral DMA STF Instruction List (continued)

Binary	Assembly	Comments
11_01_01_01 11_01_01_10 11_01_01_11	stf Rn, PDA|SZ8 |I stf Rn, PDA|SZ16|I stf Rn, PDA|SZ32|I	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA + 1, 2, or 4 after write PD.
11_01_10_01 11_01_10_10 11_01_10_11	stf Rn, PDA|SZ8 |D stf Rn, PDA|SZ16|D stf Rn, PDA|SZ32|D	• Destination is a byte, half-word, or word target at the Rn address, and any further PD write instructions will trigger byte, half-word, or word access to the destination. • Destination address is in incremented mode: PDA = PDA—1, 2, or 4 after write PD.
11_01_11_01 11_01_11_10 11_01_11_11	stf Rn, PDA|SZ8 |U stf Rn, PDA|SZ16 |U stf Rn, PDA|SZ32 |U	• *Update* destination pointer to memory, which becomes a pointer to a memory accessed in byte, half-word, or word. • PDA value is not modified by Rn • bytes present in PD are lost
11_00_10_00	stf Rn, PD	• Write “dsize” bytes of Rn in PD and automatically flush to destination target
11_11_11_11	stf Rn, PS	• Write status register
11_00_11_00	stf Rn,clrefPS	• Clear error flag if set

NOTE

When writing PD, size information is not important: It is embedded in the dsize field of PDA register. If dsize is 1, 2, or 4, then one, two, or four bytes from Rn is written to the PD register, and automatically flushed out to the destination target.

38.17.2.6 Peripheral DMA Read (ldf)—Read Mode

When received from an `ldf` instruction, the function code bits are interpreted as follows.

Register	7	6	5	4	3	2	1	0
PSA	s			ar	a			
PDA								
PD			p	cpy				
PS			pssel					

Figure 38-69. LDF Code Bits

Table 38-83. LDF Code Bits Descriptions

Field	Description
7–6 s	Functional Unit selector 11 for Peripheral DMA
5 p (PD)	Prefetch Flag 0 no prefetch 1 automatic prefetch
4 ar (PSA/PDA)	Address Register Selector 0 PSA 1 PDA
4 cpy (PD)	Copy Mode 0 standard access 1 copy mode access
3 a	Register Set selection 0 PSA or PDA 1 PD or PS
5–0 pssel	PS access selector 111111 is the only valid option to read PS

Table 38-84 provides a list of supported `ldf` instructions.

Table 38-84. Peripheral DMA LDF Instruction List

Binary	Assembly	Comments
11_0_0_0_000	ldf Rn, PSA	Reads 32-bit of PSA value
11_0_1_0_000	ldf Rn, PDA	Reads 32-bit of PDA value
11_0_0_1_000	ldf Rn, PD	Reads programmed source size bytes of PD (0-extended)
11_1_0_1_000	ldf Rn, PD|PF	Reads programmed source size bytes of PD (0-extended), and starts a prefetch at PSA address.
11_0_1_1_000	ldf Rn, PD|COPY	Starts a copy transfer from the source target at the PSA address to the destination target at the PDA address. No data transmits through Rn, but Rn contents are lost (Rn is loaded with PD temporary contents that are *not* the copied data).
11_111111	ldf Rn, PS	Reads 32-bit of PS value

NOTE

When reading PD, size information is not important: It is embedded in the ssize field of the PSA register. If ssize is 1, 2, or 4, the one, two, or four bytes is transferred from PD to Rn. Read data is 0-extended.

38.17.2.7 Copy Mode

Like burst DMA, the peripheral DMA unit has a copy mode that is used when data transfers do not involve SDMA general registers. Data is read from the source target at a PSA address, stored in PD, and then automatically flushed to the destination target at the PDA address. Copy mode is only available for transfers that involve two targets of the same data path width. Since copy mode is invoked with an `ldf`

instruction, the *loaded* general purpose register loses its previous contents. (However, the new contents are unpredictable as they depend on temporary values that are seen on the external DMA bus.)

38.17.2.8 Error Management

Peripheral DMA generates two kinds of errors: the immediate error that sanctioned incorrect register programming; and the error triggered by the previous access and stored in the error flag of PS until a DMA instruction is executed.

38.17.2.8.1 Immediate Errors

Table 38-85 lists all incorrect DMA register setups.

Table 38-85. Immediate Errors with Peripheral DMA

Rn[1:0] values	DMA Instruction	Comments
0x01 0x11	stf Rn, PSA|SZ16|F stf Rn, PSA|SZ16|I stf Rn, PDA|SZ16|F stf Rn, PDA|SZ16|I	If PSA points to a half-word peripheral or to a half-word address in memory, its value must be 0 modulo 2.
0x01 0x10 0x11	stf Rn, PSA|SZ32|F stf Rn, PSA|SZ32|I stf Rn, PDA|SZ32|F stf Rn, PDA|SZ32|I	If PSA points to a word peripheral or to a word address in memory, its value must be 0 modulo 4.
PSA[1:0]–PDA[1:0]	**DMA Instruction**	**Comments**
0x01 0x10 0x11	stf Rn, PSA|SZ32|U stf Rn, PDA|SZ32|U	When PDA or PSA is updated and becomes a pointer to a word address in memory, its content must be 0 modulo 4.
0x01 0x11	stf Rn, PSA|SZ16|U stf Rn, PDA|SZ16|U	When PDA or PSA is updated and becomes a pointer to a half-word address in memory, its content must be 0 modulo 2.
Read/Write PD Instruction	**Comments**	
stf Rn,PD ldf Rn,PD	If PDA size (dsize) has never been set up before an `stf PD` instruction (dsize=0) If PSA size (ssize) has never been set up before an `ldf PD` instruction (ssize=0)	
ldf Rn,PD|CPY	Copy mode is possible only between two targets whose data path width is identical. It is P8↔P8, P16↔P16, or P32↔P32 regardless of the way the address registers are incremented.	

38.17.2.8.2 Data Transfer Errors

When PSA and PDA are correctly set up, the only error that may arise for an `ldf PD` or `stf PD` instruction would be the error of the previous DMA cycle. Error handling is driven by a single consideration: When an error occurred during a data read on the DMA interface, this error should appear as a transfer error to the core when the core attempts to retrieve the data that was not successfully read from the accessed device (memory or peripheral). When an error occurred during a write access to the DMA interface, the data is still available in PD and should not be destroyed by subsequent core accesses: The core must be warned about the error issue.

There are three error handling mechanisms for each case: Read Error (First Phase), Write Error and Read Error (Second Phase), and Copy Mode Errors handling.

38.17.2.8.3 Read Error (First Phase)

If an error occurred during a prefetch command, the peripheral DMA enters its ERROR READ mode (PS[9:8]=11). In this mode, the error is reported on the next `ldf PD` instruction and writing PSA, PDA, or PD will cancel the error flag. The module returns no error mode and instructions are normally executed (a DMA cycle may be triggered). Similarly, initiating a copy transfer will reset the error flag and start a copy transfer. Table 38-86 details which instructions can be executed in this mode.

Table 38-86. Possibilities in ERROR READ Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA (|U |PF) stf rn, PDA ldf rn,PD|COPY	NO	Error mode is reset, PSA or PDA are updated, or a write cycle is started. For the `ldf PD|COPY`, a copy loop is executed.
stf rn, PS	NO	PS is updated.

ldf rn, PS
ldf rn, PSA
ldf rn, PDA
	NO	PS, PSA, and PDA could be read in ERROR READ mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, PD	YES	Error of the previous read access is reported here and the peripheral DMA enters its ERROR mode.

38.17.2.8.4 Write Error and Read Error (Second Phase)

The peripheral DMA enters its ERROR mode (PS[9:8]=10) when the previous DMA write cycle failed, or, as explained in Read Error (First Phase), when an `ldf PD` is executed while the module is in ERROR READ mode. When a DMA cycle failed, address registers (PSA, PDA) are not modified and continue to point to the problematic address. In ERROR mode, `stf` instructions may raise an immediate error, and `ldf` instructions will not (as detailed in Table 38-87).

Table 38-87. Possibilities in ERROR Mode

DMA Instruction	Immediate Error	Comments
stf rn, PD stf rn, PSA stf rn, PDA	YES	Any attempt to modify PD, PSA, or PDA will raise an immediate error, and the peripheral DMA stays in ERROR mode. When address registers are write accessed, an error is returned.
stf rn, PS	NO	This is the only way to exit the ERROR mode. PS[3] must be reset by an `stf PS` instruction.

ldf rn, PS
ldf rn, PSA
ldf rn, PDA
	NO	PS, PSA, and PDA could be read in ERROR mode without any side effects (for example, no DMA cycle is triggered).

ldf rn, PD	YES	Whatever the DMA direction (read or write), an `ldf rn, PD` instruction will show an immediate error.

38.17.2.8.5 Copy Mode Errors

Because copy mode is a write access that follows a read access, there are two possible cases of bus error.

When the read access incurs a bus error, the peripheral DMA behaves exactly as described in Read Error (First Phase) Section 38.17.2.8.3, “Read Error (First Phase),” and Write Error and Read Error (Second Phase) Section 38.17.2.8.4, “Write Error and Read Error (Second Phase)”: It enters its ERROR READ mode, and so on.

When the error occurred during the write access of the copy transfer, the DMA enables the core to retrieve the data that was read because it is assumed the read from the peripheral removed the data from its source device. Therefore, the data to be flushed is still in PD. Any subsequent access to PD triggers an error to the core, which should execute its error handling procedure.

Once the ERROR mode is left (after writing to PS), it is possible for the core to retrieve the data in PD with an `ldf` instruction or try to flush PD contents once again (for example, when the error was due to a full FIFO and the script waited for the FIFO to be emptied) with another `ldf` instruction in copy mode. This latter instruction detects that there is valid data in PD, tries to flush it, and thus skips the read phase of the copy instruction. This is a different behavior from the usual `stf PD` instruction that overwrites PD with the selected General Purpose register contents. The same mechanism can be used any time PD holds data that is not written because of a bus error on the DMA interface; when the data was written using a copy instruction, or using the usual `stf PD` instruction.

38.17.2.8.6 Error Check Example

Example 38-3 illustrates an example checking for both immediate and data transfer errors on a store to the PD register. The first `bdf` instruction checks for an immediate error, but if a data transfer error occurred it is reported until the next instruction to access the Peripheral DMA. A second check of the error flags is done after the `ldf PS` instruction. The value of PS here can be ignored. The act of reading any register in Peripheral DMA while it is in an error mode that returns the error to the core to set either the SF or DF flag. Any error returned on an `ldf` command sets the SF flag and any error returned on an `stf` instruction sets the DF flag. This can create a situation as shown in Example 38-3 where a bus error during a DMA write which would normally be considered as a destination fault is reported as a source fault because the error was reported to the SDMA core during an `ldf` instruction.

Example 38-3. Peripheral DMA Error Check

```
clrf    0               // Clear SF and DF flags
stf     R4, PD          // Write data to memory
bdf     error_routine   // Check for immediate error from write to PD.
ldf     r3, PS          // Read PS (PS value in R3 can be ignored)
bsf     error_routine   // Check for bus error from “stf R4,PD”
                        // SF is set because it is a ldf instruction, even though
                        // the original error was a destination fault
```

38.17.2.9 Prefetch/Flush Management

There is no flush bit because every time data is stored in PD by a `stf PD` instruction—assuming PDA is correctly programmed—it is automatically flushed to the destination. An acknowledge is returned in the

cycle of the DMA instruction, and the SDMA is only stalled by an instruction that addresses the peripheral DMA when the previous DMA access is not over.

38.17.3 CRC Unit

The Cyclic Redundancy Check (CRC) unit is connected to the SDMA core using the FUBUS. This unit can perform a CRC calculation for a set of given polynomials from degree 8 to 32.

When all CRC unit registers are loaded, the unit can process one byte of data every clock cycle. When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses.

Two 32-bit registers comprise the unit:

- The CRC algorithm CA that describes the polynomial
- The CRC checksum CS to accumulate the data after each processing

38.17.3.1 Polynomial Register (CA)

This register defines the CRC algorithm currently used in the calculation, and the management of data ordering to/from the data register CS. Before starting any CRC calculation, it must be loaded with the chosen polynomial reference number and data ordering mode as indicated in Figure 38-70 and Table 38-88.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
R	0	0	0	0	0	0	0	0	0	0	0	ri	ro	p		
W																

Figure 38-70. CA Structure

Table 38-88. CA Descriptions

Field	Description
31–5	Reserved
4 ri	Reverse bit order of data in 0 Must be used when the peripheral receives bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral receives bytes as bit streams in MSB-first order (MMC case).

Table 38-88. CA Descriptions (continued)

Field	Description
3 ro	Reverse bit order of data out 0 Must be used when the peripheral transmits bytes as bit streams in LSB-first order (UART case). 1 Selects the reverse mode, which must be used when the peripheral transmits bytes as bit streams in MSB-first order (MMC case).
2–0 p	Polynomial selection 000 CRC32 ($X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$) 001 CRC16 ($X^{16}+X^{15}+X^{2}+1$) 010 CCITT16 ($X^{16}+X^{12}+X^{5}+1$) 011 IS136 ($X^{12}+X^{10}+X^{8}+X^{5}+X^{4}+X^{3}+1$) 100 CRC10 ($X^{10}+X^{9}+X^{5}+X^{4}+X+1$) 101 CRC8 ($X^{8}+X^{2}+X+1$) 110 Parity ($X^{8}+1$) 111 Reserved

38.17.3.2 Accumulator Register (CS)

The accumulator register accumulates the division remainder during the CRC processing.

When writing the accumulator register (process bit is not set) if the *ro* bit is set, the write data has its order reversed whatever the data length (the length is specified by the selected polynomial).

When computing new CRC (writing CS and process bit set) if the *ri* bit is set in CA, any byte of write data will have its bit order reversed before storage and calculation. In the first byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on. In the second byte, bit 7 is replaced by bit 0, bit 6 is replaced by bit 1, and so on, which means in the write data, bit 15 is replaced by bit 8, bit 14 will replaced by bit 9, and so on.

If the *ro* bit is set in CA, any read data will have its bit order reversed whatever the data length. If the valid data length is 16 bits, bit 15 is replaced by bit 0, bit 14 is replaced by bit 1, and so on.

When loading new data to compute the CRC, the SDMA can perform 32-bit, 16-bit, or 8-bit accesses. The CRC is computed in four clock cycles when performing a 32-bit access, in two clock cycles when performing a 16-bit access, and in one clock cycle when performing an 8-bit access. In all cases, an immediate acknowledge is sent back to the SDMA. A wait state is inserted if the SDMA tries to perform a new access before the end of the computation.

When performing a multi-bit access, the first byte used to compute the CRC is the most significant byte of the valid data.

38.17.3.3 Write Instruction (stf)

Table 38-89 shows the `stf` instructions.

Table 38-89. Stf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	Write polynomial encoded in the General Register (CA)
1	0	0	0	0	1	0	0	Write accumulator register (right-aligned)

Table 38-89. Stf Instructions for CRC (continued)

Byte Command								Description
1	0	0	1	0	1	0	0	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	0	1	Compute the CRC with the new incoming byte (b0)
1	0	0	1	0	1	1	0	Compute the CRC with the new incoming halfword (b1 b2) The bytes are used in the following order: b1 and b0.
1	0	0	1	0	1	1	1	Compute the CRC with the new incoming word (b3 b2 b1 b0) The bytes are used in the following order: b3, b2, b1, and b0.

38.17.3.4 Read Instruction (ldf)

Table 38-90 shows the `ldf` instructions.

Table 38-90. Ldf Instructions for CRC

Byte Command								Description
1	0	0	0	0	0	0	0	read the polynomial register encoded
1	0	0	0	0	1	0	0	read the accumulator register, right aligned

38.17.3.5 Operating Mode

The following is the operating mode example:

- Load the polynomial register to select the algorithm and preset the accumulator, if required.
- Data is right-aligned in the general register.
- Input data is fed byte-by-byte into the accumulator through `stf` instructions.
- When all data is fed into the CRC unit, the CRC checksum is read with `ldf ca, sz`.

Example 38-4. 16-bit CCITT CRC with P(X) = $X^{16}+X^{12}+X^5+1$

```
.equ rL,0         // GReg[0] = rL; loop count register
.equ rA,1         // GReg[1] = rA; CCITT16 polynomial
.equ rP,2         // GReg[2] = rP; initial CRC value
.equ rT,3         // GReg[3] = rT; transferred byte register
.equ aT,4         // GReg[4] = aT; transferred byte address
                  // (MMC DAT_RX - $A003)
.equ CA,8         // CRC unit CA register
.equ CS,9         // CRC unit CS register
ldi rA,2          // init register with CCITT16 polynomial
stf rA,CA         // updates the selected polynomial
ldi rP,$ff        // initializes register with $000000ff
stf rP,CS,0       // presets the accumulator with $000000ff
ldi aT,$A0        // initializes aT with $A003: aT = $000000A0
revblo aT         // initializes aT with $A003: aT = $0000A000
ori aT,$03        // initializes aT with $A003: aT = $0000A003
ldi rL,200        // expects to read 200 bytes from MMC DAT_RX
loop 2            // executes 200 times the next 2 instructions
ld rT,(aT,0)      // loads next byte from MMC DAT_RX
stf rT,CS,1       // process checksum with new incoming byte
ldf rT,CS         // read the final checksum at the end
```

38.17.4 OnCE and Real-Time Debug

The On-Chip Emulation module (OnCE) is the debug interface to the SDMA. It supports the access to all core internal devices (registers, memory, and so on), and provides a set of mechanisms that control the core. The OnCE is accessed by JTAG ports at the chip's board level, or by the host using its peripheral bus.

To reduce the size of the hardware material involved, all tasks supported by the OnCE are performed on the SDMA core. The architecture of the SDMA OnCE is relatively simple and very flexible.

The commands supported by the SDMA OnCE are listed in the following sections.

38.17.4.1 Memory and Register Access

A set of mechanisms is provided to access SDMA memory and register locations. Both reading and writing are allowed. The access is supported if the processor is in debug mode.

Those registers can also be accessed through the AP Control interface when the OnCE is controlled by the AP, as described in Section 38.19, "Using the OnCE."

38.17.4.2 Hardware Breakpoints

An event detection unit is implemented to support memory breakpoints. The unit watches the data exchanged between the SDMA memory bus and the core. A debug request is sent to the core when matching conditions occur. The unit supports mixed conditions based on address range, access type, and data value. Event detection unit configuration registers are memory mapped in the SDMA space (see Section 38.13.3, "SDMA Core Register Descriptions"): You can modify them through a regular memory access or the AP control interface.

38.17.4.3 Watchpoints

One output pin is provided to monitor matching trigger conditions that are defined in the event detection unit.

38.17.4.4 Software Breakpoints

The SDMA instruction set contains a software breakpoint. Upon executing a software breakpoint instruction, the core suspends normal execution and enters debug mode. No hardware step execution mode is implemented in the OnCE module, but this feature may be implemented at the software level with this instruction.

38.17.4.5 Core Control

Commands are provided to monitor and control processor activity. You can halt the core, rerun the core from another address location, and get processor status. Any hardware breakpoint on the instruction bus is not supported, but this feature may be implemented by inserting a software breakpoints program.

38.18 The OnCE Controller

The OnCE controller receives commands from the AP or from the JTAG controller. Each command is interpreted before being sent to the core.

38.18.1 OnCE Commands

A small set of commands supports the communication between the OnCE module and the external world. This command set enables you to perform any of the following tasks: control processor activity, save core context, and execute an SDMA instruction from the OnCE module. Combined together, these tasks perform more complex commands.

A full OnCE command contains a 4-bit instruction (the OnCE command opcode) and a variable length data field (the OnCE data). During command execution, the OnCE data is transferred in a OnCE internal register before being exchanged with the SDMA. Some data values are also exported. This mechanism creates a link between the processor and the external world. Nine commands are defined: Table 38-91 presents their formats.

Table 38-91. OnCE Command Opcode Values

Instruction Opcode	Name	Action	Register	Data Field Size	Mode
0000	rstatus	Reads the OnCE status register	STATUS	16-bit	normal/debug
0001	dmov	Updates general register GReg1	GREG1	32-bit	debug
0010	exec_once	Runs the instruction from the SDMA instruction register	INSTRUCTION	16-bit	debug
0011	run_core	Returns to normal execution	BYPASS	1-bit	debug
0100	exec_core	Returns to normal execution using a jump instruction that specifies the new address	INSTRUCTION	16-bit	debug
0101	debug_rqst	Stops the core after execution of current instruction	BYPASS	1-bit	normal
0110	rbuffer	Reads the real time buffer	RTB	32-bit	normal/debug
0111-1110	reserved	Reserved	BYPASS	1-bit	normal/debug
1111	bypass	Bypasses TAP controller	BYPASS	1-bit	normal/debug

Each instruction corresponds to a specific action performed on the OnCE module. The nature of the associated data field is clearly identified. The `dmov` command is followed by a 32-bit data value (which is a data value for the SDMA); the `exec_once` and the `exec_core` commands are followed by a 16-bit data value (which is an instruction for the SDMA); the `rstatus` command is followed by a 16-bit control value (which is the content of the OnCE status register); the `rbuffer` command is followed by a 32-bit data value. The `debug_rqst` and the `run_core` commands are followed by a single bit data field (this is a bypass value). Finally, the bypass instruction enables the SDMA JTAG TAP controller to be daisy-chained with another JTAG TAP controller. This is a JTAG-only feature. The set of commands is simple, but enables you to perform any possible task on the SDMA during a debug process.

38.18.2 Sending Commands to the OnCE Controller

The JTAG access is the standard access to the OnCE, but sometimes the JTAG is not available to fix some bugs (if the chip is in production for instance), an additional access is then required. Therefore, one AP access to the OnCE is provided.

38.18.2.1 Using the JTAG Interface

A serial access is performed through the five JTAG pins TCK, TRST, TMS, TDI, and TDO. A Test Access Port controller is provided to decode the TMS control signal. It produces shift-enable signals (shift_ir and shift_dr), and updates enable signals (update_ir and update_dr). It is fully compatible with the IEEE Std 1149.1™ testability (JTAG) standard.

During the shift_ir state, the command opcode is shifted into the OnCE controller (for example, the signal from the TDI pin is shifted into the command register and the TDO pin receives the signal shifted out). After transferring the four bits of the command, an update_ir signal is asserted and the command is decoded. The target data register is now clearly identified and the corresponding control signal is produced, as follows: bypass enable signal (bp_en), instruction enable signal (inst_en), data enable (data_en), and status enable signal (stat_en).

During the shift_dr state, the TDI signal is shifted into one of the following target registers: bypass register (1 bit), SDMA instruction register (16 bits), SDMA data register (32 bits), or OnCE status register (16 bits). The TDO pin is connected to the output of the selected register to receive the signals shifted out.

The JTAG access is disabled when the AP access is enabled.

38.18.2.2 Using the AP

The AP access to the OnCE is not the standard access, but it is required if the JTAG is not available. For example, if the SDMA ROM is out of use on a chip in production, and the AP needs to download new code and restart the SDMA, the OnCE can easily perform this operation. This type of debug operation justifies the use of an AP access to the OnCE.

To drive the OnCE, the AP uses some registers contained in the AP Control module of the SDMA. These registers are accessed through the AP peripheral bus. Most of these registers are connected to another register in the OnCE controller. Thus, accessing one of these registers is equivalent to accessing the associated register in the OnCE controller.

The set of registers in the AP Control module is listed below:

- ONCE_ENB register (1 bit, read/write)—This 1-bit register enables the AP access to the OnCE. When this bit is set, the signals from the JTAG are ignored. When it is cleared, all writing operations to the following registers through the Host Control interface are ignored. This register is reset on a JTAG reset.
- ONCE_CMD register (4 bits, read/write)—This 4-bit register receives the command opcode. It is connected to the command register in the controller. A write access to this register causes the associated command to be executed on the OnCE. For example, after writing “0001” in this register, a `dmov` command is executed.

NOTE

On the AP side, the `rstatus` and `bypass` commands are not supported. This register is reset on a JTAG reset.

- ONCE_DATA register (32 bits, read/write)—This 32-bit register is connected to the SDMA data register. This register is used when executing a `dmov` or `rbuffer` command.

NOTE

Before requesting a `dmov` command, the 32-bit data to transfer must be written in the ONCE_DATA register. At the end of the execution, the register is updated with GReg1 former value. This register is reset on a JTAG reset.

- ONCE_INSTR register (16 bits, read/write)—This 16-bit register is connected to the SDMA instruction register. This register is used when executing an `exec_core` or an `exec_once` command.

NOTE

Before requesting an `exec_core` or an `exec_once` command, the appropriate instruction must be written in the ONCE_INSTR register. This register is reset on a JTAG reset.

- ONCE_STAT register (16 bits, read only)—A read access to the ONCE_STAT register returns the content of the OnCE status register (OSTAT). This register is read only.
- The bypass register is not useful when the AP controls the OnCE, therefore no register is defined in the AP Control module to access the bypass register.

38.18.2.3 Conflicts Between the JTAG and the AP Accesses

When AP access to the SDMA OnCE is enabled (that is, when the bit in the ONCE_ENB register is set), the JTAG access is disabled. This guarantees that the module is not accessed at the same time on both sides.

It is possible to check whether the JTAG access to the SDMA OnCE is enabled from the JTAG port. When the JTAG access is disabled, the SDMA TDO always returns 1. The check requires the following steps:

- Execute a `dmov` command from debug mode (with neither 0xffffffff nor 0x0 as `dmov` value: 0x5a5a5a5a is good).
- Execute another `dmov` command (the value here is not important).

  The returned value from the latter `dmov` command should be the original one if the JTAG access is enabled; if it is 0xffffffff instead of the original input value, this means the JTAG access is disabled.

38.18.3 Executing a Command from the OnCE

All the commands defined in Section 38.18.1, “OnCE Commands,” can be accessed through the JTAG. The AP can access all these commands except the `rstatus` command. On the AP side, the OnCE status is directly accessed by reading the ONCE_STAT register.

38.18.3.1 Nature of the Commands

Two types of commands may be distinguished. First, there are two commands that do not interact with the core: `rstatus` and `rbuffer`. Those commands may be requested at any time: They do not depend on the core status.

NOTE

Each of these commands exports a data value or a status value from the SDMA.

There are also commands that interact with the core: `dmov`, `run_core`, `exec_core`, `exec_once`, and `debug_rqst`. These commands are core status dependent, as follows:

- During user mode only the `debug_rqst` is taken into account.
- During debug mode, all these commands are taken into account except the `debug_rqst`. For example, an `exec_once` command requested while not in debug mode has no effect.

38.18.3.2 Execution Request

The SDMA starts executing a task in debug mode when requested by the OnCE controller. The execution starting time depends on the type of access used to communicate with the OnCE.

If the JTAG is used, the request is send after decoding the `update_dr` state in the TAP controller. Therefore, always cross this state when sending a command through the JTAG. If the OnCE is driven from the AP side, the request is sent after detecting a write access to the ONCE_CMD register. All the registers involved in this operation must be loaded first.

The following is an example of an `exec_core` command execution from the AP side: After writing 0b010 in the ONCE_CMD register, the OnCE controller asks the SDMA to execute the instruction contained in the ONCE_INSTR register. The instruction involved should be available in the ONCE_INSTR register before the beginning of the execution.

38.18.3.3 Command Execution

The following list shows the commands and details how each command is executed:

- `rstatus` command execution—The `rstatus` command exports the content of the OnCE status register (OSR). If the JTAG is used, the status information is captured in the OnCE status register during the capture_dr state, and shifted out after 16 TCK clock cycles in the shift_dr state. The `rstatus` command is not supported on the AP side, but a status register is provided instead. The `rstatus` may be performed in both debug and user modes.
- `dmov` command execution—The `dmov` command accesses SDMA internal registers. Executing a `dmov` instruction exchanges the 32-bit data values between the SDMA data register and the general register GReg[1].

  If the JTAG is used, the content of GReg1 is captured in the SDMA data register during the capture_dr state, then it is shifted out after 32 TCK clock cycles in the shift_dr state. During the update_dr state, GReg1 is updated with the new, shifted-in 32-bit data value. If the OnCE is driven from the AP side, the data values contained in GReg1 and the SDMA data register are exchanged

after detecting a write access to the ONCE_CMD register. The ONCE_DATA register must therefore be loaded first.

- `exec_once` command execution—The `exec_once` command executes the instruction loaded in the SDMA instruction register. The command may only be requested from debug mode. The SDMA returns to debug mode at the end of the execution.

  Change of flow instructions as well as instructions that may cause a context switch are not supported: The comprehensive list comprises `done/yield/yiedge` (except `done` 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions.

  No other command should be requested before the SDMA returns to debug mode. The SDMA status (for example, whether it is in debug mode or not) can be detected by polling with the `rstatus` OnCE command, monitoring the debug_mode pin, or checking the Section 38.11.3.19, "OnCE Status Register (ONCE_STAT)," register using the AP control interface.

  NOTE

  Most of the instructions are single-cycle, which omits the step of polling the status. Loads and stores to DMA units are typical instructions that might require this polling.

  If the JTAG is used, the 16-bit instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. A request is sent to the core when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the request is sent to the SDMA when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must be therefore be loaded first.

- `run_core` command execution—The `run_core` command leaves debug mode and resume normal program execution. The next instruction executed is the last instruction decoded before entering debug mode. Be sure to restore core context before re-running the core. This procedure is detailed in Section 38.19.5.3, "Restoring the Context."

  If the JTAG is used, a 1-bit bypass value is shifted in the bypass register in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the core is rerun when detecting a write access to the ONCE_CMD register.

- `exec_core` command execution—The `exec_core` command resumes program execution from any address. The 16-bit instruction provided with the `exec_core` overwrites the last instruction decoded before entering debug mode. This command is designed to support change of flow instructions, so that a program execution can be restarted from any address. After executing an `exec_core` command, the SDMA leaves debug mode. The `exec_core` command is usually used with a `jmp` instruction.

  If the JTAG is used, the 16-bit branch instruction is shifted in the SDMA instruction register after 16 TCK clock cycles in the shift_dr state. The SDMA is rerun when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the SDMA reruns when detecting a write access to the ONCE_CMD register. The ONCE_INSTR register must therefore be loaded first. For example, to restart the SDMA from the program address 0x100, the instruction loaded should be a `jump to address 0x100` instruction.

- `debug_rqst` command execution—The `debug_rqst` command puts the SDMA in debug mode. If the JTAG is used, a 1-bit bypass value is shifted in the bypass register during the shift_dr state. A debug

request is sent to the SDMA when the update_dr state is decoded in the TAP controller. If the OnCE is driven from the AP side, the debug request is sent when detecting a write access to the ONCE_CMD register. When the SDMA is already in debug mode, this command is simply ignored.

- `rbuffer` command execution—The `rbuffer` command exports the content of the real time buffer (RTB). If the JTAG is used, the content of the real time buffer (RTB) is captured in the SDMA data register during the capture_dr state. The register is completely shifted out after maintaining the shift_dr state during 32 TCK clock cycles. If the OnCE is driven from the AP side, the content of the RTB is captured in the ONCE_DATA register after detecting a write access to the ONCE_CMD register.
- `bypass` command execution—This command is only available from the JTAG interface. It enables daisy-chaining of the SDMA JTAG TAP controller with other JTAG TAP controllers. This command does not change the SDMA state and can be executed in any mode (run, debug, or sleep). It selects the bypass register of the TAP controller.

38.18.4 Registers Descriptions

See Section 38.13.3, "SDMA Core Register Descriptions," and Section 38.11, "AP Memory Map and Control Register Definitions," for detailed information on each register.

38.18.4.1 Event Cell Counter Register (ECOUNT)

The event cell counter register is a 16-bit register that contains the number of times minus one that an event detection occurs before generating a debug request. This register should be written before attempting to use the event detection counter during an event detection process. The event cell counter register is cleared on a JTAG reset.

38.18.4.2 Event Cell Address Registers (EAA or EAB)

The event cell contains two address registers—the event cell address register (a), called EAA, and the event cell address register (b), called EAB. Every address register is a 16-bit register that stores a user-defined address value. This value computes one of the following address conditions: addra_cond or addrb_cond. Every address register is cleared on a JTAG reset.

38.18.4.3 Event Cell Address Mask Register (EAM)

The event cell address mask register is a 16-bit register that contains a user-defined address mask value. This mask is applied to the address value latched from the memory address bus before comparing addresses.

NOTE

There is a common address mask value for the two address comparators. If bit *i* of this register is set, then bit *i* of the address value latched from the memory bus does not influence the result of the address comparison. The event cell address mask register is cleared on a JTAG reset.

38.18.4.4 Event Cell Data Register (ED)

The event cell data register is a 32-bit register that contains a user-defined data value. This data value is an input for the data comparator, which generates the data_cond condition. The event cell data register is cleared on a JTAG reset.

38.18.4.5 Event Cell Data Mask Register (EDM)

The event cell data mask register is a 32-bit register that contains a user-defined data mask value. This mask is applied to the data value latched from the memory bus before comparing data. Setting bit *i* of the event cell data mask register means that bit *i* of the data value latched from the address bus does not influence the result of the data comparison. The event cell data mask register is cleared on a JTAG reset.

38.18.4.6 Real Time Buffer Register (RTB)

The real Time Buffer register is a 32-bit register that stores and retrieves run-time information without putting the SDMA in debug mode. See Section 38.19.8.2, “Real Time Buffer,” for more details.

38.18.4.7 Event Control Register (ECTL)

The event cell control register is a 16-bit register that defines cell event occurrence conditions. The event cell control register is cleared on a JTAG reset. See also Section 38.19.6, “OnCE Event Detection Unit,” for more details.

38.18.4.8 Trace Buffer (TB)

The Trace Buffer register retrieves the information in the Trace Buffer. See Section 38.19.8.1, “Trace Buffer,” for more details.

38.18.4.9 OnCE Status Register (OSTAT)

The OnCE status register is a 16-bit register that contains processor and event detection unit status. The OSTAT is a read-only register. See Section 38.11.3.19, “OnCE Status Register (ONCE_STAT),” for detailed description of the individual fields in the OSTAT register.

Figure 38-71 shows the OSTAT structure.

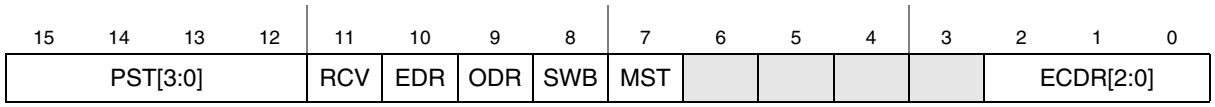


Figure 38-71. OnCE Status Register (OnCE)

Where PST[3:0] is the SDMA core state, RCV is set when the real-time buffer (RTB) is modified. EDR, ODR, and SWB are set, respectively, when the SDMA has entered debug mode because of an external debug request, a OnCE debug_rqst command, or a software breakpoint. MST is set when the OnCE is controlled from the AP control interface, and when ECDR is a three-flag set that shows the event cell condition(s) that put the core in debug mode. The OSTAT never provides more than one reason for entering debug mode.

There are two ways of accessing OSTAT content, as follows:

1. Send an `rstatus` command to the OnCE controller through the JTAG, or read the ONCE_STAT register through the AP access. Executing the `rstatus` command through the JTAG can be performed in both user and debug modes.
2. Perform an SDMA read access to the location in the SDMA core memory map (OSTAT register) debug mode using the `exec_once` command. With this method of access, the SDMA state reflected by the PST (processor status bit) is always DATA.

The register may also be accessed by a running application.

38.18.5 JTAG Interface Requirements

Because the signals received from the JTAG (running on TCK) are transferred to the OnCE controller (running on the SDMA clock), a synchronization mechanism is required.

38.18.5.1 TCK Speed Limitation

In the JTAG top-level layer, the TDO signal is always captured on a TCK falling edge. To guarantee a stable TDO signal from the SDMA during this operation, a falling edge detection is performed on TCK.

Before being latched in the *I* flip-flop (see Figure 38-72) on TCK falling edge, the TDO signal must be stable at the input of the flip-flop. This condition is verified if the TCK period is superior to the following delay:

worst-case edge detection delay + negative-edge signal propagation delay + JTAG top-level logic propagation delay

The frequency relationship, TCK < CLK/8, limitation guarantees that all operations are performed as expected.

38.18.5.2 Synchronization Implementation

Figure 38-72 shows the synchronization mechanism. Flip-flops tck0, tck1, and tck2 perform falling- and rising-edge detections on TCK. They generate the posedge_detected and negedge_detected nets that are used to sample the TDI and TMS inputs into the respective tdi and tms flip-flops, and update the tdo flip-flop to `yield` the TDO output. In the design, the only signal that might go metastable is the output of the tck0 flip-flop. This signal is captured in the tck1 flip-flop and no logical operation is performed on it to minimize a metastability propagation risk.

The TDI and TMS flip-flops also cannot go metastable: The propagation time of the rising-edge detection signal through tck0, tck1, and tck2 guarantees that the TDI and TMS inputs are stable when captured in the TDI and TMS flip-flops.

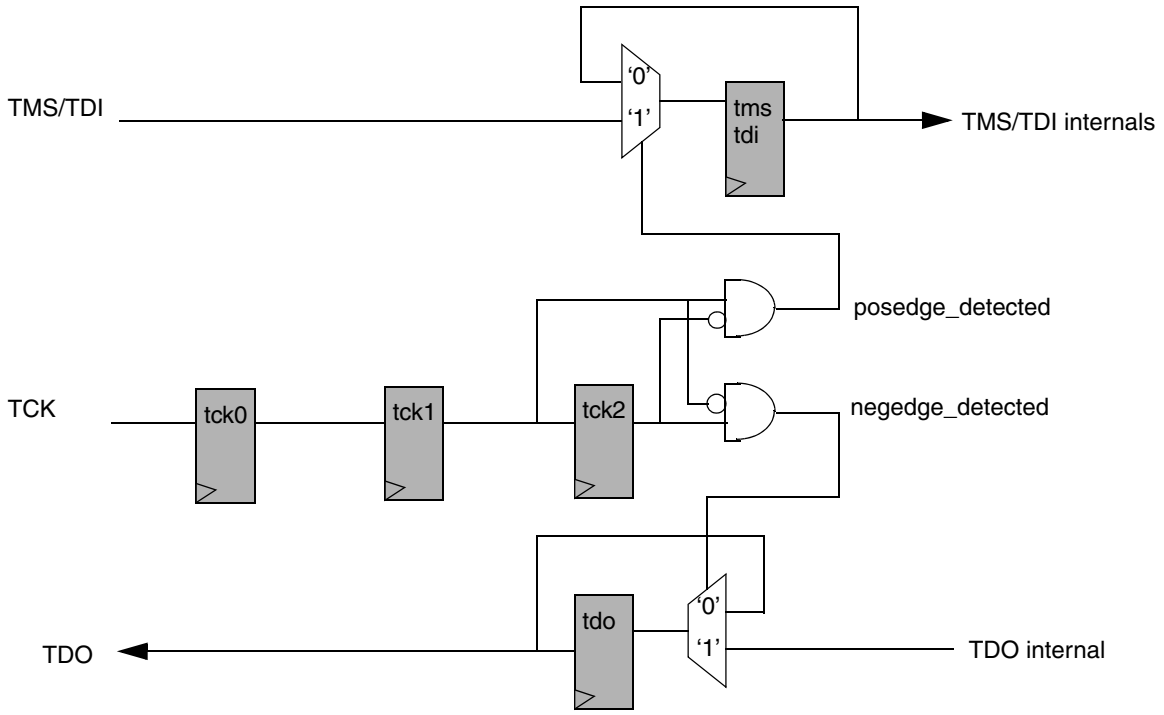


Figure 38-72. OnCE Synchronization Layer

Figure 38-73 shows synchronization timings. It takes three CLK clock cycles to synchronize TDI on the SDMA clock.

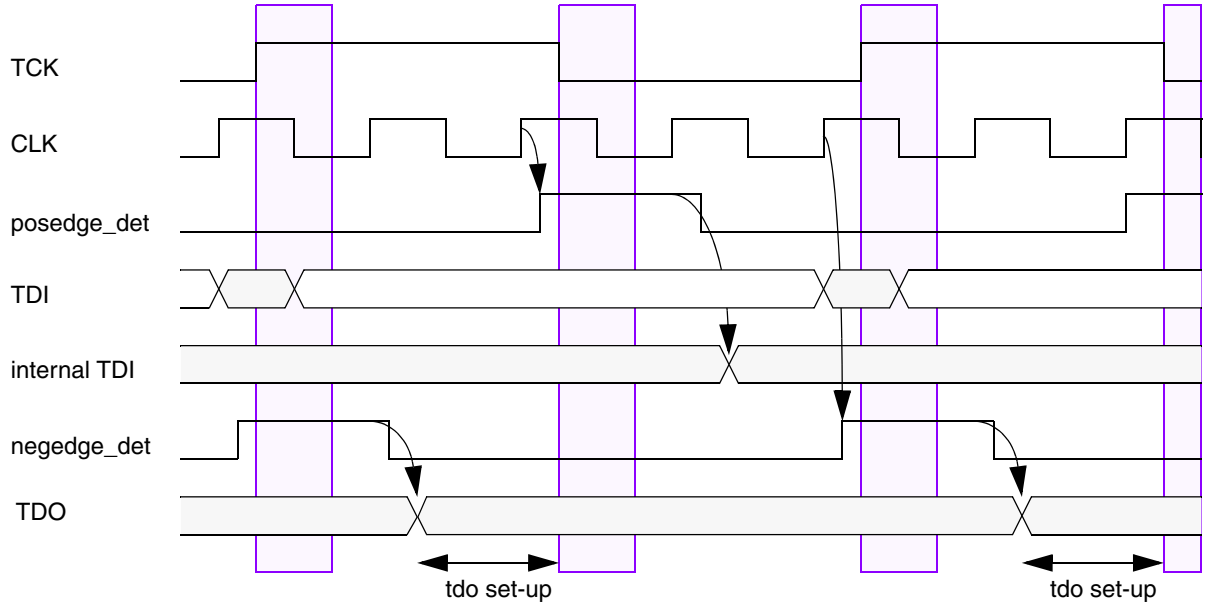


Figure 38-73. Synchronization Timings

38.18.5.3 JTAG Controller Start-Up Recommended Procedure

To ensure correct TAP controller initialization, it is recommended to use the following procedure:

1. Assert JTAG reset TRSTB (for example, set low).
2. Set TMS low.
3. Wait for 1 TCK clock.
4. Release JTAG reset TRSTB (for example, set high).
5. Wait for a minimum of five TCK cycles.

38.19 Using the OnCE

This section provides the elements necessary to run the OnCE module during a debug process. In addition to the basic set of commands described in Section 38.18.1, "OnCE Commands," more complex commands can be built to meet users' requirements.

38.19.1 Activating Clocks in Debug Mode

For power consumption issues, some clocks in the SDMA are disabled when not needed. This is the case for instance when the SDMA is in sleep mode. Clock gating management depends on the interface used to control the OnCE.

- For the JTAG access, the SDMA clock gating must be turned off using the clk_gating_off input.
- For the AP access, the SDMA clock gating is automatically turned off when the AP access is enabled (see Section 38.11.3.16, "OnCE Enable (ONCE_ENB)").

38.19.2 Getting the Current Status

Most of the commands the OnCE supports have an impact on the status of the SDMA. It is not permissible to request the execution of an instruction on the SDMA from the OnCE while the SDMA is not in debug mode. Such a violation may cause unpredictable behavior, and it might be necessary to reset the SDMA. Therefore, the value of the PST bits provided in the OnCE status register should always be checked before sending any request to the SDMA.

38.19.3 Methods of Entering Debug Mode

A debug request may be asserted at any time, but it is not always taken into account immediately. Debug mode cannot be entered in the middle of an instruction, or during the save or restore states of a context switch. The request is ignored when the core is already in debug mode. See Figure 38-4, which shows all possible transitions to the debug state, as there are several ways to enter debug mode.

38.19.3.1 External Debug Request During Reset

To enter debug mode after exiting reset, the external debug line has to be maintained high. This line is handled by the JTAG top-level block.

NOTE

The SDMA detects the debug requests only if the SDMA clock is running (see Section 38.19.1, “Activating Clocks in Debug Mode”). The debug request line should be not be maintained high when the SDMA is in debug mode.

The `debug_rqst` command (from the OnCE command set) is not supported during system reset.

38.19.3.2 Debug Request During Normal Activity

During normal activity, the SDMA enters debug mode when the following is true:

1. If the debug request line from the JTAG top-level is asserted, or
2. If the OnCE controller receives a debug_rqst command.

   The `debug_rqst` command can be sent by the JTAG access or by an access on the AP side (if the AP access is enabled).

38.19.3.3 Software Breakpoint Instruction

The SDMA enters debug mode at the end of the execution of a software breakpoint instruction. This instruction must be inserted in program flow executed by the core.

38.19.3.4 Event Detection Unit Matching Condition

If the event detection is enabled, a debug request is sent to the core after detecting a matching condition on the SDMA memory bus. See Section 38.19.6, “OnCE Event Detection Unit,” for more details.

38.19.4 Executing Instructions in Debug Mode

The OnCE supports a mechanism to execute instructions in debug mode. If the SDMA is in debug mode, then the `exec_once` command can be used to execute an SDMA instruction from the OnCE controller. The SDMA returns to debug mode at the end of each execution. Some instructions are not supported by the `exec_once` command: `done/yield/yiedge` (except `done` 5), BF, BT, BSF, BDF, JMP, JSR, JMPR, JSRR, RET, and LOOP, as well as all the illegal instructions are not supported.

NOTE

While instructions are executed in debug mode from the OnCE, the program counter of the SDMA is not incremented.

38.19.5 Command Sequences Examples

This section provides examples of command sequences that run the SDMA in debug mode. These sequences are available for both the AP and JTAG accesses.

The following presents the syntax used in this section. The data field provided with each command is put in parenthesis with the command name. A '-' is used if the data field provided is a *don't care* value.

```
my_command(data_field);    // executing my_command with a data field
my_command(-);             // executing my_command with a don't care data field
```

The value returned by the command (if there is one) is referred by an assignment. In case the value returned by the command is not used, the assignment is omitted. For an AP access, the value returned (it is always a data value) is obtained by reading back into the SDMA data register.

```
data_out = my_command(data_in);// returning a data value
```

To clarify the syntax, the instructions' opcodes are referred to by their names. In practice, use the corresponding 16-bit encoding.

38.19.5.1 Getting the SDMA Status

NOTE

Before executing any command that affects the SDMA (like `dmov` or `exec_once`), check that the SDMA is in debug mode.

Use the following snippet:

```
rstatus(); // read SDMA status until the SDMA is in debug mode
...
rstatus();
```

If the SDMA is not in debug mode, then a debug request must be generated. In this case, the SDMA enters debug mode at the end of the execution of the current instruction. Use this snippet:

```
debug_rqst(-); // debug request
```

In the following sections, it is assumed that the SDMA was successfully put into debug mode.

38.19.5.2 Saving the Context

The first debug task is to save the SDMA context, which is the content of the eight general-purpose registers, the loop and PC-related registers, and the flags. Use the general register GReg[1] as an intermediate register to export the entire context of the SDMA.

Example 38-5 shows how to save GReg[0], GReg[1], GReg[2] and GReg[3]. The sequence of commands used to export additional general registers is very similar to this.

Example 38-5. Save GReg[0], GReg[1], GReg[2], and GReg[3]

```
GReg1_data = dmov(-);              // the value exported is the content of GReg[1]
exec_once("mov GReg1,GReg0");      // puts the content of GReg[0] into GReg[1]
GReg0_data = dmov(-);              // the value exported is the content of GReg[0]
exec_once("mov GReg1, GReg2");     // puts the content of GReg[2] into GReg[1]
GReg2_data = dmov(-);              // the value exported is the content of GReg[2]
exec_once("mov GReg1, GReg3");     // puts the content of GReg[3] into GReg[1]
GReg3_data = dmov(-);              // the value exported is the content of GReg[3]
```

Get the value of the internal flags (SF, DF, T, and LM), of the loop related registers (EPC and SPC), and of the PC-related registers (PC and RPC). Use a `done 5`, which is the formatting instruction dedicated to the debug. This instruction formats the flags and the values contained in the registers. It also writes the resulting values into the channel context memory. It should not be used when entering debug from the IDLE state (for example, with no active channel script running on the SDMA), because it will update a channel context that may belong to any channel.

```
exec_once(“done 5”);      // formatting the value of flags and registers
```

At this point, the channel context should be up-to-date in memory, and debug operations should now be possible. However, the context can be exported with the following instructions:

Example 38-6. Exporting the Context

```
dmov(ctx_base_addr);                // loading GReg[1] with the channel context base address
exec_once(“ld GReg0,(GReg1,0)”);    // get RPC-PC into GReg0
exec_once(“ld GReg1, (GReg1,1)”);   // get SPC-EPC into GReg1
Loop_data = dmov(-);                // read back the value of Loop registers
exec_once(“mov GReg1, GReg0”);      // puts the PC info into GReg1
PC_data = dmov(-);                  // reads back the content of the PC registers
```

After this sequence of operations, the entire SDMA context is exported using the OnCE.

38.19.5.3 Restoring the Context

At this point in the operation, restore the context of the SDMA. It can be different from the original context located in memory, and the content previously saved into the debugging application using the OnCE. Example 38-7 shows how it is possible to modify the current channel context:

Example 38-7. Modifying the Current Channel Context

```
dmov(Loop_data);                    // put Loop former value into GReg[1]
exec_once(“mov GReg0, GReg1”);      // copy to GReg[0]
dmov(PC_data);                      // put PC former value into GReg[1]
exec_once(“mov GReg2, GReg1”);      // copy to GReg[2]
dmov(ctx_base_addr);                // put channel context base address into GReg[1]
exec_once(“st GReg0, (GReg1,1)”);   // restore Loop context
exec_once(“st GReg2, (GReg1,0)”);   // restore PC context
```

Once the context in memory is the desired context (with or without applying the previous instruction sequence), it can be restored to the *real* PC and loop registers in the SDMA core:

```
exec_once(“cpShReg”);      // restore flags and PC & loop related registers
```

After this command, the SDMA core PC, RPC, SPC, EPC registers, as well as the flags contain the same data as what is stored in the context RAM for the current channel.

Example 38-8 shows how to restore the context of general registers GReg[0], GReg[1], GReg[2] and GReg[3].

Example 38-8. Restoring the General Register Context

```
dmov(GReg3_data);          // put GReg[3] restore value in GReg[1]
exec_once("mov GReg3, GReg1");// restore GReg[3]
dmov(GReg2_data);          // put GReg[2] restore value in GReg[1]
exec_once("mov GReg2, GReg1");// restore GReg[2]
dmov(GReg0_data);          // put GReg[0] restore value in GReg[1]
exec_once("mov GReg0, GReg1");// restore GReg[0]
dmov(GReg1_data);          // restore GReg[1]
```

At this point, it is possible to restart the normal program execution.

NOTE

Every SDMA core general register value can be modified by a `mov` instruction, which makes modification of these registers easy during debug. Unfortunately, there is no such instruction as a `mov` to directly modify the contents of either PCU register or flag (PC, RPC, SPC, EPC, T, LM, SF, or DF). The `cpShReg` instruction is meant to provide a means for changing these register contents using the context memory.

38.19.5.4 Accessing the Memory

In the following example, it is assumed that the SDMA context is entirely saved. If true, it is permissible to modify the general purpose registers during debugging activity.

To perform a memory read access, the target address is stored using the OnCE in GReg[1], then the load instruction is executed on the SDMA (the data loaded from the memory overwrites the address contained in GReg[1]), and then the result value is read back using the OnCE.

```
macro READ:      dmov(target_addr);                // put the target address in GReg[1]
                 exec_once("ld GReg1,(GReg1,0)");  // execute the load instruction
                 res_data = dmov(-);               // exports the result data value
```

For a memory write access, the target address is written in GReg[0], and the value to store is written in GReg[1]. Then the store instruction is executed on the SDMA.

```
macro WRITE:     dmov(target_addr);                // puts the target address in GReg[1]
                 exec_once("mov GReg0,GReg1");     // puts the target address in GReg[0]
                 dmov(target_data);                // puts the target data in GReg[1]
                 exec_once("st GReg1,(GReg0,0)");  // performs the store operation
```

This sequence is shown as an example; however, many other sequences are possible.

NOTE

This sequence of commands can also be applied to memory-mapped registers.

38.19.5.5 Resuming Program Execution

Before resuming program execution, it is assumed that the SDMA context is properly restored. There are two ways to restart the SDMA. Start by executing the last instruction fetched before entering debug mode, as follows.

```
run_core(-);              // resume execution from where we stopped before
```

If necessary, restart the execution from a different address. In this case, use the `exec_core` command. The data field provided with this command must be the encoding of a `jump` instruction.

```
exec_core(“jmp start_addr”);// rerun the SDMA from another address
```

In these two examples, the SDMA exits debug mode and keeps executing the code fetched from the memory.

38.19.5.6 Single Stepping in RAM

To execute a program step-by-step from the RAM, insert software breakpoints in the program flow at appropriate places so that the SDMA only executes one instruction before returning to debug mode.

First, read the next instruction to execute in the RAM. Then, depending on the value of this instruction, compute the address where a software breakpoint instruction should be inserted. The instruction at the corresponding address must be saved, and, the software breakpoint instruction is inserted. After restarting the SDMA, there is only one instruction executed before meeting the software breakpoint.

Example 38-9 shows the macro functions READ and WRITE, which correspond to the sequence of commands (described above) used to access the memory.

NOTE

The data read from the memory are 32-bit values, while the instructions are 16-bit values only. This is why it is best to only use addresses divided by two when accessing the memory.

Example 38-9. READ and WRITE Macro Functions

```
next_instr = READ(run_addr/2);      // read the next instruction to execute
// the tool now has to compute the address where the breakpoint
// instruction should be inserted, this address is the “bkpt_addr”
instr_save = READ(bkpt_addr/2);     // save the instruction before overwriting
STORE(“bkpt instruction”,bkpt_addr/2);// store the bkpt instruction in memory
exec_core(“jmp run_addr”);          // rerun the SDMA
rstatus(-);                         // wait for the SDMA to enter debug mode
...
rstatus(-);
STORE(instr_save,bpkt_addr/2);      // restore the instruction overwritten
```

In case of branched conditional instructions, a breakpoint instruction should be written at the two possible target addresses.

38.19.5.7 Single Stepping in ROM

No single-step mechanism is supported in ROM. The program code can be loaded in the RAM, where the single-step mechanism can be executed.

38.19.6 OnCE Event Detection Unit

The event detection unit watches signals from the data memory bus (DMBUS), which the SDMA core uses to access its RAM, ROM, and memory mapped registers. A debug request is sent to the OnCE controller when user-defined conditions on address and/or data values are true. See Figure 38-74.

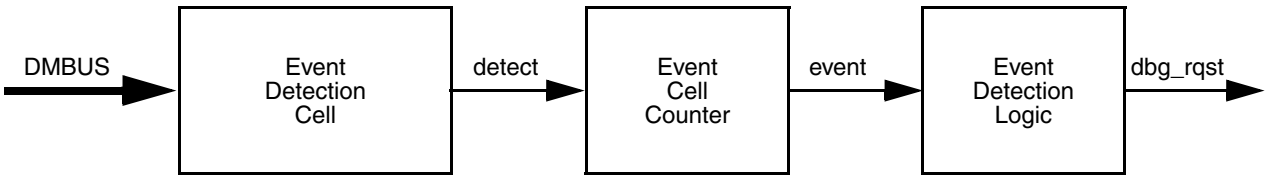


Figure 38-74. Event Detection Unit

A counter, provided with the detection cell, is decreased after an event detection. A debug request is sent to the core only when the counter reaches the value of 0. It is possible to disable the use of the counter if a debug request has to be generated after each event detection.

The event cell is the basic module that supports hardware breakpoints on an address value and/or data values coming from the SDMA memory bus. The trigger condition that generates the debug request is a mixed condition based on those values.

Figure 38-75 shows the event cell architecture. The event cell contains the address (stored in the memory address register) and the data (stored in the memory data register) used during the last memory access. There are some user-defined reference values located in memory mapped registers—the event cell addresses, the event cell address mask, the event cell data, and the event cell data mask. These registers are accessed by standard load/store instructions just like regular memory locations.

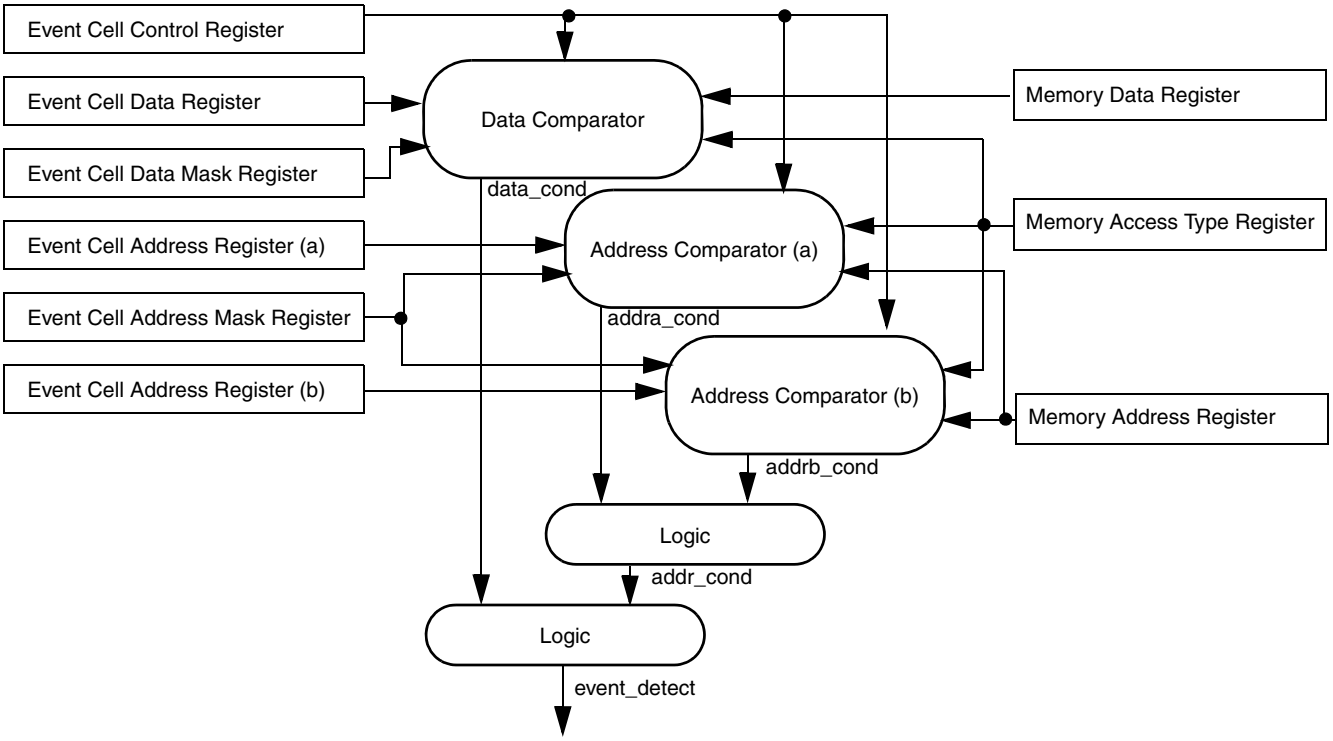


Figure 38-75. Event Cell Architecture

To define a memory breakpoint, three conditions are taken into account: The first two conditions are comparisons of the current memory address with user-defined reference addresses (these conditions are called addressA and addressB). The third condition consists of a comparison between the data received on the DMBUS and a user-defined reference data (this condition is called data). An intermediate address condition is set to express a dependency between addressA and addressB conditions.

38.19.7 Clock Gating and Reset

This section details how to use the clocks and handle the reset signals.

38.19.7.1 Clocks

Because the SDMA uses clock gating to save power, it is necessary to disable the clock gating and force the clocks to be enabled when using the OnCE. When the OnCE is accessed through its JTAG interface, clock gating must be disabled outside the SDMA using a dedicated SDMA input port clk_gating_off. The reason why detection is not performed automatically by the SDMA internal hardware is that it would cost power to monitor activity on the JTAG interface.

This clk_gating_off input is controlled by a control bit in the System JTAG Controller. See Chapter 40, “Secure JTAG Controller V1.1 (SJC),” for more information.

When the OnCE is accessed through the AP Control interface, clock gating is automatically turned off. This is done when bit 0 of the ONCE_ENB register (see Section 38.11.3.16, “OnCE Enable

(ONCE_ENB)") is set. A write access to this register is possible even when the OnCE clock is not running. If the AP access is used, the bit in the ONCE_ENB register must be set before any attempt to access any other OnCE register.

38.19.7.2 Resets

The OnCE reset is different from the SDMA main reset. The OnCE reset is connected to the TRST_B pin.

Normally, activating the SDMA reset while keeping the OnCE reset inactive (when possible) enables you to reset the core without having to reprogram the OnCE.

38.19.8 Real Time Features

To rebuild the skeleton of a program execution, it is necessary to store the addresses of the program instructions where jumps are taken: A trace buffer is therefore provided. A real time buffer has also been added to receive data values written during a program execution. The content of this register may be exported through JTAG ports without stopping the core.

38.19.8.1 Trace Buffer

The Trace Buffer is a 32-stage buffer that contains appropriate information to identify the 32 last changes of flow detected during a program execution. Figure 38-76 shows an overview of the Trace Buffer.

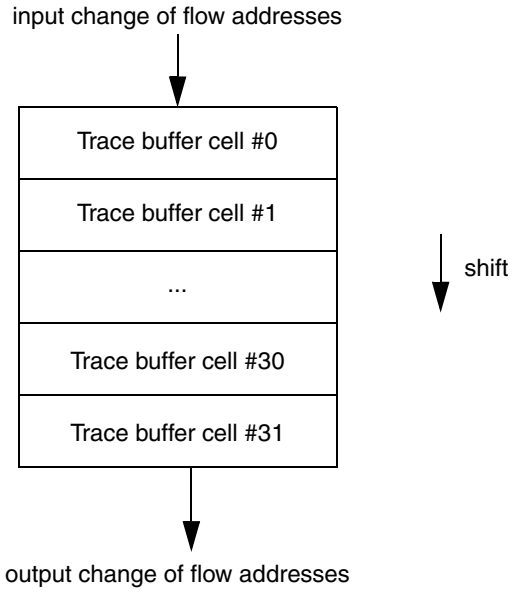


Figure 38-76. Trace Buffer

Each cell of the trace buffer contains two reference addresses and a flag. The flag is set when the addresses stored in the cell correspond to a valid change of flow; otherwise, the flag is cleared. The three most significant bits are unused.

After every change of flow detection, the address of current instruction and the address of the target instruction are stored at the top of the Trace Buffer (cell #0). The flag in the cell is set to indicate that a

valid change of flow was detected. Former cell values are shifted one level down. The Trace Buffer contains the 32 last changes of flow. All the flags are reset on a software or a hardware reset, and after each transition from debug mode to user mode.

A memory mapped register of SDMA core, the Trace Buffer register (TB), is provided to read the content of the Trace Buffer. This operation should be done in debug mode. Performing a read access to the Trace Buffer register returns the content of the bottom of the Trace Buffer (cell #31). After every read access, the trace buffer is shifted one level down, and the flag at the top of the trace buffer is cleared.

A typical OnCE command sequence that retrieves the oldest change-of-flow information is a follows:

```
exec_once(“mov r1, TB”);          // stores the oldest change-of-flow in GReg1
dmov(-);                          // retrieves GReg1 contents
```

This sequence requires the SDMA to be put in debug mode.

38.19.8.2 Real Time Buffer

The Real Time Buffer register (RTB) is a memory mapped register that can be accessed as a regular memory location by the SDMA core during program execution. This register is located in the OnCE. Executing ar `rbuffer` command (see Section 38.18, “The OnCE Controller,” for further details) exports the content of this register through JTAG ports.

When a write access is performed at the memory location corresponding to the RTB, the receive flag (for example, the RCV bit) is set in the OnCE Status Register (OSR). This flag is cleared at the end of the execution of a `rbuffer` command.

NOTE

Every write access to the RTB memory location updates the RTB register even if the RCV flag is set. The RTB is cleared on a JTAG reset.

38.19.8.3 Emulation Pin

The debug_matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Since it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

38.19.8.4 Real-Time Debug Outputs

Table 38-92 shows the debug signals that are available at the SDMA boundaries. Their availability at chip boundaries depends on the project.

Table 38-92. Real-Time Debug Output Pins

Pin	Description
debug_core_state[3:0]	The core_state bits reflect the state of the SDMA core. • The “Program” state is the usual instruction execution cycle. • The “Data” state is inserted when there are wait-states during a load or a store on the data bus (ld or st). • The “Change of Flow” state is the second cycle of any instruction that breaks the sequence of instructions (jumps and channel switching instructions). • The “Change of Flow in Loop” state is used when an error causes a hardware loop exit. • The “Debug” state means the SDMA is in debug mode. • The “Functional Unit” state is inserted when there are wait-states during a load or a store on the functional units bus (ldf or stf). • In “Sleep” modes, no script is running (this is the core idle state); the “after Reset” is slightly different because no context restoring phase will happen when a channel is triggered: The script located at address 0 is executed (boot operation). • The “in Sleep” states are the same as above except they do not have any corresponding channel: they are used when entering debug mode after reset; the reason is that it is necessary to return to the “Sleep after Reset” state when leaving debug mode. 0 Program 1 Data 2 Change of Flow 3 Change of Flow in Loop 4 Debug 5 Functional Unit 6 Sleep 7 Context Switch Saving Channel 8 Program in Sleep 9 Data in Sleep 10 Change of Flow in Sleep 11 Change of Flow in Loop in Sleep 12 Debug in Sleep 13 Functional Unit in Sleep 14 Sleep after Reset 15 Context Switch Restoring Channel
debug_yield	Pulse that is active when a `yield` (done 0) or a `yieldge` (done 1) instruction is executed. 0 - 1 `yield`/`yieldge` executed
debug_core_run	Active when the SDMA core is executing instructions. 0 Debug or sleep mode 1 Run mode
debug_event_channel_sel	Indicates if debug_event_channel displays current channel or last received event 0- debug_event_channel[5:0] gives the number of the current channel 1- debug_event_channel[5:0] gives the number of the last received event
debug_event_channel[5:0]	Gives the number of any DMA request as soon as it is received or the number of the current channel. The value of debug_event_channel_sel indicates if debug_event_channel displays the current channel or last received event. The signal debug_event_channel_sel must be observed to determine what information is provided on debug_event_chanel at any given time.

Table 38-92. Real-Time Debug Output Pins (continued)

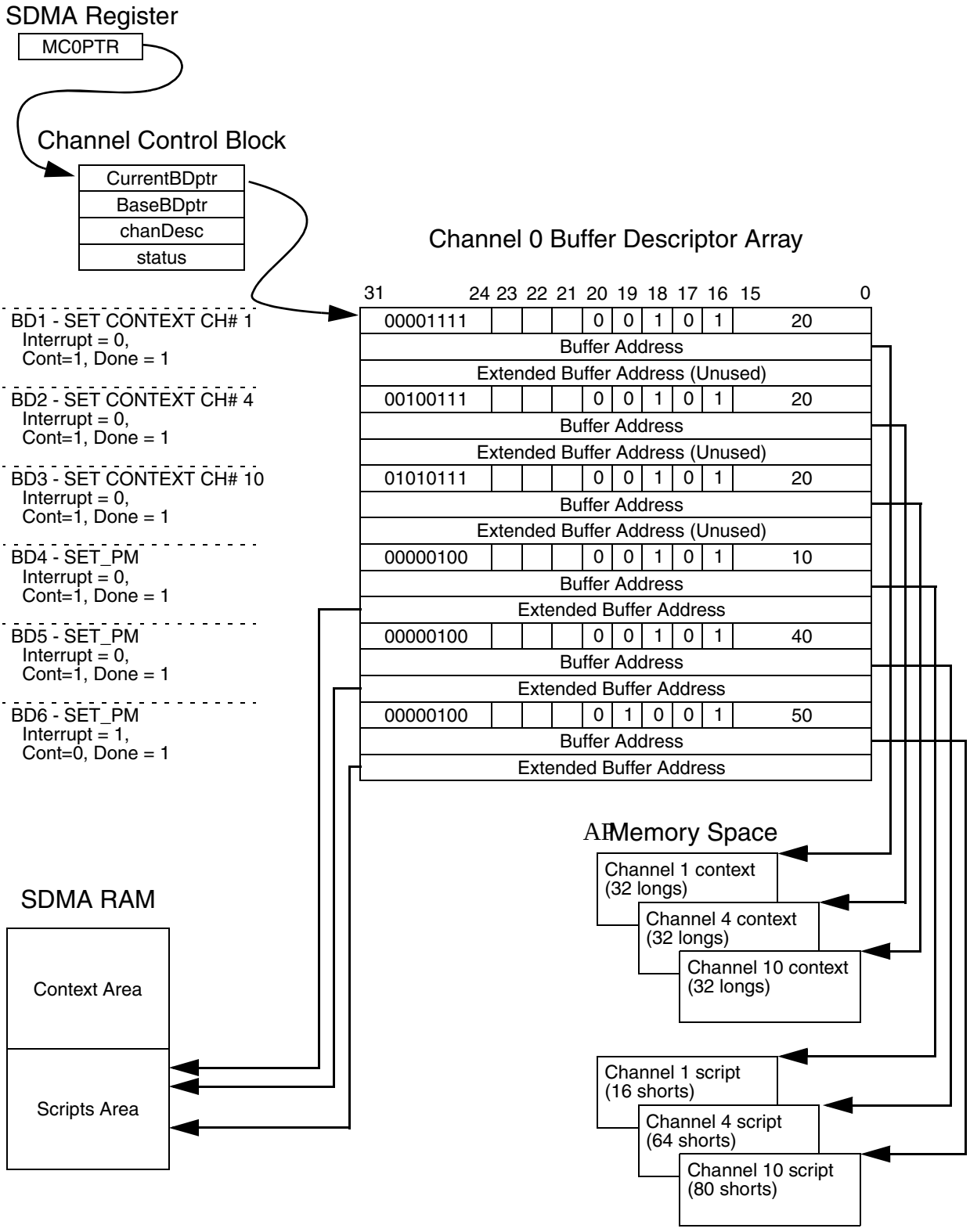
Pin	Description
debug_pc[13:0]	Program Counter value; it has a meaning when the core is in run mode.
debug_mode	Set when the core is in debug. 0 - 1 Core is in debug
debug_bus_error	Set when an error was received during a load or a store (`ld`, `st`, `ldf`, or `stf` instruction) and registered in SF or DF flag. 0 No error during last load/store 1 Error during last load/store
debug_bus_device[4:0]	Indicates the device or functional unit that is accessed by the current instruction. The debug_bus_device output is always valid when in sleep mode, debug mode, or executing any instruction that does not access the functional units or the memory mapped devices, “no access” is output. 0 No access 1 MSA 2 MDA 3 MD 4 MS 5 PSA 6 PDA 7 PD 8 PS 9 RESERVED 10 RESERVED 11 RESERVED 12 RESERVED 13 CA 14 CS 15 Reserved 16 Memory (RAM or ROM) 17 Memory mapped register 18 Peripheral #1 19 Peripheral #2 20 Peripheral #3 21 Peripheral #4 22 Peripheral #5 23 Peripheral #6 24 Peripheral #7 25 Peripheral #8 26 Peripheral #9 27 Peripheral #10 28 Peripheral #11 29 Peripheral #12 30 Peripheral #13 31 Peripheral #14
debug_bus_rwb	Indicates the direction of the access given by debug_bus_device 0 Write access (st or stf) 1 Read access (ld or ldf)

Table 38-92. Real-Time Debug Output Pins (continued)

Pin	Description
debug_matched_dmbus	Pulse indicating the OnCE event detection unit has detected a match on the data bus during an access to memory (RAM or ROM), a memory mapped register or a peripheral that is hooked to the SDMA. 0 - 1 data bus match detected
debug_rtbuffer_write	Pulse indicating when the real-time buffer is written by the core. 0 - 1 RTB was modified
debug_evt_chn_lines[7:0]	Eight lines that generate short pulses when DMA requests are received or channels are (re)started. Every line is controlled through two parameters defined in registers Cross-Trigger Events Configuration Register (1) and (2) (XTRIG_CONF1 and XTRIG_CONF2) (as described in Section 38.11, "AP Memory Map and Control Register Definitions"). The following two parameters are available for every line: • CNF—Indicates what is monitored on the line: 0 for a channel start, 1 for a DMA request reception • NUM[5:0]—Gives the number of the DMA request or channel to monitor

The matched_event emulation pin reflects the matching condition status detected by the Event Detection Unit. Because it can be necessary to detect conditions without triggering debug requests, it is possible to disable the generation of debug requests by the Event Detection Unit and still have the matching condition available on the emulation pin. This can be done by clearing the EN flag in the ECTL register.

All real-time debug outputs are disabled by default (for example, they are stuck to 0) to avoid power consumption when they are not used. They are enabled when bit 11 (RTDOBS) of the Configuration Register (CONFIG) is set. Signals provided to the system JTAG controller for SDMA debug mode status will also be enabled when the *clk_gating_off* input is asserted.


SDMA Register
MC0PTR
Channel Control Block
CurrentBDptr
BaseBDptr
chanDesc
status
Channel 0 Buffer Descriptor Array
31 24 23 22 21 20 19 18 17 16 15 0
BD1 - SET CONTEXT CH# 1
Interrupt = 0,
Cont=1, Done = 1
BD2 - SET CONTEXT CH# 4
Interrupt = 0,
Cont=1, Done = 1
BD3 - SET CONTEXT CH# 10
Interrupt = 0,
Cont=1, Done = 1
BD4 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD5 - SET_PM
Interrupt = 0,
Cont=1, Done = 1
BD6 - SET_PM
Interrupt = 1,
Cont=0, Done = 1
00001111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00100111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
01010111 0 0 1 0 1 20
Buffer Address
Extended Buffer Address (Unused)
00000100 0 0 1 0 1 10
Buffer Address
Extended Buffer Address
00000100 0 0 1 0 1 40
Buffer Address
Extended Buffer Address
00000100 0 1 0 0 1 50
Buffer Address
Extended Buffer Address
AP Memory Space
Channel 1 context
(32 longs)
Channel 4 context
(32 longs)
Channel 10 context
(32 longs)
SDMA RAM
Context Area
Scripts Area
Channel 1 script
(16 shorts)
Channel 4 script
(64 shorts)
Channel 10 script
(80 shorts)

Chapter 39
Subscriber Identification Module (SIM)

The subscriber identification module (SIM) is designed to facilitate communication to SIM cards or Eurochip pre-paid phone cards. See the SIM block diagram in Figure 39-1. The SIM module has two ports that can be used to interface with the various cards. The interface with the core is a 32-bit connection as described in the reference document IP Bus Specification.

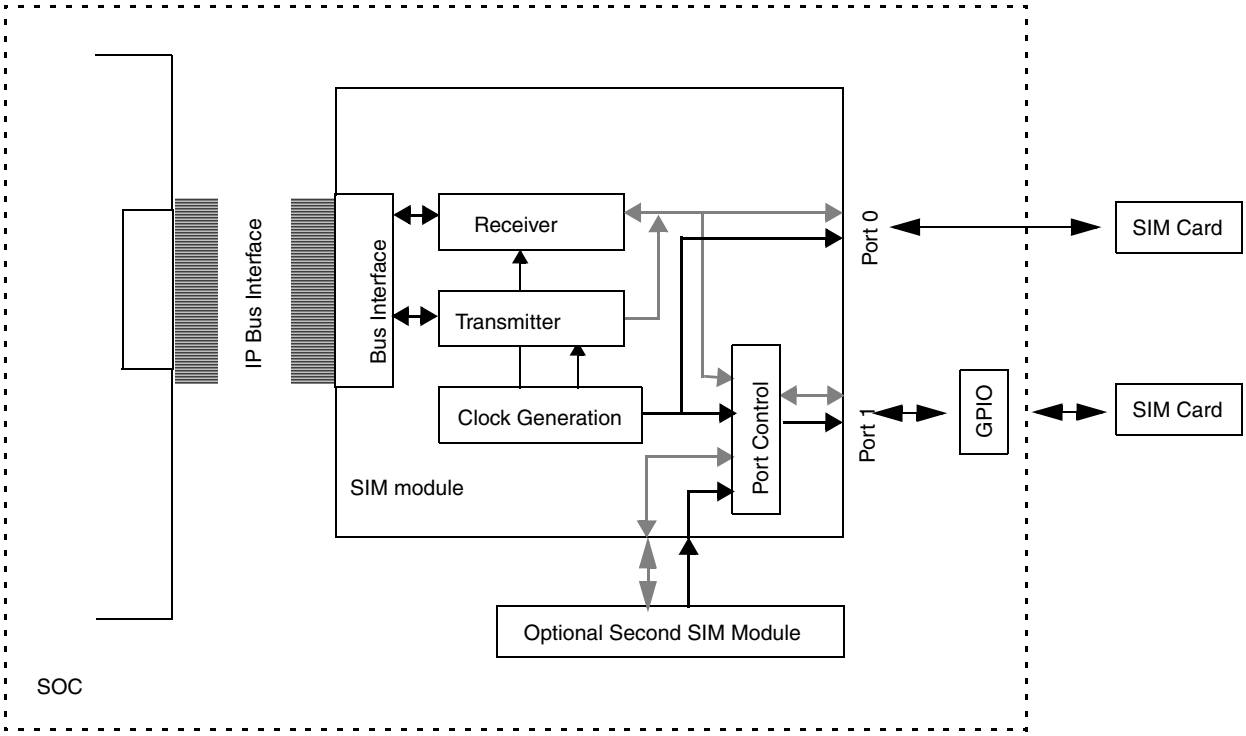


Figure 39-1. SIM Block Diagram

39.1 Overview

The SIM design is summarized in the following sections.

39.1.1 Features

The SIM design is summarized in the following sections:

- Section 39.1.3, “SIM Bus Interface Overview”
- Section 39.1.4, “SIM Clock Generator Overview”
- Section 39.1.5, “SIM Transmitter Overview”

- Section 39.1.6, “SIM Receiver Overview”
- Section 39.1.7, “SIM Port Control Overview”
- Section 39.1.8, “SIM General Purpose Counter Overview”
- Section 39.1.9, “SIM LRC Block Overview”
- Section 39.1.10, “SIM CRC Block Overview”

39.1.2 Modes of Operation

The SIM module I/O interface can be operated in one of three modes of operation summarized below.

- Two wire interface. In this mode both the IC pin RX and TX are used to interface to the SmartCard. This is activated by resetting the 3VOLTx bit in the port control register to a “0”.
- External one wire interface. In this mode the IC pins RX and TX are tied together external to the IC and routed to the SmartCard. The 3VOLT bit in the port control register is reset to a “0” and the OD_Px bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin (RX-TX) must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.
- Internal one wire interface. In this mode the IC pin TX is routed to the SmartCard. The receive pin RX is connected to the TX pin internal to the IC. The 3VOLTx bit in the port control register is reset to a “1” and the OD_Px bit in the OD_CONFIG register is set to a “1”. For this interface to work properly the IC pin TX must be pulled high by a resistor. The value should be selected small enough to give a fast enough rise time.

CAUTION:

For the internal 1-Wire interface to work, the bond pad must be capable of simultaneous input and output. (The input path should not be disabled when the output is enabled)

39.1.3 SIM Bus Interface Overview

The SIM module is designed with a separate block that encompasses the interface to the IP Bus. The bus interface is built in such a way as to be easily modified to other bus definitions. The SIM module is to be considered a 32-bit peripheral. To avoid endian issues with register access, all read/writes should be done as word access(32 bit). The bus interface has the following features:

- Peripheral Address/chip select decoding
- Illegal Address generation
- Dynamic wait state generation (tied inactive)
- Register organization
- Register bit implementations

See Table 39-1 for summary of SIM top level interrupts.

Table 39-1. SIM Top Level Interrupt Summary

Flag	Interrupt Sources	Interrupt Masks	Description
SIMIRQ_N	XTE, OEF, SDI0, SDI1, TFO, GPCNT, BWT, BGT, CWT, RTE	XTM, OIM, SDIM0, SDIM1, TFOM, GPCNTM, BWTM, BGTM, CWTM, RTM	SIM General Interrupt
SIMDAT_N	TC, ETC, TFE, RDRF, TDTF	TCIM, ETCIM, TFEIM, RIM, TDTFM	SIM Data Interrupt

39.1.4 SIM Clock Generator Overview

The SIM module contains a block designed specifically for generating the clocks used internal to the SIM module, and the clocks provided to the SIM cards. The clock generator has the following features:

- Programmable Divisor (CLK_PRESCALER[7:0]) for SIM card clock generation (***45%~55% duty cycle except for divider by 3, 5, 7, 9***)
- Receive clock generation (/1, /2, /4, /8, /16, /32, /31) from first stage
- Transmit clock generation (/12, /8) from second stage
- Programmable Divisor for Receive clock generation
- There are no interrupt sources generated by the SIM clock generator block

39.1.5 SIM Transmitter Overview

The SIM Transmitter block contains the transmit state machine, transmit shift register, and transmit FIFO. The transmitter has the following features.

- 16 bytes deep transmit FIFO
- Automatic NACK generation on parity and overrun errors
- Hardware data format support (inverse convention or direct convention)
- Retransmission of data upon SIM card NACK request with programmable maximum threshold of retransmissions
- Programmable guard time between transmitted bytes
- Six interrupt sources (see Table 39-2)

Table 39-2. SIM Transmitter Interrupt Summary

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill Error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag

39.1.6 SIM Receiver Overview

The SIM Receiver block contains the receive state machine, receive FIFO, and control logic. The receiver has the following features.

- 285 bytes deep receive FIFO
- Decoding of initial character mode for setting data format
- Hardware data format support (inverse convention or direct convention)
- NACK detection
- 11 ETU character support
- Character Wait Time Counter
- Six interrupt sources (see Table 39-3)

Table 39-3. SIM Receiver Interrupt Summary

Flag	Flag register	Mask	Mask register	Description
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive Nack threshold
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time flag
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive Data Register Full

39.1.7 SIM Port Control Overview

The SIM module supports numerous port control functions necessary for SIM card interaction. Each of the two SIM card ports has the following I/O: CLK, $\overline{\text{RST}}$, DATA_RCV, DATA_XMT, SIMPD (SIM Presence Detect), and SVEN (SIM Vcc enable). The following list gives a number of the important features of the port logic.

- Open-drain or push pull DATA_XMT pin
- Port 1, port 0, or alternate Port selection
- SIM card presence detect with interrupt capability
- Deep sleep wake-up using SIM card presence detect interrupt
- Manual control of all SIM card interface signals
- Automatic power down of port logic on SIM card presence detect
- Two interrupt sources (see Table 39-4)

Table 39-4. SIM Card Detect Interrupts

Flag	Flag register	Mask	Mask Register	Description
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT	SIM Detect Interrupt for port 0

39.1.8 SIM General Purpose Counter Overview

The SIM module provides a 16-bit counter that can be configured to count using clock sources from the card clock, receive clock, or transmitter clock (ETU Clock). A programmable 16-bit register is provided to compare with the counter for interrupt generation:

- Selectable clock source
- 16-bit counter and comparator
- One Interrupt source (see Table 39-5)

Table 39-5. SIM General Purpose Counter Interrupts

Flag	Flag Register	Mask	Mask Register	Description
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	GPCNT Comparator Interrupt

39.1.9 SIM LRC Block Overview

The SIM module contains a block designed to generate Linear Redundancy Checking (LRC) information for both received and transmitted characters. The LRC block has the following features:

- 8-bit LRC value
- Valid LRC Detector
- There are no interrupt sources generated by the SIM LRC block

39.1.10 SIM CRC Block Overview

The SIM module contains a block designed to generate Cyclic Redundancy Checking (CRC) information for both received and transmitted characters. The CRC block has the following features:

- 16-bit CRC value
- 16-bit CRC Polynomial Generator
- Valid CRC Detector
- There are no interrupt sources generated by the SIM CRC block

39.2 External Signal Description

39.2.1 Overview

See Table 39-6 for summary of the SIM module signals.

Table 39-6. Signal Properties

Name	Port	Function	Reset State	Pull-up
External Signals (Pads)				
sim_pin_sclk0	out	Clock for the smartcard on port0	0	Active

Table 39-6. Signal Properties (continued)

Name	Port	Function	Reset State	Pull-up
sim_pin_srst0	out	Reset signal for port0	0	Active
sim_pin_sven0	out	Vcc enable signal for port0	0	Active
sim_pin_data0_tx_out	out	Transmit data signal for port0	0	Active/Passive
pin_sim_rcvd0_in	in	Receive data signal for port0	—	—
pin_sim_simpd0	in	Card insertion detect signal for port0	—	—
sim_pin_sclk1	out	Clock for the smartcard on port1	0	Active
sim_pin_srst1	out	Reset signal for port1	0	Active
sim_pin_sven1	out	Vcc enable signal for port1	0	Active
sim_pin_data1_tx_out	out	Transmit data signal for port1	0	Active/Passive
pin_sim_rcvd1_in	in	Receive data signal for port1	—	—
pin_sim_simpd1	in	Card insertion detect signal for port1	—	—
Module Interrupts				
ipi_int_sim_ipb_int	out	Active high SIM Interrupt. Composed of OEF, XTE, SDI1 and SDI0.	0	—
ipi_int_sim_data_irq	out	Active high SIM Data Interrupt. Composed of TC, ETC, TFE, and RDRF.	0	—

39.2.2 Detailed Signal Descriptions

39.2.2.1 sim_pin_sclk0

The sim_pin_sclk0 is an output from the chip to the SmartCard. This signal is the clock that the SIM module provides for the SmartCard. Typical frequencies are 1 MHz to 5 MHz. This clock is 372 times the data rate that is on pin sim_pin_data0_tx_out. There is no required timing relationship between this clock signal and any of the other data signals. This is because of the asynchronous nature of the protocol.

39.2.2.2 sim_pin_srst0

The sim_pin_srst0 is the reset signal from the SIM to the SmartCard.

39.2.2.3 sim_pin_sven0

The sim_pin_sven0 is the SmartCard power supply enable control signal.

39.2.2.4 sim_pin_data0_tx_out

The sim_pin_data0_tx_out is the data transmitted from the SIM module to the SmartCard. In a 1-Wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

39.2.2.5 pin_sim_rcvd0_in

The pin_sim_rcvd0_in is the receive data coming from the SmartCard to the SIM module.

39.2.2.6 pin_sim_simpd0

The pin_sim_simpd0 is the SmartCard insertion detect.

39.2.2.7 sim_pin_sclk1

The sim_pin_sclk1 is an output from the chip to the smartcard. This signal is the clock that the SIM module provides for the smartcard. Typical frequencies are 1MHz to 5MHz. This clock is 372 times the data rate that is on pin sim_pin_data1_tx_out. There is no required timing relationship between this clock signal and any of the other data signals. This is because of the asynchronous nature of the protocol.

39.2.2.8 sim_pin_srst1

The sim_pin_srst1 is the reset signal from the SIM to the SmartCard.

39.2.2.9 sim_pin_sven1

The sim_pin_sven1 is the SmartCard power supply enable control signal.

39.2.2.10 sim_pin_data1_tx_out

The sim_pin_data1_tx_out is the data transmitted from the SIM module to the SmartCard. In a one wire mode of operating, this output port must be bidirectional with a pull-up resister off chip.

39.2.2.11 pin_sim_rcvd1_in

The pin_sim_rcvd1_in is the receive data coming from the SmartCard to the SIM module.

39.2.2.12 pin_sim_simpd1

The pin_sim_simpd1 is the SmartCard insertion detect.

39.3 Memory Map and Register Definition

Section 39.3.3, “Register Descriptions,” provides the detailed descriptions for all of the SIM registers.

39.3.1 Memory Map

Table 39-7 shows the SIM memory map.

Table 39-7. SIM Memory Map

Address	Register	Access	Reset Value	Section/Page
0x5002_4000 (PORT1_CNTL)	SIM Port1 Control register	R/W	0x0000_0000	39.3.3.1/39-13
0x5002_4004 (SETUP)	SIM Setup register	R/W	0x0000_0000	39.3.3.2/39-14
0x5002_4008 (PORT1_DETECT)	SIM Port 1 Detect register	R/W	0x0000_————	39.3.3.3/39-15
0x5002_400C (PORT1_XMT_BUF)	SIM Port1 Transmit Buffer register	R/W	0x0000_0000	39.3.3.4/39-16
0x5002_4010 (PORT1_RCV_BUF)	SIM Port 1 Receive Buffer register	R	0x0000_0000	39.3.3.5/39-17
0x5002_4014 (PORT0_CNTL)	SIM Port0 Control register	R/W	0x0000_0000	39.3.3.6/39-18
0x5002_4018 (CNTL)	SIM Control register	R/W	0x0000_0006	39.3.3.7/39-19
0x5002_401C (CLK_PRESCALER)	SIM Clock Prescaler register	R/W	0x0000_0002	39.3.3.8/39-21
0x5002_4020(RCV_THRESHOLD)	SIM Receive Threshold register	R/W	0x0000_0001	39.3.3.9/39-22
0x5002_4024 (ENABLE)	SIM Enable register	R/W	0x0000_0000	39.3.3.10/39-23
0x5002_4028 (XMT_STATUS)	SIM Transmit Status register	R/W	0x0000_00B8	39.3.3.11/39-24
0x5002_402C (RCV_STATUS)	SIM Receive Status register	R/W	0x0000_0040	39.3.3.12/39-26
0x5002_4030 (INT_MASK)	SIM Interrupt Mask register	R/W	0x0000_1FFF	39.3.3.13/39-28
0x5002_4034 (PORT0_XMT_BUF)	SIM Port0 Transmit Buffer register	R/W	0x0000_0000	39.3.3.14/39-30
0x5002_4038 (PORT0_RCV_BUF)	SIM Port0 Receive Buffer register	R	0x0000_0000	39.3.3.15/39-31
0x5002_403C (PORT0_DETECT)	SIM Port0 Detect register	R/W	0x0000_————	39.3.3.16/39-32
0x5002_4040 (DATA_FORMAT)	SIM Data Format register	R/W	0x0000_0000	39.3.3.17/39-33
0x5002_4044(XMT_THRESHOLD)	SIM Transmit Threshold register	R/W	0x0000_0000	39.3.3.18/39-34
0x5002_4048 (GUARD_CNTL)	SIM Transmit Guard Control register	R/W	0x0000_0000	39.3.3.19/39-35
0x5002_404C (OD_CONFIG)	SIM Open Drain Configuration Control register	R/W	0x0000_0000	39.3.3.20/39-36
0x5002_4050 (RESET_CNTL)	SIM Reset Control register	R/W	0x0000_0000	39.3.3.21/39-37
0x5002_4054 (CHAR_WAIT)	SIM Character Wait Time register	R/W	0x0000_FFFF	39.3.3.22/39-38
0x5002_4058 (GPCNT)	SIM General Purpose Counter register	R/W	0x0000_FFFF	39.3.3.23/39-39
0x5002_405C (DIVISOR)	SIM Divisor register	R/W	0x0000_00FF	39.3.3.24/39-40
0x5002_4060 (BWT)	SIM Block Wait Time register	R/W	0x0000_FFFF	39.3.3.25/39-40
0x5002_4064 (BGT)	SIM Block Guard Time register	R/W	0x0000_0000	39.3.3.26/39-41
0x5002_4068 (BWT_H)	SIM Block Wait Time register HIGH	R/W	0x0000_FFFF	39.3.3.27/39-42
0x5002_406C (XMT_FIFO_STAT)	SIM Transmit FIFO Status register	R	0x0000_0000	39.3.3.28/39-43
0x5002_4070 (RCV_FIFO_CNT)	SIM Receive FIFO Counter register	R	0x0000_0000	39.3.3.29/39-44

Table 39-7. SIM Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x5002_4074 (RCV_FIFO_WPTR)	SIM Receive FIFO Write Pointer register	R	0x0000_0000	39.3.3.30/39-44
0x5002_4078 (RCV_FIFO_RPTR)	SIM Receive FIFO Read Pointer register	R	0x0000_0000	39.3.3.31/39-45

39.3.2 Register Summary

Figure 39-2 shows the key to the register fields and Table 39-8 shows the register figure conventions.

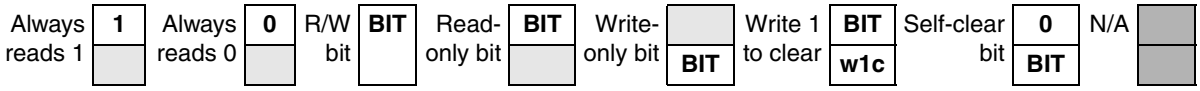


Figure 39-2. Key to Register Fields

Table 39-8. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 39-9 shows SIM register summary.

Table 39-9. SIM Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5002_4000 (PORT1_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0		3VOLT1	SCSP1	SCEN1	SRST1	STEN1	SVEN1	SAPD1
	W									SFPD1							
0x5002_4004 (SETUP)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SPS	AMODE
	W																
0x5002_4008 (PORT1_DETECT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS1	SPDP1	SDI1	SDIM1
	W															w1c	
0x5002_400C (PORT1_XMT_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PORT1_XMT[7:0]							
	W																
0x5002_4010 (PORT1_RCV_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	CWT	FE	PE	PORT1_RCV[7:0]							
	W																
0x5002_4014 (PORT0_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0		3VOLT0	SCSP0	SCEN0	SRST0	STEN0	SVEN0	SAPD0
	W									SFPD0							
0x5002_4018 (CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BWTEN	XMT_CRC_LRC	CRCEN	LRCEN	CWTEN	GPCNTSEL[1:0]		CLKBAUD_SEL[2:0]				SAMPLE12	ONACK	ANACK	ICM	0
	W																
0x5002_401C (CLK_PRESCALER)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	CLK_PRESCALER[7:0]							
	W																
0x5002_4020(RCV_THRESHOLD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	RTH[3:0]				RDT[8:0]								
	W																
0x5002_4024 (ENABLE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XMTEN	RCVEN
	W																

Table 39-9. SIM Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5002_4028 (XMT_STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	GPCNT	TDTF	TFO	TC	ETC	TFE	0	0	XTE
	W								w1c	w1c	n/a	w1c	w1c	w1c			w1c
0x5002_402C (RCV_STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	BGT	BWT	RTE	CWT	CRCOK	LRCOK	RDRF	RFD	0	0	0	OEF
	W					w1c	w1c	w1c	w1c								w1c
0x5002_4030 (INT_MASK)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
	W																
0x5002_4034 (PORT0_XMT_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PORT0_XMT[7:0]							
	W																
0x5002_4038 (PORT0_RCV_BUF)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	CWT	FE	PE	PORT0_RCV[7:0]							
	W																
0x5002_403C (PORT0_DETECT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS0	SPDP0	SDI0	SDIM0
	W															w1c	
0x5002_4040 (DATA_FORMAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IC
	W																
0x5002_4044(XMT_THRESHOLD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	XTH[3:0]				TDT3:0]			
	W																
0x5002_4048 (GUARD_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RCVR11	GETU[7:0]							
	W																
0x5002_404C (OD_CONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OD_P1	OD_P0
	W																

Table 39-9. SIM Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x5002_4050 (RESET_CNTL)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	DBUG	STOP	DOZE	KILL CLOCK		FLUSH XMT	FLUSH RCV
	W														SOFT RESET		
0x5002_4054 (CHAR_WAIT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CHARACTER WAIT TIME[15:0]															
	W																
0x5002_4058 (GPCNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	GENERAL PURPOSE COUNTER[15:0]															
	W																
0x5002_405C (DIVISOR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	DIVISOR[7:0]							
	W																
0x5002_4060 (BWT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BLOCK WAIT TIME[15:0]															
	W																
0x5002_4064 (BGT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BLOCK GUARD TIME[15:0]															
	W																
0x5002_4068 (BWT_H)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BLOCK WAIT TIME HIGH[15:0]															
	W																
0x5002_406C (XMT_FIFO_STAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	XMT_CNT[3:0]				XMT_WPTR[3:0]				XMT_RPTR[3:0]			
	W																
0x5002_4070 (RCV_FIFO_CNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RCV_CNT[8:0]								
	W																
0x5002_4074 (RCV_FIFO_WPTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RCV_WPTR[8:0]								
	W																

Table 39-9. SIM Register Summary (continued)

Name		31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x5002_4078 (RCV_FIFO_RPTR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	RCV_RPTR[8:0]								
	W																

39.3.3 Register Descriptions

This section contains the detailed descriptions for the SIM registers.

39.3.3.1 SIM Port1 Control Register (PORT1_CNTL)

See Figure 39-3 for illustration of valid bits in the SIM Port1 Control Register and Table 39-10 for description of the bit fields.

0x5002_4000 (PORT1_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	3VOLT1	SCSP1	SCEN1	SRST1	STEN1	SVEN1	SAPD1
W									SFPD1							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-3. SIM Port1 Control Register

Table 39-10. SIM Port1 Control Register Field Descriptions

Field	Description
31–8	Reserved
7 SFPD1	Auto Power Down port1. Writing a “1” to this location will start the auto power down sequence for port 1. This bit will autoclear. A read of this bit gives a “0”. 0 No effect 1 Start Auto Power down
6 3VOLT1	External one wire interface for SIM Card port1. Used to configure the Port 1 transmit pin as bi-directional. This allows the Port 1 Receive pin to be re-used as General Purpose I/O. This operation is restricted to bi-directional data SIM cards only. This bit should not be changed while the receiver or transmitter is enabled! 0 Port1 uses both RCV and XMT pins 1 Port1 XMT pin bidirectional. Port1 RCV PIN unused

Table 39-10. SIM Port1 Control Register Field Descriptions (continued)

Field	Description
5 SCSP1	SIM Card Clock Stop Polarity Port1. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN1. It will be forced low by hardware during the auto power down sequence. This forces the clock be a logic 0 when stopped by auto power down as required by ISO 7816 specification. 0 Clock is logic 0 when stopped by SCEN1 1 Clock is logic 1 when stopped by SCEN1
4 SCEN1	SIM card Clock Enable Port 1. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Enabled Port1 1 SIM Card Clock Disabled Port1
3 SRST1	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port1 inactive (default) 1 SIM Card reset Port1 active
2 STEN1	SIM card Transmit Enable Port 1. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port1 Transmit Data is forced to zero (default) 1 Port 1 Transmit Data controlled by SIM module
1 SVEN1	SIM card Vcc Enable Port 1. Used to control the state of the SVEN1 pin on SIM card port 1. The SVEN1 pin controls the SIM card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 1 disabled (default) 1 SIM card Voltage Port 1 enabled
0 SAPD1	SIM card Auto Power Down Port 1. Used to enable/disable the auto power down function for port 1. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 1 disabled (default) 1 Auto power down Port 1 enabled

39.3.3.2 SIM Setup Register (SETUP)

See Figure 39-4 for illustration of valid bits in the SIM Setup Register and Table 39-11 for description of the bit fields.

0x5002_4004 (SETUP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SPS	AMODE
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-4. SIM Setup Register

Table 39-11. SIM Setup Register Field Descriptions

Field	Description
31–2	Reserved
1 SPS	SIM card Port Select. Controls which port the SIM interface uses. **Note:** The AMODE bit must be zero when the SPS bit is set to 1. 0 Port 0 Enabled (default) 1 Port 1 Enabled
0 AMODE	Alternate SIM Card Mode enable. Enables an alternate SIM module to control SIM card Port 1. **Note:** The SPS bit must be 0 to give the alternate SIM module control. 0 Alternate Port Disabled (default) 1 Alternate Port Enabled

39.3.3.3 SIM Port1 Detect Register (PORT1_DETECT)

See Figure 39-5 for illustration of valid bits in the SIM Port1 Detect Register and Table 39-12 for description of the bit fields.

0x5002_4008 (PORT1_DETECT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS1	SPDP1	SDI1	SDIM1
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	—	0	1

Figure 39-5. SIM Port 1 Detect Register

Table 39-12. SIM Port 1 Detect Register Field Descriptions

Field	Description
31–4	Reserved
3 SPDS1	SIM Presence Detect Select Port 1. Controls which edge of the SIMPD1 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD1 Input (default) 1 Rising edge of SIMPD1 Input
2 SPDP1	SIMPD1 input pin status. This bit reflects the state of the SIMPD1 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD1 pin itself. 0 SIMPD1 pin is logic low 1 SIMPD1 pin is logic high

Table 39-12. SIM Port 1 Detect Register Field Descriptions (continued)

Field	Description
1 SDI1	SIM Detect Interrupt Flag Port 1. Status flag to indicate the insertion or removal of a SIM card has been detected on port 1. Can create an interrupt to the core if SDIM1 is low. Write a “1” to this bit to clear. 0 No insertion or removal of SIM card detected on Port 1(default) 1 Insertion or removal of SIM card detected on Port 1
0 SDIM1	SIM Detect Interrupt Mask Port 1. Interrupt mask for the SDI1 interrupt flag. 0 SDI1 enabled 1 SDI1 masked (default)
Summary: SPDS1 determines which edge transition of the SIMPD1 pin is used for SIM card presence detection. Presence detection can be used to determine if the card has been inserted or removed. The occurrence of the SIMPD1 edge specified by SPDS1 will cause the following: SDI1 to be set; if the SDIM1 mask is clear, an interrupt on SIMIRQ_N; and if SAPD1 in the PORT1_CNTL register is set, an auto power down sequence to begin. If SIM card insertion is expected, SAPD1 can be set low to avoid the auto power down sequence. There is no auto power up sequence. The bit SPDP1 can be used to determine the current state of the SIMPD1 pin.	

39.3.3.4 SIM Port1 Transmit Buffer Register (PORT1_XMT_BUF)

See Figure 39-6 for illustration of valid bits in the SIM Port1 Transmit Buffer Register and Table 39-13 for description of the bit fields.

0x5002_400C (PORT1_XMT_BUF) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PORT1_XMT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-6. SIM Port1 Transmit Buffer Register

Table 39-13. SIM Port1 Transmit Buffer Register Field Descriptions

Field	Description
31–8	Reserved
7–0 PORT1_XMT	Port1 Transmit Buffer. Write to the next available location in the transmit buffer. Writes to this register are ignored when the SPS bit is zero.

39.3.3.5 SIM Port1 Receive Buffer Register (PORT1_RCV_BUF)

See Figure 39-7 for illustration of valid bits in the SIM Port1 Receive Buffer Register and Table 39-14 for description of the bit fields.

0x5002_4010 (PORT1_RCV_BUF) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CWT	FE	PE	PORT1_RCV[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-7. SIM Port 1 Receive Buffer Register

Table 39-14. SIM Port 1 Receive Buffer Register Field Descriptions

Field	Description
31–11	Reserved
10 CWT	Port1 CWT flag. The CWT indicates that this byte was late. It is not necessary to clear the byte since it is overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	Port1 Frame Error flag. The PORT1 FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the PORT1 RCV field. The PORT1 FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	Port1 Parity Error flag. The PORT1 FE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the PORT1 RCV field. The PORT1 PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
7–0 PORT1_RCV	Port1 Receive buffer. Read from the next location in the receive buffer. Reads from this register return zero when the SPS bit is zero.

39.3.3.6 SIM Port0 Control Register (PORT0_CNTL)

See Figure 39-8 for illustration of valid bits in the SIM Port0 Control Register and Table 39-15 for description of the bit fields.

0x5002_4014 (PORT0_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0		3VOLT0	SCSP0	SCEN0	SRST0	STEN0	SVEN0	SAPD0
W									SFPD0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-8. SIM Port0 Control Register

Table 39-15. SIM Port0 Control Register Field Descriptions

Field	Description
31–8	Reserved
7 SFPD0	Auto Power Down port0. Writing a “1” to this location will start the auto power down sequence for port 0. This bit will autoclear.A read of this bit will give a “0”. 0 No effect 1 Start Auto Power down
6 3VOLT0	External one wire interface for SIM Card port0. Used to configure the Port 0 transmit pin as bi-directional. This allows the Port 0 Receive pin to be re-used as General Purpose I/O. This operation is restricted to bi-directional data SIM cards only. This bit should not be changed while the receiver or transmitter is enabled! 0 Port0 uses both RCV and XMT pins 1 Port0 XMT pin bidirectional. Port0 RCV pin unused
5 SCSP0	SIM Card Clock Stop Polarity port0. Used to control the polarity of the idle SIM clock when the clock is disabled by SCEN0. It will be forced low by hardware during the auto power down sequence. This forces the clock to be a logic 0 when stopped by auto power down as required by ISO 7816 specification. 0 Clock is logic 0 when stopped by SCEN0 1 Clock is logic 1 when stopped by SCEN0
4 SCEN0	SIM card Clock Enable Port 0. Used to enable/disable the clock to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 SIM Card Clock Disabled Port 0 1 SIM Card Clock Enabled Port 0
3 SRST0	SIM card Reset. Used to control state of reset line to the SIM card. It can be forced low by hardware during the auto power down sequence. SIM card reset signals are active low. 0 SIM Card reset Port0 inactive (default) 1 SIM Card reset Port0 active

Table 39-15. SIM Port0 Control Register Field Descriptions (continued)

Field	Description
2 STEN0	SIM card Transmit Enable Port 0. Used to enable/disable the XMT data to the SIM card. It can be forced low by hardware during the auto power down sequence. 0 Port0 Transmit Data is forced to zero (default) 1 Port 0 Transmit Data controlled by SIM module
1 SVEN0	SIM card Vcc Enable Port 0. Used to control the state of the SVEN0 pin on SIM card port 0. The SVEN0 pin controls the SIM card Vcc enable in the power management chip. It can be forced low by hardware during the auto power down sequence. 0 SIM card Voltage Port 0 disabled (default) 1 SIM card Voltage Port 0 enabled
0 SAPD0	SIM card Auto Power Down Port 0. Used to enable/disable the auto power down function for port 0. It will be forced low at the end of the auto power down sequence. 0 Auto power down Port 0 disabled (default) 1 Auto power down Port 0 enabled

39.3.3.7 SIM Control Register (CNTL)

See Figure 39-9 for illustration of valid bits in the SIM Control Register and Table 39-16 for description of the bit fields.

0x5002_4018 (CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWTEN	XMT_CRC_LRC	CRCEN	LRCEN	CWTEN	GPCNT_CLK_SEL[1:0]			BAUD_SEL[2:0]		0	SAMPLE12	ONACK	ANACK	ICM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 39-9. SIM Control Register

Table 39-16. SIM Control Register Field Descriptions

Field	Description
31–16	Reserved
15 BWTEN	Block wait time enable. Writing a “1” to this bit will enable the BWT and BGT functions. The BWT and BGT functions can then be individually selected using the interrupt mask. 0 Disable BWT, BGT 1 Enable BWT, BGT

Table 39-16. SIM Control Register Field Descriptions (continued)

Field	Description
14 XMT_CRC_LRC	Transmit CRC or LRC. This bit specifies whether or not to transmit the redundancy checking data at the end of a transmission (that is, when the FIFO becomes empty). 0 No Redundancy check info transmitted (default) 1 Transmit LRC or CRC info when FIFO empties (whichever is enabled)
13 CRCEN	CRC Enable. This bit enables the calculation of the 16-bit CRC value for both receiver and transmitter. The result of the calculation is continuously compared to the expected remainder and reflected in the CRCOK bit in the RCV_STAT register. Clearing this bit resets the current CRC residual value in the SIM hardware. 0 16-bit Cyclic Redundancy Checking disabled (default) 1 16-bit Cyclic Redundancy Checking enabled
12 LRCEN	LRC Enable. This bit enables the calculation of the 8-bit LRC value for both receiver and transmitter. The result of the calculation is continuously compared to zero and reflected in the LRCOK bit in the RCV_STAT register. Clearing this bit resets the current LRC value in the SIM hardware. 0 8-bit Linear Redundancy Checking disabled (default) 1 8-bit Linear Redundancy Checking enabled
11 CWTEN	Character Wait Time Counter Enable. Enables the character wait time counter. Clearing this bit resets the counter to zero. 0 Character Wait time Counter off (default) 1 Character Wait time counter on
10–9 GPCNT_CLK_SEL[1:0]	General Purpose Counter Clock Select. Selects which clock source is used by SIM Module general purpose counter. The only way to reset the counter is to set these bits to zero. The counter will begin counting as soon as the clock input is selected and the clocks are enabled. These input clocks are enabled through other register bits of the SIM module (KILL_CLOCK, RCV_EN, and XMT_EN respectively). 00 Disabled / Reset 01 Card Clock 10 Receive Clock 11 ETU Clock (transmit clock)
8–6 BAUD_SEL[2:0]	SIM Baud Rate Select. Selects the asynchronous baud rate divisor of the clock When set to “111”, the divisor is set to the value programmed in the DIVISOR register. This allows for more flexible baud rate determination. 000 31 (372/1 Fi/Di) 001 32 (512/2 Fi/Di) 010 16 (512/4 Fi/Di) 011 8 (512/8 Fi/Di) 100 4 (512/16 Fi/Di) 101 2 (512/32 Fi/Di) 110 1 (512/64 Fi/Di) 111 DIVISOR Reg
5	Reserved
4 SAMPLE12	Sample12. Set the third stage divider. This sets the corresponding sample rate which is the number of times a bit being received is sampled. 0 divide by 8(default) 1 divide by 12
3 ONACK	Overrun NACK Enable. Enables overrun NACK generation. 0 NACK generation on overrun is disabled (default) 1 NACK generation on overrun is enabled

Table 39-16. SIM Control Register Field Descriptions (continued)

Field	Description
2 ANACK	Automatic NACK Enable. Enables nack generation for parity errors or invalid initial characters when in ICM mode. 0 NACK generation on errors disabled 1 NACK generation on errors enabled (default)
1 ICM	Initial Character Mode. Enables initial character mode. Will be automatically cleared by hardware once a valid initial character is received. 0 Initial Character Mode disabled 1 Initial Character Mode enabled (default)
0	Reserved

39.3.3.8 SIM Clock Prescaler Register (CLK_PRESCALER)

See Figure 39-10 for illustration of valid bits in the SIM Clock Prescaler Register and Table 39-17 for description of the bit fields.

0x5002_401C (CLK_PRESCALER) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	CLK_PRESCALER[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 39-10. SIM Clock Prescaler Register

Table 39-17. SIM Clock Prescaler Register Field Descriptions

Field	Description
31–8	Reserved
7–0 CLK_PRESCALER	Clock prescaler divisor register. The value written to this register will determine the first stage divider setting. If the ipg_clk is 66Mhz, a typical setting would be 0x0e. This would set the SIM card clock to 66Mhz/14 = 4.7Mhz. The duty cycle of divided clock will be between 45% and 55% according to ISO7816 requirement. For the odd divider factor (2K+1), the duty cycle will be K/(2K+1) and (K+1)/(2k+1). So for 0x03, the duty cycle is 33%-66%. For 0x05, the duty cycle is 40%-60%. For 0x07, the duty cycle is 43%-57%, and for 0x09, the duty cycle is 44%-56%. For all other values, the clock duty cycle can meet the ISO7816 requirement. 0x00~0x02 ipg_clk/ 2 0x03 ipg_clk / 3 0x04 ipg_clk / 4 0x05 ipg_clk / 5 0x06 ipg_clk / 6 ... 0xFD ipg_clk / 253 0xFE ipg_clk / 254 0xFF ipg_clk / 255

39.3.3.9 SIM Receive Threshold Register (RCV_THRESHOLD)

See Figure 39-11 for illustration of valid bits in the SIM Receive Threshold Register and Table 39-18 for description of the bit fields.

0x5002_4020(RCV_THRESHOLD) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	RTH[3:0]					RDT[8:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 39-11. SIM Receive Threshold Register

Table 39-18. SIM Receive Threshold Register Field Descriptions

Field	Description
31–13	Reserved
12–9 RTH[3:0]	Receive Nack Threshold. Used to specify the number of consecutive NACK's transmitted by the SIM module, for a given character, before the receive threshold error (RTE) flag is triggered. A value of 0 indicates that RTE is never set. When a valid character is received by the SIM, the internal counter keeping track of the NACK count resets to zero for the subsequent byte being received. If the ANACK bit is clear in the CNTL register, RTH has no effect.
8–0 RDT[8:0]	Receive Data Threshold. Determines the number of unread bytes that must exist in the FIFO to trigger the receive data register full (RDRF) interrupt flag. If the number of unread bytes in the receive FIFO is greater than or equal to the value in RDT, the RDRF flag in the RCV_STATUS register will be set. A value of zero indicates that there must be 285 unread bytes in the FIFO to trigger RDRF. The RDT value can be altered at any time, and the RDRF flag will be updated accordingly. If a value of more than 285 is entered in this register, the register value will remain 285.

39.3.3.10 SIM Enable Register (ENABLE)

See Figure 39-12 for illustration of valid bits in the SIM Enable Register and Table 39-19 for description of the bit fields.

0x5002_4024 (ENABLE) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XMTEN	RCVEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-12. SIM Enable Register

Table 39-19. SIM Enable Register Field Descriptions

Field	Description
31–2	Reserved

Table 39-19. SIM Enable Register Field Descriptions (continued)

Field	Description
1 XMT_EN	SIM Transmit Enable. Used to enable/disable the SIM transmit state machine. When the SIM is being used to receive data, XMT_EN should be deasserted. This bit also enables the xmt_clk and rcv_clk inputs to the General Purpose Counter. **Note:** Setting this bit (transition from 0 to 1) will reset the CRC and LRC values. 0 SIM Transmitter disabled (default) 1 SIM Transmitter enabled
0 RCV_EN	SIM Receiver Enable. Used to enable/disable the SIM receive state machine. The RCV_EN bit should be left high whenever the SIM module is in use. The SIM module has an automatic receive mode operation that disables the reception of characters when the transmitter is operational. Once the transmitter has completed sending the last character, the receiver is automatically enabled. This bit also enables the RCV_CLK input to the General Purpose Counter. 0 SIM Receiver disabled (default) 1 SIM Receiver enabled

39.3.3.11 SIM Transmit Status Register (XMT_STATUS)

See Figure 39-13 for illustration of valid bits in the SIM Transmit Status Register and Table 39-20 for description of the bit fields.

0x5002_4028 (XMT_STATUS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	GPCNT	TDTF	TFO	TC	ETC	TFE	0	0	XTE
W								w1c	w1c	n/a	w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0

Figure 39-13. SIM Transmit Status Register

Table 39-20. SIM Transmit Status Register Field Descriptions

Field	Description
31–9	Reserved
8 GPCNT	General purpose Counter Flag. Used to indicate when the General purpose counter has reached the value in the GPCNT register. 0 GPCNT time not reached, or bit has been cleared. (default). 1 General Purpose counter has reached the GPCNT value.

Table 39-20. SIM Transmit Status Register Field Descriptions (continued)

Field	Description
7 TDTF	Transmit Data Threshold Flag. Used to indicate when the number of bytes in the transmit FIFO is less than or equal to the value programmed in the TDT[3:0] bits in the XMT_THRESHOLD register. 0 Number of bytes in FIFO is greater than TDT[3:0], or bit has been cleared. 1 Number of bytes in FIFO is less than or equal to TDT[3:0] (default)
6 TFO	Transmit FIFO Overfill Error. Used to indicate when the Transmit FIFO has been written with more than 16 bytes. The TFO bit can only be cleared by setting the FLUSH_XMT or SOFT_RESET bit in the RESET_CNTL register. 0 No transmit FIFO overfill error has occurred (default). 1 A Transmit FIFO overfill error has occurred.
5 TC	Transmit Complete. Used to indicate whether the SIM transmitter is ready for a new transmission. The TC flag becomes set after the guard time has expired for the last byte in the transmit FIFO. The TC flag will create an interrupt if TCIM in the INT_MASK register is low. The TC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
4 ETC	Early Transmit Complete. Used to indicate that the SIM transmitter has finished sending the current byte and the transmit FIFO is empty. This bit differs from the TC bit in that it is set before the guard time of the last byte has elapsed. The ETC flag will create an interrupt if ETCIM in the INT_MASK register is low. The ETC bit is a write-one-to-clear bit. 0 Transmit pending or in progress 1 Transmit complete (default)
3 TFE	Transmit FIFO Empty. Used to indicate that the SIM transmit FIFO has emptied. This bit will be set when the last byte in the transmit FIFO has been transferred to the SIM transmitter shift register. The TFE flag will create an interrupt if TFEIM in the INT_MASK register is low. The TFE bit is a write-one-to-clear bit. 0 Transmit FIFO is not empty 1 Transmit FIFO is empty (default)
2–1	Reserved
0 XTE	Transmit NACK Threshold Error. Used to indicate the transmit NACK threshold has been reached. When XTE is high, no further transmissions will be done until the XTE flag is cleared. Any data transmissions still pending in the transmit FIFO will be aborted, and the TC, ETC, and TFE flags will be set. The XTE flag will create an interrupt if XTM in the IMT_MASK register is low. The XTE bit is a write-one-to-clear bit. 0 Transmit NACK threshold has not been reached (default) 1 Transmit NACK threshold reached; transmitter frozen

39.3.3.12 SIM Receive Status Register (RCV_STATUS)

See Figure 39-14 for illustration of valid bits in the SIM Receive Status Register and Table 39-21 for description of the bit fields.

0x5002_402C (RCV_STATUS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	BGT	BWT	RTE	CWT	CRC OK	LRCO K	RDRF	RFD	0	0	0	OEF
W					w1c	w1c	w1c	w1c								w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 39-14. SIM Receive Status Register

Table 39-21. SIM Receive Status Register Field Descriptions

Field	Description
31–12	Reserved
11 BGT	Block guard time error flag. Used to indicate if the block guard time was too small. The threshold is set by the block guard time register. 0 Block guard time was sufficient. 1 Block guard time was too small.
10 BWT	Block wait time error flag. Used to indicate if the block wait time has been exceeded. The threshold is set by the block wait time registers. 0 Block wait time not exceeded. 1 Block wait time was exceeded.
9 RTE	Receive NACK threshold error flag. Used to indicate whether the number of consecutive NACK's generated by the SIM module in response to receive parity errors, for the byte being received, equals the value programmed in the RTH[3:0] in the RCV_THRESHOLD register. This bit would never be set unless the ANACK bit is set in the CNTL register. The SAPDx bit in the PORTx_CNTL resister must be set to enable the threshold error to trigger the auto power down sequence. RTE is a write once to clear bit. Clearing this bit also resets the internal counter for consecutive NACK's being transmitted for a given byte. 0 Number of NACKs generated by the receiver is less than the value programmed in RTH[3:0] 1 Number of NACKs generated by the receiver is equal to the value programmed in RTH[3:0]
8 CWT	Character Wait Time Counter Flag. Used to indicate when the time between received characters is equal to or greater than the value programmed in the CHAR_WAIT register. 0 No CWT violation has occurred (default). 1 Time between two consecutive characters exceeded the value in CHAR_WAIT.

Table 39-21. SIM Receive Status Register Field Descriptions (continued)

Field	Description
7 CRCOK	Cyclic Redundancy Check Okay flag. Used to indicate when the calculated 16-bit CRC value matches the expected value for the current input data stream. The value is calculated across all received characters from the point the CRCEN bit is set in the CNTL register. The current CRC residual can be reset by three mechanisms: • Clear CRCEN bit in CNTL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current CRC value does not match remainder. 1 Current calculated CRC value matches the expected result.
6 LRCOK	Linear Redundancy Check Okay flag. Used to indicate when the calculated 8-bit LRC value is zero value for the current input data stream. The value is calculated across all received characters from the point the LRCEN bit is set in the CNTL register. The current LRC residual can be reset by three mechanisms: • Clear LRCEN bit in CNTL register • Set XMT_EN bit in ENABLE register • Automatically by hardware when ETC flag is set at the end of a transmission. 0 Current LRC value does not match remainder. 1 Current calculated LRC value matches the expected result (that is, zero).
5 RDRF	Receive Data Register Full. Used to indicate whether the SIM receive FIFO has reached the threshold level set by RDT[8:0] in the RCV_THRESHOLD register. The RDRF flag will be set any time the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT[8:0]. The flag can be cleared by reading enough bytes out of the receive FIFO so as to bring the number of bytes left in the FIFO below the RDT[8:0] level. Another way to clear the flag is to set the RDT[8:0] level higher than the number of unread bytes currently in the FIFO. The RDRF flag will create an interrupt if the RIM bit in the INT_MASK register is cleared. 0 Number of unread bytes in receive buffer < value set by RDT[8:0] (default). 1 Number of unread bytes in receive buffer >= value set by RDT[8:0].
4 RFD	Receive FIFO has unread Data. Used to indicate that there is at least one unread byte in the receive data FIFO. Can only be cleared by reading all bytes out of the receive FIFO. The RFD bit cannot be used to create an interrupt. Normally, the SIM triggers the interrupt with RDRF and software uses RFD to read all of the bytes out of the receive FIFO. 0 There are no unread bytes in the receive FIFO (default). 1 There is at least one unread byte in the receiver FIFO.
3–1	Reserved
0 OEF	Overrun Error Flag. Used to indicate that the SIM was unable to store received data due to already having 285 unread bytes in the FIFO. It does not necessarily indicate that data has been lost. If the ONACK control bit in the CNTL register is set, there will be a NACK pulse generated on bytes that would otherwise cause a loss of data due to a full FIFO. These bytes should be retransmitted by the SIM card which implies that no data has actually been lost. In this case, the OEF flag is just an indicator that this situation has occurred which may be helpful in system debug. For the case where ONACK is not set, a set OEF flag does indicate a loss of data since all bytes received with the OEF flag set will indeed be lost (including the byte that caused the bit to be set). The OEF flag will cause an interrupt if the OIM bit in the INT_MASK register. The OEF flag is a write-one-to-clear bit. 0 No overrun error has occurred (default). 1 A byte was received when the received FIFO was already full.

39.3.3.13 SIM Interrupt Mask Register (INT_MASK)

See Figure 39-15 for illustration of valid bits in the SIM Interrupt Mask Register and Table 39-22 for description of the bit fields.

0x5002_4030 (INT_MASK) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	BGTM	BWTM	RTM	CWTM	GPCNTM	TDTFM	TFOM	XTM	TFEIM	ETCIM	OIM	TCIM	RIM
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 39-15. SIM Interrupt Mask Register

Table 39-22. SIM Interrupt Mask Register Field Descriptions

Field	Description
31–13	Reserved
12 BGTM	Block guard time interrupt mask. Used to enable/disable the ability of the BGT flag in the RCV_STATUS register to generate SIM interrupts. 0 BGT interrupt enabled 1 BGT interrupt masked (default)
11 BWTM	Block wait time interrupt mask. Used to enable/disable the ability of the BWT flag in the RCV_STATUS register to generate SIM interrupts. 0 BWT interrupt enabled 1 BWT interrupt masked (default)
10 RTM	Receive Nack threshold interrupt mask. Used to enable/disable the ability of the RTE flag in the RCV_STATUS register to generate SIM interrupts. 0 RTE interrupt enabled 1 RTE interrupt masked (default)
9 CWTM	Character Wait Time Interrupt Mask. Used to enable/disable the ability of the CWT flag in the RCV_STATUS register to generate SIM interrupts. 0 CWT interrupt enabled 1 CWT interrupt masked (default)
8 GPCNTM	General Purpose Counter Interrupt Mask. Used to enable/disable the ability of the GPCNT flag in the XMT_STATUS register to generate SIM interrupts. 0 GPCNT interrupt enabled 1 GPCNT interrupt masked (default)
7 TDTFM	Transmit Data Threshold Interrupt Mask. Used to enable/disable the ability of the TDTF flag in the XMT_STATUS register to generate SIM interrupts. 0 TDTF interrupt enabled 1 TDTF interrupt masked (default)

Table 39-22. SIM Interrupt Mask Register Field Descriptions (continued)

Field	Description
6 TFOM	Transmit FIFO Overfill Error Interrupt Mask. Used to enable/disable the ability of the TFO flag in the XMT_STATUS register to generate SIM interrupts. 0 TFO interrupt enabled 1 TFO interrupt masked (default)
5 XTM	Transmit Threshold Interrupt Mask. Used to enable/disable the ability of the XTE flag in the XMT_STATUS register to generate SIM interrupts 0 XTE interrupt enabled 1 XTE interrupt masked (default)
4 TFEIM	Transmit FIFO Empty Interrupt Mask. Used to enable/disable the ability of the TFE flag in the XMT_STATUS register to generate SIM interrupts. 0 TFE interrupt enabled 1 TFE interrupt masked (default)
3 ETCIM	Early Transmit Complete Interrupt Mask. Used to enable/disable the ability of the ETC flag in the XMT_STATUS register to generate SIM interrupts. 0 ETC interrupt enabled 1 ETC interrupt masked (default)
2 OIM	Overrun Interrupt Mask. Used to enable/disable the ability of the OEF flag in the RCV_STATUS register to generate SIM interrupts. 0 OEF interrupt enabled 1 OEF interrupt masked (default)
1 TCIM	Transmit Complete Interrupt Mask. Used to enable/disable the ability of the TC flag in the XMT_STATUS register to generate SIM interrupts. 0 TC interrupt enabled 1 TC interrupt masked (default)
0 RIM	Receive Interrupt Mask. Used to enable/disable the ability of the RDRF flag in the RCV_STATUS register to generate SIM interrupts. 0 RDRF interrupt enabled 1 RDRF interrupt masked (default)

39.3.3.14 SIM Port0 Transmit Buffer Register (PORT0_XMT_BUF)

See Figure 39-16 for illustration of valid bits in the SIM Port0 Transmit Buffer Register and Table 39-23 for description of the bit fields.

0x5002_4034 (PORT0_XMT_BUF) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PORT0_XMT[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-16. SIM Port0 Transmit Buffer Register

Table 39-23. SIM Port0 Transmit Buffer Register Field Descriptions

Field	Description
31–8	Reserved
7–0 PORT0_XMT	Port0 Transmit Buffer. Register to write when transmitting using port 0. To use this register to initiate transmissions to the SIM, set SPS = 0. When SPS = 1, a write to this register has no effect. A read of this register will produce the last thing written, or 0x00 if when SPS = 1. **Note:** Writing more data to the transmit FIFO than it can hold (16 bytes) will cause a transmit FIFO Overfill error.

39.3.3.15 SIM Port0 Receive Buffer Register (PORT0_RCV_BUF)

See Figure 39-17 for illustration of valid bits in the SIM Port0 Receive Buffer Register and Table 39-24 for description of the bit fields.

0x5002_4038 (PORT0_RCV_BUF) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	CWT	FE	PE	PORT0_RCV[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-17. SIM Port0 Receive Buffer Register

Table 39-24. SIM Port0 Receive Buffer Register Field Descriptions

Field	Description
31–11	Reserved
10 CWT	Port0 CWT flag. The CWT indicates that this byte was late. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte was on time 1 Byte was late
9 FE	Port0 Frame Error flag. The PORT0 FE flag indicates whether a frame error was detected during the reception of the corresponding byte read in the PORT0 RCV field. The PORT1 FE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. 0 Byte contains no framing error (default) 1 Byte contains a framing error
8 PE	Port0 Parity Error flag. The PORT0 PE flag indicates whether a parity error was detected during the reception of the corresponding byte read in the PORT0 RCV field. The PORT0 PE flag cannot create an interrupt. It need not be cleared since it will be overwritten by the next byte received into that location of the FIFO. A parity error can create a NACK pulse. 0 Byte contains no parity error (default) 1 Byte contains a parity error
7–0 PORT0_RCV	Port0 Receive buffer. Read from the next location in the receive buffer. Reads from this register return zero when the SPS bit is 1.

39.3.3.16 SIM Port0 Detect Register (PORT0_DETECT)

See Figure 39-18 for illustration of valid bits in the SIM Port0 Detect Register and Table 39-25 for description of the bit fields.

0x5002_403C (PORT0_DETECT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	SPDS0	SPDP0	SDI0	SDIM0
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	—	0	1

Figure 39-18. SIM Port0 Detect Register

Table 39-25. SIM Port0 Detect Register Field Descriptions

Field	Description
31–4	Reserved
3 SPDS0	SIM Presence Detect Select Port 0. Controls which edge of the SIMPD0 pin is used to detect the presence of the SIM card. 0 Falling edge of SIMPD0 Input (default) 1 Rising edge of SIMPD0 Input
2 SPDP0	SIMPD0 input pin status. This bit reflects the state of the SIMPD0 pin. It is not a latched register bit, but instead a synchronized version of the state of the SIMPD0 pin itself. 0 SIMPD0 pin is logic low 1 SIMPD0 pin is logic high
1 SDI0	SIM Detect Interrupt flag Port 0. Status flag to indicate the insertion or removal of a SIM card has been detected on port 0. Can create an interrupt to the core if SDIM0 is low. Write a “1” to this bit to clear. 0 No insertion or removal of SIM card detected on Port 0 (default) 1 Insertion or removal of SIM card detected on Port 0
0 SDIM0	SIM Detect Interrupt Mask Port 0. Interrupt mask for the SDI0 interrupt flag. 0 SDI0 enabled 1 SDI0 masked (default)

39.3.3.17 SIM Data Format Register (DATA_FORMAT)

See Figure 39-19 for illustration of valid bits in the SIM Data Format Register and Table 39-26 for description of the bit fields.

0x5002_4040 (DATA_FORMAT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-19. SIM Data Format Register

Table 39-26. SIM Data Format Register Field Descriptions

Field	Description
31–1	Reserved
0 IC	Inverse Convention. Used to configure the SIM to use either inverse convention or direct convention for its data format. The IC bit can be controlled by software, but it is normally set by hardware as a result of the interpretation of the initial character when in ICM mode. 0 Direction convention transfers enabled (default). 1 Inverse convention transfers enabled.

39.3.3.18 SIM Transmit Threshold Register (XMT_THRESHOLD)

See Figure 39-20 for illustration of valid bits in the SIM Transmit Threshold Register and Table 39-27 for description of the bit fields.

0x5002_4044(XMT_THRESHOLD) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	XTH[3:0]				TDT[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-20. SIM Transmit Threshold Register

Table 39-27. SIM Transmit Threshold Register Field Descriptions

Field	Description
31–8	Reserved
7–4 XTH[3:0]	Transmit NACK Threshold. Used to set the NACK threshold for the transmitter. Once the threshold number set by XTH has been reached for the current byte being transmitted, the error flag XTE in the XMT_STATUS register will be set. Setting of XTE causes the remaining transmissions queued in the transmit FIFO to be aborted and no more transmissions to occur until software clears XTE. To trigger XTE, a given byte being transmitted must reach the XTH threshold itself. Transmit NACKs accumulated on one byte are not carried over to the next. 0x0XTE will never be set; retransmission after NACK reception is disabled. 0x1XTE will be set after 1 nack is received; 0 retransmissions occurs. 0x2XTE will be set after 2 nacks are received; at most 1 retransmission occurs. 0x3XTE will be set after 3 nacks are received; at most 2 retransmissions occurs. ... ... 0xXXTE will be set after X nacks are received; at most (X - 1) retransmissions occurs. ... ... 0xFXTE will be set after 15 nacks are received; at most 14 retransmissions occurs.
3–0 TDT	Transmit Data Threshold. Used to set the threshold value for the Transmit FIFO at which the TDTF bit in the XMT_STATUS register will be set. When the number of bytes in the Transmit FIFO is less than or equal to TDT[3:0], TDTF will be set.

39.3.3.19 SIM Transmit Guard Control Register (GUARD_CNTL)

See Figure 39-21 for illustration of valid bits in the SIM Transmit Guard Control Register and Table 39-28 for description of the bit fields.

0x5002_4048 (GUARD_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RCV R11	GETU[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-21. SIM Transmit Guard Control Register

Table 39-28. SIM Transmit Guard Control Register Field Descriptions

Field	Description
31–9	Reserved
8 RCVR11	Receiver use 11 ETUs. Used to configure the SIM module receiver for 11 ETU operation (that is, 1 Stop bit). This bit is provided for support of T=1 cards. 0 Receiver configured for 12 ETU operation (default) 1 Receiver configured for 11 ETU operation
7–0 GETU	Transmit Guard ETUs. Used to control the number of additional Elementary Time Units (ETUs) inserted between bytes transmitted by the SIM transmitter. An ETU is equivalent to one bit time at the given baud rate (for example, the length of a START bit). The guard time has no effect on the SIM receiver. A value of 0x00 inserts no additional ETUs, while a value of 0xFE inserts 254 additional ETUs. A value of 0xFF subtracts one ETU by reducing the number of STOP bits from two to one.

39.3.3.20 SIM Open Drain Configuration Control Register (OD_CONFIG)

See Figure 39-22 for illustration of valid bits in the SIM Open Drain Configuration Control Register and Table 39-29 for description of the bit fields.

0x5002_404C (OD_CONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OD_PI	OD_P0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-22. SIM Open Drain Configuration Control Register

Table 39-29. SIM Open Drain Configuration Control Register Field Descriptions

Field	Description
31–2	Reserved
1 OD_P1	Open Drain control for Port 1. Used to control whether the XMT data line on port 1 is open-drain. If AMODE bit in SETUP register is set, this bit will have no effect. 0 XMT pin on port 1 is push-pull (default). 1 XMT pin on port 1 is open-drain.
0 OD_P0	Open Drain control for Port 0. Used to control whether the XMT data line on port 0 is open-drain. 0 XMT pin on port 0 is push-pull (default). 1 XMT pin on port 0 is open-drain.

39.3.3.21 SIM Reset Control Register (RESET_CNTL)

See Figure 39-23 for illustration of valid bits in the SIM Reset Control Register and Table 39-30 for description of the bit fields.

0x5002_4050 (RESET_CNTL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	DBUG	STOP	DOZE	KILL CLOCK	0	FLUSH XMT	FLUSH RCV
W														SOFT RST		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-23. SIM Reset Control Register

Table 39-30. SIM Reset Control Register Field Descriptions

Field	Description
31-7	Reserved
6 DBUG	DEBUG. Used to configure the operation of the SIM module when a debug event occurs. When set, a debug event (generated by such causes as OnCE module and external control) will make the receive FIFO read pointer to be frozen. 0 Debug event has no affect on SIM module (default). 1 Debug event prohibits read pointer changes for receive FIFO.
5 STOP	STOP. Used to configure the operation of the SIM module when a processor STOP instruction is executed. This bit is added to provide support for SIM cards that do not allow the SIM Card clock to be stopped while power is applied. See Figure 39-39. 0 STOP instruction shuts down all SIM clocks (default). 1 STOP instruction shuts down all clocks except for the BAUD_CLK (clock provided to SIM Card).
4 DOZE	DOZE. Used to configure the operation of the SIM module when a processor DOZE instruction is executed. See Figure 39-39. 0 DOZE instruction has no effect on SIM module (default). 1 DOZE instruction will cause SIM module to gate SIM clocks when the transmit FIFO is empty.
3 KILL_CLOCK	Kill SIM Clock. Used to enable/disable the SIM clock input to the SIM module. This bit will gate all SIM clocks including the SIM card clock regardless of the state of the STOP bit described above. 0 SIM input clock enabled (default). 1 SIM input clock disabled.
2 SOFT_RST	Software Reset. Used to reset the entire SIM module. This acts the same as a hardware reset for the SIM module. This bit is self-clearing. **Note:** Software should allow a minimum of 4 reference clock cycles (CKIH) before attempting to access the SIM module after a software reset. 0 SIM Normal operation (default). 1 SIM held in Reset.

Table 39-30. SIM Reset Control Register Field Descriptions (continued)

Field	Description
1 FLUSH_XMT	Flush Transmitter. This bit operates as a SIM transmitter reset. The receive portion of the SIM module is not affected. The software must clear this bit before the SIM transmitter can operate. 0 SIM Transmitter normal operation (default). 1 SIM Transmitter held in Reset.
0 FLUSH_RCV	Flush Receiver. This bit operates as a SIM receiver reset. The transmit portion of the SIM module is not affected. The software must clear this bit before the SIM receiver can operate. 0 SIM Receiver normal operation (default). 1 SIM Receiver held in Reset.

39.3.3.22 SIM Character Wait Time Register (CHAR_WAIT)

See Figure 39-24 for illustration of valid bits in the SIM Character Wait Time Register and Table 39-31 for description of the bit fields.

0x5002_4054 (CHAR_WAIT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CWT[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 39-24. SIM Character Wait Time Register

Table 39-31. SIM Character Wait Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 CWT	Character Wait Time. The value written to this register will specify the number of ETU times allowed between characters. Default is 0xFFFF

39.3.3.23 SIM General Purpose Counter Register (GPCNT)

See Figure 39-25 for illustration of valid bits in the SIM General Purpose Counter Register and Table 39-32 for description of the bit fields.

0x5002_4058 (GPCNT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPCNT[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 39-25. SIM General Purpose Counter Register

Table 39-32. SIM General Purpose Counter Register Field Descriptions

Field	Description
31–16	Reserved
15–0 GPCNT	General Purpose Counter. The value written to this register will be used to compare to the general purpose counter in the SIM module. Once the General purpose counter reaches this value, the GPCNT flag in the XMT_STATUS register will be set. This counter is intended to be used for any events that must be monitored for duration based on the card clock, receiver sample rate, or ETU rate (transmit clock). Example: ATR arrival time and ATR duration. Default is 0xFFFF

39.3.3.24 SIM Divisor Register (DIVISOR)

See Figure 39-26 for illustration of valid bits in the SIM Divisor Register and Table 39-33 for description of the bit fields.

0x5002_405C (DIVISOR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	DIVISOR[7:0]							
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 39-26. SIM Divisor Register

Table 39-33. SIM Divisor Register Field Descriptions

Field	Description
31–8	Reserved
7–0 DIVISOR	DIVISOR Register. The value written to this register will be used to generate the SIM Receive clock. The BAUD_SEL[2:0] bits in the CNTL register must be set to ‘111’ in order to control the divisor value using the DIVISOR register. Default is 0xFF

39.3.3.25 SIM Block Wait Time Register (BWT)

See Figure 39-27 for illustration of valid bits in the SIM Block Wait Time Register and Table 39-34 for description of the bit fields.

0x5002_4060 (BWT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWT[15:0]															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 39-27. SIM Block Wait Time Register

Table 39-34. SIM Block Wait Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 BWT	BWT Register 16 LSB. The value in this register is the block wait time 16 LSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag will be set. Default is 0xFFFF

39.3.3.26 SIM Block Guard Time Register (BGT)

See Figure 39-28 for illustration of valid bits in the SIM Block Guard Time Register and Table 39-35 for description of the bit fields.

0x5002_4064 (BGT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BGT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-28. SIM Block Guard Time Register

Table 39-35. SIM Block Guard Time Register Field Descriptions

Field	Description
31–16	Reserved
15–0 BGT	BGT Register. The value in this register is the block guard time. Time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be greater than this value. If it is not, then the BGT flag will be set. Default is 0x0000

39.3.3.27 SIM Block Wait Time Register HIGH (BWT_H)

See Figure 39-29 for illustration of valid bits in the SIM Block Wait Time Register HIGH and Table 39-36 for description of the bit fields.

0x5002_4068 (BWT_H) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWT_H[15:0]															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 39-29. SIM Block Wait Time Register HIGH

Table 39-36. SIM Block Wait Time Register HIGH Field Descriptions

Field	Description
31–16	Reserved
15–0 BWT_H	BWT Register 16 MSB. The value in this register is the block wait time 16 MSB. The time from START bit of last byte sent from the SIM module to the START bit of the first byte sent from the SmartCard must be less than the value formed by the 32-bit register composed by BWT_H and BWT registers. If it is not, then the BWT flag is set. Default is 0xFFFF

39.3.3.28 SIM Transmit FIFO Status Register (XMT_FIFO_STAT)

See Figure 39-30 for illustration of valid bits in the SIM Transmit FIFO Status Register and Table 39-37 for description of the bit fields.

0x5002_406C (XMT_FIFO_STAT) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	XMT_CNT[3:0]				XMT_WPTR[3:0]				XMT_RPTR[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-30. SIM Transmit FIFO Status Register

Table 39-37. SIM Transmit FIFO Status Register Field Descriptions

Field	Description
31–12	Reserved
11–8 XMT_CNT	These bits indicate the number of Bytes in the transmit FIFO. ‘0’ value means FIFO is empty or full.
7–4 XMT_WPTR	These bits indicate the transmit FIFO Write Pointer.
3–0 XMT_RPTR	These bits indicate the transmit FIFO Read Pointer.

39.3.3.29 SIM Receive FIFO Counter Register (RCV_FIFO_CNT)

See Figure 39-31 for illustration of valid bits in the SIM Receiver FIFO Counter Register and Table 39-38 for description of the bit fields.

0x5002_4070 (RCV_FIFO_CNT) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RCV_CNT[8:0]								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-31. SIM Receive FIFO Counter Register

Table 39-38. SIM Receive FIFO Counter Register Field Descriptions

Field	Description
31–9	Reserved
8–0 RCV_CNT	These bits indicate the number of Byte can be written into the receive FIFO. 0 value means FIFO is empty or full.

39.3.3.30 SIM Receive FIFO Write Pointer Register (RCV_FIFO_WPTR)

See Figure 39-32 for illustration of valid bits in the SIM Receive FIFO Write Pointer Register and Table 39-39 for description of the bit fields.

0x5002_4074 (RCV_FIFO_WPTR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RCV_WPTR[8:0]								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-32. SIM Receive FIFO Write Pointer Register

Table 39-39. SIM Receive FIFO Write Pointer Register Field Descriptions

Field	Description
31–9	Reserved
8–0 RCV_WPTR	These bits indicate the receive FIFO Write pointer.

39.3.3.31 SIM Receive FIFO Read Pointer Register (RCV_FIFO_RPTR)

See Figure 39-33 for illustration of valid bits in the SIM Receive FIFO Read Pointer Register and Table 39-40 for description of the bit fields.

0x5002_4078 (RCV_FIFO_RPTR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RCV_RPTR[8:0]								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-33. SIM Receive FIFO Read Pointer Register

Table 39-40. SIM Receive FIFO Read Pointer Register Field Descriptions

Field	Description
31–9	Reserved
8–0 RCV_RPTR	These bits indicate the receiver FIFO read pointer.

39.4 Functional Description

To best describe the organization of the SIM module from a user's point of view, it is instructive to view the SIM at a number of different levels of hierarchy. See Figure 39-34 for illustration of the organization of SIM and connection of the signals to the two available serial ports.

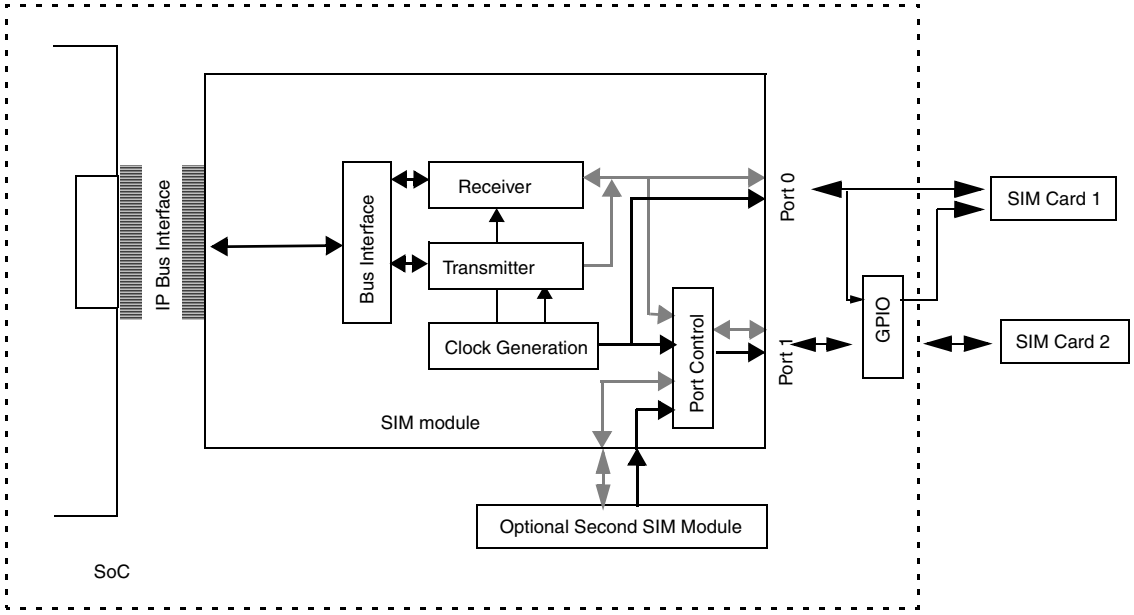


Figure 39-34. Block Diagram for SIM Module

The SIM module is essentially a standard UART with some special provisions made for SIM card communication. The SIM consists of nine main parts:

- IP Bus Interface
- Bus interface
- Clock generator
- Transmitter
- Receiver
- Port controller
- General purpose counter
- LRC blocks
- CRC blocks

See Figure 39-35 for illustration of the block diagram in detail, specifically the nine main parts.

39.4.1 SIM Detail Block Diagram

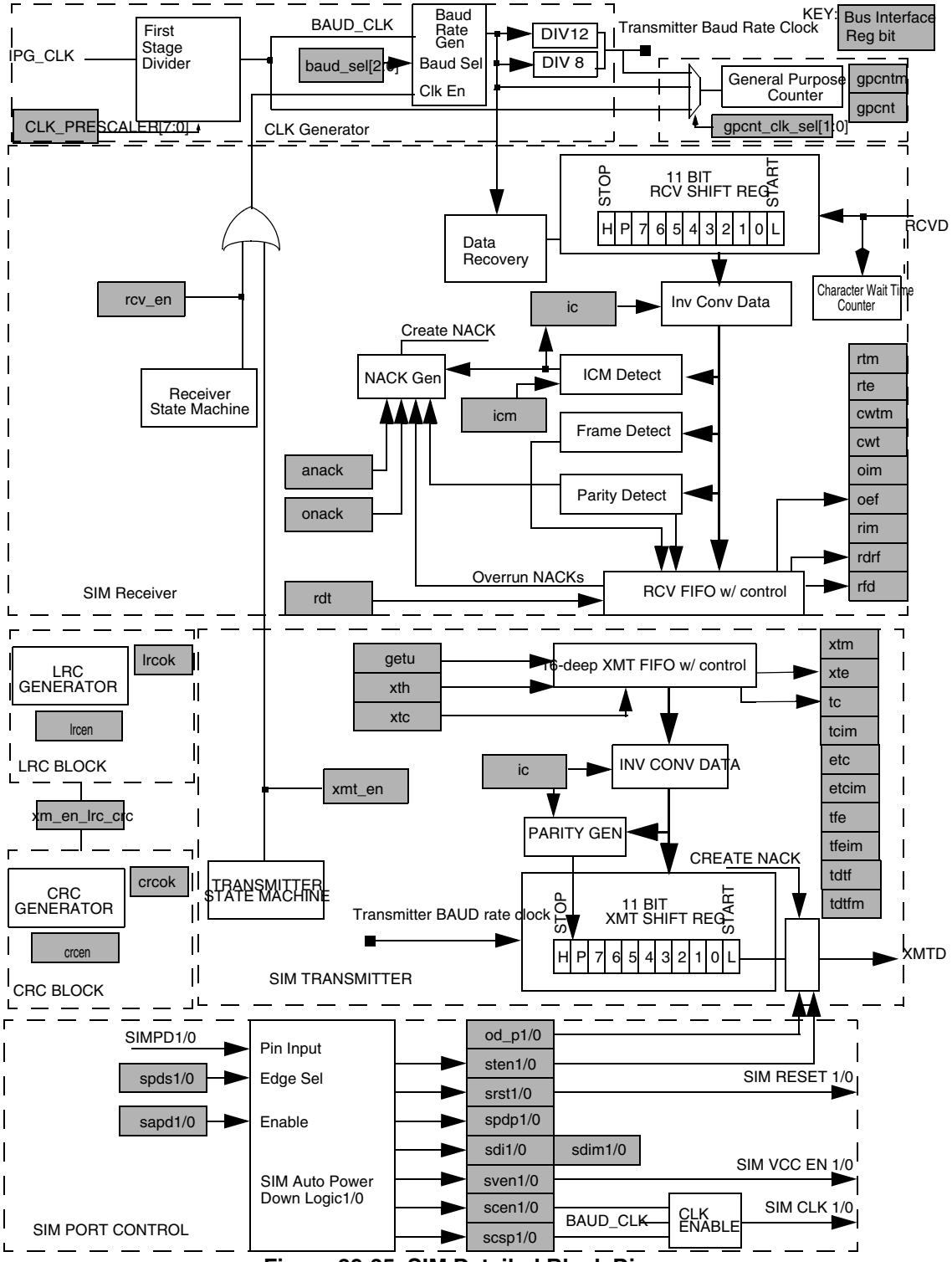


Figure 39-35. SIM Detailed Block Diagram

39.4.2 SIM Bus Interface

The SIM Bus interface block has been designed to enable the SIM module to be easily ported to other cores. The bus interface block contains all of the logic that uses the bus interface signals. See Figure 39-36.

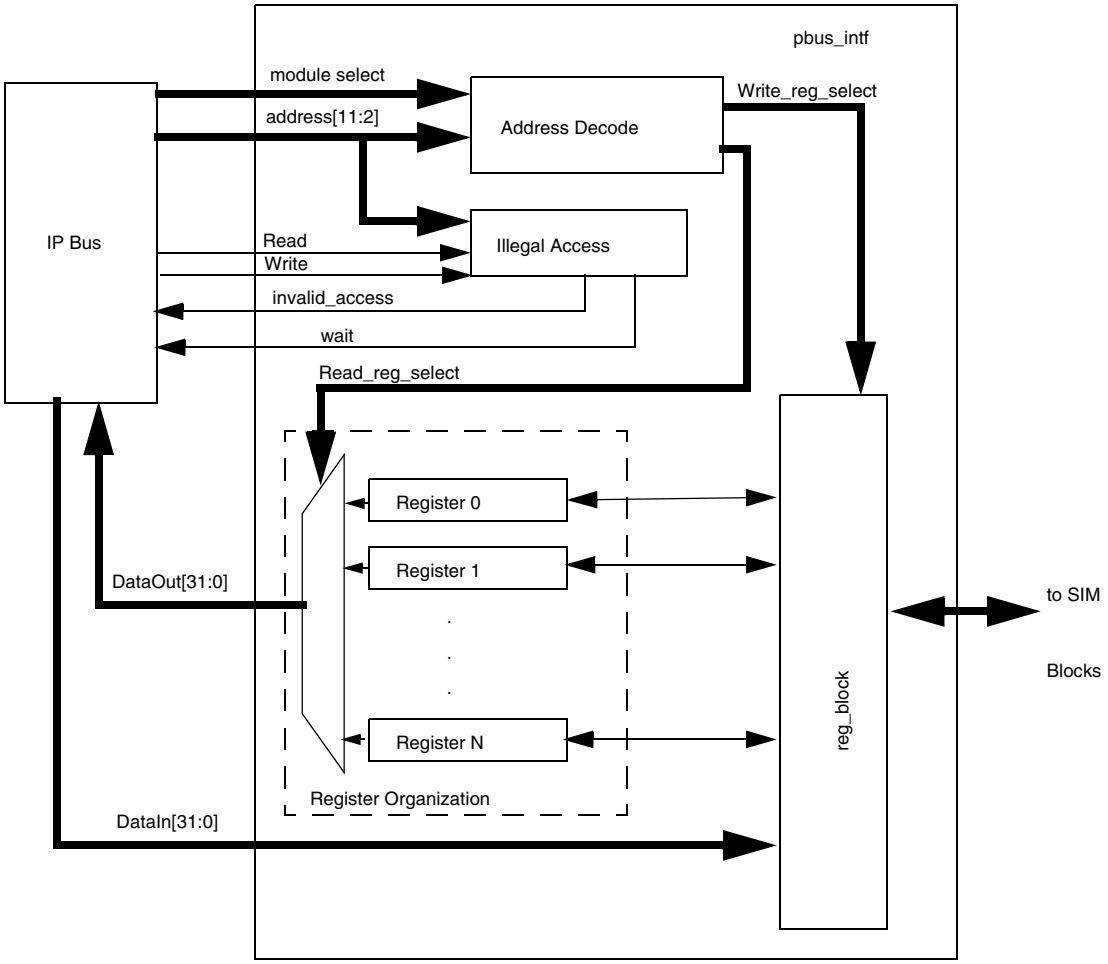


Figure 39-36. SIM Bus Interface

The bus interface block is responsible for peripheral bus address decoding, illegal access detection, dynamic wait-state requests, register organization, and register bit implementations. The address decoding uses the module address bus input along with the module select output of the IP bus to decode if the SIM module is being addressed. This decoding of these signals is done asynchronously. The output of the address decode is used with the read/write signals from the IP bus to determine the action requested by the bus master. If the read signal is active, the data_out bus will be driven with the register selected by the address decoding. If the write signal is active, the data_in bus will be latched into the register selected by the address decoder at the rising edge of the data_strobe signal.

The illegal access detection is implemented similarly to the address decoding. If the module_select signal becomes active while the address inputs are pointing to an unimplemented address, the invalid signal is asserted asynchronously to the bus master. The invalid signal can also be generated under certain register

access conditions such as writing to a full transmit FIFO, or writing to a read-only register. See Figure 39-37.

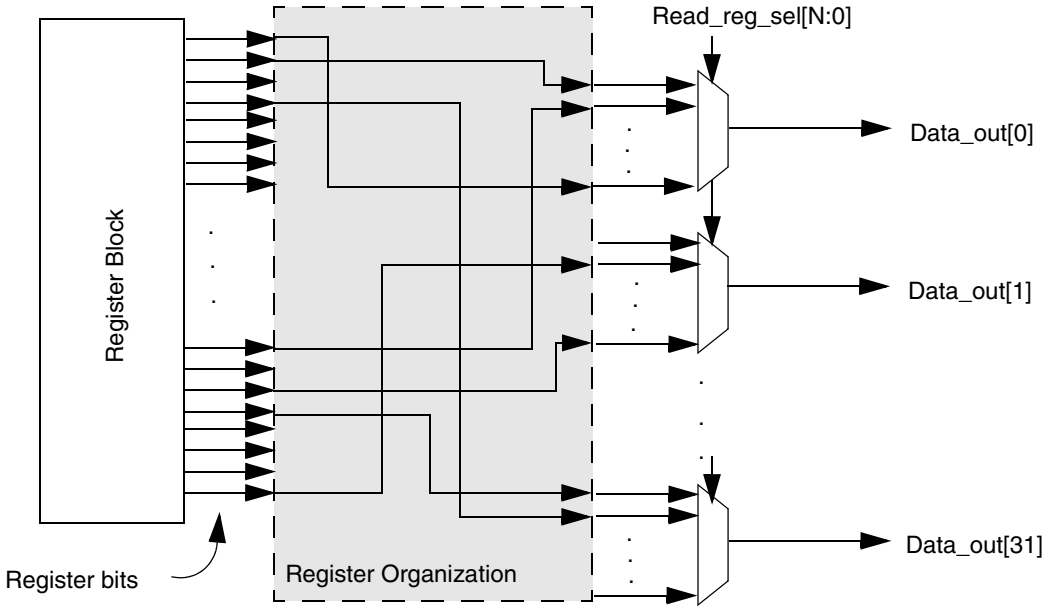


Figure 39-37. SIM Read Registers

The register bit implementations are done inside the reg_block sub-module. This block accepts the decoded read and write signals for the register bits generated in the address decoder. The data_strobe output is used as the clock input to the register bits in order to clock in the new data during a write access. The write_reg_sel signals are used to gate the write action with the address decoding. The clearing mechanisms for status bits are implemented as write-one-to-clear. See Figure 39-38.

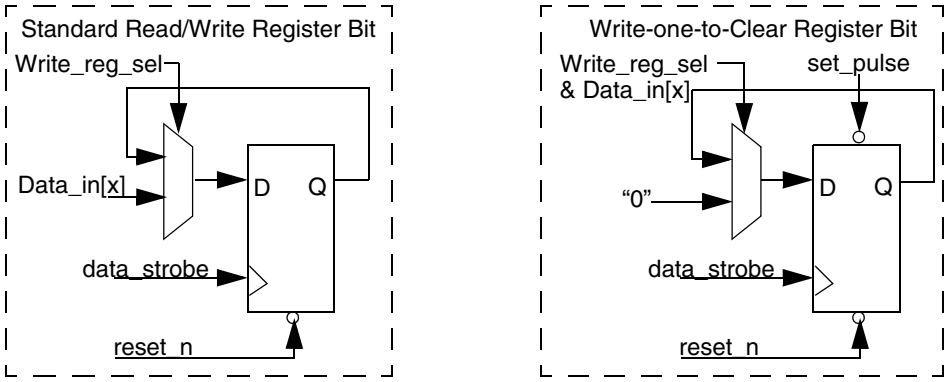


Figure 39-38. Register Bit Diagram

In the "write-one-to-clear" register bit example, "set_pulse" represents a pulsed signal generated in the data_strobe clock domain to set the register output.

39.4.3 SIM Clock Generator

The clock generator is responsible for implementing clock tree synthesis constructs, scan clock muxing, baud rate clock (BAUD_CLK) generation, and providing clocks to the transmitter, receiver, and port controller sections of the SIM module. See Figure 39-39 for the schematic of the SIM clock generator. The dividers outlined in bold, generate a pulsed (gated) clock of the desired frequency. The pulse is equal in duration to one half the IPG_CLK period. This clock is used internal to the SIM module to drive the transmit and receive sections of the module.

The dividers outline in normal, generate a clock of 45%~55% duty cycle to drive the clock pin of the SmartCard. This clock is not used internal to the SIM module

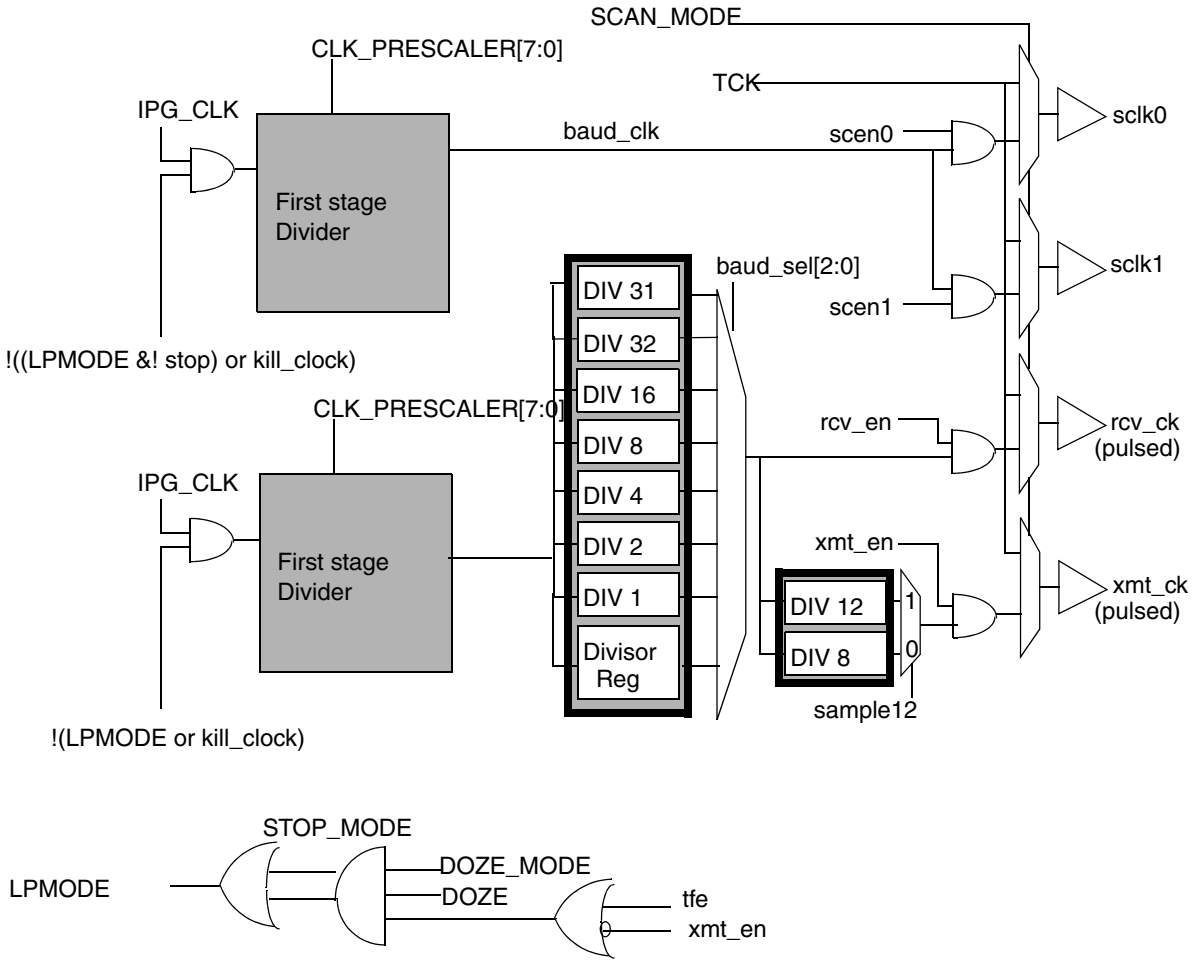


Figure 39-39. SIM Clock Generator

39.4.3.1 Clock Tree Synthesis

The clock tree synthesis constructs used in the current implementation follow the current clock methodology(srs3.1 clock methodology). This can be seen in the RTL of the sim_clocks.v module. There is an instance of the special cell (clk_driver_pos.v) to drive each clock. This cell can then be used to control the clock skew during the layout process.

39.4.3.2 Scan Test

The scan clock muxing is implemented according to the design for testability rules.

39.4.3.3 Baud Clock Generation

The baud rate clock generation performed by the clock generator results in different frequencies. The default frequency is a divide by two of the IPG_CLK input. The baud rate can be programmed to be IPG_CLK divided by the even value using the CLK_PRESCALER[7:0] bits as shown in the CLK_PRESCALER register. The baud rate clock is generated in two forms in the design. The BAUD_CLK that is used by the SIM cards (SCLK0, SCLK1) is generated so that it must be 45%~55% duty at the divide values. This is necessary to meet the requirements of the ISO 7816 specification. The BAUD_CLK that is used internal to the SIM module is generated as a gated version of the IPG_CLK clock. The resultant clock will be a pulse of one-half IPG_CLK period in width with the expected frequency. The gated baud clock complies with the clock methodology initiative. The 50% duty cycle clock does not comply with the clock methodology initiative but this clock is not used as a clock internal to the SIM module. It is only used as a data path.

39.4.3.4 Transmitter Clock Generation

The transmitter clock (xmt_clk) is generated by the clock generator based on the values passed to it for the baud rate select (baud_sel[2:0]) and sample12. The transmit clock is gated by the transmit enable (XMT_EN) register bit. When the transmitter is enabled, the clock generator counts the appropriate number of receive clock (rcv_clk) positive edges to determine when to toggle the transmitter clock output. The transmitter clock is always based upon the receive clock.

39.4.3.5 Receiver Clock Generation

The receiver clock (rcv_clk) is generated by the clock generator based on the value passed to it for the baud rate select (baud_sel[2:0]). The receiver clock is gated by the receiver enable (RCV_EN) register bit. When the receiver is enabled, the clock generator counts the appropriate number of BAUD_CLK positive edges to determine when to toggle the receiver clock output. The number of BAUD_CLK positive edges is determined by the value of the baud rate select input (baud_sel[2:0]) and is programmable to these divisors: 31 (slowest because used with the 372/1 Fi/Di rate), 32, 16, 8, 4, 2, 1, and divisor[7:0]. The programmable divisor (divisor[7:0]) is programmable using the DIVISOR_REG register.

39.4.3.6 Port Control Clock Generation

The port controller clocks are provided by the clock generator for use on the SIM card ports. These clocks are equivalent in frequency to the BAUD_CLK and are gated by the SIM clock enable (scen0 and scen1) signals. The level at which the card clocks (SCLK0, SCLK1) are stopped when disabled is determined by the SIM clock select polarity (SCSP0, SCSP1) inputs to the clock generator. Synchronizers are implemented to ensure glitch free operation of the card clocks when enabling/disabling, or changing the clock stopped polarity.

39.4.3.7 Low Power Mode Clock Control

The clock generator block is responsible for gating the clocks to the SIM module appropriately whenever a low power mode instruction is decoded. The IP interface provides two signals that encode the two available low-power states of the processor. These states are: STOP and DOZE. The response to the DOZE instruction is controlled through the use of the DOZE bit in the RESET_CNTL register. See the RESET_CNTL register description. The response to the STOP instruction is controlled through the use of the STOP bit in the RESET_CNTL register. See the RESET_CNTL description

39.4.4 SIM Transmitter

The SIM Transmitter block has been designed to localize all transmit related circuitry into one section of hierarchy. The Transmitter block comprises the following sections of logic: transmit state machine, transmit shift register, transmit FIFO, guard time generator, transmit NACK control, and transmit data convention.

39.4.4.1 Transmit State Machine

The Transmit state machine is the heart of the transmitter block. The state machine is responsible for sequencing through a transmit operation while reacting to inputs from the receiver, the transmit FIFO, and the guard time circuit. See Figure 39-40 for flow diagram of the transmit state machine.

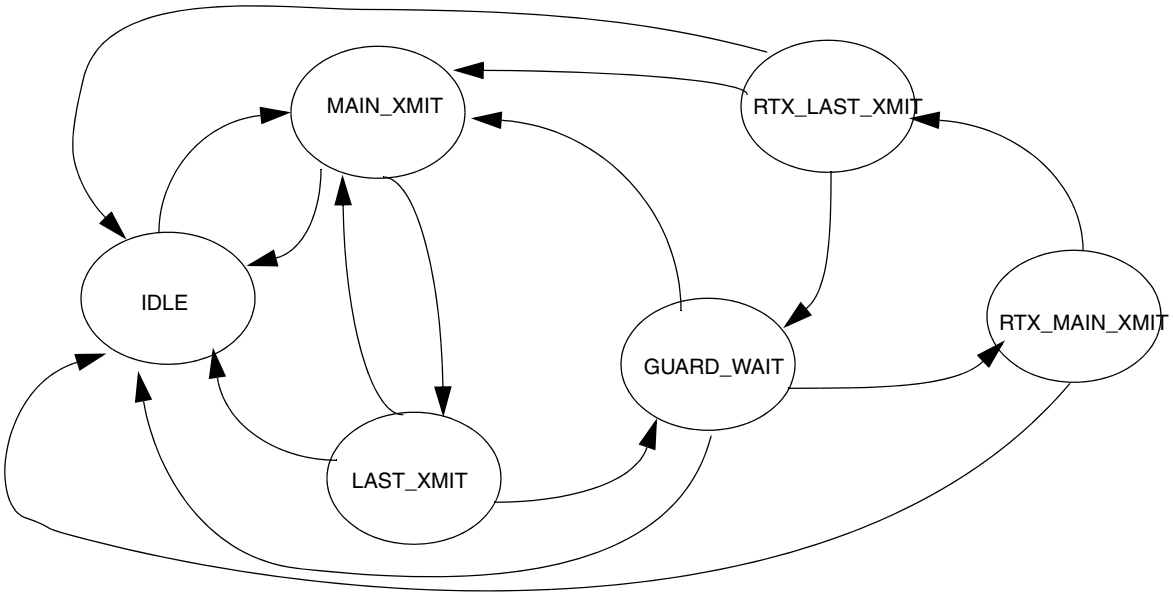


Figure 39-40. Transmit State Machine

The functions performed by each state are:

- IDLE
  - This is the initial state. The state machine waits here until it detects XMT_EN is set and a write to the transmit FIFO has occurred. The data pointed to by the transmit read pointer is loaded into the shift register, and the state machine transitions to the MAIN_XMIT state. Any time XMT_EN is cleared, the state machine returns to this state.

- MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the LAST_XMIT state.
- LAST_XMIT
  - This state transmits the last bit of the current transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in GETU[7:0], jump to MAIN_XMIT state to retransmit the current byte.
    - If no transmit NACK, and GETU[7:0] is zero, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.
- GUARD_WAIT
  - The state machine remains in this state until the Guard time counter has expired.
    - If a transmit NACK error occurred on last transmission, jump to RTX_MAIN_XMIT and re-transmit
    - If no transmit NACK and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte.
    - If no transmit NACK and the FIFO is empty, return to the IDLE state.
    - If transmit NACK threshold is detected, stop transmitter, set XTE flag, and jump to the IDLE state.
- RTX_MAIN_XMIT
  - The transmitter is operating normally in this state. The data in the shift register is shifting once every transmit clock cycle. When the second to last bit of the current transmission is about to be sent, the state machine transitions to the RTX_LAST_XMIT state. This state is identical to the MAIN_XMIT state except that it retransmits the previously NACKed byte.
- RTX_LAST_XMIT
  - This state transmits the last bit of the current re-transmission and determines the next operation. One of the following will occur.
    - If GETU[7:0] is non-zero, jump to the GUARD_WAIT state.
    - If a transmit NACK error occurred, with a zero in GETU[7:0], jump to GUARD_WAIT state to check Transmit NACK threshold.
    - If no transmit NACK, GETU[7:0] is zero, and the FIFO is not empty, load the shift register, jump to MAIN_XMIT to transmit the next byte
    - If no transmit NACK, GETU[7:0] is zero, and the FIFO is empty, jump to IDLE state; set the transmit complete (TC) flag.

39.4.4.2 Transmit Shift Register

The transmit shift register is 11 bits wide and controlled by the transmit state machine described previously. The shift register shifts out data at the transmit clock frequency (xmt_ck).

39.4.4.3 Transmit FIFO

The transmit FIFO is implemented inside the transmitter block. The FIFO depth is 16 bytes. The FIFO block is shared by both SIM module ports. The transmit FIFO cannot be accessed by the alternate SIM module through the alternate port. Each write to the transmit FIFO increases the transmit FIFO write pointer. Each time the transmit shift register is loaded from the transmit FIFO, the transmit FIFO read pointer is increased. When the read and write pointers are equal, the transmit FIFO empty flag (TFE) is set. Software has no visibility of the transmit FIFO pointers, but a transmit FIFO threshold value can be set to alert the software when the number of bytes in the FIFO has reached a specified level. A read of the transmit FIFO register (PORTx_XMT_BUF) will return the last byte written to the FIFO. Writes to the transmit FIFO can occur at any time.

The transmit FIFO can be flushed by setting the XMT_FLUSH bit in the RESET_CNTL register. A transmit NACK threshold error (XTE) will halt the transmitter, and flush the transmit FIFO. The flush operation resets the transmit read and write pointers to equal values. Everything in the transmitter block is reset by the transmit flush operation. This does not include the control registers associated with the transmitter. The Transmit data threshold flag (TDTF) does not get cleared by the XMT_FLUSH operation.

39.4.4.4 Transmit Guard Time Generator

The guard time generator is simply a counter that is clocked by the transmit clock in order to delay the beginning of the next transmission and the setting of the transmit complete interrupt flag (TC) by a programmable amount of transmit bit widths (ETUs). The duration of the count is controlled by the GUARD_CNTL register (GETU[7:0]). See Figure 39-41 for depiction of three transmit operations in order to show the effect of the guard time generator logic

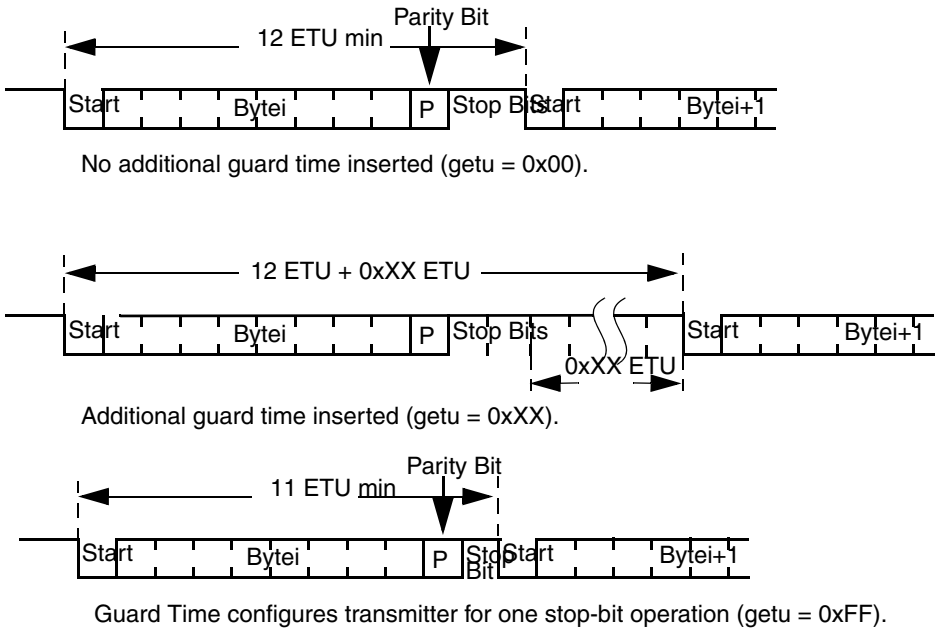


Figure 39-41. Transmit Guard Time

39.4.4.5 Transmit NACK Generator

The transmit NACK generator is responsible for driving the transmitter output low during the STOP bit time to signify an error was detected in the received data from the SIM card. This logic responds to a NACK request generated by the receiver block. See Figure 39-42 shows a typical SIM transaction with the NACK pulse inserted.

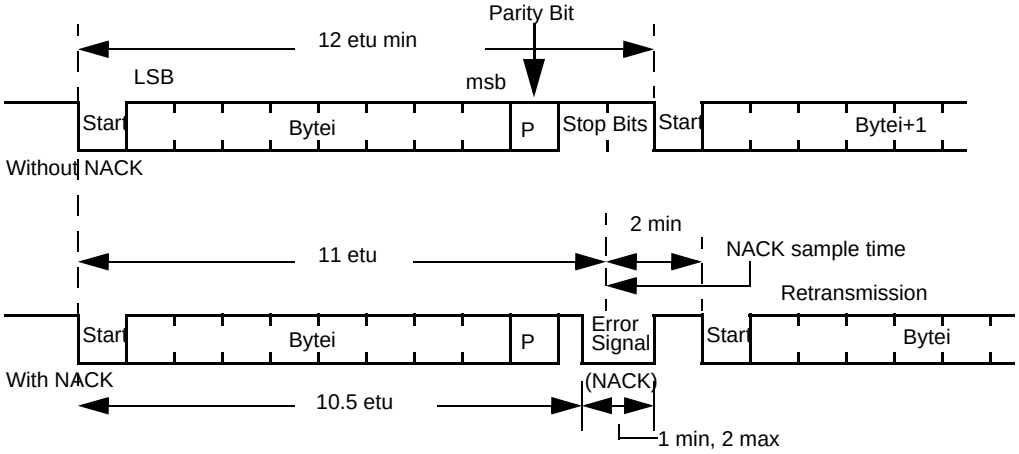
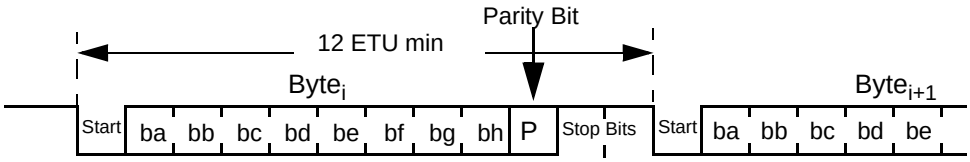


Figure 39-42. Transmit NACK Generator

The transmit NACK generator is also responsible for keeping track of the number of NACKs received during a transmit operation. The SIM receive state machine detects NACKs generated by the SIM card, and reports them to the transmit NACK logic. Once the number of detected NACKs has reached the programmed threshold (XTH[3:0]), an interrupt flag is generated, the transmit FIFO is flushed, and the transmitter is disabled.

39.4.4.6 Transmit Data Convention Logic

The transmit data convention logic provides the support for the two different data conventions available in SIM cards. See Figure 39-43 for illustration of SIM data conventions.



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM before being transmitted.

Direct Convention: *ba* is lsb of data byte to be sent. *bh* is msb.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is msb of data byte to be sent. *bh* is lsb.
Both the data bits and parity bit are logically inverted by hardware.

Figure 39-43. SIM Data Conventions

The direct data convention is the default. If the inverse convention bit (IC) in the DATA_FORMAT register is set, then the transmit data convention logic will convert the output of the transmit FIFO to the inverse convention before sending it to the transmit shift register.

39.4.5 SIM Receiver

The SIM Receiver block has been designed to localize all receive related circuitry into one section of hierarchy. The receiver block is comprised of the following functions: receive state machine and the receive FIFO.

39.4.5.1 Receive State Machine

The receive state machine is responsible for sampling the receive data pin and capturing the bit value into the receive shift register. Additionally, the receive state machine can detect the START bit, parity errors, framing errors, and initial character when operating in initial character mode.

Once enabled by the RCV_EN bit in the ENABLE register, the receive state machine sequences through the states as shown in Figure 39-44.

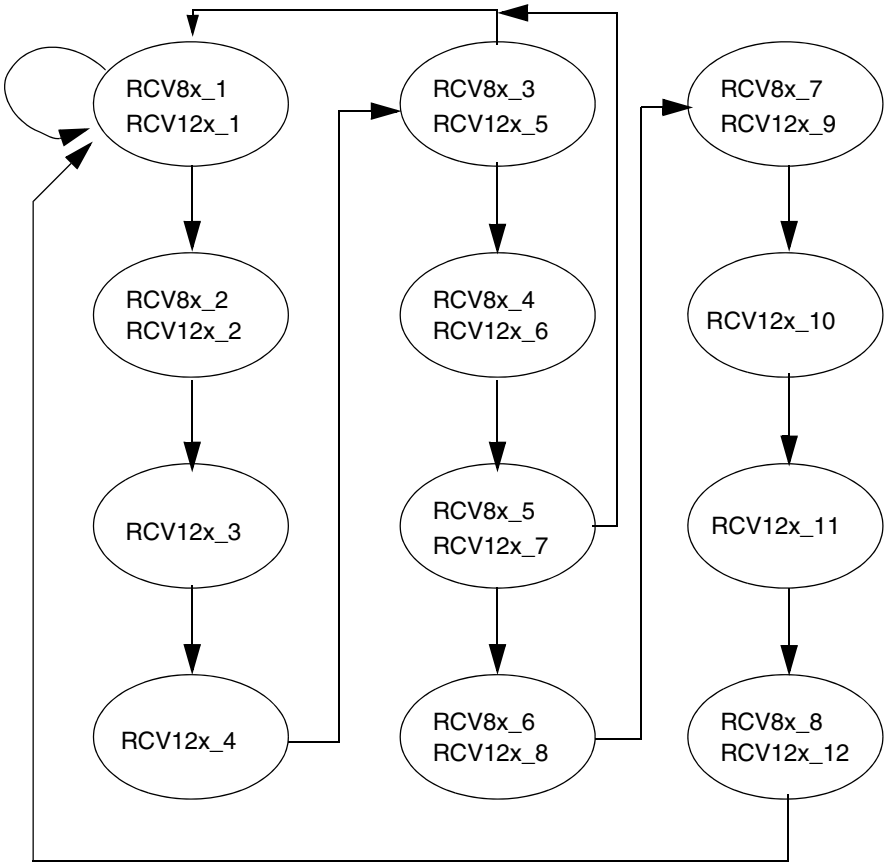


Figure 39-44. Receive State Machine

The states identified with a “8x” are used when operating in a 8x oversampling mode. The states identified with a “12x” are used when operating in a 12x oversampling mode. This mode is only used when sample12 is set to “1”. The number following the oversampling mode identifier represents the state number in the current mode. There are 12 states in “12x” mode, and 8 states in “8x” mode. Some states simply implement a one RCV_CK delay. States that perform additional functions are:

- RCV8x_1, RCV12x_1
  - This is the initial state of the receive state machine. If the first bit has not been received, the state machine remains in this state until a valid START bit is detected. For every subsequent bit, this state is simply a one RCV_CK cycle delay.
- RCV8x_2, RCV12x_2
  - This state captures the first sample of the current receive data input.
- RCV12x_3
  - This state captures the second sample of the current receive data input.
- RCV12x_4
  - This state captures the third sample of the current receive data input.

- RCV8x_3, RCV12x_5
  - This state checks if we are receiving a correct START Bit. If not, Then we return to state 1.
- RCV8x_4, RCV12x_6
  - If we are in the 11th Bit, We check for Parity or Initial Character errors and send NACK if needed.
- RCV8x_5, RCV12x_7
  - Store current Bit value in Shift Register. If this is the first bit and the value is not 0, restart the State Machine.
- RCV8x_6, RCV12x_8
  - If the current bit is the last bit of the transfer, Set flag to transfer Shift Register to Receive Buffer.
- RCV8x_7, RCV12x_9
  - Clear flag for transferring Shift Register to Receive Buffer.
- RCV12x_10
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV12x_11
  - If the current bit is the last bit of the transfer (first stop bit), this state takes the sample of the NACK window.
- RCV8x_8, RCV12x_12
  - This state represents the end of the current receive input bit. Several operations occur during this state, including:
    - Increment bit counter
    - Perform a majority vote on the NACK samples and notify the transmitter if a NACK pulse was detected.

39.4.5.2 Data Sampling / Voting

The receive state machine runs at the receive clock rate (RCV_CK). This clock is used to oversample the received data at either a 8X or 12X sample rate. For each input bit, the receive state machine captures 3 samples. A majority voting algorithm is then applied to determine the value of the bit received. The value common to 2 or more of the samples is determined to be the bit value in the receive shift register.

39.4.5.3 Start Bit Detection

The SIM receive input is defined to be high when not active. The data transmission is defined to begin with a low pulse for a bit duration. This is called the START bit. The receive state machine is responsible for detecting and validating the START bit. The receive state machine samples the START bit three times using a majority voting scheme to determine if the START bit is valid. This will effectively filter out any low receive inputs shorter than one RCV_CK period. See Figure 39-45 for illustration of a typical SIM data transaction with the START bit identified.

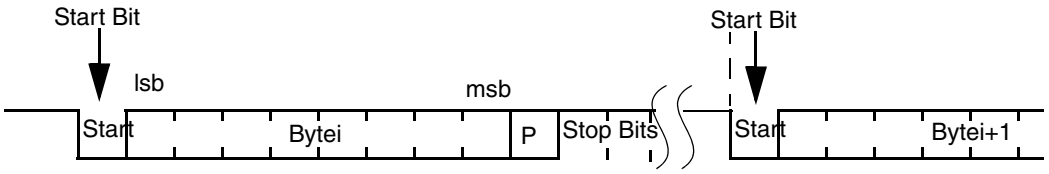


Figure 39-45. Start Bit Diagram

39.4.5.4 Parity Error Detection

The receive state machine is responsible for detecting parity errors in the received data. Data is always transmitted with even parity, except when in inverse convention mode. In inverse convention mode, all data bits and the parity bit are complemented making the data appear to be odd parity. The parity bit is defined as the 10th bit of the received data. The parity of the 2nd through 10th received bits is calculated by the receiver parity logic. This logic determines if the parity of the 9 received bits is correct. See Figure 39-46 for illustration of a typical SIM data transaction with the parity bit identified.

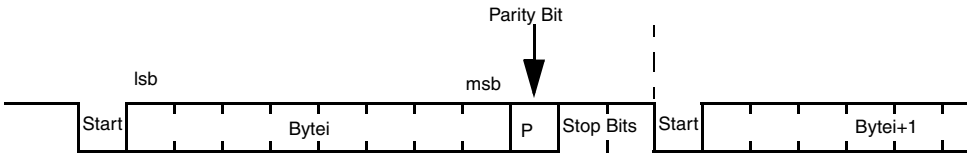


Figure 39-46. Parity Bit Diagram

When a parity error is detected on a given byte, the RCV_PF bit for that byte is set in the receive FIFO. A parity error cannot cause an interrupt, but it can signal the SIM transmitter to create a NACK pulse to the SIM card asking for a retransmission of the corrupted data. NACK generation upon a parity error is enabled by setting the ANACK bit in the CNTL register.

39.4.5.5 Framing Error Detection

The receive state machine is responsible for detecting framing errors in the received data. A SIM data transaction is defined as 11 or 12 bits long consisting of the START bit, 8 data bits, 1 parity bit and 1 or 2 STOP bits. A framing error occurs when the STOP bit is not detected during the 11th bit time of a data transaction. The STOP bit is generally defined as two bit times (ETUs) of a high pulse following the parity bit. When the GUARD_CNTL register is programmed to 0xFF, the STOP bit is defined as one bit time. A framing error can only occur when the parity bit of the current byte is low, and the STOP bit arrives late. See Figure 39-47 for illustration of a typical SIM data transaction with the STOP bits identified. Also, shown is a SIM data transaction with a late arriving STOP bit indicating a framing error.

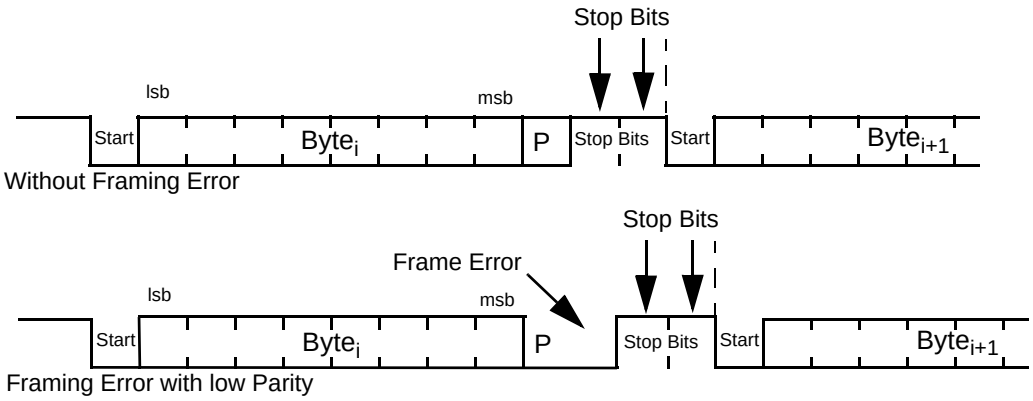


Figure 39-47. Framing Error Diagram

When a framing error is detected on a given byte, the RCV_FE bit for that byte is set in the receive FIFO. A framing error cannot cause an interrupt, nor can it create a NACK pulse to the SIM card asking for a retransmission of the corrupted data.

39.4.5.6 NACK Detection

The existence of the NACK pulse is sampled by the receive state machine at 11 Elementary Time Units (ETUs) after the falling edge of the START bit. An ETU is equivalent in time to 1 transmit clock period. Once the receiver detects a NACK, it signals the transmitter that an error occurred. The transmitter will not initiate retransmission for at least another two ETU times as required by the ISO 7816 specification.

39.4.5.7 Initial Character Detection

The SIM receive state machine supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential characters that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Figure 39-48 and Figure 39-49 illustrate the differences between the two formats. Essentially, inverse convention differs from direct convention in that the order of the data is flipped MSB for LSB and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

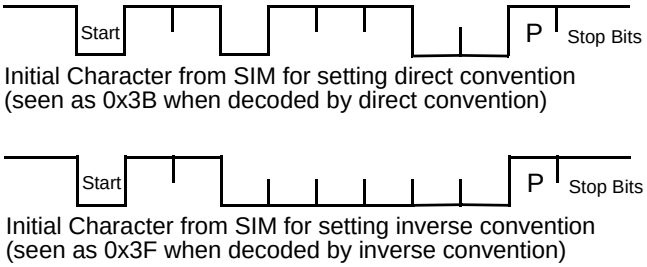
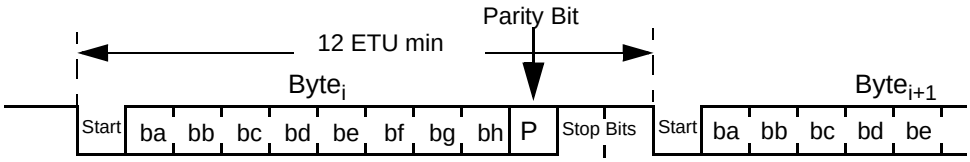


Figure 39-48. Valid Initial Characters



Parity Bit: if configured for even parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be even
if configured for odd parity, total number of logic ones in the 9 bits (8 bits data, 1 parity bit) will be odd
When configured for inverse convention, the parity bit is inverted by SIM card before being transmitted.

Direct Convention: *ba* is LSB of data byte to be sent. *bh* is MSB.
Neither the data bits nor parity bit is logically inverted.

Inverse Convention: *ba* is MSB of data byte to be sent. *bh* is LSB.
Both the data bits and parity bit are logically inverted by SIM card.

Figure 39-49. Inverse Convention versus Direct Convention

39.4.5.8 Receive FIFO

The receive FIFO is implemented inside a sub-block of the receiver block. The FIFO depth is 285 bytes. The FIFO block is shared by both SIM module ports. The receive FIFO cannot be accessed by another SIM module through the alternate port. Only the port1 IO pins can be accessed through the alternate port. The secondary SIM would then uses its own internal FIFO while using the primary SIM IO ports. This is only done if it is required to control two SIM cards at the same time. It the two SIM card access only occurs one at a time, then a single SIM module can control both SIM cards (one on port0 and one on port1). The receive FIFO is accessed through the PORTx_RCV_BUF registers.

The receive FIFO is loaded from the receive shift register after the final bit of the current SIM card transmission has been received. The FIFO contains 10 bits per transmission. The lower eight bits contain the received data byte. Bits 8 and 9 contain the parity and framing status for the received byte.

Each read from the receive FIFO increases the receive FIFO read pointer. Each time the receive shift register is transferred to the receive FIFO, the receive FIFO write pointer is increased. When the difference between the read and write pointers equals the programmed threshold value (RDT), the receive data register full flag (RDRF) will be set. An interrupt can be generated by the RDRF flag if the RIM bit in the INT_MASK register is cleared. Software has no visibility of the receive FIFO pointers. A write to the receive FIFO register (PORTx_XMT_BUF) will generate an invalid access exception to the processor.

The receive FIFO can be flushed by setting the RCV_FLUSH bit in the RESET_CNTL register. The flush operation resets the receive read and write pointers to equal values. All logic associated with the receiver will be reset by the flush operation except for receiver control registers.

39.4.5.9 Overrun Detection

The receive FIFO logic is responsible for detecting an overrun condition. When a received byte is transferred from the receive shift register to a receive FIFO that contains 285 unread bytes, the SIM receiver will flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 285 unread bytes in the FIFO unaltered. The SIM module will generate a NACK to the SIM card on an overrun condition if the ONACK

bit in the CNTL register is set. The SIM module will continually NACK SIM card transmissions until a read of the receive FIFO occurs.

39.4.5.10 Character Wait Time Counter

The SIM receiver block includes a 16-bit counter that counts the number of bit times (ETUs) between received characters. When enabled, the Character Wait Time Counter (CWT) will not start counting until after the START bit(s) of a valid character has been received. The counter is synchronized to the receive character bit positions so that an accurate count of the number of ETUs between characters can be made. The Character Wait Time Counter has a 16-bit programmable comparator. Software can write the expected number of ETUs between characters to the comparator. If the time between characters exceeds this value, an interrupt flag will be set and an interrupt generated if the mask is clear.

39.4.6 SIM Port Control

The SIM Port Control block has been designed to localize all port related circuitry into one section of hierarchy. The port control block comprises the following functions: SIM card interface, SIM Card presence detect, and SIM card auto-power down.

39.4.6.1 SIM Card Interface

The SIM module allows for direct control of two separate SIM cards. The SIM module does not support simultaneous communication with two SIM cards. To achieve simultaneous communication with two SIM cards, a second SIM module must be used. The SIM module can relinquish control of the inactive port to a secondary SIM module by setting the AMODE bit in the SETUP register.

The SIM card clock is generated in the SIM clock generator. The SIM card reset and SIM card voltage enable are controlled by software through the PORTx_CNTL registers.

See Figure 39-50 for illustration of a example SIM module hookup to two SIM cards. The power management chip shown is used to provide Vcc for the SIM card, and level translators for the remaining signals when interfacing to a SIM card operating at a different voltage than SIM module. The SIM module can directly access a SIM card if it is operating at the same voltage. In this case, the 3VOLTx bit in the PORTx_CNTL registers can be configured so the SIM Port transmit output as bi-directional. This frees up the SIM port receive pin to be used as general purpose I/O. If the SIM module and the SIM card are operating at different voltages, then the power management level translators must be used. In this case, the 3VOLTx bit must be cleared and then both the RX and TX pins will be used.

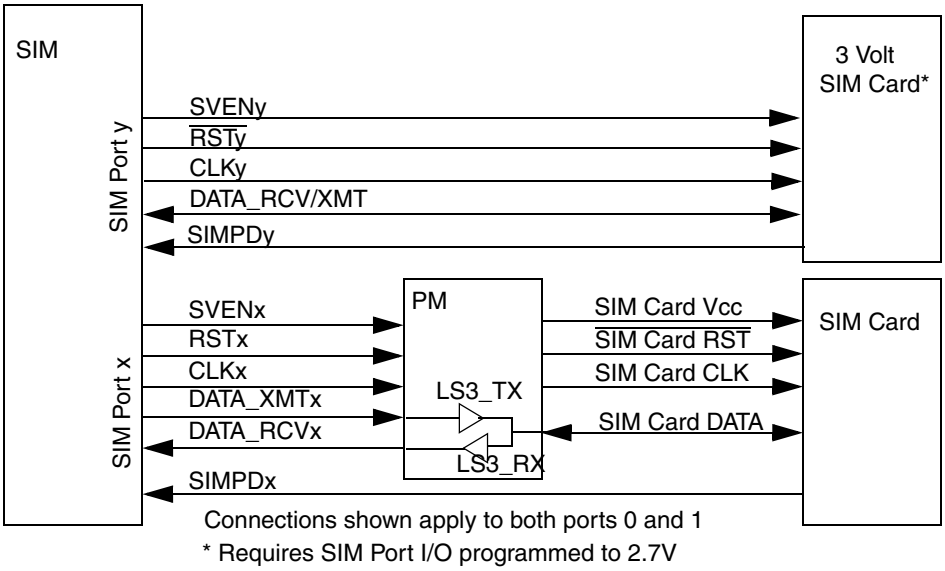


Figure 39-50. SIM Card Hookup

39.4.6.2 SIM Card Presence Detect

The SIMPDx input allows for detection of the insertion or removal of a SIM card. The SPDSx control bit in the PORTx_DETECT register allows the software to configure which edge of the SIMPDx pin will cause a presence detect event. A maskable interrupt can be generated when a SIMPDx event occurs.

An internal pull-down device is present on the SIMPDx pins. This will provide for a high to low transition on the SIMPDx pin when a SIM card is removed.

39.4.6.3 SIM Card Automatic Power Down

When interfacing to the SIM cards, it is necessary to follow a particular sequence when powering them up and down. The SIM port control block contains hardware that provides the correct sequence to power down a SIM card (see Figure 39-51). The power up sequence must be done manually by the software using the pin control bits supplied in the PORTx_CNTL registers.

The power down sequence is specified in ISO 7816 as:

1. RST transitions from high to low
2. CLK is turned off to a low
3. I/O transitions from high impedance to low
4. SIM Vcc is turned off

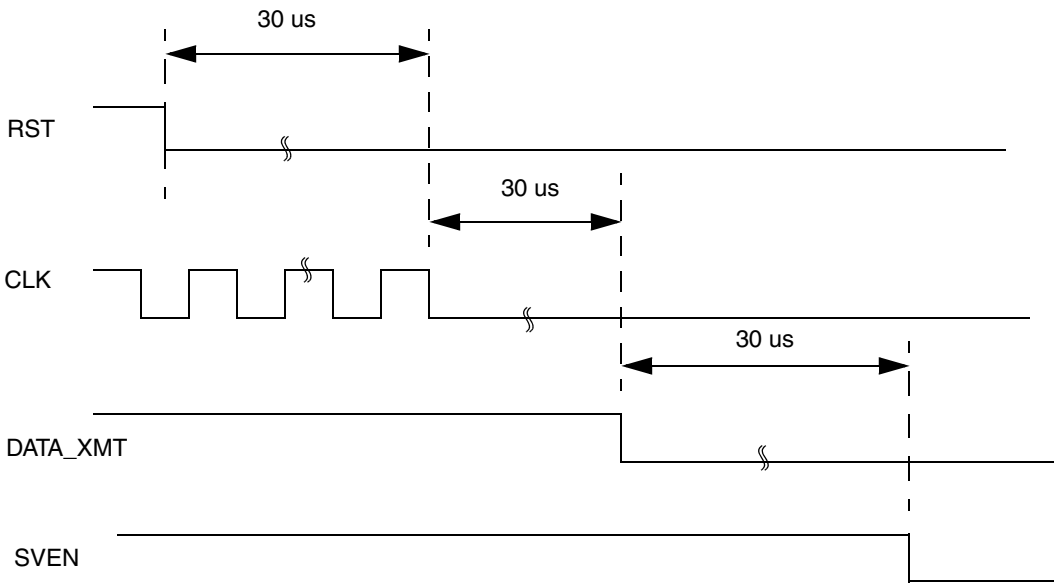


Figure 39-51. Auto Power Down Sequence

39.4.7 SIM General Purpose Counter

The SIM module provides a 16-bit counter for use when timing events during SIM card communication. The clock source for the counter is selectable between three sources: BAUD_CLK, RCV_CLK, or XMT_CK (ETU clock). The GPCNT_CLKSEL[1:0] bits in the CNTL register are used to select the clock input. The counter is enabled as soon as the input clock is selected. The starting of the counter is immediate once the input clock is running. Software can control the three input clock sources by using the KILL_CLOCK, RCV_EN and XMT_EN bits provided in the RESET_CNTL and ENABLE registers.

To run the counter from the card clock source the following conditions must be met:

1. ‘KILL_CLOCK = 0’ in the RESET_CNTL register.
2. ‘GPCNT_CLKSET[1:0] = 01’ in the CNTL register

The counter will begin to count at the card clock rate as soon as these conditions are met.

To run the counter from the receive clock source the following conditions must be met:

1. ‘KILL_CLOCK = 0’ in the RESET_CNTL register.
2. ‘RCV_EN = 1’ or XMT_EN = 1 in the ENABLE register
3. ‘GPCNT_CLKSET[1:0] = 10’ in the CNTL register

The counter will begin to count at the receive clock rate as soon as these conditions are met.

To run the counter from the ETU (transmit) clock source the following conditions must be met:

1. ‘KILL_CLOCK = 0’ in the RESET_CNTL register.
2. Either one of the following:
   “RCV_EN = 1’ in the ENABLE register and ‘CWT_EN = 1’ in the CNTL register
   ‘XMT_EN = 1’ in the ENABLE register

3. ‘GPCNT_CLKSET[1:0] = 11’ in the CNTL register

The counter will begin to count at the ETU clock rate as soon as these conditions are met.

The counter can be reset by setting GPCNT_CLKSEL[1:0] to 00. A 16-bit comparator value is provided that allows the software to select a count value at which an interrupt flag can be set and an interrupt generated if the mask is clear.

39.4.8 SIM LRC Block

The SIM module provides an 8-bit Linear Redundancy Check (LRC) generator / checker. The block is provided for use with T=1 SIM cards that support LRC. This block can be enabled through the LRCEN bit in the CNTL register. This block performs an 8-bit exclusive-OR on all received or transmitted characters. At the end of the reception of a block of characters, the result is expected to be 00. If so, the LRCOK bit is set in the RCV_STAT register. During transmission, the LRC block Exclusive-ORs each character that is transmitted with the current value of the LRC. If the XMT_CRC_LRC bit in the CNTL register is set, the LRC value will automatically be sent by the SIM transmitter as the final character when the transmit FIFO empties.

The LRC value can be reset in multiple ways. Clearing the LCREN bit in the CNTL register will reset the LRC value. At the end of a transmission (either after the LRC byte is transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the LRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the LRC value.

39.4.9 SIM CRC Block

The SIM module provides an 16-bit Cyclic Redundancy Check (CRC) generator / checker. The block is provided for use with T=1 SIM cards that support CRC. This block can be enabled through the CRCEN bit in the CNTL register. This block performs a polynomial based check on all received or transmitted characters. The polynomial description is shown in Figure 4-17. The polynomial used is the standard CRC-CCITT where g(x) = x^16 + x^12 + x^5 + 1. The CRC register is initialized to all “1” before the data is shifted in. Before transmission the resulting CRC is inverted. For example, transmitting a 0xFA results in the following:

- Data = 0x5F (bit reversal of 0xFA)
- crc = 0x4AEA
- invert the crc = 0xB515 (bit reversal of 0xADA8
- data transmitted = 0xFA, 0xAD, 0xA8

See Figure 39-52 for illustration of the SIM CRC block.

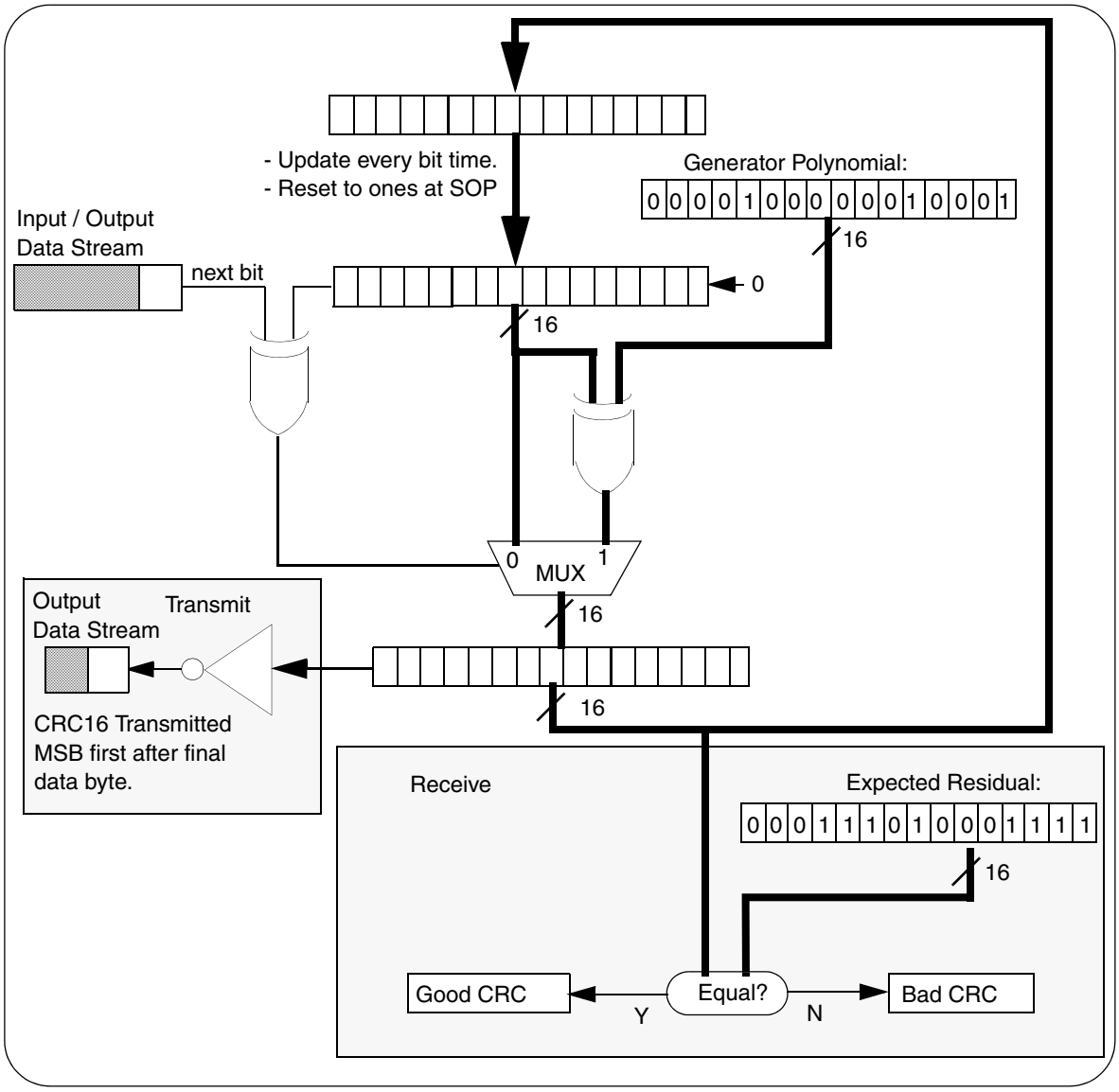


Figure 39-52. CRC Block Diagram

At the end of the reception of a block of characters, the residual from the CRC calculation is compared to 0x1D0F. If so, the CRCOK bit is set in the RCV_STAT register. During transmission, the CRC block updates the current value of the CRC residual using each character. If the XMT_CRC_LRC bit in the CNTL register is set, the CRC value will automatically be inverted and sent by the SIM transmitter as the final two characters when the transmit FIFO empties.

The CRC value can be reset in multiple ways. Clearing the CRCEN bit in the CNTL register will reset the CRC value. At the end of a transmission (either after the CRC characters are transmitted, or after the last character in the transmit FIFO is sent when XMT_CRC_LRC is clear), the CRC value is automatically reset by the SIM hardware. Finally, when setting the XMT_EN bit, the SIM hardware resets the CRC value.

39.4.10 Module Interrupts

See Table 39-41 for the list of all possible interrupt sources and their corresponding mask bits. All SIM interrupts are logically ORed to create the active low irq_n and data_irq_n signals that go to the ITC module. All mask bits are such that a logic 1 implies that the corresponding interrupt is disabled (masked), whereas a 0 implies that the interrupt enabled (unmasked). All of these mask bits are logic 1 out of reset (all interrupts are masked).

Table 39-41. SIM Module Interrupts

Flag	Flag Register	Mask	Mask Register	Description
TC	XMT_STATUS	TCIM	INT_MASK	Transmit Complete
ETC	XMT_STATUS	ETCIM	INT_MASK	Early Transmit Complete
TFE	XMT_STATUS)	TFEIM	INT_MASK	Transmit FIFO Empty
XTE	XMT_STATUS	XTM	INT_MASK	Transmit Threshold Error
TFO	XMT_STATUS	TFOM	INT_MASK	Transmit FIFO Overfill error
TDTF	XMT_STATUS	TDTFM	INT_MASK	Transmit Data Threshold Flag
GPCNT	XMT_STATUS	GPCNTM	INT_MASK	General Purpose Counter Comparator Flag
RDRF	RCV_STATUS	RIM	INT_MASK	Receive data register full (FIFO threshold level reached)
OEF	RCV_STATUS	OIM	INT_MASK	Overrun Error Flag
CWT	RCV_STATUS	CWTM	INT_MASK	Character Wait Time Counter Comparator Flag
BWT	RCV_STATUS	BWTM	INT_MASK	Block Wait Time Counter Comparator Flag
BGT	RCV_STATUS	BGTM	INT_MASK	Block Guard Time Counter Comparator Flag
RTE	RCV_STATUS	RTM	INT_MASK	Receive NACK Threshold Error
SDI1	PORT1_DETECT	SDIM1	PORT1_DETECT	SIM Detect Interrupt for port 1
SDI0	PORT0_DETECT	SDIM0	PORT0_DETECT)	SIM Detect Interrupt for port 0

39.5 Initialization/Application Information

This section describes the intended programming model for using the SIM module. The section describes how to begin a typical mode of operation using the registers.

39.5.1 Configuring SIM for Operation

The following is a list of items that need to be performed in order to configure the SIM module for operation:

1. Port selection in the SETUP register.
   a) Select which SIM module port is active by writing the SPS bit.
   b) Select whether the second SIM module port is active at the same time under the control of an alternate SIM module by writing the AMODE bit.

2. Enable the selected port (for port 1, see PORT1_CNTL register; for port 0, see PORT0_CNTL) following the SIM power up procedure specified in ISO 7816.
   a) Enable power to the SIM card using the SVEN1 or SVEN0 bit.
   b) Enable SIM module transmit data output using the STEN1 or STEN0 bit. This is required to allow the SIM receiver to create NACK pulses.
   c) Enable SIM card clock using the SCEN1 or SCEN0 bit.
   d) Release the SIM card reset using the SRST1 or SRST0 bit (assert SRSTx high).
3. Select XMT port driver type (open-drain or push-pull) in the OD_CONFIG register.
   a) Configure the selected port to be open drain or push-pull by using the OD_P1 or OD_P0 bit.
4. Choose the port Baud rate in the CNTL register.
   a) Select the SIM card clock rate by using the CLK_PRESCALER[7:0] register
   b) Select the SIM card baud rate by using the BAUD_SEL[2:0] bits and SAMPLE12

NOTE

Follow the ISO 7816 specification with regards to card clock frequencies to ensure the maximum frequency specification is not violated.

5. Select Data format type, or place SIM receiver in initial character mode.
   a) Select inverse convention or direct convention by using the IC bit in the DATA_FORMAT register, or
   b) Enable initial character mode by using the ICM bit in the CNTL register

39.5.1.1 Configuring SIM Receive

The following is a list of items that need to be performed in order to configure the SIM receiver for operation:

1. Enable NACK capability in CNTL register.
   a) Select NACK generation on parity errors, or invalid initial character by using the ANACK bit.
   b) Select NACK generation on overrun conditions by using the ONACK bit.
2. Select the desired Receive NACK threshold and receive FIFO threshold in RCV_THRESHOLD register.
   a) Program the threshold at which the RDRF flag will be set by using the RDT bits.
   b) Program the threshold at which the RTE flag will be set by writing to the RTH[3:0] bits. If an auto power down after RTE flag is set, is desired, then enable the SIM card auto power down by setting the SAPD0,1 bits in the port0,1_cntl register.
3. Configure the Character Wait Time Counter, the Block Wait Timer Counter and the Block Guard Time Counter.
4. Enable interrupts in the INT_MASK register.
   a) Enable the receive data register full interrupt by using the RIM bit
   b) Enable the receive threshold error interrupt by using the RTM bit.
   c) Enable the overrun condition interrupt by using the OIM bit

d) Enable the Character Wait Time interrupt by using the CWTM bit

39.5.1.2 Configuring SIM Transmitter

The following is a list of items that need to be performed to configure the SIM transmitter for operation:

1. Select desired re-transmission threshold for NACKed characters in XMT_THRESHOLD register.
   a) Program the threshold at which the XTE flag will be set by using the XTH[3:0] bits
2. Select the guard time between transmissions in the GUARD_CNTL register.
   a. Program the desired guard time between characters transmitted by the SIM module by using the GETU[7:0] bits
3. Select the desired Transmit FIFO threshold level in the XMT_THRESHOLD register.
   a) Program the desired threshold using the TDT[3:0] bits
4. Enable interrupts in the INT_MASK register.
   a) Enable the transmit complete interrupt by using the TCIM BIT
   b) Enable the early transmit complete interrupt by using the ETCIM bit
   c) Enable the transmit FIFO empty interrupt by using the TFEIM bit
   d) Enable the transmit threshold error interrupt by using the XTM bit
   e) Enable the transmit FIFO threshold interrupt by using the TDTFM bit
   f) Enable the transmit FIFO overfill interrupt by using the TFOM bit

39.5.1.3 Configuring SIM General Purpose Counter

The following is a list of items that need to be performed in order to configure the SIM General Purpose Counter for operation:

1. Select desired clock source for the General Purpose Counter using the CNTL register.
   a) Use the GPCNT_CLKSEL[1:0] bits to select the desired clock source for the counter
2. Program counter comparator using the GPCNT register.
   a) Use the GPCNT[15:0] bits to select the desired count value at which the GPCNT interrupt flag will be set.
3. Enable the selected clock source for the General Purpose Counter using either the RESET_CNTL or ENABLE registers.
   a) If the GP Counter is configured for the card clock, enable the clock by clearing the KILL_CLOCK bit in the RESET_CNTL register.
   b) If the GP Counter is configured for the receive oversample clock, enable this clock by setting the RCV_EN bit or XMT_EN bit in the ENABLE register.
   c) If the GP Counter is configured for the transmit oversample clock, enable this clock by setting the XMT_EN bit in the ENABLE register.
4. Enable interrupts in the INT_MASK register.
   a) Enable the general purpose counter interrupt by using the GPCNTM BIT

39.5.1.4 Configuring SIM to Measure WWT (Work Wait Time) for Type=0 Smartcards

The following is a list of items that need to be performed in order to configure the SIM to measure work wait time. This is intended for type=0 SmartCards. Work wait time is a combination of BWT and CWT. The CWT timer is to measure the time between the start bits of two consecutive characters in the block received from the SmartCard. If this time is larger that the value programmed in the cwt[15:0] register, then a flag will be set and an interrupt generated. This timer can be used for both type=0 and type=1 SmartCards. The BWT and BGT timer are used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If you want the SIM to enforce a WWT of 100 bits. Then, you must activate both the CWT and BWT, and program both of them for a value of 100 bits. When measuring WWT, the BGT timer will not be used so it will be masked (inactive) by the BGTM bit. Below is the sequence to program the SIM to send and receive data while checking WWT.

1. Program both the CWT and the BWT (low and high) registers to the WWT (work wait time) value that needs to be enforced. Set BGT register to 0 (this is the default value).
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT and BWT interrupts by clearing the CWTM and BWTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to PORT0,1_XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the PORT0,1_XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the PORT0,1_XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should consider that a WWT violation. The software should clear the CWT or BWT interrupt by writing a “1” to the BWT or CWT bit in the RCV_STATUS register.

39.5.1.5 Configuring SIM to measure CWT, BWT, BGT for type=1 SmartCards

The following is a list of items that need to be performed in order to configure the SIM to measure CWT, BWT, BGT. This is intended for type=1 SmartCards. The CWT timer is to measure the time between start bits in the consecutive characters received from the SmartCard. If this time is larger that the value programmed in the CWT[15:0] register, then a flag will be set and an interrupt generated. This timer can

be used for both type=0 and type=1 SmartCards. The BWT, BGT timer is used to measure the delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block sent by the card. If this time is larger than the value in the both BWT registers (BWT and BWT_H), then the BWT flag will be set and an interrupt generated. If the time is less than the value in the BGT register, then the BGT flag will be set and an interrupt generated.

1. Program the CWT, BWT, BWT_H and BGT registers to the value that needs to be enforced.
2. Activate both the CWT and BWT functions by setting the CWTEN and BWTEN bit in the CNTL register.
3. Enable the CWT, BWT, BGT interrupts by clearing the CWTM, BWTM, BGTM bits in the INT_MASK register.
4. Program the data to send to the SmartCard by writing data to PORT0_XMT_BUFFER or PORT1_XMT_BUFFER
5. Activate the SIM transmit and receive by setting bits XMT_EN and RCV_EN in the ENABLE register.
6. If the software has more data to send, then as the FIFO starts to get near empty, it should write more data to the PORT0,1_XMT_BUFFER. If the software does not have any more data to send then proceed to the next step.
7. When SmartCard has completed its transmission to the SIM module, disable the BWT by clearing the bit BWTEN in the CNTL register. This will prepare SIM to measure the next block wait time.
8. Disable the SIM transmitter by clearing the XMT_EN bit in the ENABLE register.
9. Program the next data to send by writing data to the PORT0,1_XMT_BUFFER.
10. Enable the BWT function by setting the BWTEN bit in the CNTL register.
11. Enable the SIM transmitter by setting the XMT_EN bit in the ENABLE register.
12. Steps 6 to 10 can be repeated for each transmission to the SmartCard.
13. If a CWT or a BWT interrupt occurs, then the software should read the RCV_STATUS register to determine if the error was caused by a CWT, BWT, or a BGT violation. The software should clear the CWT, BWT, BGT interrupt by writing a “1” to the CWT, BWT, BGT bit in the RCV_STATUS register.

39.5.1.6 Configuring SIM Linear Redundancy Check (LRC) Block

The following is a list of items that need to be performed in order to configure the SIM Linear Redundancy Check Block for operation:

1. Enable the LRC block by using the CNTL register
   a) Use the LRCEN bit to enable the LRC block
   b) Use the XMT_CRC_LRC bit to enable the transmission of the LRC Character after the last character in the Transmit FIFO is sent. See the T=1 programming model for more details.

39.5.1.7 Configuring SIM Cyclic Redundancy Check (CRC) Block

The following is a list of items that must be performed to configure the SIM Cyclic Redundancy Check Block for operation:

1. Enable the CRC block by using the CNTL register.
   a) Use the CRCEN bit to enable the CRC block
   b) Use the XMT_CRC_LRC bit to enable the transmission of the CRC Characters after the last character in the Transmit FIFO is sent. See the T=1 Programming model for more details.

39.5.2 Using the SIM Receiver

Once the SIM has been properly configured (such as correct baud rate and correct data format), SIM receptions can be enabled by setting the receive enable bit, RCV_EN, in the ENABLE register. As bytes are received, they will be placed in the 285 byte deep receive data FIFO. Unread bytes can be accessed from this FIFO at any time. There is no need to disable the receiver to access the FIFO. The FIFO should only be read when the receive FIFO data flag, RFD, in the RCV_STATUS register is set. The RFD flag, which cannot create an interrupt, is high any time there is at least one unread byte in the receive FIFO. If the receive FIFO is read when RFD is low, it will simply produce the last byte read.

The correct address to read the data contained in the receive FIFO depends on which SIM port is selected. The SPS control bit in the SETUP register controls port selection. If SPS = 0, read the data from the PORT0_RCV_BUF register; if SPS = 1, read the data from the PORT1_RCV_BUF register.

The receive data register full flag, RDRF, in the RCV_STATUS register can be used to determine when the receive FIFO has reached a given threshold value. This flag will create an interrupt if the RIM bit in the INT_MASK register is clear. To control at which point RDRF is set, program the receive data threshold, RDT, in the RCV_THRESHOLD register. If the number of unread bytes in the receive FIFO is equal to or greater than the value set by RDT, RDRF will be set.

NOTE

A value of 0x0 in RDT implies that there must be 285 unread bytes in the receive FIFO to trigger RDRF.

The value in RDT can be changed at any time to alter this threshold level. The comparison between the number of unread bytes in the FIFO and the value set by RDT is continuously updated so that any change in either will be immediately reflected in the state of RDRF. For instance, if RDT is set to 5, and there are 3 unread bytes in the FIFO, changing RDT to 2 will immediately cause RDRF to be set. Likewise, setting RDT back to 5 will cause RDRF to clear. Similarly, if there are 5 unread bytes in the receive FIFO and RDT is set to 3, RDRF will remain high until 3 reads are complete, assuming RDT is left as a constant and no new data is received.

The standard flow for receiving bytes from the SIM card is to set RDT to the appropriate value, wait for RDRF to cause an interrupt (RIM clear), and then read bytes out of the receive FIFO as long as RFD is high. In addition to checking RFD between every byte, it is also recommended that software check for the existence of a set OEF flag as well.

39.5.2.1 Receive Parity Errors and Parity NACK Generation

The SIM receiver checks every byte received for proper parity. The IC control bit in the DATA_FORMAT register controls whether it checks for odd parity or even parity. When checking for odd parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be odd. Likewise, when checking for even parity, the number of logic ones contained in the 9 received bits (8 data bits and 1 parity bit) should be even.

When a parity error is detected on a given byte, the PORTx_PE bit for that byte is set in the FIFO. The PORTx_PE flag for each byte is read out of the FIFO when the data itself is read. There is no need to attempt to clear the parity error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A parity error cannot cause an interrupt.

It is possible for the SIM to automatically request that the SIM card re-send a byte found to have a parity error by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse on a byte received with a parity error if the control bit ANACK in the CNTL register is set. Bytes with parity errors that cause a NACK pulse are still placed into the FIFO just as bytes that do not cause a NACK pulse. It is up to the software to discard data bytes with parity errors.

To control NACK generation by the SIM receiver use the RTH[3:0], Receive NACK threshold, in the RCV_THRESHOLD register. This set of bits specify the number of consecutive NACK's generated by the SIM module on a received byte, before generating an RTE (receive threshold interrupt flag) in the RCV_STATUS register. The RTE flag would also force the SIM port to power down the card if the SAPD0,1 bit is set in the PORT0,1_CNTL register.

NOTE

The ANACK bit must be set in the CNTL register to enable this feature. The control bit ANACK is also used in initial character mode to enable the retransmission of initial characters in the event that an invalid initial character is received.

When a valid character has been received by the SIM, the internal counter keeping track of the number of NACK's transmitted on the current byte resets to zero. Clearing the receive threshold error (RTE) bit would also clear that counter.

When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

39.5.2.2 Receive Frame Errors

The SIM receiver checks every byte received for a proper stop bit. A stop bit should exist during at least the first half of the 11th ETU time after the start of the character. If this is not true, a frame error is flagged. When a frame error is detected on a given byte, the PORTx_FE bit for that byte is set in the FIFO. The PORTx_FE flag for each byte is read out of the FIFO when the data itself is read. There is no need to clear the frame error flag in the FIFO. It is simply overwritten the next time a byte is received into that position of the FIFO. A frame error cannot cause an interrupt, nor can it create a NACK pulse to the receiver asking for a retransmission of the corrupted data.

39.5.2.3 Receive Overrun Errors and Overrun NACK Generation

When there already exists 285 unread bytes in the FIFO, a received character will cause the SIM receiver to flag an overrun condition. This condition will always set the overrun error flag, OEF in the RCV_STATUS register. The received byte will be discarded leaving the 285 unread bytes in the FIFO unaltered.

It is possible for the SIM to automatically request that the SIM card re-send the byte that caused the overrun condition by generating a NACK pulse on the SIM XMT pin. The SIM will generate a NACK pulse for the byte that caused the overrun condition if the control bit ONACK in the CNTL register is set. In this case, the existence of an OEF flag does not indicate the loss of data, but rather a NACK (that is, retransmission request) due to a full receive FIFO. As opposed to transmit NACK generation, there is no limit to the number of times an overrun condition will cause a NACK other than to disable ONACK itself. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETU.

If the control bit ONACK is clear, the existence of a high OEF flag indicates the loss of data. The OEF flag can create an interrupt if the OIM bit in the INT_MASK register is clear.

To clear the OEF flag, software must simply write a “one” to the OEF bit position in the RCV_STATUS register. A high OEF flag has no effect on the operation of the SIM receiver other than to create an interrupt if OIM is clear.

39.5.2.4 Using Initial Character Mode and Resulting Receive Data Formats

The SIM receiver supports the detection of special characters that allow it to determine what data format is being used by the connected SIM card. When placed in initial character mode, the SIM expects to receive one of two potential values that it will use to set the data format control bit, IC, in the DATA_FORMAT register.

The two possible data formats are inverse convention and direct convention. Essentially, inverse convention differs from direct convention in that the order of the data is flipped MSB for LSB, and the data bits and parity bit are logically inverted. When receiving inverse convention data, the transformation of the data back to direct convention format is done in hardware, including the inversion of the data and parity bits.

To place the SIM into initial character mode, set the ICM bit in the CNTL register. Once a valid initial character is received, the IC bit in the DATA_FORMAT register will be set accordingly by the hardware, and the ICM bit will be cleared. Software can read the state of the IC bit to determine which mode the SIM is currently using.

The 0x3B (as decoded by direct convention) with parity bit high will cause direct convention to be used (IC set to logic 0), whereas a 0x3F (as decoded by inverse convention) with parity bit high will cause inverse convention to be used (IC set to logic 1).

When the receiver is in initial character mode, all received bytes will continue to be placed into the receive FIFO whether they be valid initial characters or not. If a valid initial character is received that causes the data format being used to change, all subsequent bytes will be decoded with that format before being placed into the FIFO, including the initial character byte itself. That is, if the IC bit is low, and the correct

initial character for setting inverse convention is received, that character and all subsequently received characters will be stored in the FIFO after having been decoded using inverse convention (for example, the initial character will be stored as 0x3F).

If the receiver is in initial character mode (ICM is high), and an invalid initial character is received, the SIM can be configured to automatically request that the initial character be retransmitted by setting the ANACK control bit in the CNTL register. When generating a NACK pulse, the SIM will generate the low pulse starting at 10.5 ETUs and lasting for 1 ETUs. The invalid initial character will be placed into the receive FIFO and marked with a parity error to signify that this is an invalid initial character.

39.5.2.5 Initial Character Mode Programming Notes

The usage of the initial character mode requires close attention to the programming model. There is a condition where a parity error in the initial byte for direct convention (0x3B) could be decoded as what appears to be a valid initial character for inverse convention (that is, 0x3F). The SIM module will not recognize this as a valid initial character for inverse convention and will mark the character by setting the parity error flag. The software must look for the existence of a parity error before recognizing a character as a valid initial character.

39.5.2.6 Automatic Receiver Mode

The SIM module has an automatic receive mode that inhibits the data being transmitted by the SIM module from entering the SIM receive buffer through the feedback path of the SIM data pin. The SIM module receiver should normally be enabled while the transmitter is operational. Automatic receive mode saves the software from having to actively manage the transition from transmitter to receiver. The auto receive mode is always active whenever the receiver is enabled.

39.5.2.7 Using the SIM Receiver with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM receiver are as follows:

- 11 ETU Characters
  - “T=1” cards can transmit with character lengths of 11 ETUs (that is, one STOP bit). The SIM module provides the RCVR11 bit in the GUARD_CNTL register in order to configure the receiver state machine to accept 11 ETU characters.
- Character Wait Time Counter
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters received from the SmartCard. The value of CWT can range from 12 ETU to 32779 ETU. The SIM module provides a 16-bit counter with programmable comparator clocked at the ETU bit rate to identify when the CWT has been exceeded by the SIM card.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The block wait time must

not exceed a value that is programmable. The SIM module provides a 32-bit Block Wait Timer that can be used to identify when the BWT has been violated (divided in two registers).

- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the leading edges of two consecutive characters sent in opposite directions. The block guard must be greater than a value that is programmable. The SIM module provides a 16-bit Block Guard Timer that can be used to identify when the BGT has been violated.
- Error Detection Code
  - "T=1" cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operations.

39.5.3 Using SIM Transmitter

Once the SIM has been properly configured (such as data XMT enabled, correct baud rate, and correct data format), the transmitter can be enabled by setting the XMT_EN bit in the ENABLE register. If data had been previously written to the transmit FIFO, the transmitter will begin to send the first character. If no data is written to the transmit FIFO before enabling the transmitter, then the transmitter will wait until the first character is written before beginning transmission. Clearing the XMT_EN bit while the transmitter is in operation, will halt any transmission in progress, flush the transmit FIFO, and reset the transmit state machine.

Data can be written to the transmit FIFO at any time.

The transmit data threshold flag, TDTF, in the XMT_STATUS register can be used to determine when the transmit FIFO has reached a given threshold value. This flag will create an interrupt if the TDTFM bit in the INT_MASK register is clear. To control at which point TDTF is set, program the transmit data threshold, TDT[3:0], in the XMT_THRESHOLD register. If the number of bytes remaining in the transmit FIFO is equal to or less than the value set by TDT[4:0], TDTF will be set.

NOTE

A value of 0x0 in TDT[3:0] implies that the transmit FIFO must be empty to trigger TDTF.

The value in TDT[3:0] can be changed at any time to alter this threshold level. The comparison between the number of remaining bytes in the transmit FIFO and the value set by TDT[3:0] is continuously updated so that any change in either will be immediately reflected in the state of TDTF. Unlike the RDRF flag for the receive FIFO, TDTF is latched and remains set until the software writes a 1 to the TDTF bit location in the XMT_STATUS register. For instance, if TDT[3:0] is set to 5, and there are 6 bytes remaining in the FIFO, changing TDT[3:0] to 6 will immediately cause TDTF to be set. However, setting TDT[3:0] back to 5 will not cause TDTF to clear.

The standard flow for transmitting bytes from the SIM card is to set TDT[3:0] to the appropriate value, write up to 16 bytes to the transmit FIFO, enable the transmitter, wait for TDTF to cause an interrupt (TDTFM clear), and then write additional bytes to the transmit FIFO.

NOTE

For the SIM module to transmit successfully, the transmit pin must be connected to the receive pin of the same port. This connection is necessary for the SIM to decode transmit NACKs sent to it by the SIM card. Without this connection, all transmissions will appear to have a NACK thereby causing the first byte to be transmitted continuously. When operating in external one wire interface mode (3VOLT0 or 3VOLT1), this connection is made internal to the SoC.

39.5.3.1 Transmit Data Formats

There are two possible data formats that the SIM module uses when transferring data to the SIM: card — inverse convention or direct convention. The format used depends on the state of inverse convention bit (IC in the DATA_FORMAT register). Software can set the IC bit, or it can be set automatically by hardware when using the initial character mode.

39.5.3.2 Transmit NACK

The SIM transmitter can respond to NACKs created by the SIM card. A NACK will be decoded if the SIM card creates a logic low level on the SIM receive pin during the STOP bit time at the end of a transmitted byte. To prevent a situation where the SIM interface is stalled by an infinite number of NACK pulses on a given byte, the SIM module can be configured to limit the number of times it will respond to NACKs. The XTH[3:0] control field in the XMT_THRESHOLD register allows software to set a threshold on the number of times a given byte will be retransmitted. If the threshold is reached, a transmit threshold error, XTE in the XMT_STATUS register, is asserted.

When XTE is set, the SIM transmitter is halted, and all pending transfers are aborted, and the TC, ETC, AND TFE flags are set. All bytes remaining in the transmit FIFO are lost. There is no way to restart the transmission on the next byte in the FIFO. The transmitter remains frozen until XTE is cleared by software. The only way to clear XTE is to write a 1 to the XTE bit in the XMT_THRESHOLD register. The XTE flag can create an interrupt if the XTM mask in the XMT_THRESHOLD register is clear.

It is possible to disable the detection of NACKs from the SIM card by setting XTH[3:0] to 0x0. By setting XTH[3:0] to 0x1, it is possible to disable all retransmissions while still setting XTE on the first NACK received. In general, XTE is set on the NACK that causes the threshold set by XTH[3:0] to be reached. This final NACK will not cause a retransmission, whereas all previous NACKs will cause a retransmission.

39.5.3.3 Transmit Guard Time

The time between data bytes sent from the SIM transmitter can be altered using the transmit guard time control. By default, the minimum time between start bits of successive transmitted bytes is 12 ETUs (Elementary Time Units). An ETU is equivalent in time to 1 bit time. Therefore, the amount of time an ETU consumes is determined by the baud rate chosen. The 12 ETUs that form the default character transmission time consist of a start bit, 8 data bits, a parity bit and two stop bits. The number of stop bits (idle bits) can be extended by an integer number of ETUs. The number of additional ETUs can be programmed directly into the GETU[7:0] bits of the GUARD_CNTL register. Programming the

GETU[7:0] bits to 0xFF will configure the SIM transmitter to use only one stop bit for each character transmission.

39.5.3.4 Using SIM Transmit with “T=1” SIM Cards

The SIM module provides hardware support for “T=1” type SIM cards. These type of cards present several requirements above and beyond the standard “T=0” cards. The features provided to meet the requirements that pertain to the SIM transmitter are as follows:

- 11 ETU Characters
  - The SIM module transmitter has a programmable guard time register that allows the programmer to specify the number of ETUs between character transmissions. Programming a value of 255 (0xFF) in the GETU[7:0] bits in the GUARD_CNTL register will set the number of ETUs per character transmitted to 11.
- Character Waiting Time
  - The character waiting time (CWT) is defined as the time between the start bits of two consecutive characters. The value of CWT can range from 12 ETU to 32779 ETU. The time between transmitted characters is controlled by the programmable guard time in the GUARD_CNTL register. However, the time between the last byte in the transmit FIFO, and the next transmitted byte can be largely affected by software response time to the transmit interrupts. The SIM transmitter provides a Transmit FIFO threshold (TDTF) interrupt to signal the system when the expected number of characters have been transmitted from the transmit FIFO. The minimum CWT is achieved only if the software can respond to the TDTF interrupt and write new data to the transmit FIFO before the last character in the Transmit FIFO has been sent.
- Block Waiting Time
  - The block waiting time (BWT) is defined as the time between the start bits of the last character of a received block and the first character of the next received block. The value of BWT is always greater than 1800 ETU. The SIM transmitter provides a General Purpose Counter that can be used to track the BWT. The BWT is purely determined by software response time to the transmit interrupts.
- Block Guard Time
  - The block guard time (BGT) is defined as the minimum delay between the leading edges of two consecutive characters sent in opposite directions. The value of BGT is 22 ETU. The SIM module supports the BGT by providing the ability to generate an interrupt when the last byte is received, and transmitting within 2 ETU after the XMT_EN bit is set. The BGT will be determined by the speed at which the software can react to an interrupt and enable the transmitter.
- Error Detection Code
  - “T=1” cards can specify LRC or CRC error detection codes to be used. The SIM module provides hardware support for both the LRC and CRC operation.

39.5.4 Suggested “T=1” Compliant Programming Model

This section describes the suggested programming model for supporting “T=1”, “T=0”, and known “special” cards using the SIM module on the SoC. This should be used as a rough guide for how to configure the SIM module for use with SIM cards specified by ISO 7816-3 and EMV. Some details are not addressed. Other uses for some of the SIM features are not included (for example, GP Counter uses for some ISO timing requirements).

39.5.4.1 Answer To Reset (ATR) Detection

The first step to communicating with a SIM card is to provide power and a clock signal to the card. Once the card is detected as present (using the presence detect features, or some other method), the SIM card should be powered up according to the power up sequence specified in the ISO 7816-3 specification.

1. Apply voltage to the SIM card by setting the SVEN0 or SVEN1 bit in the PORTx_CNTL register
2. Select the appropriate clock frequency for the SIM card by programming the CLK_PRESCALER[7:0] register.
3. Enable the clock to the SIM card by setting the SCEN0 or SCEN1 bit in the PORTx_CNTL register
4. Remove the card from reset by setting the SRST0 or SRST1 bit in the PORTx_CNTL register

The first communication between the SIM card and the SIM module will be a block of data sent from the SIM card to the SIM module after the card is powered and the card reset is removed. This block is called the Answer To Reset (ATR). To receive the ATR, the SIM module should be configured for 12 ETU character reception. According to the ISO 7816-3 specification, both “T=0” and “T=1” cards will communicate initially using 12 ETU character durations.

NOTE

We are aware of some card manufacturers that communicate at 11.5 ETU character durations (Geldkarte). This will complicate the initial card detection sequence shown below.

5. Clear RCVR11 bit in the GUARD_CNTL register

The next item to configure for ATR reception is the NACK capability of the SIM module. The ISO 7816-3 specification allows the SIM module to NACK any communication errors that occur during the initial communication at 12 ETU.

NOTE

The Europay Mastercard and VISA (EMV) cards are similar to “T=1”, but do not allow the SIM module to NACK during the initial communication. This will again complicate the initial card detection sequence shown below.

6. Set ANACK bit in the CNTL register to enable NACK generation

In order for the SIM module to notify when characters are received, the receive interrupt(s) can be enabled. A threshold can be set for the number of characters to receive before generating an interrupt.

7. Enable RDRF and OEF interrupts by setting RIM and OIM bits in INT_MASK register
8. Set desired threshold for received characters by writing RDT in RCV_THRESHOLD register

The SIM should be setup to perform in initial character mode. This will cause the hardware to identify the first valid character sent during the ATR as an initial character. This character will automatically configure the hardware for the data convention used by the SIM card.

9. Set Initial Character Mode by setting the ICM bit in the CNTL register

The ISO 7816-3 specification requires that SIM cards meet certain timing restrictions. One of these is the time from the deassertion of the card reset to the beginning of the ATR sequence. The SIM module General Purpose Counter can be used to verify that the SIM card begins its ATR within the 400 to 40,000 clock cycle range.

1. Set General Purpose Counter Comparator to 0x9C40 using GPCNT register.
2. Enable General Purpose Counter Interrupt by clearing GPCNTM in INT_MASK register.
3. Enable General Purpose Counter by programming the GPCNT_CLKSEL[1:0] bits to 01 so the card clock is used for counting.

The ISO7816-3 specification states that the maximum allowed time between two characters during the ATR is 9600 ETUs (Initial Waiting Time). The Character Wait Time Counter should be setup to detect any errors for this condition.

1. Set Character Wait Time Counter Comparator to 9600 using the CHAR_WAIT register
2. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register
3. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register

The last step in preparing for ATR reception is to enable the receiver.

4. Set RCV_EN bit in ENABLE register

The SIM module will generate interrupts once a threshold number of characters is received. The software should react to these interrupts and read the characters from the receive FIFO (PORTx_RCV_BUF) until the complete ATR has been received. If a General Purpose Counter interrupt occurs before the final ATR character is received, then the card should be deactivated according to the ISO 7816-3 specification. Otherwise, once a valid ATR is received, the software will know from the ATR information the specific characteristics for this card (See the ISO 7816-3 specification for details).

39.5.4.2 Programming Considerations for Geldkarte Cards

There is at least one manufacturer of SIM cards (Geldkarte) that we know of that does not send the ATR in 12 ETU mode or support NACKs. This creates an issue with detecting a valid ATR from a SIM card. Since any kind of card can be attached to the SIM module, the software and hardware do not know how to begin communication. Basically, if the card fails to send a valid ATR on the first try, disable NACKs, configure the SIM module for 11 ETU and try again. The software should toggle between 12 and 11 ETU modes with and without NACKs enabled until a valid ATR is received, or the number of attempts to communicate passes a predetermined error threshold. Figure 39-53 shows the flow chart for the suggested Geldkarte Compliant SIM Initialization.

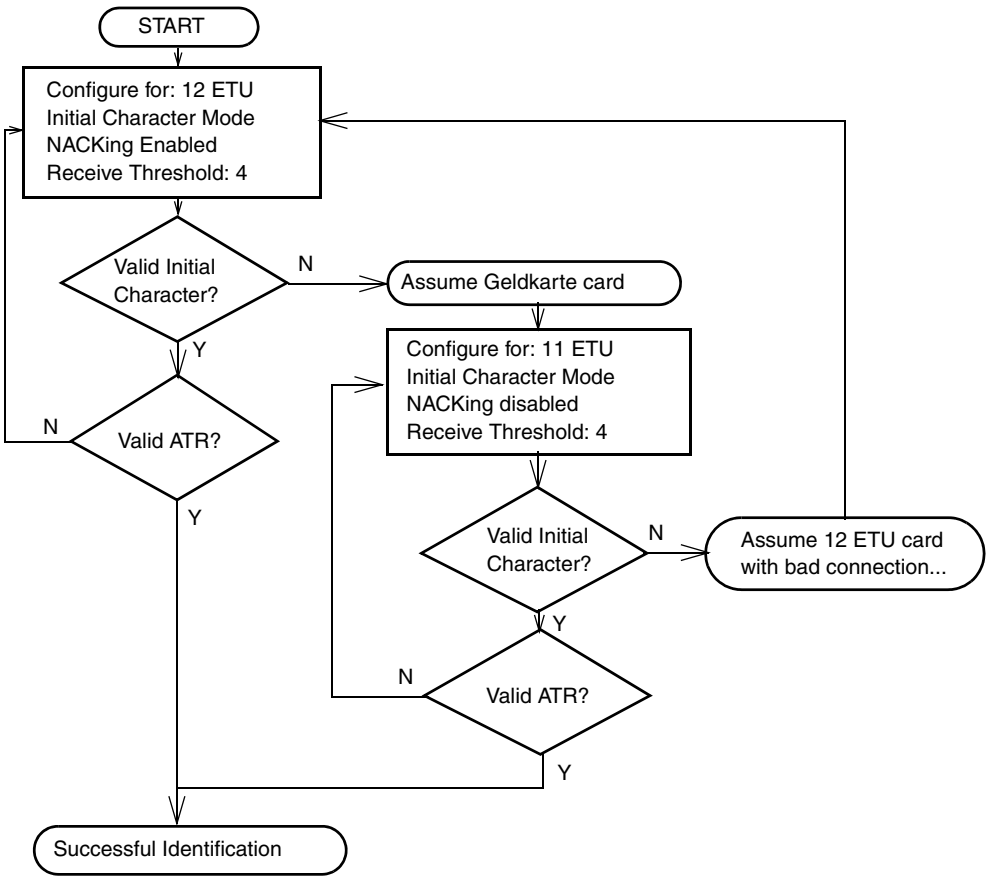


Figure 39-53. Suggested "T=1", EMV, Geldkarte Compliant SIM Initialization

39.5.4.3 Programming Considerations for T=0 SIM Cards

If the card is of type T=0, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register
3. Adjust NACK capability by modifying the values of the ONACK and ANACK bits in the CNTL register
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Adjust the level of transmit NACK re-transmissions allowed by modifying the value of the XTH[3:0] bits in the XMT_THRESHOLD register
6. Adjust the level for the Receive NACK threshold by modifying the RTH[3:0] bits in the RCV_THRESHOLD register.

If a negotiation with the SIM card is desired, the software sends a PPS response to the SIM card. To send the response, the following steps should be performed:

1. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
2. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register
3. Clear all transmit interrupt flags in the XMT_STAT register by writing a one to them
4. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
5. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=0 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

39.5.4.4 Programming Considerations for T=1 SIM Cards

If the card is of type T=1, the software should adjust the following parameters according to the information in the ATR:

1. Adjust the baud rate by changing the values of BAUD_SEL[2:0] and SAMPLE12 in the CNTL register
2. Adjust the guard time between characters by changing the value of GETU[7:0] in the GUARD_CNTL register. Setting GETU[7:0] to 0xFF configures the SIM transmitter for 11 ETU transmissions
3. Disable NACK capability by clearing the ONACK and ANACK bits in the CNTL register. T=1 cards do not allow NACKs.
4. Adjust the stop clock polarity by modifying the values of the SCSP0 or SCSP1 bits in the PORTx_CNTL register
5. Set Character Wait Time Counter Comparator to value specified in the ATR by using the CHAR_WAIT register
6. Enable the Character Wait Time Counter Interrupt by clearing CWTM in INT_MASK register
7. Enable the Character Wait Time Counter by setting the CWTEN bit in the CNTL register
8. Enable CRC or LRC error checking according to the ATR information by setting either the CRCEN or LRCEN bit in the CNTL register. These bits will never be set at the same time!

For T=1 cards, the ATR is sent using a T=0 type of structure (12 ETU, no LRC or CRC). If a negotiation with the SIM card is desired, the software will send a PPS response to the SIM card. Otherwise, the

protocol is initiated with a block transfer from the SIM module. In order to send the response or the first block, the following steps should be performed:

9. Set the desired transmit FIFO threshold level by writing the TDT[3:0] bits in the XMT_THRESHOLD register
10. Write the characters to be sent as response (max 16) to the transmit FIFO using the PORTx_XMT_BUF register
11. Clear all transmit interrupt flags in the XMT_STAT register by writing a 1 to them
12. Enable the transmit interrupts desired by clearing the mask bits in the INT_MASK register. If more than 16 character are to be sent, it is suggested that the TDTF interrupt be used to signify when to write more characters to the transmit FIFO. This results in the most efficient transfer times to the SIM card.
13. Enable the transmission of the error checking characters (LRC or CRC) by setting the XMT_CRC_LRC bit in the CNTL register.

NOTE

If the card supports PPS, the software may not be allowed to send the LRC/CRC information until the PPS exchange is completed. If so, do not set the XMT_CRC_LRC bit during the PPS exchange.

14. Enable the transmitter by setting the XMT_EN bit in the ENABLE register

At this point, the SIM module will transmit the characters in the transmit FIFO. If more than 16 characters are to be sent, the transmit threshold interrupt will be set when the threshold number of characters are remaining in the FIFO. The software can then write an additional number of characters to be sent without interrupting transmission to the SIM card.

Once the transmission is complete, the SIM module should be completely configured for standard operation with the T=1 SIM card. The software can continue to service RDRF interrupts for received characters, and TDTF interrupts for transmitted characters.

39.6 Definitions, Acronyms, and Abbreviations

CRC	Cyclic Redundancy Checking
LRC	Linear Redundancy Checking

ISO/IEC 7816 Smart Card Standards

Chapter 40
Secure JTAG Controller V1.1 (SJC)

This chapter presents SJC, an enhanced version of the original, first-generation Secure JTAG Controller.

40.1 SJC Overview

JTAG access is important to many different users, including:

- Software developers depend on JTAG-based development and profiling tools.
- Silicon validation engineers—similar to Software Developers
- SOC Design teams—when debug needs to be performed on silicon
- Production Engineering—manufacturing tests depend heavily on the JTAG interface for pre-configuration of SOCs and test pattern download
- Failure Analysis Engineering
- End Customers—in final applications, JTAG may be the only debug and monitoring interface
- End Customers—boundary scan for final PCB-level product test

The SJC module supports the IEEE P1149.1 v2001 standard. Features include access to the embedded debug control and observability logic on each core. In addition, SJCv1.1 includes security features to prevent improper use of end products.

SJC is a modified version of the original SJC module that has been used on many of the FSL wireless parts to date. It addresses a known issue with previous generation SOC-level JTAG implementations involving certain ARM-supplied IP (ARM 9 core and DAP). These IP blocks depends not only on TCLK, but also on the functional clock to perform JTAG shift operations. When the functional clock is off or the block is powered down, the JTAG shift chain cannot propagate into or through these modules. With the new bypass capability in SJC, the problem-blocks can now be bypassed during clock-off or power-down scenarios. JTAG shift chain continuity is then assured, allowing other SOC-related JTAG operations to take place. Use-case scenarios include the following:

- Debug during power-down of ARM IP blocks
- Debug during clock-off of ARM IP blocks (low power modes, PLL re-lock scenarios, etc)
- Debug during other scenarios in which the ARM core is not responding to JTAG/debug control
- Need guaranteed read-access to the ExtraDebug registers, particularly status registers, even when a hardware fault exists

In addition to the ARM IP bypass capability, the SJC also provides the following capabilities:

1. Supports JTAG IEEE 1149.1 mandatory instructions, see Section 40.4.3.3, “EXTEST Instruction,” Section 40.4.3.2, “SAMPLE/PRELOAD Instruction,” Section 40.4.3.5, “BYPASS Instruction”

2. Supports JTAG IEEE 1149.1 optional instructions, see Section 40.4.3.1, “ID_CODE Instruction,” Section 40.4.3.4, “HIGHZ Instruction.”
3. Provides means of accessing each OnCE/ICE TAP controller independently to control a target system (see section Section 40.2.1, “Modes of Operation”).
4. Provides means of accessing the ExtraDebug logic (see Section 40.4.3.6, “ENABLE_ExtraDebug Instruction”).
5. SJC operates at maximum 1/8 the slowest frequency of the accessed OnCE/ICE. For example in normal operation (no core in low-power mode), this frequency will be 1/8 of the SDMA frequency if this core is present in the TDI-TDO chain (serially connected with other cores and not bypassed).
6. Cores compliant mode to support standalone core debuggers (see Section 40.2.1, “Modes of Operation”).
7. Multi-cores daisy chained mode (default one) to support multi-core debuggers (see Section 40.2.1, “Modes of Operation”).

40.2 SJCv1.1 Block Diagram

Figure 40-1 is a block diagram of SJC, configured in daisy chain mode and including the new ARM TAP controller bypass block. Shown in red are the bypassed TAPs in the i.MX25. Within the SJC block is the support/interface for up to three CPU cores, the ARM core trace buffer, and ARM’s Debug Access Port (DAP) as follows:

1. Smart DMA controller (SDMA), which includes a 32-bit micro-RISC core
2. ARM 9 core with or without a JTAG-accessed Embedded Trace Buffer (ETB)
(note: Coresight ETB on later platforms does not include JTAG access)
3. ARM’s CoreSight Debug Access Port (DAP)

The SJC and each of these JTAG-accessible blocks contains its own Test Access Port (TAP) interface.

40.2.1 Modes of Operation

SJC includes two primary configuration options as follows:

1. SJC primary TAP only (single IR, single DR with one-bit DR-BYPASS)
   — referred to as “Compliant mode” for IEEE P1149.1 specification compliance
   — typically used for end-product boundary scan and other IEEE P1149.1-compatible applications
2. All TAPS enabled and connected in series - SJC TAP, SDMA TAP, DSP TAP, ARM MCU TAP, and either an ETB or the DAP TAP
   — referred to as “daisy chain mode”
   — typically used for debug- and development mode operation
   — all TAPs are connected in series and accessed using a single, long JTAG scan chain

The desired mode listed above is controlled through the value of the *sjc_mode* input pin on the SOC. This pin is sampled at TRST and selects between the two modes as follows:

- Negating it (the default state) will enable all TAPs (SJC, SDMA, DSP, and ARM/ETB/DAP) to be connected in the TDI-TDO chain (daisy chain mode).

- Asserting it will enable only the SJC TAP, bypassing all other TAPs (compliant-mode).

When operating in the daisy chain mode, additional options exist - the SDMA core may be bypassed within the SJC. Bits located in the SJC TAP Select Register shown in Table 40-1.

Table 40-1. TAP Select Register Configuration Options

Tap Select Bit	Bit Name	Reset Value	Affected TAP
Bit 0	Connect_SDMA	0 - not connected	SDMA

Conceptually, the TAP Select Register is a data register which is accessed through “Access TSR” IR instruction of SJC TAP.

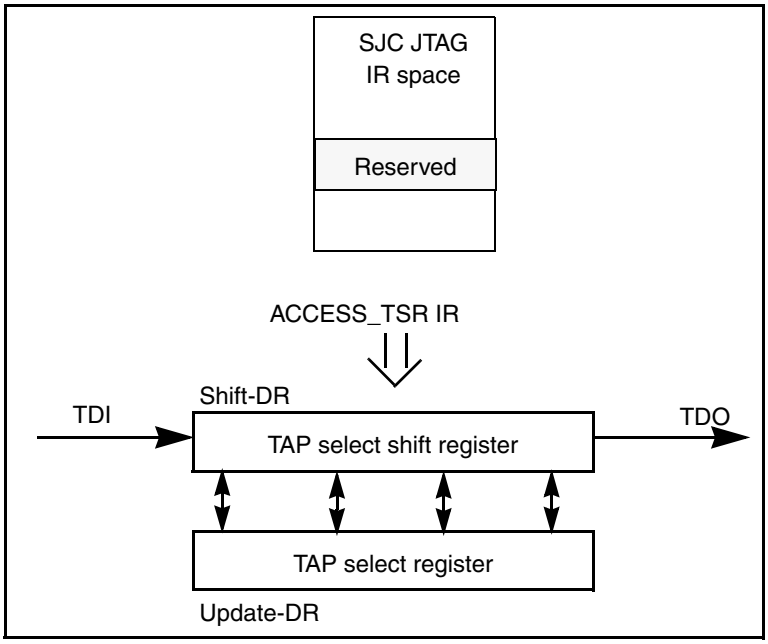


Figure 40-1. Using Reserved IR to Access TAP Select Register

The TAP Select Register is only allowed to change during the update-DR state of the TSB JTAG state machine. This is required to prevent a TAP that is being selected from losing synchronization with the TSB state machine as the TSB state machine returns to run-test-idle. Therefore, there is an associated shift register for the TAP Select Register which is loaded into the TAP Select Register during the update-DR state (see Figure 40-1). The shift register must also capture the state of the TAP Select Register when in the Capture-DR state for visibility of the contents of the TAP Select Register. See Chapter 40.4.3.8, “TAP Select Instruction” for more information.

Bit 0 of the TAP Select Register controls bypass of the SDMA core as follows:

- When negated (should be the default state), SDMA TAP is bypassed with a single D-FF during Shift-Dr path.
- When asserted SDMA TAP is connected inside the chain.

SDMA bypass is handled within the block inside the SJC that is labelled “SDMA TAP Ctlr w/Bypass” in Figure 40-1.

When operating in the daisy chain mode the ARM domain may be bypassed within the SJC. Bits located in the SJC Extra Debug DCR[8] is used in the i.MX25 to enable or disable bypass for each block shown in Table 40-2.

Table 40-2. DCR Register Configuration Options

DCR Bit	Bit Name	Reset Value	Affected TAP
Bit 8	Connect_MCU/DAP	0 - connected	ARM 9 Platform - Core and ETB DAP or DAP_SYS

Conceptually, the DCR[8] is a data register which is accessed through “Extra Debug” IR instruction of SJC TAP.

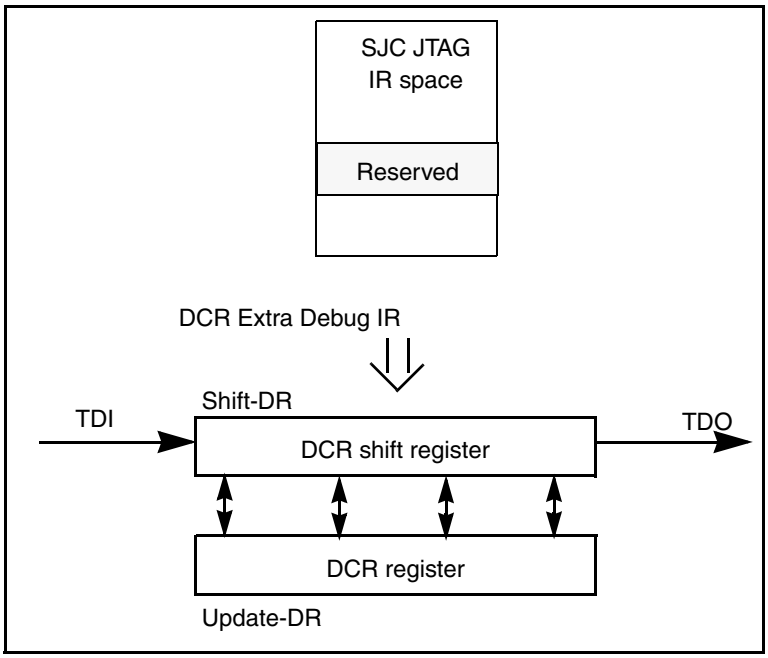


Figure 40-2. Using Reserved IR to Access DCR Extra Debug Register

The DCR Extra Debug is only allowed to change during the update-DR state of the TSB JTAG state machine. This is required to prevent a TAP that is being selected from losing synchronization with the TSB state machine as the TSB state machine returns to run-test-idle. Therefore, there is an associated shift register for the DCR Extra Debug which is loaded into the TAP Select Register during the update-DR state (see Figure 40-2). The shift register must also capture the state of the DCR Extra Debug when in the Capture-DR state for visibility of the contents of the DCR Extra Debug. See Chapter 40.4.3.6, “ENABLE_ExtraDebug Instruction” for more information.

Bits 8 of the DCR Extra Debug Register control bypass of the ARM domain blocks. The need for bypass of the ARM domain blocks (cores, ETB, and/or DAP) and the mechanism for doing so is described in the following sections.

Figure 40-3 describes the SJC instantiation connectivity in case of TAPs bypass option required, the GPUCR3 register bit 0 and 1 connected to arm_bypass and dap_bypass according to Table 40-2.

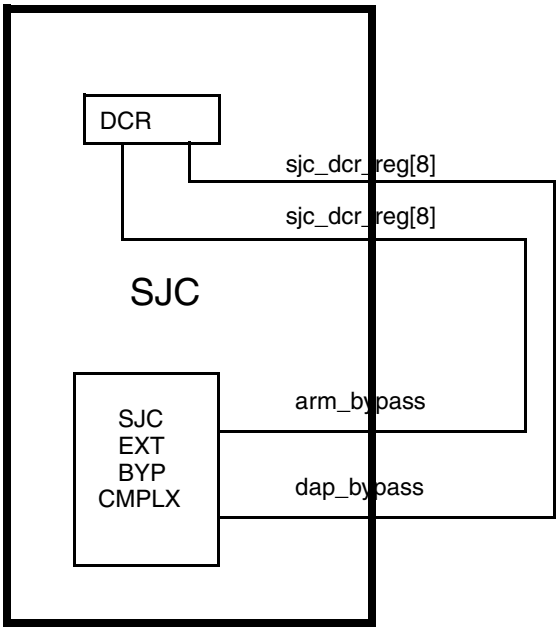


Figure 40-3. DCR Connectivity for TAPs Bypass

40.2.2 ARM IP Dependency on Functional Clock and RTCK

To illustrate the need to bypass the ARM domain blocks, first consider the JTAG clocking mechanism that is common to the ARM blocks. As mentioned earlier, the ARM IP blocks depend not only on TCLK to perform JTAG operations, but also the functional clock. This is illustrated in ARM 9 core documentation. The TCLK signal is translated to the functional clock domain, after which all JTAG operations inside of the ARM IP blocks take place in the functional clock domain.

The signal “RTCK” is referred to as Return-TCK and is an ARM-defined extension to the standard JTAG interface. This signal is routed directly from the ARM platform to the SOC pin and is not affected by or used by the SJCv1.1 module. RTCK may be used by development tools to control the TCK clock rate such that it does not overrun the clock synchronization logic. This mode of operation is referred to as adaptive clocking and is useful when the ARM core frequency changes dynamically, usually associated with low power modes. However, RTCK only reflects the relationship between the ARM core’s functional clock rate and TCK. The clock rates of other cores is not taken into account. Therefore, the end user is advised to use RTCK and adaptive clocking only when the ARM core is active with all other cores in bypass mode.

Note that the bypass capability added to SJC does not fix the ARM IP dependency on a running functional clock and thus does not enable JTAG operations to ARM IP when the functional clock is stopped to those blocks. It only provides a bypass mechanism for the ARM domain which enables JTAG operations on other parts of the SOC.

40.2.3 ARM Core/ETB/DAP Bypass

When in daisy-chain mode, the three basic bypass scenario options exist as follows:

- Platform-level bypass for the ARM 9 platform

- IP block-level bypass for DAP_SYS, which exists at the SOC level and not within a platform

NOTE

The i.MX25 only uses the ARM 9 platform bypass and DAP_SYS block-level bypass scenarios.

For all scenarios, “bypass” applies to both the IR- and the DR-shift paths. The IR-shift path is bypassed through a shadow register inside of the SJC. That shadow register matches the length of the IR register in the bypassed block/platform. To development tools, the IR shift path length is the same, whether in core-bypassed mode or not. The IR lengths are shown in Table 40-3.

Table 40-3. ARM IP JTAG TAP IR Lengths

ARM IP block	IR Length
ARM 9 core	4 bits
DAP	4 bits
ETB	4 bits

The bits control TAP bypass as follows

- When asserted, all ARM domain IR- and DR-paths are bypassed if their associated JTAG shadow IR contains the IEEE P1149.1-mandatory BYPASS instruction. The IR bypass logic mimics the exact length of the ARM domain’s IR register length to minimize impact to development tools JTAG algorithms. The DR path is bypassed with the traditional single D-FF register stage.
  - Note that if bypass is desired, but the corresponding shadow IR register does not contain the BYPASS instruction, that instruction can be shifted into the shadow register. While shifting the BYPASS instruction into the shadow IR is guaranteed under all conditions, depending on the state of the ARM IP, it may or may not be shifted into the actual ARM IP block at the same time. Once the BYPASS instruction has been shifted into the shadow registers, and the ARM/DAP bypass bit is asserted in the DCR[8] register, the ARM and DAP TAP’s configured in this manner will be bypassed.
- When negated (which is the default state), the ARM domain IR- and DR-paths are not bypassed. The JTAG shift chain routes through the ARM IP blocks and all JTAG-based resources are accessible, assuming no issues exist with the ARM IP’s functional clock.
- When this bit is set or cleared, the SJC JTAG TAP controller must transition to the Run-Test-Idle state for the actual change to take effect.

40.2.4 ARM Bypass Use-Case Scenarios

Figure 40-4 is a pseudo-schematic of the logic and signal paths associated with bypassing the ARM cores, ETB, and/or DAP. Following this are several pages depicting the paths for various use-cases. The use-cases are outlined in two tables - one for the ARM 9 Platform scenarios (Table 40-4), which is the i.MX25 concerns. Some general notes:

- During SHIFT-IR operations, the shadow-IR’s in the SJC always receive the shift stream input in parallel with the IR’s of the ARM core, ETB, and/or DAP.
  - The shift path halts at the end of the shadow-IR when the external logic is not bypassed

  - The path extends to the SJC's shadow TDO when the external logic is bypassed
- SHIFT-DR when in BYPASS always routes through the 1-bit DR-BYPASS reg inside of SJC for each ARM TAP that is bypassed.
  - 1 bit for the ARM core (ARM 9)
  - 1 bit for the ETB
  - 1 bit for the DAP
- The logic/path used for IR-shadow and DR-bypass for the ETB is also used for the DAP since they have identical IR-lengths and are in the same TAP position in the daisy chain topology.

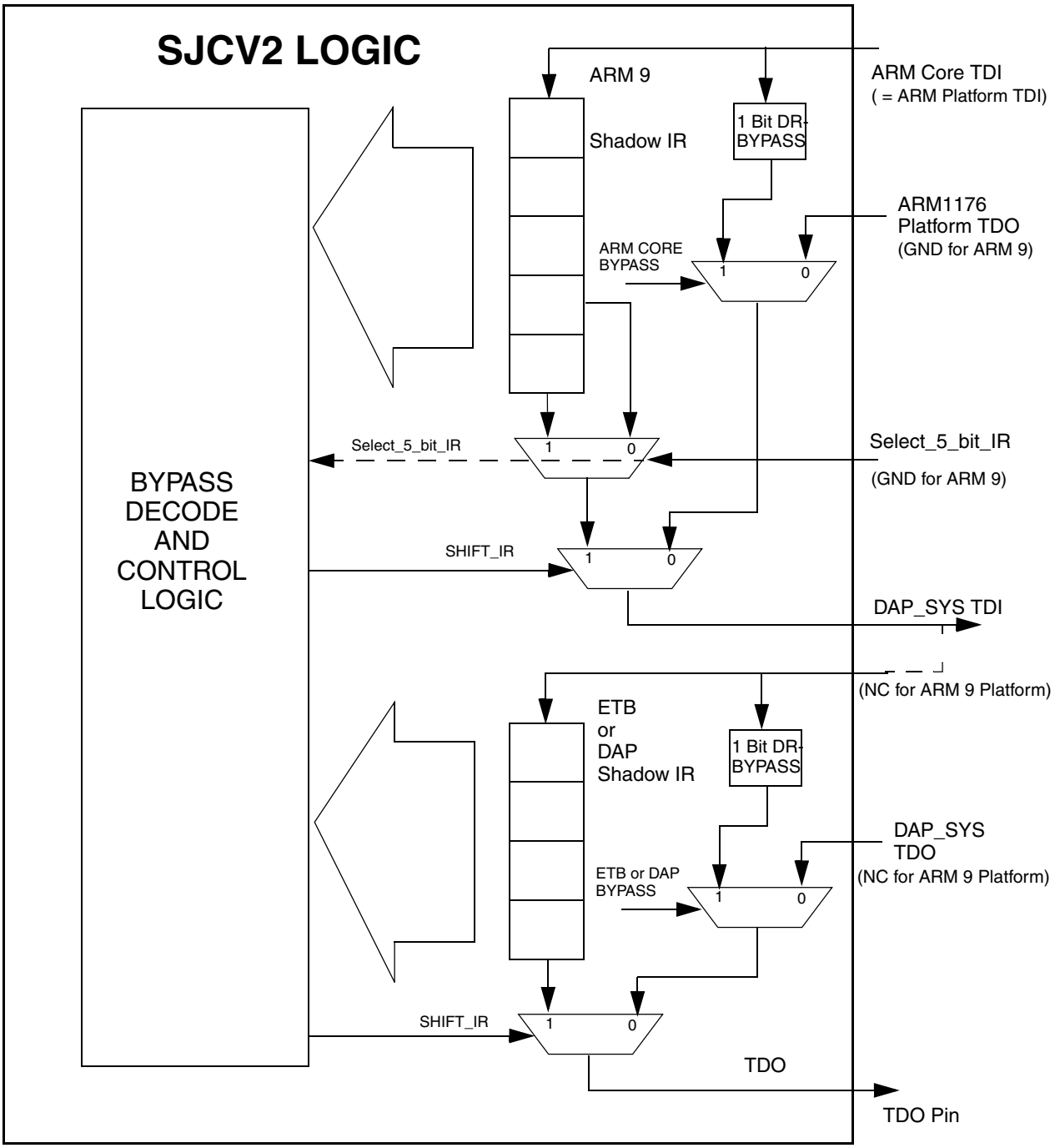


Figure 40-4. ARM Domain Bypass Mechanism

Table 40-4. ARM 9/ Platform—ARM 9+ ETB

Shift State	ARM Bypassed ?	ETM bypassed?	Figure	Comments
Shift-IR	NO	NO	Figure 40-5	Note: The ARM 9 Platform includes an internal tie between the ARM core and the internal ETB. This node is not available externally, and no current implementation supports the possibility of the ARM core and the ETB not being hard-wired in series or having different clock and power states. Therefore, scenarios separating the states of the two are not directly supported. They are treated the same as the first case, in which no bypass is enabled within the SJC and the full scan path routes through the external IP blocks.
Shift-IR	NO	YES	Same as Figure 40-5	
Shift-IR	YES	NO	Same as Figure 40-5	
Shift-IR	YES	YES	Figure 40-6	
Shift-DR	NO	NO	Figure 40-7	
Shift-DR	NO	YES	Same as Figure 40-7	
Shift-DR	YES	NO	Same as Figure 40-7	
Shift-DR	YES	YES	Figure 40-8	

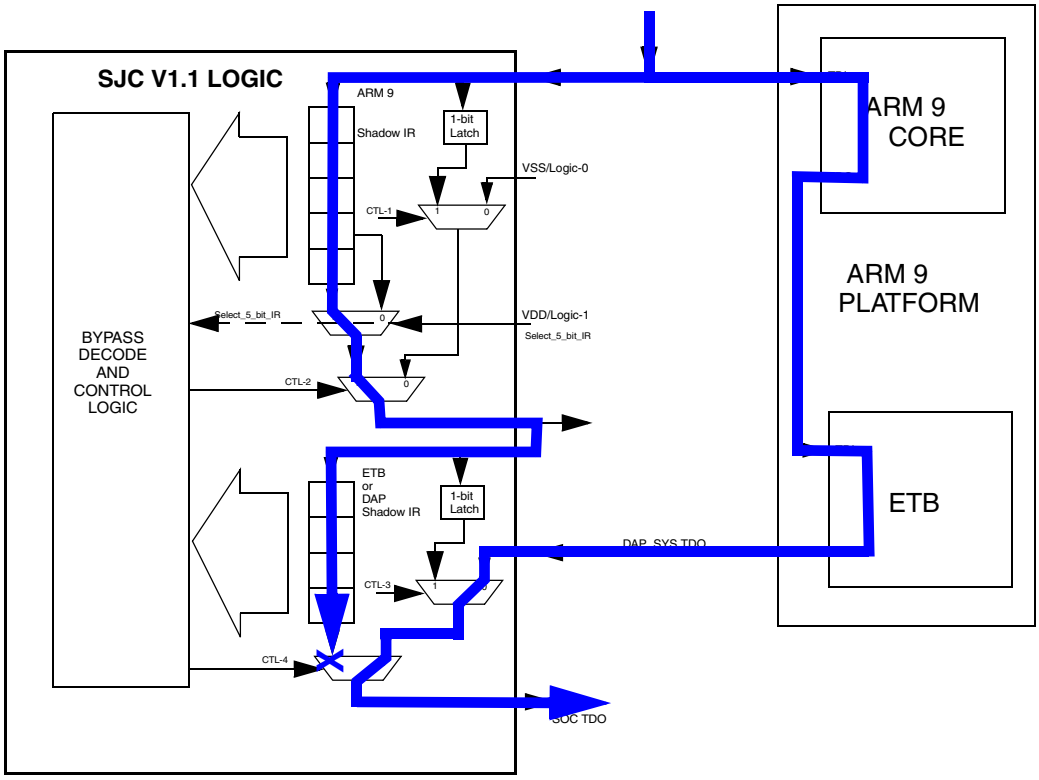


Figure 40-5. ARM 9 PLATFORM SHIFT_IR PATH ARM CORE = NOT-BYPASSED ETB = NOT-BYPASSED

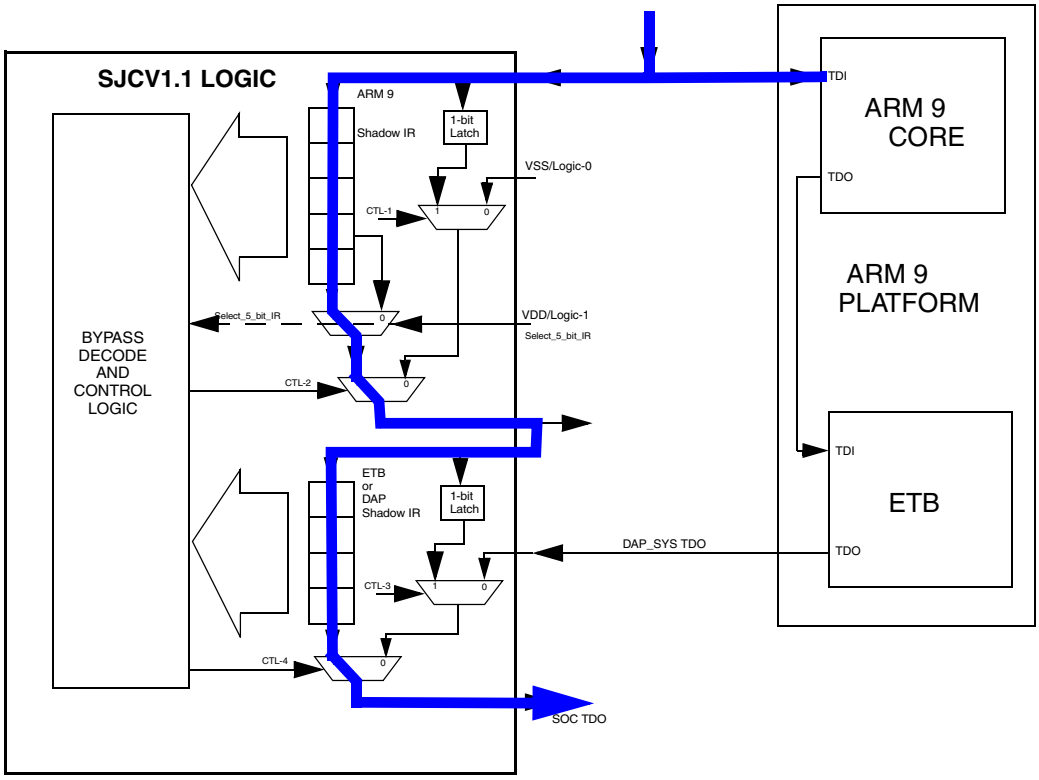


Figure 40-6. ARM 9 PLATFORM SHIFT_IR PATH ARM CORE = BYPASSED ETB = BYPASSED

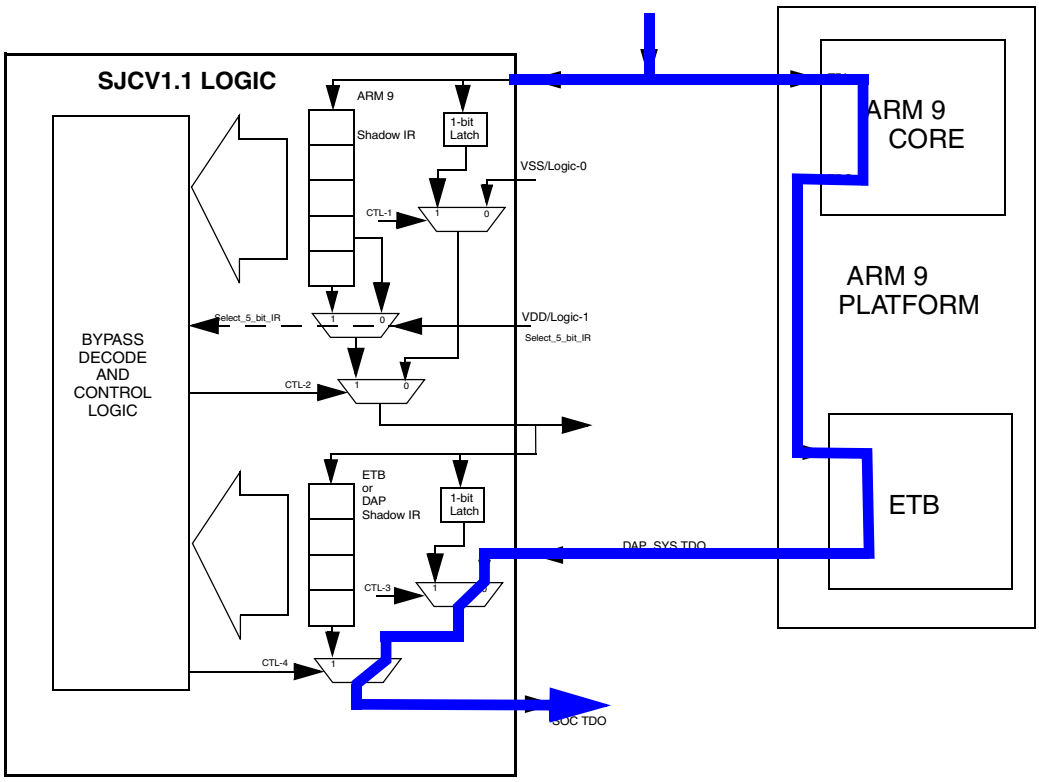


Figure 40-7. ARM 9 PLATFORM SHIFT_DR PATH ARM CORE = NOT-BYPASSED ETB = NOT-BYPASSED

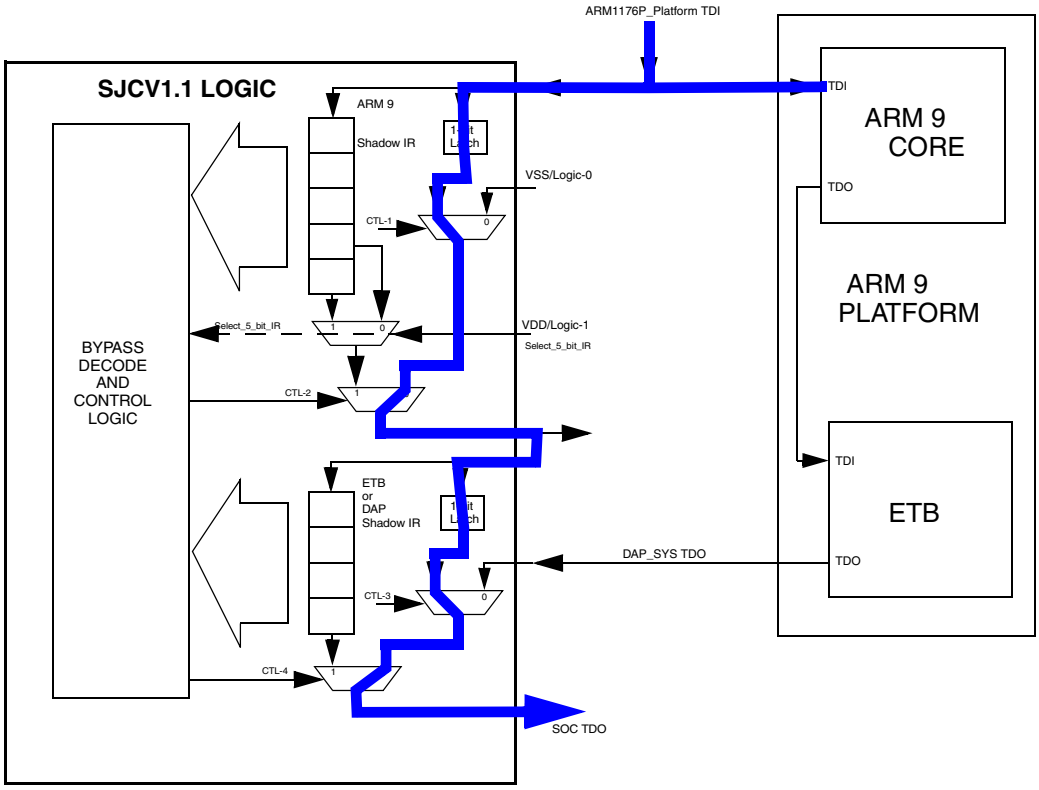


Figure 40-8. ARM 9 PLATFORM SHIFT_DR PATH ARM CORE = BYPASSED DAP_SYS = BYPASSED

40.3 External Signal Description

40.3.1 Overview

The SJC provides test and debug control with the minimum pincount. Figure 40-9 shows SJC connections external and to SoC sub-blocks. See integration guide for more detailed list of internal connections in the SoC from SJC to other modules.

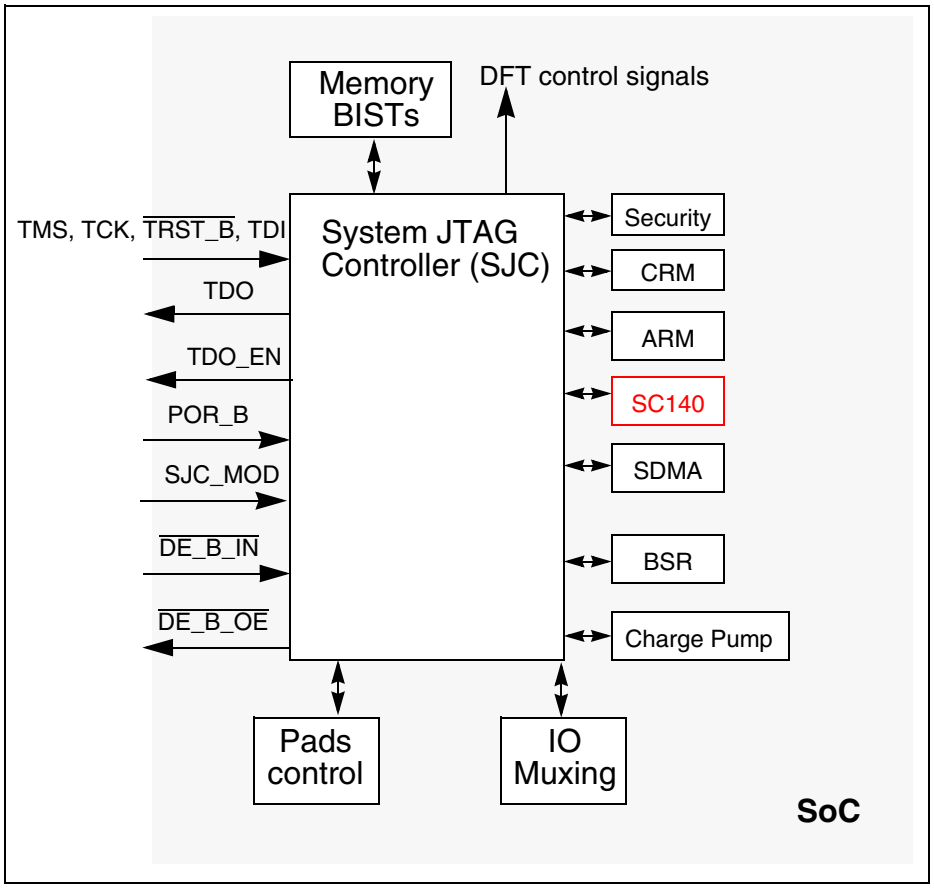


Figure 40-9. SJC Connections in an SoC

SJC external connections are described with signal properties in Table 40-5.

Table 40-5. Signal Properties

Name	Port	Function	Reset State	Pull Up
POR_B	POR_B	POR reset input. This input can arrive from either pad of IC or from IC's internal logic, e.g. Clock Controller.	0	—
TCK	TCK	Test Clock (TCK): used to synchronize the test logic and includes an internal pull-up resistor	0	—
TDI	TDI	Test Data Input (TDI): serial test instruction and data are received through the test data input (TDI) pin. TDI is sampled on the rising edge of TCK and includes an internal pull-up resistor	1	Active
TDO	TDO	Test Data Output (TDO): serial output for test instructions and data. TDO is tri-statable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK	—	—
TDO_EN	TDO_EN	Test Data Output Enable (TDO_EN). This signal enables tri-state buffer which output is connected to TDO pad and input connected to IC internal SJC generated TDO signal. Whenever TDO_EN is negated TDO will be tri-stated.	—	—
TMS	TMS	Test Mode Select (TMS): used to sequence the test controller's state machine. TMS is sampled on the rising edge of TCK and includes an internal pull-up resistor	1	Active

Table 40-5. Signal Properties (continued)

Name	Port	Function	Reset State	Pull Up
TRST_B	TRST_B	Test Reset (TRST): used to asynchronously initialize the test controller. The TRST pin has an internal pull-up resistor	1	Active
SJC_MOD	SJC_MOD	SJC mode selection. This pin is sampled at TRST reset to determine two possible modes for the TAP connection configuration.	11	Active
DE_IN_B	DE_IN_B	SoC debug request/acknowledge pins. These pins are used to propagate a debug request to the core(s) after programming of the SoC JTAG, they can also reflect debug acknowledge from the cores. See Figure 40-9 for schematic description.	1	Active
DE_B_OE	DE_B_OE		—	—

40.3.2 TAP controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in Figure 40-10. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of TCK signal. For a description of the TAP controller states, see the IEEE 1149.1 document.

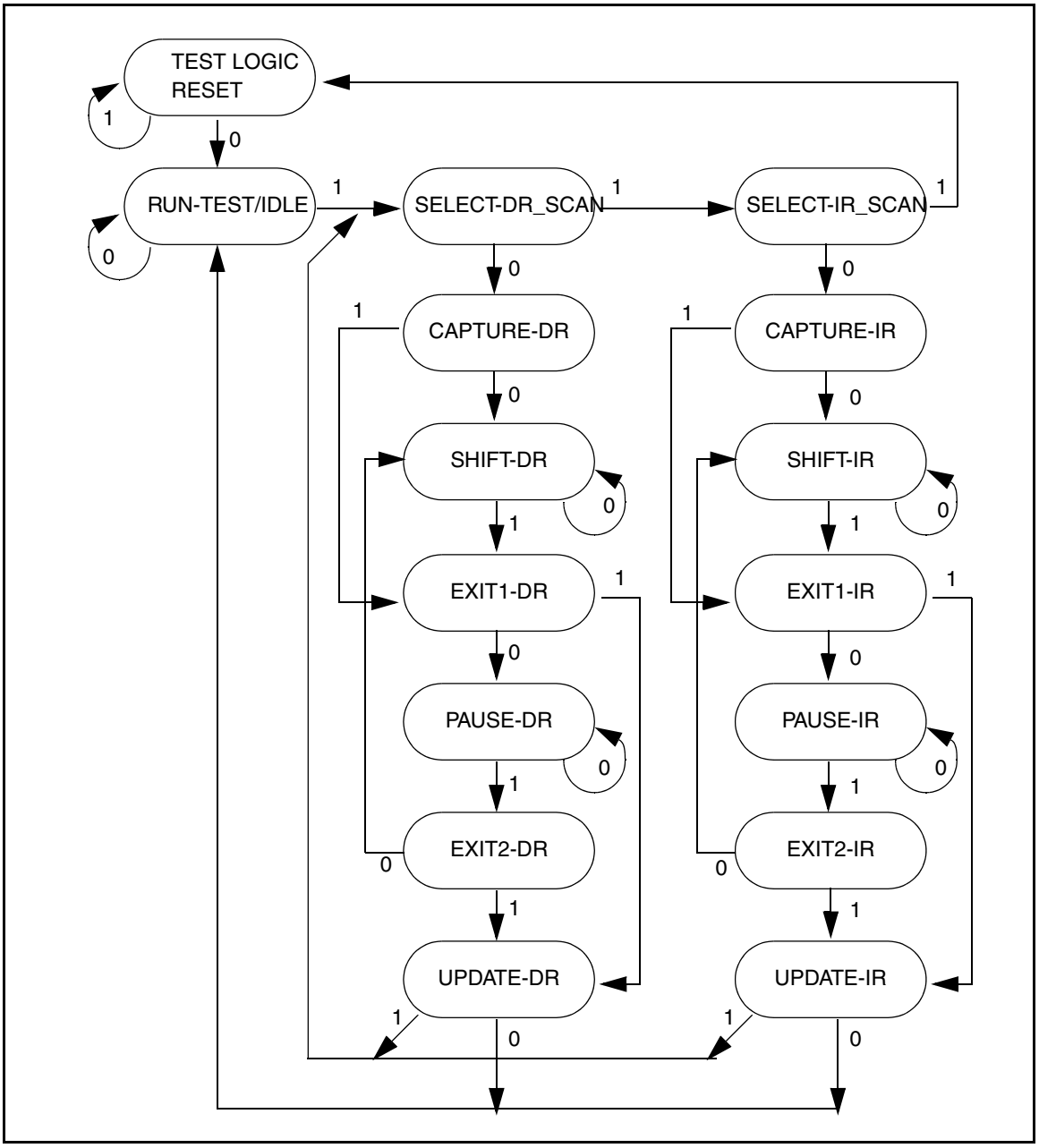


Figure 40-10. TAP Controller State Machine

The change of the JTAG state machine occurs on the rising edge of TCK. TMS and TDI changes on the falling edge of TCK. Also TDO changes on the falling edge of TCK following entry into the Shift_DR or Shift_IR states (TDO_EN is the enable of the tristate buffer driving TDO output).

The Figure 40-11 shows the timings of the SJC signals.

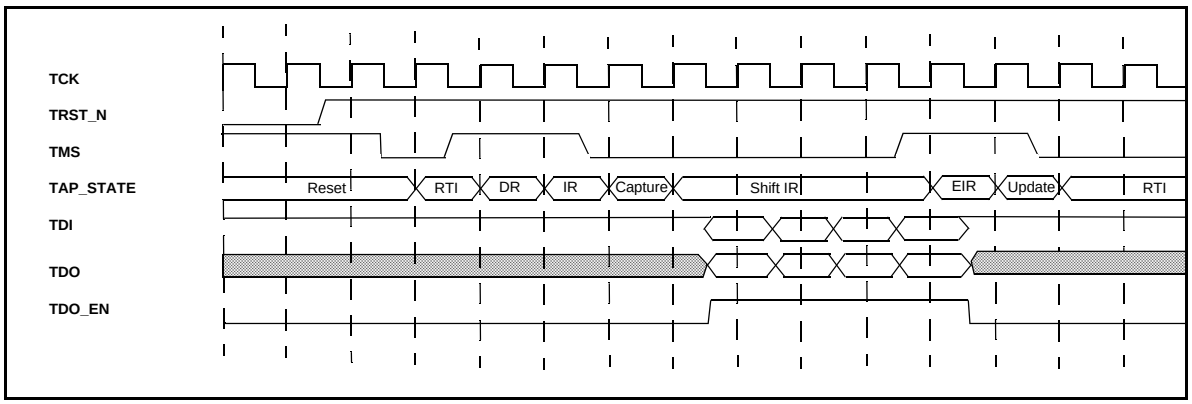


Figure 40-11. SJC Signals Timing Diagram

40.4 SoC JTAG

40.4.1 Register Summary

The $BASE value used for address calculation is necessary for automatic tests generation of the JTAG design. This variable is equal to 0 for the JTAG module. Following registers are accessed using the extradebug register, see Section 40.4.2, “Accessing Extradebug Registers,” for more information.

KEY:

Always Reads One	1

Always Reads Zero	0

Read/ Write Bit	bit

Read-Only Bit	bit

Write-Only Bit	
	bit

Write 1 to Clear	bit
	W1C

Self-Clear Bit	0
	bit

N/A	

Table 40-6. JTAG Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPUSR1 ($BASE + 0x00)	R	GPUSR1[31:16]															
	W																
	R	GPUSR1[15:10].						DPLL_stat	APLL_stat	UPLL_stat	Q_STAT[1:0]		S_STAT[2:0]			A_WFI	A_DBG
	W																
GPUSR2 ($BASE + 0x01)	R	GPUSR2[31:16]															
	W																
	R	GPUSR2[15:4]												S_STAT[3:0]			
	W																
GPUSR3 ($BASE + 0x02)	R	GPUSR3[31:16]															
	W																
	R	GPUSR3[15:0]															
	W																
GPSSR ($BASE + 0x03) secured	R	GPSSR1[31:16]															
	W																
	R	GPSSR1[15:0]															
	W																

Table 40-6. JTAG Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCR ($BASE + 0x04) secured	R	DCR[31:16]															
	W																
	R	DCR[15:7]									direct_arm_req_en	direct_sdma_req_en	direct_dsp_req_en	debug_obs	DE_to_DSP	DE_to_SDMA	DE_to_MCU
	W																
SSR ($BASE + 0x05)	R	SSR[31:16]															
	W																
	R	SSR[15:13]			RSSTAT[1:0]		SJM[1:0]		FT	BSF	RSF	EBG	EBF	SWE	SWF	KTA	KTF
	W																
GPCCR ($BASE + 0x07)	R	GPCCR[31:16]															
	W																
	R	GPCCR[15:3]													SWRES	ACLKOFF	SCLKR
	W																
GPUCR1 ($BASE + 0x09)	R	GPUCR1[31:16]															
	W																
	R	GPUCR1[15:0]															
	W																
GPUCR2 ($BASE + 0x0A)	R	GPUCR2[31:16]															
	W																
	R	GPUCR2[15:0]															
	W																
GPUCR3 ($BASE + 0x0B)	R	GPUCR3[31:16]															
	W																
	R	GPUCR3[15:0]															
	W																
GPSCR ($BASE + 0x0C) secured	R	GPSCR[31:16]															
	W																
	R	GPSCR[15:0]															
	W																

40.4.2 Accessing Extradebug Registers

Accessed through the Select-DR-Scan path, the ExtraDebug shift register consists of 38 bits (maximum) comprising a 32 bit data field (max length, see extradebug register description), a 5 bit address field and read/write bit. The write actually takes place when the JTAG TAP controller enters the Update-DR state. On a read, the data field is ignored (the user should shift only 5 times to enter Read=1 and the address), the read will take place on the next path through DR at the Capture-DR state, the data will be shifted-out during the Shift-DR state.

On the second path for a read access, simultaneous write access is not supported: command converter software will shift in 0s so the TAP will decode a write to the CSR (read-only register) which won't have any effect on the circuit.

The number of shift depends on the width of the accessed register as explained in the following diagrams.

First a write access (one path through Select-DR-Scan):

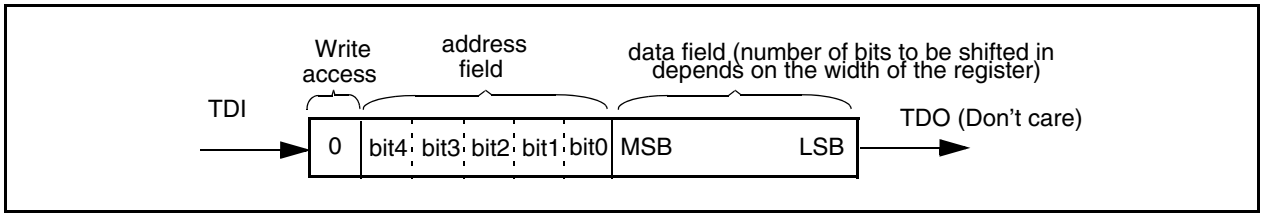


Figure 40-12. TDI/TDO on write access

Then a read access (requires two paths through JTAG DR Scan path):

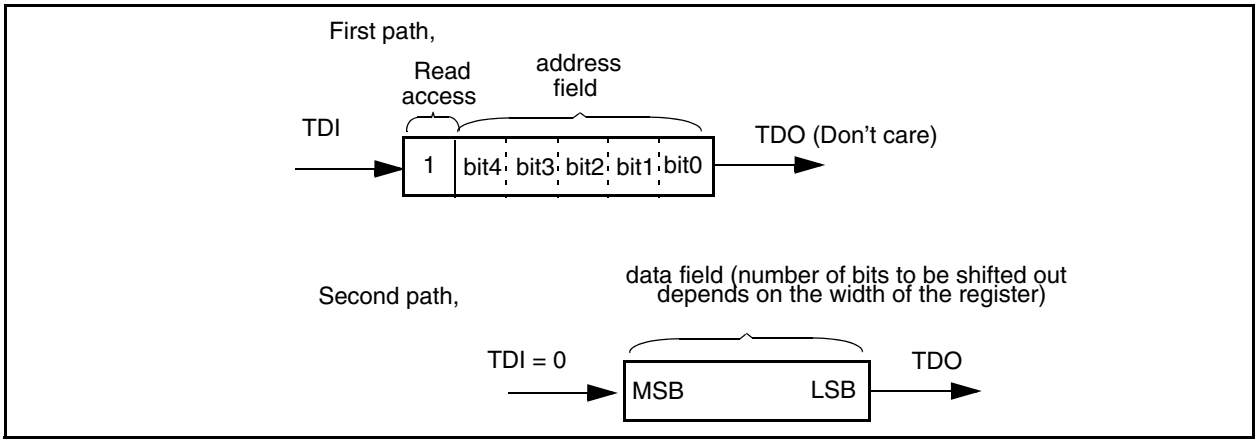


Figure 40-13. TDI/TDO on Read Access

Example: Write value 0b1010_1100 to Debug Control Register (address=0b00110).

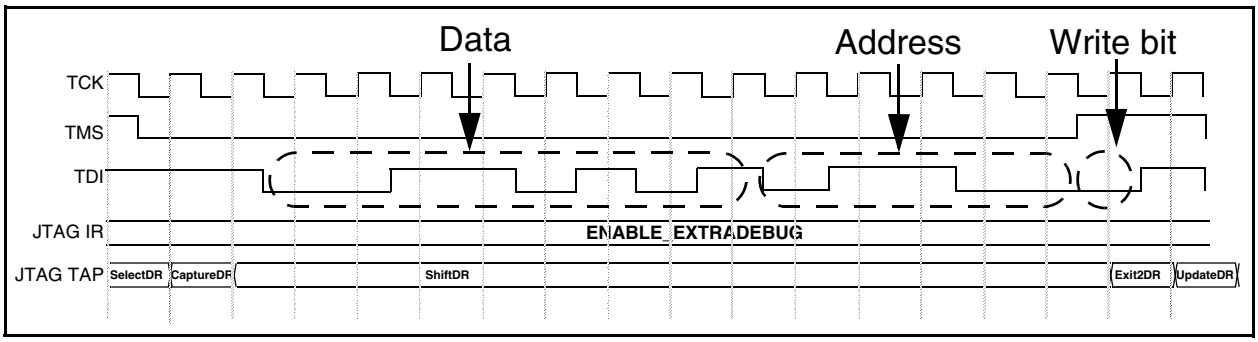


Figure 40-14. Example: write access to DCR

The SJC registers have different levels of security (see Section 40.5.3, "JTAG Security Modes"):

- **Secured**: accessible only in modes 2 (supposed correct response entered), mode 3 and mode 4.
- **Unsecured** (accessible in all modes)

This information is displayed in the detailed register description below. In the "Note" rows the letter "s" means a secured bit and "u" an unsecured bit.

40.4.2.1 General Purpose Unsecured Status Register 1

The General Purpose Unsecured Status Register 1 is a read only registers used to check the status of the different Cores and of the PLL. The rest of its bits are for general purpose use. See the SOC-specific integration guide for detailed connection information.

GPUSR — **General Purpose Unsecured Status Register 1** — **Addr $BASE + 0x00**

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPUSR1[31:16]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPUSR1[15:10]						DPLL_stat	APLL_stat	UPLL_stat	Q_STAT[1:0]		S_STAT[2:0]			A_WFI	A_DBG
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 40-7. General Purpose Unsecured Status Register 1 Description

Name	Description	Settings
GPUSR1 Bits 31-10	**General Purpose Unsecured Status Register 1** - Project specific bits. For the i.MX25 bit 15 - ECT CTI1 trigger out 4 status bits [11:10] - CRM boot_reg	bit 15: when high the CTI1 trigger out4 is asserted bits [11:10]: current boot mode 00: internal boot 01: functional test 10: external boot 11: boot strap
DPLL_stat Bits 9	DSP PLL status bit This bit indicates the status of the DSP PLL. It is reserved in the i.MX25 since there is no DSP PLL available	When HIGH, the DSP PLL is locked.
MPLL_stat Bits 8	MCU PLL status bit — This bits indicates the status of the MCU PLL.	When HIGH, the MCU PLL is locked.
UPLL_stat Bits 7	USB PLL status bit — This bits indicates the status of the USB PLL.	When HIGH, the USB PLL is locked.
Q_STAT Bits 6–5	Star*Core140v3 status bits — These bits indicates the status of the StarCore DSP. They are reserved in the i.MX25 since there is no DSP available	00 - SC140 is executing instructions. 01 - SC140 is in Wait or Stop mode. 10 - SC140 is waiting for bus. 11 - SC140 is in Debug Mode.

Table 40-7. General Purpose Unsecured Status Register 1 Description (continued)

Name	Description	Settings
S_STAT Bits 4-2	3 LSBits of SDMA core status (ipc_cstatus[2:0]). For whole bus value see GPUSR2 register. NOTE - The bits are read '0' unless visibility of SDMA Debug signals is enabled (SDMA Config Register, RTDOBS bit)	000 - PGM 001 - DATA 010 - Change Flow 011 - Set wake-up 100 - DEBUG 101 - FU Bus 110 - SLEEP 111 - Reserved
A_WFI Bit 1	ARM core wait-for interrupt bit — Bit 1 is the ARM core standbywfi (stand by wait-for interrupt).	When HIGH, ARM core is in wait for interrupt mode
A_DBG Bit 0	ARM core debug status bit — Bit 0 is the ARM core DBGACK (debug acknowledge) DBGACK can be overwritten in the ARM core DCR to force a particular DBGACK value. Consequently interpretation of the DBGACK value is highly dependent on the debug sequence	When HIGH, ARM core is in debug

40.4.2.2 General Purpose Unsecured Status Register 2

The General Purpose Unsecured Status Register 2 is read only general purpose registers.See the SOC-specific integration guide for detailed connection information.

GPUSR2	General Purpose Unsecured Status Registers 2																Addr $BASE + 0x01
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16	
	GPUSR2 [31:16]																
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0	
	GPUSR2 [15:0]																
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	

Table 40-8. General Purpose Unsecured Status Register 2 Description

Name	Description	Settings
GPUSR2 Bits 31-0	**General Purpose Unsecured Status Register** - Project specific bits. For i.MX25 bits [3:0]—SDMA core status	0000: PGM 0001: DATA 0010: Change Flow 0011: Set wake-up 0100: DEBUG 0101: FU Bus 0110: SLEEP ... See SDMA spec for more details.

40.4.2.3 General Purpose Unsecured Status Registers 3

The General Purpose Unsecured Status Register 3 is a read only general purpose registers. See the SOC-specific integration guide for detailed connection information.

GPUSR3	General Purpose Unsecured Status Registers 3															Addr $BASE + 0x02
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPUSR3 [31:16]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPUSR3 [15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 40-9. General Purpose Unsecured Status Register 3 Description

Name	Description	Settings
GPUSR3 Bits 31-0	**General Purpose Unsecured Status Register** - See Project specific integration guide and Test Guide for more information.	—

40.4.2.4 General Purpose Secured Status Register (Secured)

The General Purpose Secured Status Register is a read only register used to check the status of the different critical information in the SoC. This register cannot be accessed in secure modes. See the SOC-specific integration guide for detailed connection information.

GPSSR	General Purpose Secured Status Register															Addr $BASE + 0x03
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPSSR[31:16]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPSSR[15:0]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

Table 40-10. General Purpose Secured Status Register Description

Name	Description	Settings
GPSSR Bits 31-0	**General Purpose Secured Status Register** - See Project specific integration guide and Test Guide for more information.	—

40.4.2.5 Debug Control Register (Secured)

This register is used to control propagation of debug request from DE_B pad to the cores and debug signals from embedded cross-trigger IP to the DE_B pad.

DCR	Debug Control Register															Addr $BASE + 0x04
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	DCR[31:16]															
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	DCR[15:7]									direct_arm_req_en	direct_sdma_req_en	direct_dsp_req_en	debug_obs	DE_to_DSP	DE_to_SDMA	DE_to_ARM
TYPE:	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

Table 40-11. Debug Control Register Description

Name	Description	Settings
DCR Bits 31-7	**General Purpose Unsecured Status Register** - Project specific bits. For i.MX25: bit 8 - bypass bit for ARM 9 and DAP	When high, ARM 9 platform and DAP are bypassed, use shadow logic of ARM and DAP to substitute TAP of ARM 9 and ETB
direct_arm_req_en Bits 31-7	**Debug enable of the arm debug request** — This bit controls the propagation of debug request DE_B to the arm.	0 - Disable propagation of system debug to (DE pin) to arm. 1 - Enable propagation of system debug to (DE pin) to arm.
direct_sdma_req_en Bits 31-7	**Debug enable of the sdmadebug request** — This bit controls the propagation of debug request DE_B to the sdma.	0 - Disable propagation of system debug to (DE pin) to sdma. 1 - Enable propagation of system debug to (DE pin) to sdma.
direct_dsp_req_en Bits 31-7	**Debug enable of the dspdebug request** — This bit controls the propagation of debug request DE_B to the dsp.	0 - Disable propagation of system debug to (DE pin) to dsp. 1 - Enable propagation of system debug to (DE pin) to dsp.
debug_obs Bits 3	**Debug observability —** This bit controls propagation of the system debug signal from Embedded Cross Trigger Block (ECT) to DE_B pad (can propagate debug acknowledge for example).	0 - Disable propagation of system debug to DE pin 1- Enable propagation of system debug to $\overline{DE}$ pin
DE_to_DSP Bits 2	**DSP debug request input propagation —** This bit controls propagation of the debug request to the DSP. This bit is reserved since there is no DSP in the i.MX25	0 - Disable propagation of debug request to DSP 1- Enable propagation of debug request to DSP
DE_to_SDMA Bits 1	**SDMA debug request input propagation —** This bit controls propagation of the debug request to the SDMA.	0 - Disable propagation of debug request to SDMA 1- Enable propagation of debug request to SDMA
DE_to_MCU Bits 0	**ARM debug request input propagation —** This bit controls propagation of the debug request to the ARM.	0 - Disable propagation of debug request to ARM 1- Enable propagation of debug request to ARM

For pin count restriction requirements, and knowing that the JTAG/OnCE/ICE architecture is pretty flexible to send debug request and check core status through SJC interface, only one external pin is dedicated for debug request input / system debug output, $\overline{DE}$, bidirectional open drain (including an internal pull-up device).

The main function of the Embedded Cross Trigger is to pass debug events from one core to another. For example this can be programmed to pass debug request from one IPas debug request to ARM so that ARM enters debug mode when the debug request is generated. ECT system_debug is connected to SJC and can be programmed for observability on $\overline{DE}$ pad.

The bits 6:4 define the propagation of external debug request to each Core, and bit 3 controls propagation of system debug signal from Embedded Cross Trigger Block (ECT) on $\overline{DE}$. It must be noted that I/O circuitry depicted in the figure is the recommended one and may vary from IC to IC.

The bits 2:0 define the propagation of IR debug request to each Core.

Note in below Figure 40-15, ignore all DSP related functionality regarding the i.MX25 usage only.

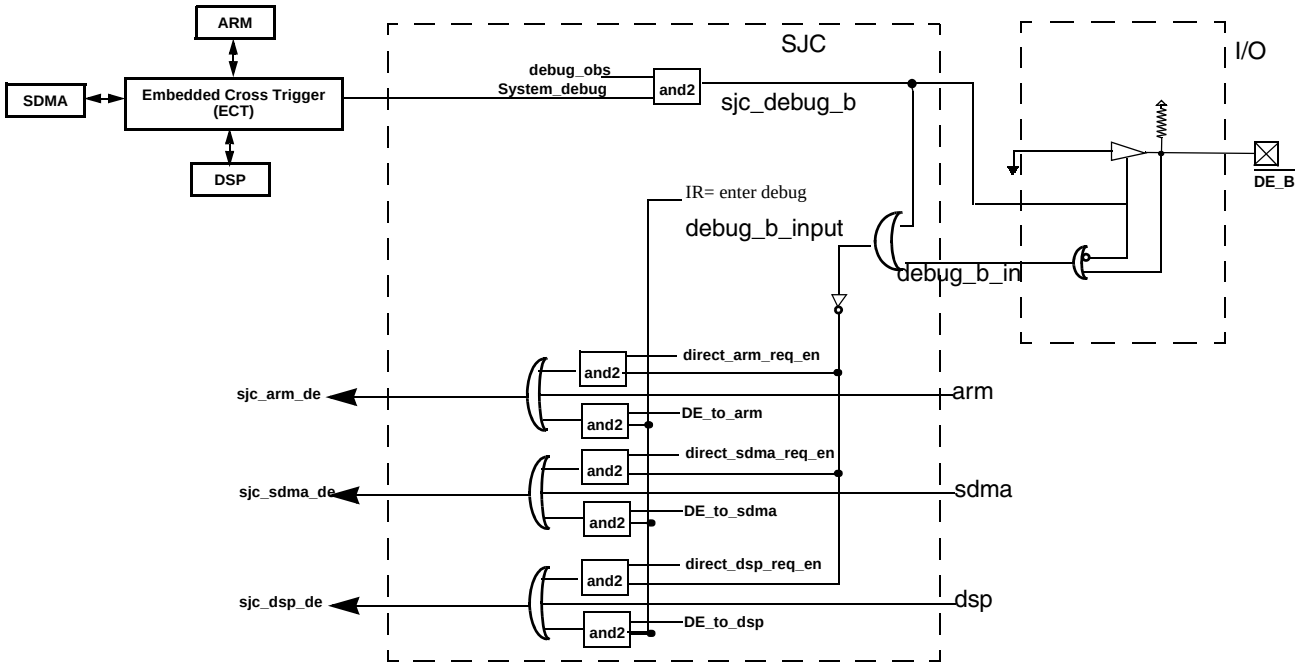


Figure 40-15. $\overline{DE}$ Pin Select Logic

NOTE

For security reasons bits for output and input propagation control will be at their negated values after reset: a user will not be able to put the cores in debug mode through $\overline{DE}$ without any JTAG access.

According to the values of those bits, the user can do the following:

1. Get the two cores to enter Debug Mode from an external command controller.
2. Acknowledge entrance in Debug mode from one Core or two Cores or all of them. If two debug acknowledges arrive at the same time from two different cores, the user might only see one low pulse on $\overline{DE}$. This has to be defined in the Embedded Cross Trigger integration and programming.

Note that if one Core enters Debug, it can force other ones to enter Debug too if I/O connection scheme supports that, like one depicted in Figure 40-15. A debug acknowledge will propagate to the other Cores debug request lines. That should normally be handled through Embedded Cross Trigger Block (ECT) though.

The configuration after reset prevents propagation of debug requests / acknowledges to or from the cores.

40.4.2.6 Security Status Register

This register is used to reflect IC security status and is accessible in all the security modes. The assumption is that the information contained in this register does not pose any security breach for the system. See the SOC-specific integration guide for detailed connection information. See your Freescale representative for details on the Security Status Register.

40.4.2.7 General Purpose Clocks Control Register

This register is used to configure clock related modes in SOC. See the SOC-specific integration guide for detailed connection information. Those bits will be directly connected to JTAG outputs.

GPCCR	General Purpose Clocks Control Register														Addr $BASE + 0x07	
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPCCR[31:16]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPCCR[15:1]															SCLKR
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

Table 40-12. General Purpose Clocks Control Register Description

Name	Description	Settings
GPCCR Bits 31-1	**General Purpose Clocks Control Register** - See Project specific integration guide and Test Guide for more information.	—
SCLKR Bit 0	**SDMA Clock ON Register** - This bit will force the clock on of the SDMA	0 - SDMA clock is not forced ON. 1 - Force SDMA clock ON

Bit 0 of GPCCR controls SDMA clocks invocation. When out of reset, the SDMA is in sleep mode with no SDMA clock running. Unlike events, debug requests does not wake SDMA if it is in sleep mode. The debug request will only be recognized by the SDMA when it exits sleep mode upon reception of an event. To be able to enter debug mode even if no event is triggered, the SDMA clock on bit needs to be set prior to sending the debug request (clear at reset).

40.4.2.8 General Purpose Unsecured Control Registers 1,2,3 (Three Registers)

This register is used to configure SJC for special Test or debug modes. See the SOC-specific integration guide for detailed connection information. These registers are not secured (accessible in all jtag security modes). The bits of these registers are directly connected to SJC outputs.

GPUCR(1,2,3) General Purpose Unsecured Control Register

Addr
$BASE + 0x09
$BASE + 0x0A
$BASE + 0x0B

	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPUCR[31:16]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPUCR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 40-13. General Purpose Unsecured Control Register Description

Name	Description	Settings
GPUCR Bits 31-0	**General Purpose Unsecured Control Register** - See Project specific integration guide and Test Guide for more information.	—

40.4.2.9 General Purpose Secured Control register (Secured)

This register is used to configure SJC for special Test or debug modes. See the SOC-specific integration guide for detailed connection information. This register is secured (accessible in secure jtag mode #3, #4 and #2 with response entered). Those bits will be directly connected to SJC outputs.

GPSCR	General Purpose Secured Control Register															Addr $BASE + 0x0C
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	GPSCR[31:16]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	GPSCR[15:0]															
TYPE:	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Note:	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

Table 40-14. General Purpose Secured Control Register Description

Name	Description	Settings
GPSCR Bits 31-0	**General Purpose Control Register** - See Project specific integration guide and Test Guide for more information.	—

40.4.3 SoC JTAG Instruction Register

The SoC JTAG Instruction register is 5 bits wide.

Table 40-15. JTAG Instruction Register

Code					SJC IR
B4	B3	B2	B1	B0	
0	0	0	0	0	IDCODE
0	0	0	0	1	SAMPLE/PRELOAD
0	0	0	1	0	EXTEST
0	0	0	1	1	HI-Z
0	0	1	0	0	ENABLE_ExtraDebug
0	0	1	0	1	ENTER_DEBUG (secured)
0	0	1	1	0	Serial Access
0	0	1	1	1	TAP select
0	1	0	0	0	Rsvd.
0	1	0	0	1	Rsvd.
0	1	0	1	0	Rsvd.
0	1	0	1	1	Rsvd.
0	1	1	0	0	Security Output challenge
0	1	1	0	1	Security Enter response
.....					Rsvd.
1	1	1	1	1	BYPASS

The instruction register is reset to 0b00000 in the test-logic-reset controller state which is equivalent to the IDCODE instruction.

During the capture-IR controller state, the parallel inputs to the instruction register are loaded with the code 01 in the least significant bits as required by the standard, the most significant bits are loaded with the values 00, leading to a capture value of 0b00001.

40.4.3.1 ID_CODE Instruction

Selects the ID register, and the system logic controls the I/O pins. This instruction is provided as a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP. The following table shows the ID register configuration.

IDCODE				ID Configuration Register												
	BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
	Version Information[3:0]				Design Center Number						Core number					
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	**0**	**0**	**0**	**0**	**z**	**z**	**z**	**z**	**z**	**z**	**y**	**y**	**y**	**y**	**y**	**x**
Note:																
	BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
	Chip Derivative Number				Manufacturer Identity											1
TYPE	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
RESET	**x**	**x**	**x**	**x**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
Note:																

Table 40-16. ID Configuration Register Description

Name	Description	Settings
Version Information Bits 31-28	Version information number - This number is incremented for each metal fix of the IC.	Bits [31:28] - 0000 (for initial SoC, 0001 for first metal fix, etc. See i.MX25 Reference Manual Book I for details on specific SOC.
Customer Part Number Bits 27-12	**Customer Part Number —** The Customer Part Number consists of two parts: Bits [27-22] - Freescale Design Center Number Bits [21:12] - A sequence number divided into two parts: Bits [21-17] - Core Number (SoC project) Bits [16-12] - Chip Derivative Number	See i.MX25 Reference Manual Book I for details on specific SOC. (for zzzzzz, yyyyy,x defaults).
Manufacturer Identity Bits 11-1	**Manufacturer Identity —** Freescale's Manufacturer Identity code.	Bits [11:1] - 00000001110
Bit 0	Tied to logic 1.	1

NOTE

The IDCODE value is programmed by SOC integration of the SJC module. See Chapter 7, "System Boot," for exact register value for a specific SOC.

One application of the ID register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge which conform to IEEE 1149.1, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design in order to determine the

type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

Once the IDCODE instruction is decoded, it will select the ID register which is a 32 Bit data register. Since the bypass register loads a logic 0 at the start of a scan cycle, whereas the ID register loads a logic 1 into its least significant bit, examination of the first bit of data shifted out of a component during a test data scan sequence immediate following exit from Test-Logic-Reset controller state will show whether such a register is included in the design. When the IDCODE instruction is selected, the operation of the test logic has no effect on the operation of the on-chip system logic as required by IEEE 1149.1.

40.4.3.2 SAMPLE/PRELOAD Instruction

Selects the boundary scan register and the system logic controls the I/O pins. The SAMPLE/PRELOAD instruction provides two separate functions:

1. First, it provides a means to obtain a snapshot of system data and control signals. The snapshot occurs on the rising edge of TCK in the capture-DR controller state. The data can be observed by shifting it transparently through the boundary scan register.
2. The second function of SAMPLE/PRELOAD is to initialize the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction.

NOTE

Since there is no internal synchronization between the JTAG clock (TCK) and the system clock (CLK), the user must provide some form of external synchronization to achieve meaningful results.

For more details on the function and use of SAMPLE/PRELOAD, see the IEEE 1149.1 document.

40.4.3.3 EXTEST Instruction

Selects the boundary scan register, and the 1149.1 test logic has control of the I/O pins.

By using the TAP controller, the register is capable of:

1. Scanning user-defined values into the output buffers,
2. Capturing values presented to input pins
3. Controlling the direction of bidirectional pins,
4. Controlling the output drive of tri-statable output pins.

For more details on the function and use of EXTEST, see the IEEE 1149.1 document.

The EXTEST instruction also asserts internal reset for the cores (through CRM, see Figure 40-17) to force a predictable internal state while performing external boundary scan operations.

40.4.3.4 HIGHZ Instruction

All output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register. In this mode, all internal pullup resistors on all the pins (except for the TMS

TDI TCK $\overline{\text{TRST}}$ pins) will be disabled. This “disabling” functionality is not built into SJC, but should be implemented by some logic in the SOC/IO Pads.

For more details on the function and use of HIGHZ, see the IEEE 1149.1 document.

The HIGHZ instruction also asserts internal reset for the cores (through CRM, see Figure 40-17) to force a predictable internal state while performing external boundary scan operations.

40.4.3.5 BYPASS Instruction

Selects the single Bit bypass register and the system logic controls the I/O pins. This creates a shift-register path from TDI to the bypass register and, finally, to TDO, circumventing the boundary scan register. This instruction is used to enhance test efficiency when a component other than the SoC Core based device becomes the device under test.

When the bypass register is selected by the current instruction, the shift-register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

For more details on the function and use of BYPASS, see the IEEE 1149.1 document.

40.4.3.6 ENABLE_ExtraDebug Instruction

The TDI and TDO pins are connected directly to the ExtraDebug registers, the SJC TAP controller remaining connected to TDI and TMS. The ExtraDebug shift register consists of 38 bits (maximum) comprising a 32 bits data field (maximum length, see Section 40.4.2, “Accessing Extradebug Registers”), a 5 bits address field and read/$\overline{\text{write}}$ bit. On a register read, the data field doesn’t need to be filled in. The particular ExtraDebug register connected between TDI and TDO at a given time is selected by the ExtraDebug controller depending on the ExtraDebug Address being currently decoded. All communication with the ExtraDebug controller is done through the Select-DR-Scan path of the JTAG TAP Controller.

40.4.3.7 ENTER_DEBUG Instruction

Generates a debug request to the three cores “simultaneously” (or “Pseudo simultaneously” as the Cores run at different frequencies), the TDI and TDO are connected to the Instruction Register. After the acknowledgment of the Debug Mode is received (can be checked by reading the Core Status Register part of the ExtraDebug logic), the user can perform system debug functions on the three Cores.

NOTE

In case the user needs only one of the cores in Debug Mode, he should not issue the ENTER_DEBUG instruction but rather access the preferred core and issue a debug request only to this core.

The user need to be careful to shift another IR value (like IDCODE) before trying to get the Cores out of debug as the debug request signals to the Cores stay asserted until IR is no longer equals to ENTER_DEBUG.

The user need to check that cores are in debug mode (watching debug acknowledge signal) before leaving ENTER_DEBUG instruction, otherwise debug request might not have been seen by the core.

40.4.3.8 TAP Select Instruction

By means of “TAP select” instruction a user can access TAP select register and by controlling the bits for SDMA Bypass.

Note ARM MCU/ETB bypass, and ARM DAP bypass are performed by writing to the DCR register bit.

TAP Select Register

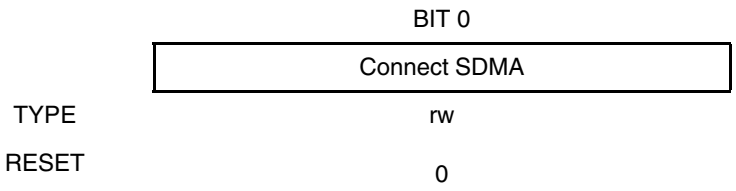


Table 40-17. TAP select Register Description

Name	Description	Settings
SDMA Bypass Bits 0	Connect SDMA — control whether SDMA TAP will be bypassed or not.	• When negated (should be the default state), SDMA TAP is bypassed with a single D-FF. • When asserted SDMA TAP is connected inside the chain.

40.5 Security

40.5.1 Introduction

JTAG manipulation is one of the known hackers’ ways of executing unauthorized program code, getting control over the OS and run code in privileged modes.

The system JTAG controller provides a debug access to several H/W blocks including the ARM processor and the system bus. This allows for program control and manipulation as well as visibility into system peripherals and memory. The ETM and NEXUS interfaces allow bus transactions to be traced. Together these tools provide the hacker all the access needed to completely comprise the system. Means must be provided to block any malicious JTAG access.

The SJC provides a way of regulating the JTAG access.

There are four different JTAG security modes:

- Mode #1: No Debug - Maximum Security. All security sensitive JTAG features are permanently blocked.
- Mode #2: Secure JTAG - High security. JTAG use is regulated by secret key based authentication mechanism.
- Mode #3: JTAG Enabled - Low security. JTAG always enabled.

- Mode #4: SCC JTAG - No Security. The SCC forced to remain in its secure state during JTAG operation.Parts in this mode are to be used only under tight control, for secure operation debug.

The JTAG security modes are configured using eFUSEs. eFUSEs can be burned after packaging, just by applying electrical signals. Fuse burning is an irreversible process; after an eFUSE is burned, it is impossible to change it back to the unburned state.

The eFUSE can be burned at any time and can provide flexibility to the OEM vendor regarding the security level of its products.

40.5.2 Fuses Programming

For electric fuses (eFUSE) technology and programming description, see the IC Identification Module (IIM) block guide.

40.5.3 JTAG Security Modes

JTAG can be in one of four JTAG security modes. The specific security mode is determined by the SJC fuses configuration[1].

40.5.3.1 Mode #1: No Debug—Maximum Security

“No Debug” JTAG security mode provides the highest security level. In this mode, all JTAG features are disabled except for:

- Scan.
- Boundary Scan.
- MBIST, all modes except for debug modes which enable controlled memory contents output.
- PLL BIST.
- BIST monitor mode[2].
- PLL bypass[3]
- Visibility of the following status bits: power mode - normal, standby, stop, shutdown, etc.

These features do not reduce the security level of the product, and they allows to perform important tests and board connectivity checks.

NOTE

If external boot is done (enabled by a corresponding fuse and requested) then it is possible to use all JTAG features even if “No Debug” JTAG security mode is chosen, as described in Chapter 40.5.6, “External Boot Fuse”

1. Physical location of the fuses is not in the SJCv1.1.
2. Allowing routing to external pins BIST pass/fail/invoke information
3. Bypass ARM or/and USB PLL.

NOTE

See your Freescale representative for details about the operation of the security features of this modules.

40.5.4 Software Enabled JTAG

To increase the flexibility of the SJC, an option to enable the JTAG using software is added and is available only in Secure JTAG mode. This feature can be permanently blocked by burning the dedicated eFUSE. By writing to SW_*JTAG_EN* bit in the system control module (IIM), the JTAG will be opened, regardless of its security mode. It is the responsibility of the HAB (High Assurance Boot) software to assert or negate this bit. The *JTAG_SW_LOCK* bit of the system control (IIM) makes sure that only the HAB will be able to set the SW_*JTAG_EN* bit. When *JTAG_SW_LOCK* sets, no future change of SW_*JTAG_EN* is possible, unless after asserting POR (power-on-reset). The HAB sets the *JTAG_SW_LOCK* bit before transferring control to the application code. It is also automatically set whenever the device transitions into a non-secure state. Through this means, the HAB ensures that only it may modify SW_*JTAG_EN*. The *JTAG_SW_LOCK* is write only, and sticky bit. It can only be cleared through POR.

The S/W JTAG enable allows JTAG enabling without activating the challenge-Response mechanism (which requires JTAG access tool enhancement or special H/W). The JTAG S/W enable doesn’t allow debug in case of boot or memory fault as it requires reset before entering debug.

NOTE

The S/W enabled JTAG feature reduces the overall security level of the system as it relies on S/W protections. If this feature is not required, it is strongly recommended to burn a dedicated eFUSE which disables this feature.

40.5.5 ETM Kill Trace

The kill trace signal disables any output from the Nexus and ETM modules. The ETM can be accessed using JTAG port and by direct software address. When JTAG port access is blocked the “kill trace” signal will be active, i.e. trace module’s output is blocked. The intention of this action is to block any attempt to break into the system using ETM software configuration. The “kill trace”, when active, prevents trace output even in case where it can be activated using chip pin.

The “kill trace” feature needs to be activated by burning a dedicated eFUSE. If the fuse is left intact, “kill trace” will never be activated as seen in Figure 40-16.

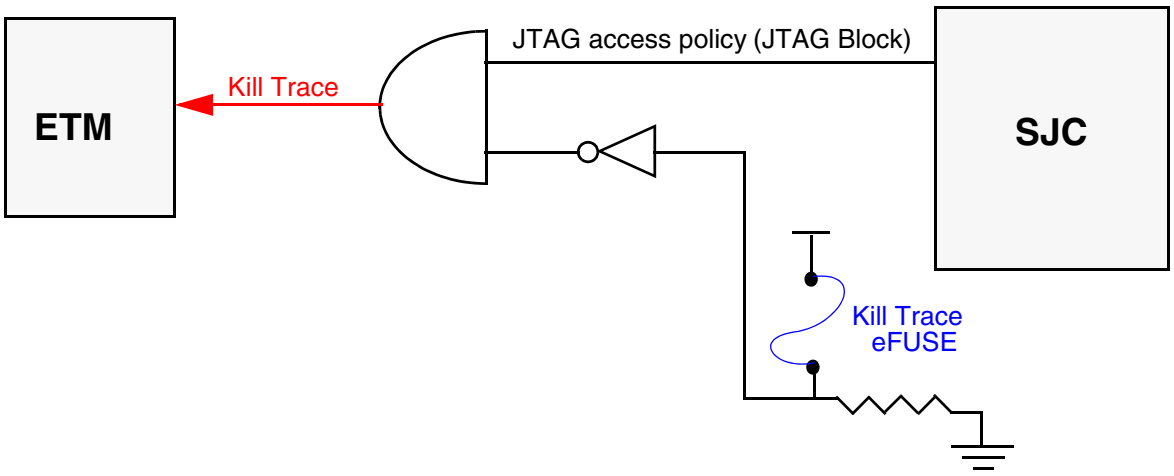


Figure 40-16. Kill Trace eFUSE

The “kill trace” is asserted when “ipt_enable_kill_trace_fuse” fuse is burned and “ipt_secur_block” signal in SJC is asserted which happens when at least one of the following is true:

- “iim_fuses_latched” signal come from IIM to Secure JTAG is not asserted which tells that Secure JTAG can’t consider fuses values to be valid
- Mode #2 (Secure JTAG) and no code has been entered
- Mode #2 (Secure JTAG) with burned “Bypass” and “Re-enable” fuses
- Mode #2 (Secure JTAG) with incorrect response entered
- Mode #1 (No debug)
- “trst_b” signal is active
- POR has not ever been asserted

40.5.6 External Boot Fuse

External boot can be a security hazard. Using external boot, it is possible to run any code without having it passes the authentication checks. However, sometimes, external boot is required in debug processes. The purpose of the External Boot eFUSE is to disable the external boot feature. When burned, no external boot is possible. If external boot is enabled (by the fuse, and configured using boot mode pins, all JTAG features are enabled for all JTAG security modes.

40.6 Functional Description

40.6.1 Cores Static Debug

The SJC can interface with 3 Cores on the same die: an ARM core with ETM real-time trace interface, a Star*Core140v3 DSP with IEEE-ISTO 5001 Nexus interface and a SDMA RISC machine as described in Figure 40-1. i.MX25 has employed 2 of them, ARM platform and SDMA interfaces.

A SoC JTAG TAP controller fully compatible with the *IEEE1149.1a-2001 IEEE Standard Test Access Port and Boundary Scan Architecture specification* is implemented at the top level.

The ARM core has an integrated JTAG interface and a TAP controller to manage its own ICE. Also it can access an embedded trace ETM interface, see ARM core and ETM Technical reference guide for more information.

The SDMA has a TAP controller to manage its own OnCE but is a JTAG free Core, see SDMA OnCE spec for more information.

While the JTAG port provides board test capability, the OnCE and ICE ports provide emulation and debug capability to the user. The JTAG port conforms to the IEEE1149.1a-2001 IEEE Standard Test Access Port and Boundary Scan Architecture specification defined by the Join Test Action Group (JTAG). The TAP (Test Access Port) uses a boundary scan technique to test the interconnections between integrated circuits after they are assembled onto a printed circuit board. Boundary scan allows a tester to observe and control signal levels at each component pin through a shift register placed next to each pin. This is important for testing continuity and determining if pins are stuck at a one or zero level.

The OnCE and ICE provide a mean of interacting with the Cores and their peripherals non-intrusively so that a user may examine registers, memories to facilitate hardware and software development.

40.6.2 Reset Mechanism

In the actual application, all logic is initialized by a POR (Power-On Reset). In a tester or debug setting, the TRST and RESET_IN pins can be used to re-establish the initialized state as desired.

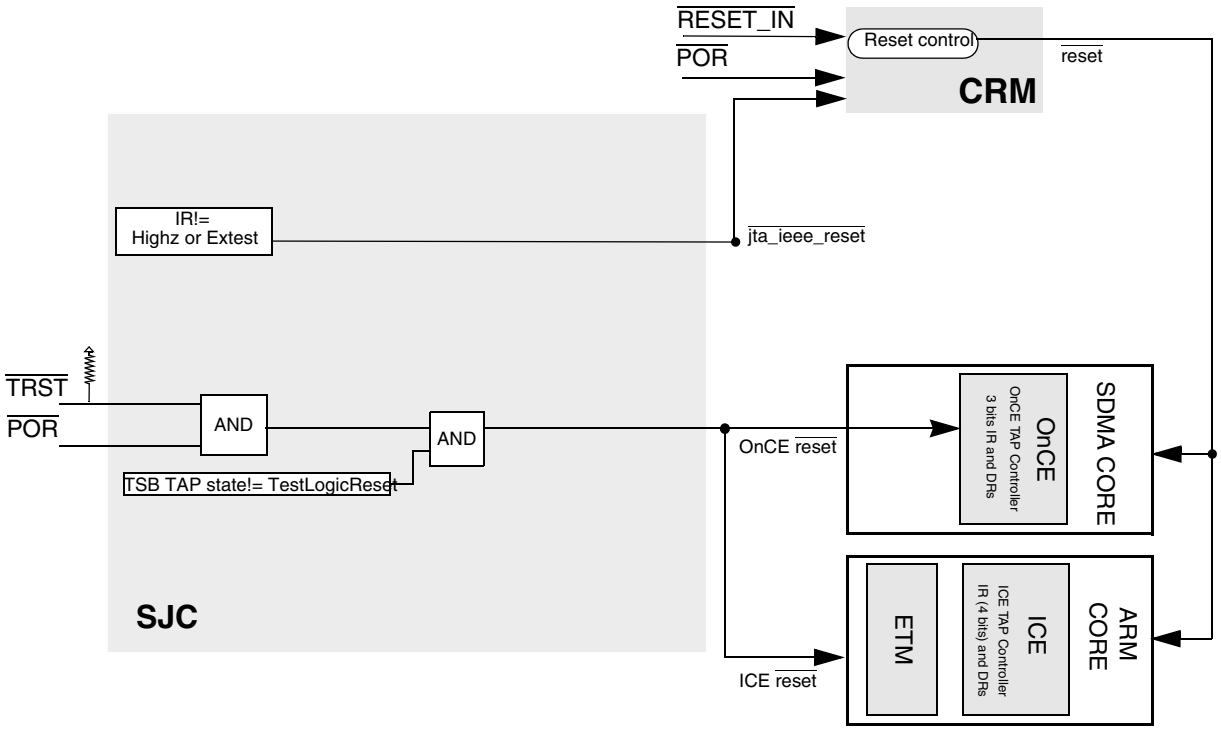


Figure 40-17. SJC Reset Logic

NOTE

- Asserting TRST in any scan mode will reset the TCR thus will loose the testmode configuration and will select default TAP.
- SJC generates an IEEE reset signal to the CRM when in one of the IEEE modes HIGHZ or EXTEST. This signal will generate a system reset to the cores until exit from one of these modes.
- The TAP Selection Block will generate Once/ICE reset (either TRST if implemented or other) when its TAP state will reach Test-Logic-Reset (meaning that TAP accessed is also reaching Test-Logic-Reset).
- When the SJC is in scan mode reset lines will be controlled from RESET_IN for manufacturing test purpose.

40.7 Initialization/Application Information

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the SJC output drivers are enabled into actively driven networks.

There are two constraints related to the JTAG interface:

- Ensure that the JTAG test logic is kept transparent to the system logic by forcing TAP into the Test-Logic-Reset controller state. During power-up, SJC's internal $\overline{\text{TRST}}$ is asserted as IC's $\overline{\text{POR}}$ is asserted which forces the TAP controller into this state. After that, if TMS either remains unconnected or is connected to VCC, then the TAP controller cannot leave the Test-Logic-Reset state, regardless of the state of TCK.
- $\overline{\text{DE\_B}}$ is an IO pin with pullup and care must be taken of the direction when driving this signal.

Chapter 41
Smart Liquid Crystal Display Controller (SLCDC)

The SLCDC module transfers data from display memory buffer to external display device. DMA transfers data transparently with minimal software intervention. DMA Bus utilization is controllable and deterministic.

As cellular phone displays become larger and more colorful, demands on the processor increase. More CPU power is needed to render and manage the image. The role of the display controller is to reduce the CPU involvement in the data transfer from memory to the display device so that CPU can concentrate on image rendering. DMA is used to optimize the transfer. Embedded control information needed by the display device is automatically read from a second buffer in system memory and inserted into the data stream at the proper time to completely eliminate the CPU role in the transfer.

A typical scenario for a cellular phone display is to have display image rendered in main system memory. After the image is complete, CPU triggers the SLCDC module to transfer the image to the display device. Image transfer is accomplished by burst DMA which steals bus cycles from CPU. Cycle stealing behavior is programmable and bus use can be kept within predefined bounds. After transfer is complete, a maskable interrupt is generated indicating the status. For animated displays, it is suggested to implement a 2-buffer ping-pong scheme so that when DMA is fetching data from 1 buffer, next image is rendered into the other.

Several display sizes and types are used in various products that uses SLCDC. SLCDC module has the capability of directly interfacing to the selected display devices. Both serial and parallel interfaces are supported. SLCDC module only supports write to the display controller. SLCDC read operations from the display controller are not supported.

41.1 SLCDC Module Pin List

Table 41-1 is a list of the SLCDC module pins.

Table 41-1. SLCDC Module Pin List

Pin Name	Direction	Description
LCD Interfaces		
SLCDC_LCD_DATA[15:0]	Output	Data bus used to write information to external LCD controller.
SLCDC_LCD_CS	Output	Used as a chip select for external display controller in serial mode. In parallel mode, used as write strobe for the external display controller. This signal polarity is programmable.
SLCDC_LCD_RS	Output	LCD Register Select signal that indicates to the LCD device whether data being written is display data or control data. This signal polarity is programmable.

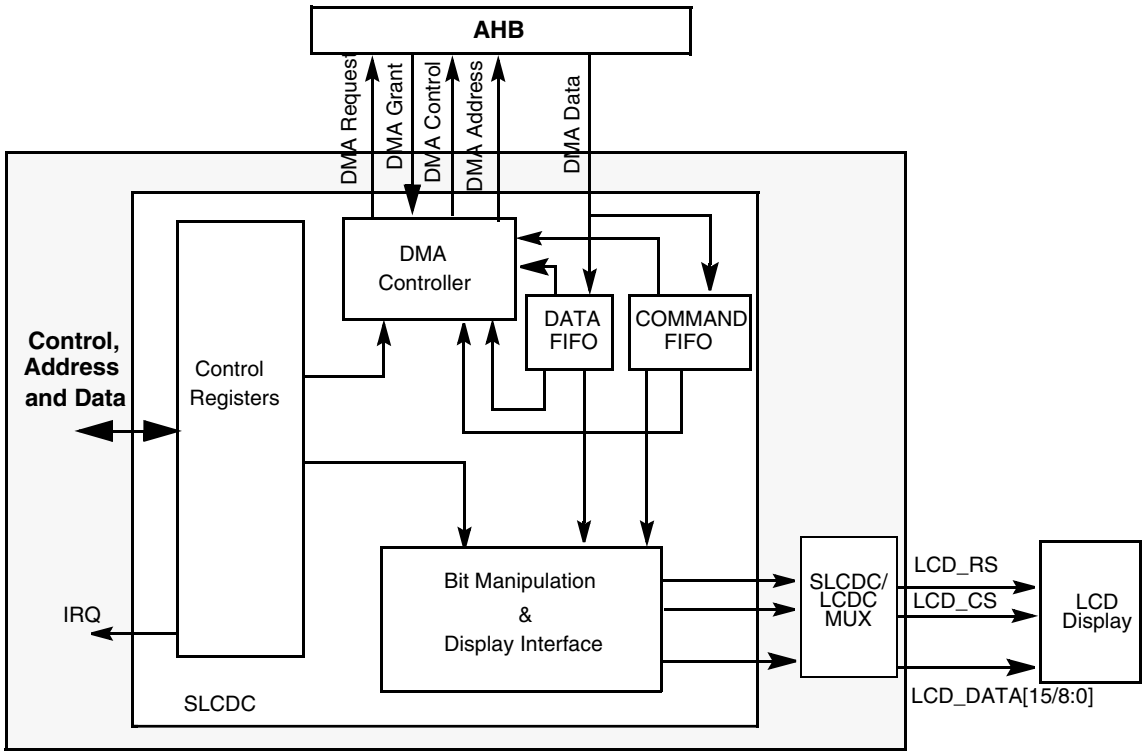


Figure 41-1. SLCDC System Diagram

41.2 Functional Description

The purpose of SLCDC is to transparently and efficiently transfer image data from system memory to an external LCD controller. System memory can be either internal or external memory. Figure 41-1 shows the block diagram of SLCDC module within the system. SLCDC module contains a DMA controller that is designed to operate as an alternate master. SLCDC DMA interface requests control of AHB to do 32-bit reads from system memory.

The purpose of DMA within SLCDC is to transfer image and control data from system memory to SLCDC FIFO where it is formatted and sent out to the external LCD controller. DMA only performs read accesses from system memory. The data is collected in 32-bit words and stored in two internal FIFOs. Data is pulled from the FIFOs, as needed, and put in correct format for the external LCD controller.

SLCDC has both control and status registers which are accessible using IP bus. These registers are used to store information about the type of LCD controller attached to the system. SLCDC can be configured to write image data to an external LCD controller using a 4-line serial, 3-line serial, an 8-bit or 16-bit parallel interface.

During automatic transfers, SLCDC is responsible for properly sequencing display data and control strings in a data stream that is sent to the LCD controller to fill its internal display RAM. Display data is typically written to the external controller in pages. Each write to the controller fills one column of the current page. SLCDC assumes that the controller automatically increments its display RAM pointers after each write.

NOTE

Changing SLCDC configuration in middle of transfer can cause corruption of data stream output to LCD. SLCDC configuration must only be changed when GO bit and BUSY bit of SLCDC control/status register are cleared.

41.2.1 Word Size Definition

Due to increased complexity of color displays, SLCDC supports 16- or 8-bit transfers of command or display data. This is controlled by WRD_DEF_COM and WRD_DEF_DAT bits in LCD transfer configuration register. All registers in SLCDC are defined in words.

WRD_DEF_DAT controls definition of word for all data registers:

- DATA_BUFFER_SIZE
- LCD_CONFIG

WRD_DEF_COM controls the definition of word for all command registers:

- COMMAND_BUFFER_SIZE

WRD_DEF_WRITE controls the definition of LCDDAT for the register:

- LCD_WRITE_DATA

41.2.2 Image Endianness

In the examples of automatic transfers shown in this document, all memory images are either big endian or 32-bit little endian. However, it is possible that user may have an image that is stored in half word (16-bit) or byte (8-bit little endian). SLCDC can compensate for this by using the IMGEND bits in the LCD_TRANSFER_CONFIG register. These bits causes SLCDC to convert 16- or 8-bit little endian image to a big endian image. The resultant big endian decode is used for the rest of the data processing.

Table 41-2. Image Endianness

IMGEND	Conversion	32-Bit Word Data Stored in Memory	32-Bit Data After Conversion
00	Big endian or 32-bit little endian to big endian	0x12345678	0x12345678
01	16-bit little endian to big endian	0x56781234	0x12345678
10	8-bit little endian to big endian	0x78563412	0x12345678

41.2.3 Accessing the LCD Controller

SLCDC provides two methods of accessing the external display device.

- The first, and preferred method, is through automatic writes handled by SLCDC.
- The second, is through direct IP register access using LCD write data register.

41.2.3.1 Automatic SLCDC Transfers

SLCDC is designed to take a block of data from system memory and write it, without software intervention, to an external display controller. This is done by giving SLCDC a starting address and a data block size (in words). This method can be used for both LCD display data as well as command data. Transfers are initiated by setting the GO bit in SLCDC control/status register. Data transfer is done in the background. Upon completion, a maskable interrupt is generated. Automatic transfers are accomplished in one of three ways:

- A block of data is transferred to the LCD controller with control strings automatically inserted to navigate through the LCD controller's internal RAM. This mode makes use of a second buffer in system memory to store the software commands necessary for LCD controller RAM addressing.
- A block of data is transferred from system memory to the display controller while tying the LCD_RS pin low.
- A block of data is transferred from system memory to the display controller while tying the LCD_RS pin high.

Automatic transfer mode control bits (AUTOMODE[1:0]) of SLCDC control/status register configures the SLCDC for one of the three supported automatic transfer modes.

41.2.3.1.1 Automatic Display Data Transfers (AUTOMODE[1:0][1]=10)

SLCDC is designed to display a frame buffer from system memory to external LCD controller without any software intervention while transferring. To accomplish this, SLCDC must automatically transmit control strings for LCD controller RAM addressing at appropriate times in the frame buffer data stream. LCD controller RAMs are typically organized into pages. After RAM page is filled, controller's page address must be set to the next page. SLCDC contains counters that monitor the number of display data words written to the LCD controller to determine when the current page of RAM is filled. After this page is filled, SLCDC inserts a control string into the data stream to set controller RAM's page address to next page. SLCDC requires this control string be stored in a command buffer in system memory. The organization of command buffer is discussed later in this section. To correctly move a complete frame of image data from system memory to the LCD controller, SLCDC must be programmed with the following LCD information:

- Command word definition (8-bit or 16-bit words)
- Data word definition (8-bit or 16-bit words)
- Number of words in a page of LCD image data
- Number of words in the LCD frame data buffer
- Type of LCD interface (serial or parallel)
- Data clock polarity of the LCD (for serial interfaces only)
- Endianness of the image (big endian, little 32-bit, little 16-bit, or little 8-bit)
- Chip-select polarity for the LCD
- Transfer clock speed requirements of the LCD
- Control string needed at the start of each page of LCD display data (stored in a separate command buffer in system memory)

1. Automode [1:0] = 11 is reserved.

SLCDC uses number of words per page of LCD image data information (stored in WORDPPAGE[12:0] bits of LCD CONFIG register) to determine when a set of addressing commands must be sent to the LCD. When the current display RAM page is full, SLCDC must increment the page address and reset the column address of the LCD controller to 0. This is accomplished by reading the next control string from the command buffer and transmitting it to the LCD controller. SLCDC continues to send display data and command strings as needed until the controller's display RAM is filled. Transfer is finished when the number of data words transferred equals the number of words in the buffer as defined by DATABUFSIZ bits in DATA BUFFER SIZE registers.

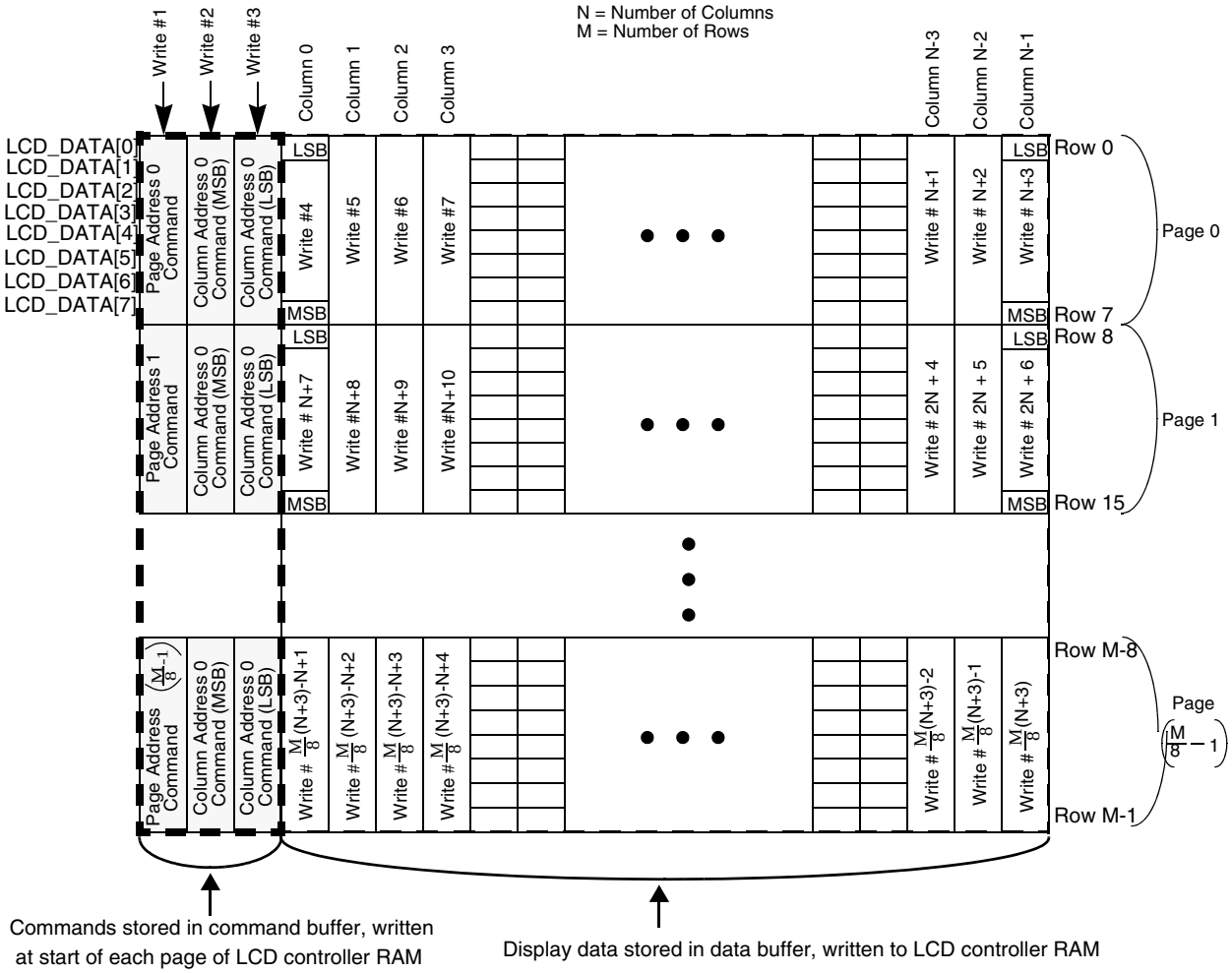


Figure 41-2. Sample LCD Controller Memory Mapping (Monochrome) (8-Bit Command/8-Bit Data)

Figure 41-2 shows an example of how an external LCD controller RAM is filled during automatic SLCDC writes for a monochrome display. Command and data word sizes are both defined as bytes. In this example, LCD controller requires three "command" words (bytes) to set the page and column address. Setting GO bit of control/status register starts the data transfer from system memory to the LCD controller. The first word written to LCD sets the page address to 0. The next two words written are control strings that set the LCD column address to 0. Word write #4 starts transfer of image data to the LCD. The first word of display data maps to the display column 0. The MSB of the word maps to row 7 and the LSB maps to row 0.

Words of display data continue to be written to LCD until the SLCDC internal word counter equals the value stored in WORDPPAGE[12:0] bits of LCD CONFIG register. At this point, SLCDC issues a command string, taken from the command buffer, to increment the page address and reset the column address of LCD. This sequence continues until LCD controller's data RAM is filled. After display buffer has been filled, IRQ bit of SLCDC control/status register is set. Also, BUSY bit and GO bit is cleared. An interrupt is generated if IRQEN bit is set and the system is configured for SLCDC interrupts.

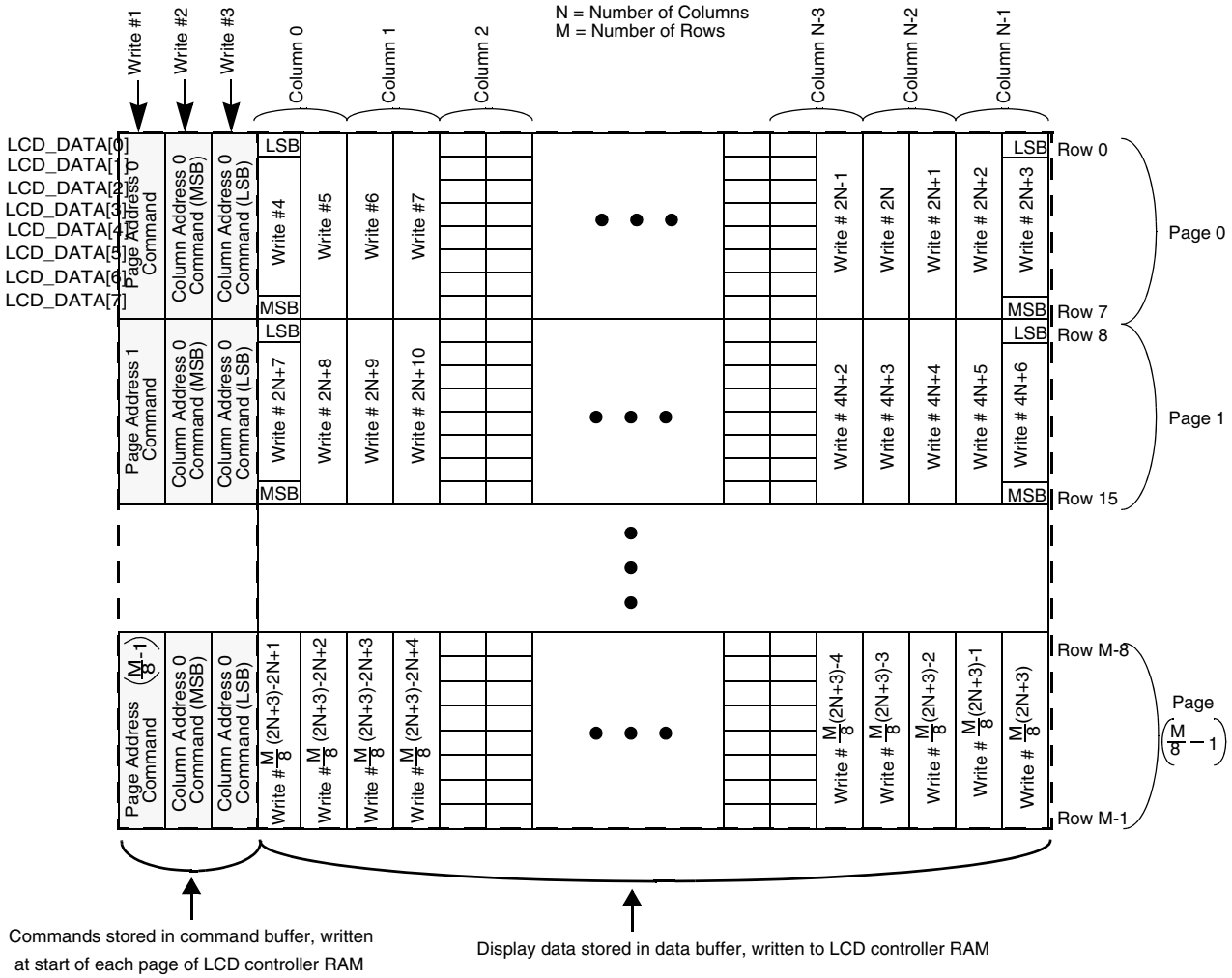


Figure 41-3. SLCDC LCD Controller Memory Mapping (2-Bit Color/Gray Scale) (8-Bit Command/8-Bit Data)

Figure 41-3 shows the mapping for a 2-bit color/gray scale display. Command and data word sizes are defined as bytes. For this configuration, each column of data within the page requires 2 word writes. The sequence remains the same as 1-bit per pixel case shown in Figure 41-2, however twice as many display data writes are required to fill the page.

SLCDC is designed to deal with a large variety of LCD controllers from multiple vendors. While the hardware interface to the external controller is somewhat standard, software interface can vary from chip to chip. To deal with the software differences between LCD controllers, SLCDC makes use of a separate

command buffer stored in system memory to go along with the display data buffer. Figure 41-4 shows the organization of the information in system memory. LCD page and column command buffer should be set up to contain the commands necessary to navigate through the LCD controller's internal RAM. Each command in the buffer is preceded by a tag. SLCDC uses the tags to determine the LCD_RS pin value during the command word transmission.

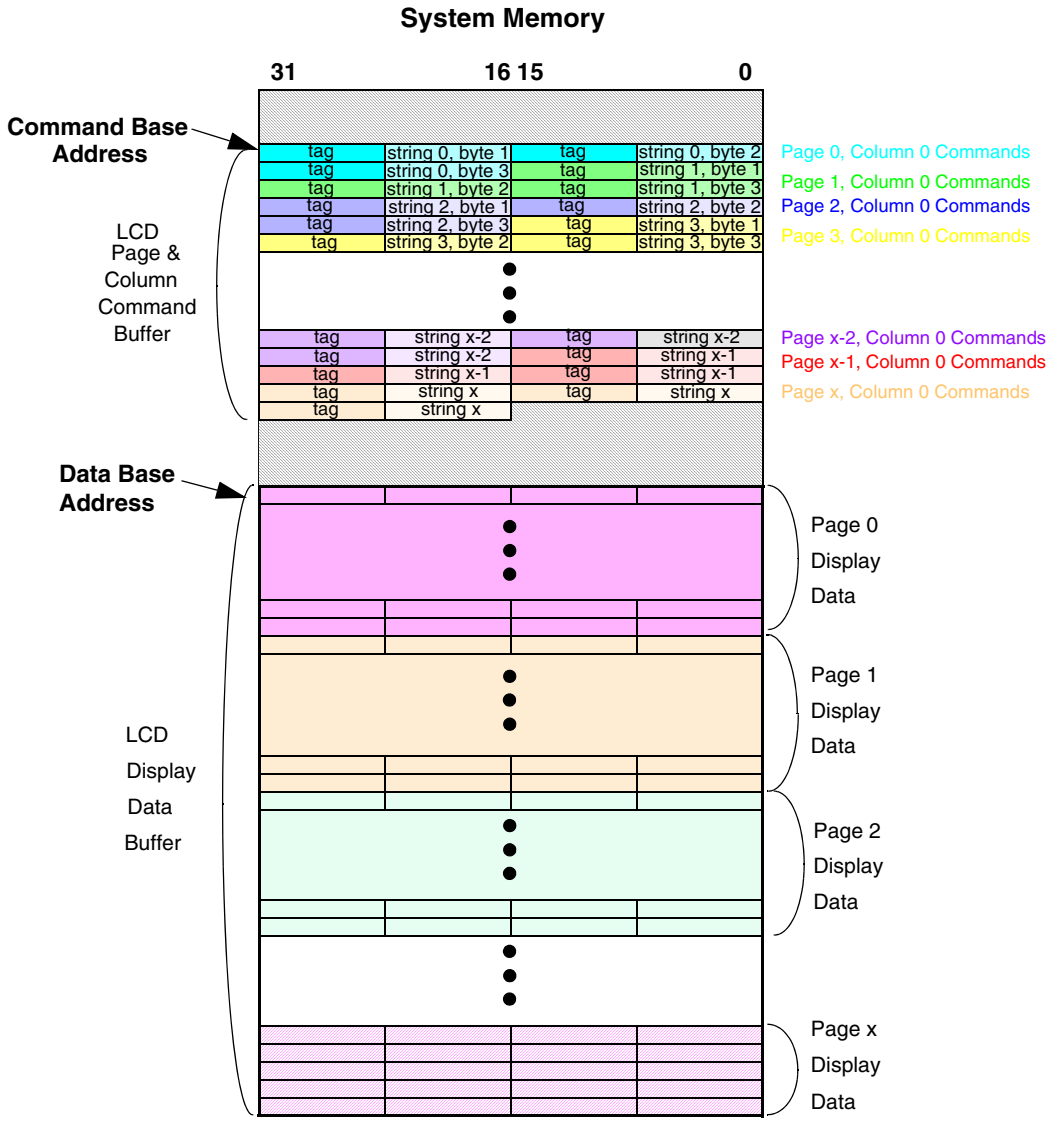


Figure 41-4. Automatic Display Data Transfer Memory Configuration (8-Bit Command/8-Bit Data)

Figure 41-5 shows how the commands and tags are organized in system memory. In this example, command string length is 3, that is, three words are required to be transmitted to the LCD controller at the start of each page of LCD RAM. Each of the command words have a tag placed adjacent to it in the memory. The command is placed immediately after its tag in the memory. Currently SLCDC only uses the least-significant bit of the tag field. This bit, labeled "RS" in Figure 41-5, sets the LCD_RS pin value while the command word is being transferred to the LCD controller. If RS tag bit is set to "1", LCD_RS is driven

to 1 during the transfer of the corresponding command byte. Similarly, if RS tag bit is set to “0”, LCD_RS is driven to 0 during the transfer of the corresponding command byte.

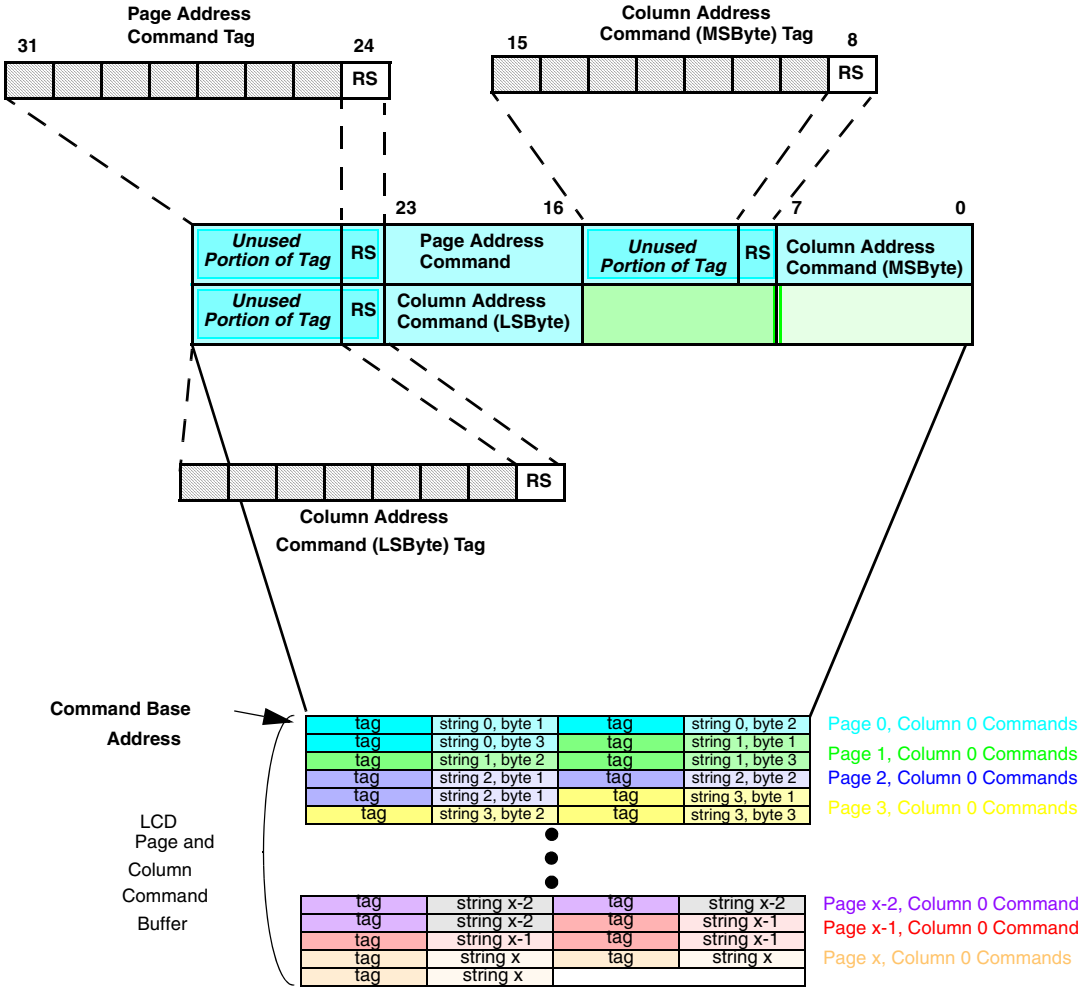


Figure 41-5. Command Buffer Tag Organization (8-Bit Words)

Figure 41-6 shows a 16-bit command and 16-bit data transaction for a 16-bit color display. In this example, command length is 3. Each LCD data transfer is for one display pixel. The mapping of display pixels for 16-bit color is generally “RRRRGGGGBBBBXXX” (R=red G=green B=blue X=ignored). Check your display to see what the 16-bit color mapping is. In this case, a page consists of a row of pixels. Three 16-bit words of command data are transferred to the display first, then 16-bit transfers of data will occur until the WORDPPAGE are transferred. Three more command words are sent designating the next page, and the cycle repeats itself.

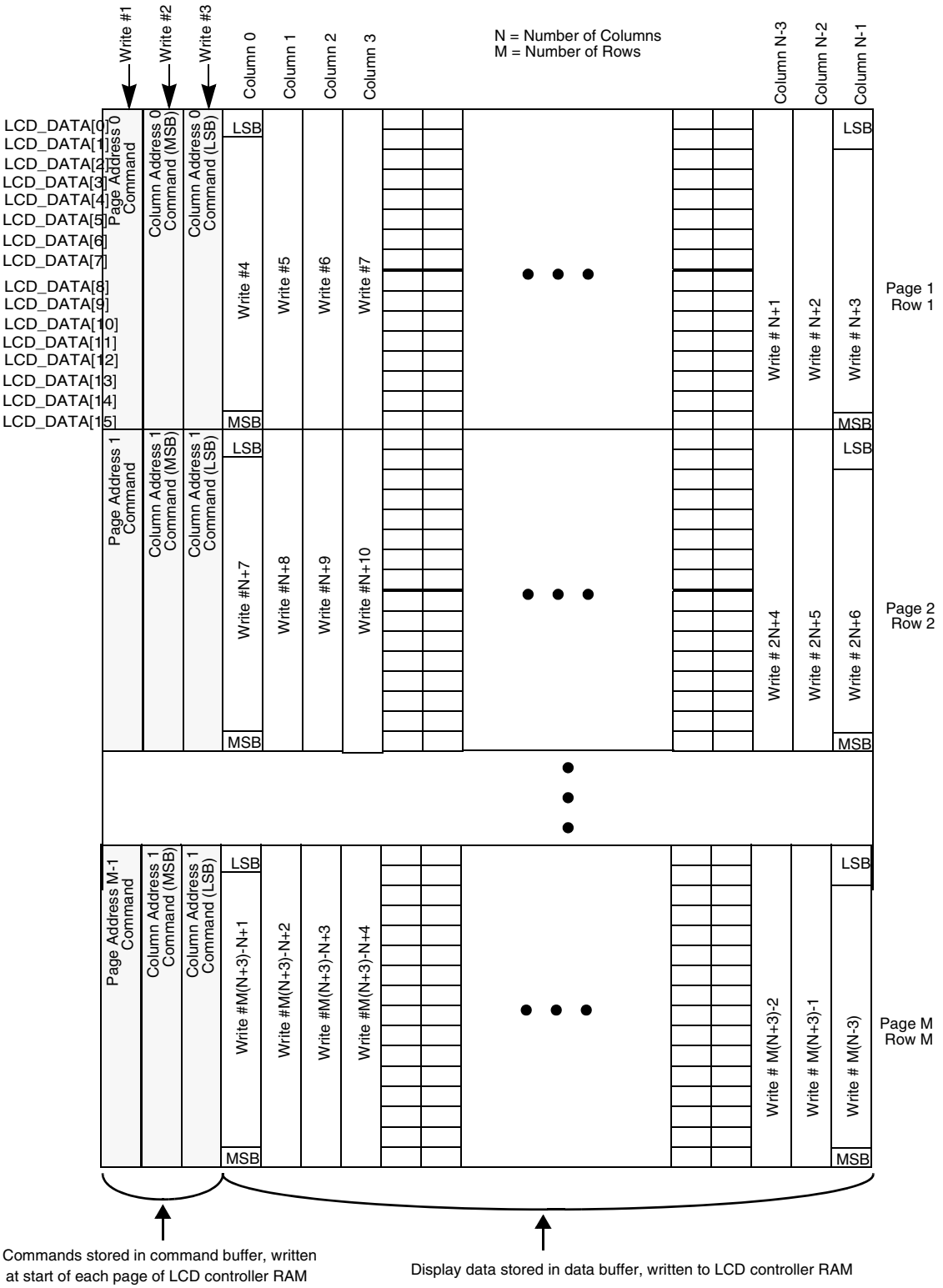


Figure 41-6. Sample LCD Controller Memory Map (16-Bit Color) (16-Bit Command/16-Bit Data)

Figure 41-7 shows how a 16-bit command word is stored in system memory. The 16-bit command is stored in the least significant half-word of memory (15–0). Tag is stored in the most significant half-word of memory (31–16). Currently only bit 16 of the tag is used for RS.

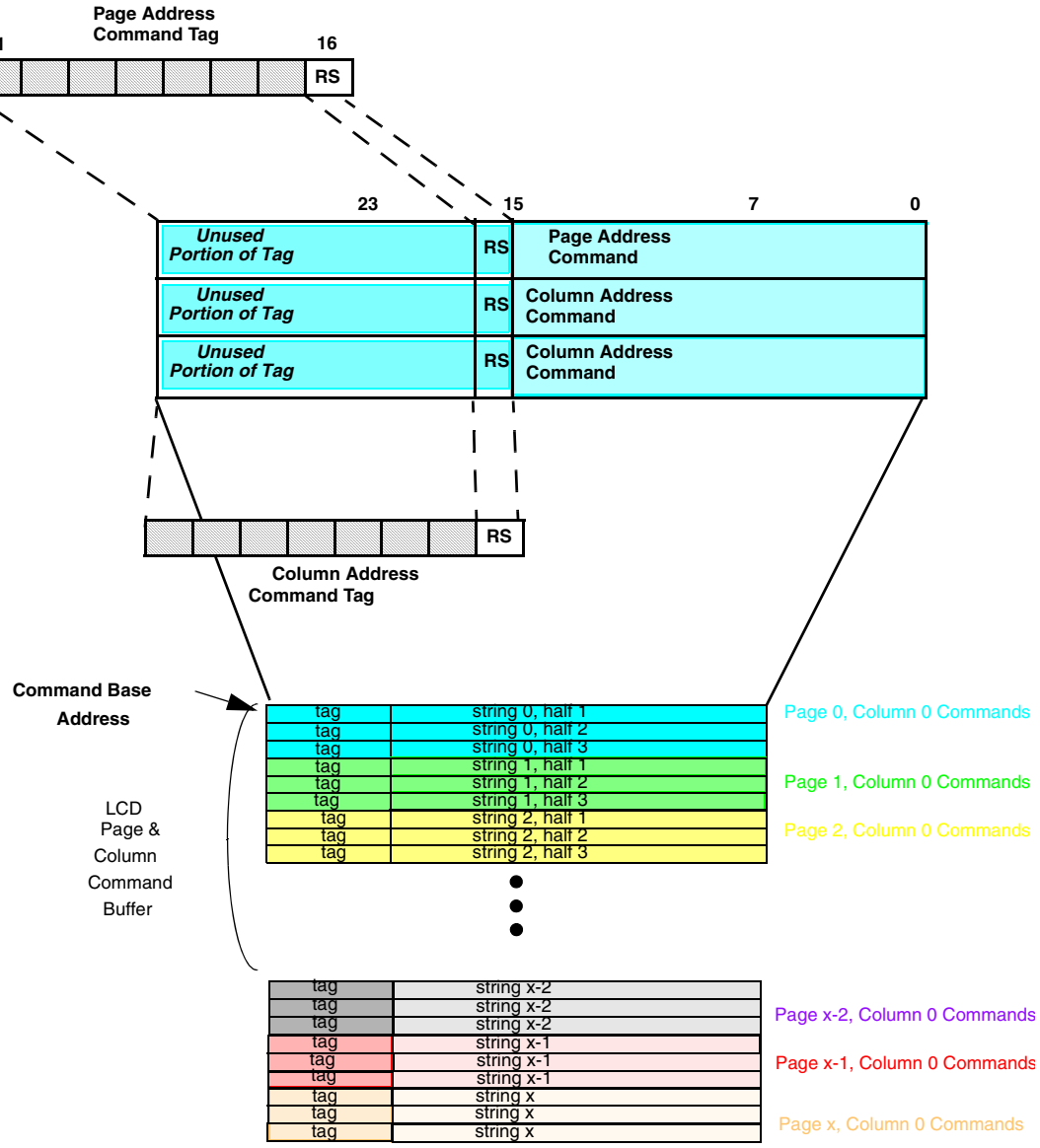


Figure 41-7. Command Buffer Tag Organization (16-bit Words)

Figure 41-8 shows how 16-bit word commands and 16-bit word data is stored in system memory. Because command data is 17-bits long (16-bits of command + 1-bit tag), each command takes up 1 word of system memory. Each 16-bit word of data can be placed in a half-word of system memory.

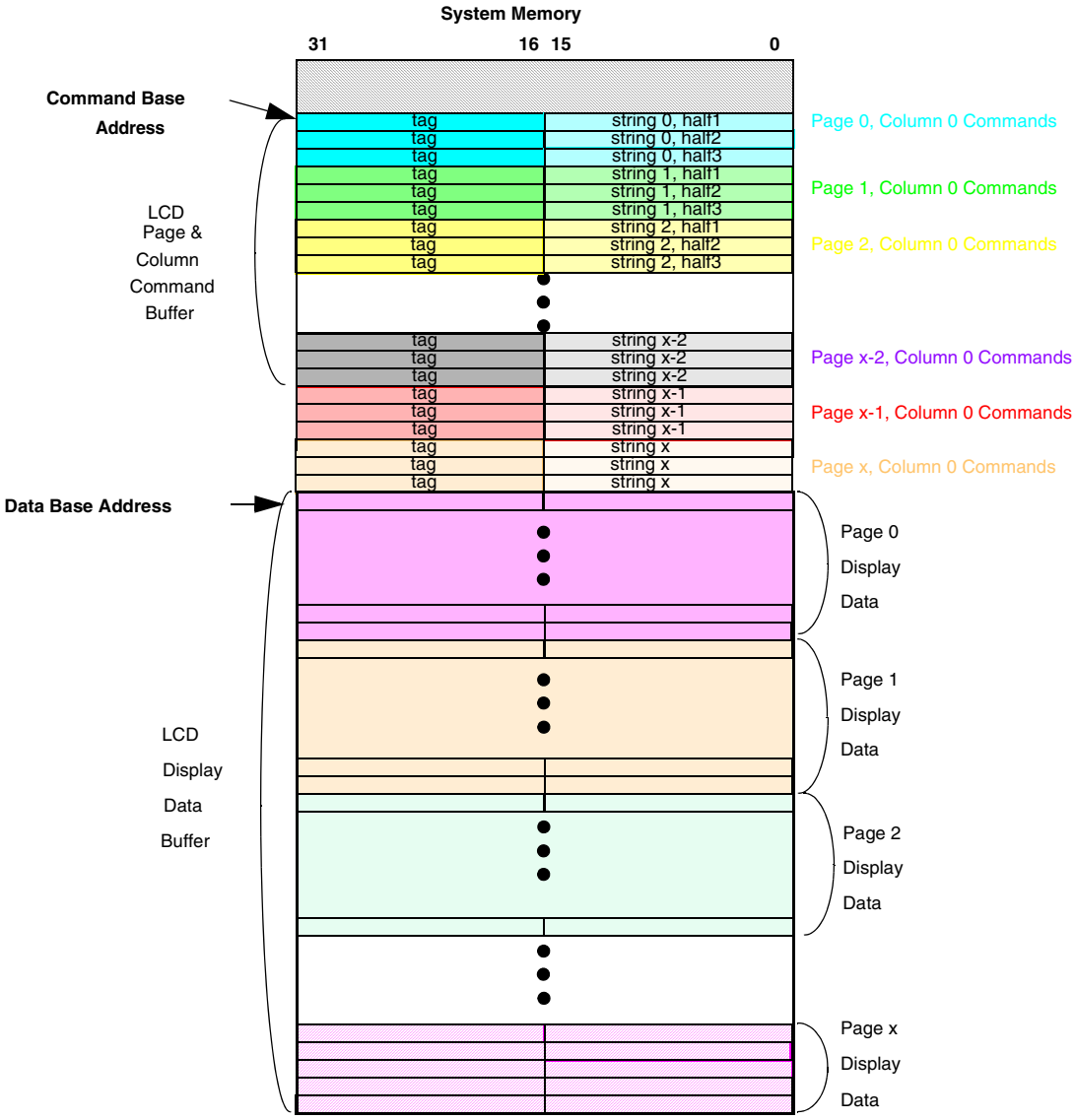


Figure 41-8. Automatic Display Data Transfer Memory Configuration (16-Bit Command/16-Bit Data)

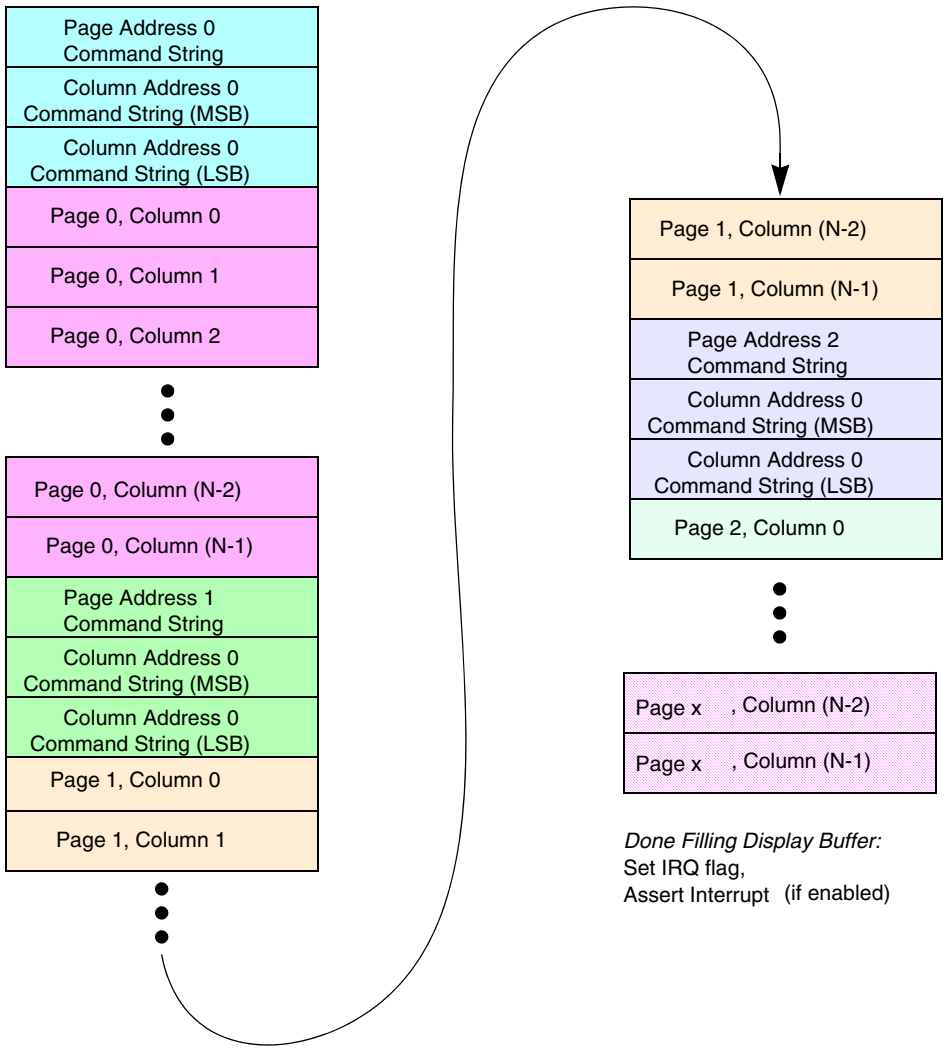


Figure 41-9. SLCDC Automatic Mode Write Sequence (Monochrome Display)

Figure 41-9 and Figure 41-10 show the sequence in which the data words are written to the LCD controller when AUTOMODE[1:0] =10. These figures illustrate how the command strings, taken from the command buffer, are interleaved with display data taken from the data buffer.

It is important to note that different combinations of word definitions are allowed. For example, there can be 8-bit command words and 16-bit data words. In this case, commands would be stored in system memory much like Figure 41-5, and data would be stored like Figure 41-8. See Section 41.3, "LCD Controller Interface," to determine how mixed transfers occur.

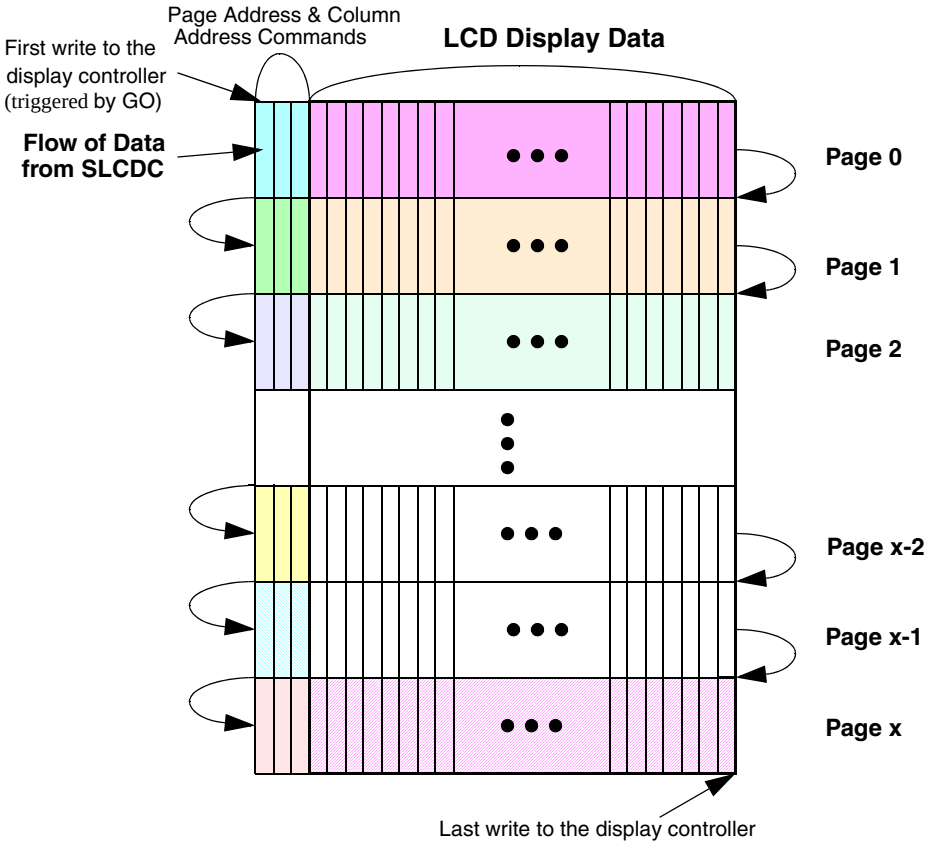


Figure 41-10. SLCDC Automatic Mode Data Flow (AUTOMODE[1:0] = 10)

41.2.3.1.2 Automatic Command Data Transfers (AUTOMODE[1:0]=00)

SLCDC provides a convenient method of moving data from system memory to an external device, like a LCD controller. When AUTOMODE[1:0] bits = 00, SLCDC ignores the command buffer and transfers the contents of specified data buffer to SLCDC output pins, without inserting the page and column addressing control strings. When AUTOMODE[1:0] = 00, LCD_RS output is held to 0 during the entire transfer. This method is used for sending a block of commands to the LCD controller for initialization. The buffer transmitted is defined by DATABASEADR[16:0] bits of DATA BASE ADDRESS register and DATABUFSIZE[16:0] bits of DATA BUFFER SIZE register. Setting GO bit of SLCDC control/status register starts the transfer and sets the BUSY status bit. After the number of words transferred to the external device is equal to the buffer size defined by the DATA BUFFER SIZE register, BUSY and GO bit will be cleared, and IRQ flag will be set. A system interrupt is generated if IRQEN bit is set.

41.2.3.1.3 Automatic Command Data Transfers (AUTOMODE[1:0]=01)

SLCDC provides a convenient general purpose method of moving data from system memory to an external device, like a LCD controller. When AUTOMODE[1:0] bits = 01, SLCDC ignores the command buffer and transfers the contents of the specified data buffer to the SLCDC output pins without inserting page and column addressing control strings. When AUTOMODE[1:0] = 01, LCD_RS output is held to 1 during the entire transfer. The buffer transmitted is defined by DATABASEADR[31:0] bits of DATA BASE ADDRESS register and DATABUFSIZ[16:0] bits of DATA BUFFER SIZE register. Setting GO bit of

SLCDC control/status register starts the transfer and sets the BUSY status bit. After the number of words transferred to the external device is equal to the buffer size defined by DATA BUFFER SIZE register, BUSY and GO bit is cleared, and IRQ flag is set. A system interrupt is generated if IRQEN bit is set.

41.2.3.2 Direct Register Access

Single words are written to the external LCD controller using 9-bit LCD WRITE DATA register. Any write to the LCD WRITE DATA register results in a write to the external display controller, provided that SLCDC is not currently in middle of a transfer (indicated by BUSY=1 in control/status register). The value written to LCDDAT[15:0] bits of LCD write data register are transmitted according to settings stored in LCD transfer configuration register, that is, transfer is done according to the settings of WORDDEFWRITE, XFRMODE, CSPOL, and SCKPOL bits of LCD transfer configuration register.

Typical LCD controllers use a register select signal to distinguish between display data writes and control command writes. RS bit of LCD WRITE DATA register determines whether a write is presented to the external controller as a command string or a display data string. LCD_RS pin is held to the value stored in the RS bit during byte transfer. Since transfer to the external device is slow compared to the system clock rates, BUSY bit in the control/status register is used to indicate that a transfer is in progress. A transfer initiated by a write to the WRITE DATA register causes an interrupt if IRQEN bit of SLCDC control/status register is set.

WORDDEFWRITE in LCD transfer configuration register determines if an 8- or 16-bit transfer occurs. An 8-bit transfer causes LCDDAT[7:0] bits to be transferred, and LCDDAT[15:8] to be ignored. Another direct write cannot be done until the previous transfer has completed (BUSY=0).

41.2.4 Aborting SLCDC Transfers

ABORT bit in SLCDC control/status register is used to terminate a SLCDC transfer that is in progress. Termination occurs gracefully. Any byte transfer that is in progress when ABORT is asserted is completed before the SLCDC goes to an idle state. BUSY status bit clears upon entry to the idle state. IRQ flag will also assert, indicating that the abort is complete. A SLCDC interrupt is generated, if enabled.

41.2.5 Low-Power Mode Operation

SLCDC does not contain any control registers to program SLCDC behavior during low-power modes. Any power savings in the usage of SLCDC are gained by reducing the system clock to a lower rate.

41.2.6 Memory Map

Table 41-3. SDLC Memory Map

Address	Register	Access	Reset Value	Section/Page
0x1002_2000 (DATABASEADR)	Data Buffer Base Address Register	R/W[1]	0x0000_0000	41.2.9/41-18
0x1002_2004 (DATABUFSIZE)	Data Buffer Size register	R/W	0x0000_0000	41.2.10/41-18
0x1002_2008 (COMBASEADR)	Command Base Address register	R/W	0x0000_0000	41.2.11/41-19

Table 41-3. SDLC Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0x1002_200C (COMBUFSIZ)	Command Buffer Size register	R/W	0x0000_0000	41.2.12/41-19
0x1002_2010 (COMSTRINGSIZ)	Command String Size register	R/W	0x0000_0000	41.2.13/41-20
0x1002_2014 (FIFOCONFIG)	FIFO Configuration register	R/W	0x0000_0000	41.2.14/41-21
0x1002_2018 (LCDCONFIG)	LCD Controller Configuration register	R/W	0x0000_0000	41.2.15/41-21
0x1002_201C (LCDTRANSCONFIG)	LCD Transfer Configuration register	R/W	0x0000_0000	41.2.16/41-22
0x1002_2020 (SLCDCCONTROL/STATUS)	Smart LCD Control/Status register	R/W	0x0000_0000	41.2.17/41-23
0x1002_2024 (LCDCLOCKCONFIG)	LCD Clock Configuration register	R/W	0x0000_0000	41.2.18/41-26
0x1002_2028 (LCD WRITE DATA)	LCD Write register	R/W	0x0000_0000	41.2.19/41-26

Note:

1 Some R/W registers may contain some read-only or write-only bits.

41.2.7 Register Summary

The conventions in Figure 41-11 and Table 41-4 serve as a key for register summary and individual register diagrams.

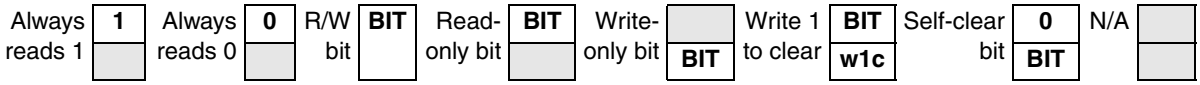


Figure 41-11. Key to Register Fields

Table 41-4. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.

Table 41-4. Register Conventions (continued)

Convention	Description
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

41.2.8 SLDC Register Descriptions

This section provides detailed descriptions of various SLCDC registers. Table 41-5 provides the detail summary for SLCDC registers.

Table 41-5. SLCDC Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1002_2000 (DATABASEADR)	R	DATABASEADR[31:2]															
	W																
	R	DATABASEADR[31:2]														0	0
	W																
0x1002_2004 (DATABUFSIZE)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	W																
	R	DATABUFSIZ[16:0]															
	W																
0x1002_2008 (COMBASEADR)	R	COMBASEADR[31:2]															
	W																
	R	COMBASEADR[31:2]														0	0
	W																
0x1002_200C (COMBUFSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COMBUFSIZ[16:0]
	W																
	R	COMBUFSIZ[16:0]															
	W																
0x1002_2010 (COMSTRINGSIZ)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	COMSTRINGSIZ[7:0]							
	W																

Table 41-5. SLCDC Register Summary (continued)

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1002_2014 (FIFOCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BURST	
	W																
0x1002_2018 (LCDCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	WORDPPAGE[12:0]												
	W																
0x1002_201C (LCDTRANSCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IMGEND	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WORD DEF WRITE	WOR DDEF DAT	WOR DDEF COM	XFR MOD E	CS POL	SCK POL
	W																
0x1002_2020 (SLCDCCONTROL/STATUS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	AUTOM ODE [1:0]		0	0	PR OT1	IRQ EN	IRQ	UNDR FLOW	IRQ	0	BUS Y	0	0
	W										W1C	W1C	W1C			ABO RT	GO
0x1002_2024 (LCDCLOCKCONFIG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DIVIDE[5:0]					
	W																
0x1002_2028 (LCD WRITE DATA)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RS
	W																
	R	LCDDAT[15:0]															
	W																

41.2.9 Data Buffer Base Address Register (DATABASEADR)

Address 0x1002_2000 (DATABASEADR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATABASEADR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATABASEADR														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-12. Data Buffer Base Address register

Table 41-6. Data Buffer Base Address Register Field Description

Field	Description
31–2 DATABASEADR	Data Buffer Base Address. This register contains the pointer in R-AHB address space to the start of LCD data buffer. This value is used to form the 32-bit data buffer base address. Data buffer base address value is forced to be word-aligned because the two LSBs are forced to 0. For example, data buffer base address [1:0] = 00.
1–0	Reserved.

41.2.10 Data Buffer Size Register (DATABUFSIZE)

Address 0x1002_2004 (DATABUFSIZE) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATABUFSIZE(17:0)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-13. Data Buffer Size register

Table 41-7. Data Buffer Size Register Description

Field	Description
31–17	Reserved.
16–0 DATABUFSIZ	Data Buffer Size. This register defines the length (in words) of LCD image or control buffer stored in the system memory. This value is used to determine when DMA transfer of data buffer from system memory to LCD controller is complete.

NOTE

SLCDC DMA address generator width is 17-bits. Therefore, it is possible that a data buffer can be incorrectly addressed by SLCDC, if it extends beyond a 128 kbyte boundary. Therefore, it is recommended that data buffer base address be set at the beginning of a 128 kbyte boundary. The sum of DATABASEADR[16:0] and DATABUFSIZ[16:0] must not exceed a multiple of 128K.

41.2.11 Command Buffer Base Address Register (COMBASEADR)

Address 0x1002_2008 (COMBASEADR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COMBASEADR(31:2)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMBASEADR(31:2)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-14. Command Buffer Base Address Register

Table 41-8. Command Base Address Register Field Description

Field	Description
31–2 COMBASEADR	Command Buffer Base Address. This register contains the pointer in R-AHB address space to the start of LCD command buffer that contains the LCD controller page and column address commands. This buffer is used when SLCDC is configured for transfers with automatic page address and column address command insertion. The command buffer base address value is forced to be word-aligned because the two LSBs are forced to 0. For example, command buffer base address [1:0] = 00.
1–0	Reserved.

41.2.12 Command Buffer Size Register (COMBUFSIZ)

Address 0x1002_200C (COMBUFSIZ) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COMB UFSIZ (16:0)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMBUFSIZ (16:0)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-15. Command Buffer Size Register

Table 41-9. Command Buffer Size Register Field Description

Field	Description
31–17	Reserved.
16–0 COMBUFSIZ	Command Buffer Size. This register defines the length (in words) of LCD control buffer stored in system memory. This value is used to determine when DMA transfer of command buffer from system memory to the LCD controller is complete.

NOTE

SLCDC DMA address generator width is 17-bits. Therefore, it is possible that a command buffer can be incorrectly addressed by the SLCDC if it extends beyond a 128 kilobyte boundary. Therefore, it is recommended that the command buffer base address be set at the beginning of a 128 kilobyte boundary. The sum of COMBASEADR[16:0] and COMBUFSIZ[16:0] must not exceed a multiple of 128K.

41.2.13 Command String Size Register (COMSTRINGSIZ)

Address 0x1002_2010 (COMSTRINGSIZ) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	COMSTRINGSIZ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-16. Command String Size Register

Table 41-10. Command String Size Register Field Description

Field	Description
31–8	Reserved.
7–0 COMSTRINGSIZ	Command String Size. Length (in words) of the LCD command string made up of LCD controller page and column address commands. This command string is used when SLCDC begins to fill a new page of the LCD controller's RAM. **Note:** These strings are only transmitted to the LCD controller when SLCDC is configured for transfers with automatic page and column address command insertion. Command strings are stored along with tags in the LCD command buffer located in system memory. These bits represent the number of words that must be sent to the LCD controller to set the page and column address, not the number of words needed to store the corresponding commands and tags in system memory.

41.2.14 FIFO Configuration Register (FIFOCONFIG)

Address 0x1002_2014 (FIFOCONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	BURST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-17. FIFO Configuration Register

Table 41-11. FIFO Configuration Register Description

Field	Description
31–3	Reserved.
2–0 BURST	DMA Burst Length. Length (in 32-bit words) of the DMA burst. that is, these bits determine the number of 32-bit word reads that occur when SLCDC has ownership of R-AHB bus. 000 Burst length set to 1 32-bit word 001 Burst length set to 2 32-bit word ... 111 Burst length set to 8 32-bit words

41.2.15 LCD Controller Configuration Register (LCDCONFIG)

Address 0x1002_2018 (LCDCONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	WORDPPAGE												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-18. LCD Controller Configuration Register

Table 41-12. LCD Controller Configuration Register Field Description

Field	Description
31–13	Reserved.
12–0 WORDPPAGE	LCD Bytes Per Page. Number of words per page for external LCD controller connected to SLCDC module. These bits are used by SLCDC to determine when control characters must be sent to the controller. 0 = 0 words per LCD page, 1 = 1 word per LCD page … 8191 = 8191 words per LCD page

41.2.16 LCD Transfer Configuration Register (LCDTRANSCONFIG)

Address 0x1002_201C (LCDTRANSCONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IMGEND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	WORDDEFWRITE	WORDDEFDAT	WORDDEFCOM	XFRMODE	CSPOL	SKCPOL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-19. LCD Transfer Configuration Register

Table 41-13. LCD Transfer Configuration Register Field Description

Field	Description
31–18	Reserved.
17–16 IMGEND	Image Endianness. This field defines the image endianness. 00 Big endian or 32-bit little endian 01 16-bit little endian 10 8-bit little endian 11 Reserved
15–6	Reserved.
5 WORDDEFWRITE	Word Define, Write Data. It defines word for the LCD WRITE DATA register transfers. 0 8-bit data words 1 16-bit data words
4 WORDDEFDAT	Word Define, Data. It defines word for LCD data registers and data transfers. All registers associated with data transfers (DATA BUFFER SIZE, LCD CONFIG) must take this into consideration. 0 8-bit data words 1 16-bit data words
3 WORDDEFCOM	Word Define, Command. It defines word for LCD command registers and command data transfers. All registers associated with data transfers (COMMAND BUFFER SIZE) must take this into consideration. 0 8-bit command words 1 16-bit command words
2 XFRMODE	Image Data Transfer Width. This bit selects the width of LCD display interface data bus. Data transfers to the LCD controller can be done in a serial or parallel manner. 0 word serial transfers 1 word parallel transfers

Table 41-13. LCD Transfer Configuration Register Field Description (continued)

Field	Description
1 CSPOL	Chip Select Polarity. Selects LCD_CS signal polarity. It affects the LCD_CS pin behavior. 0 LCD_CS is asserted low. Therefore, SLCDC will drive LCD_CS to 0 during serial transfer of data to external LCD controller. In parallel mode, LCD_CS will normally be high. LCD data will change when LCD_CS goes low. LCD_CS rising edge is used by the LCD controller to latch the parallel data. 1 LCD_CS is asserted high. Therefore, SLCDC will drive LCD_CS to 1 during serial transfer of data to external LCD controller. In parallel mode, LCD_CS will normally be low. LCD data will change when LCD_CS goes high. LCD_CS falling edge is used by LCD controller to latch the parallel data.
0 SCKPOL	Serial Data Clock Polarity. It selects whether serial data transitions on rising/falling edge of serial data clock during transfers of data in serial mode. It has no effect during parallel data transfers. 0 Serial data will transition on the rising edge of serial clock. Therefore, data should be latched by the display device on the falling edge of serial clock. 1 Serial data will transition on the falling edge of serial clock. Therefore, data should be latched by the display device on the rising edge of serial clock.

41.2.17 SLCDC Control/Status Register (SLCDCCONTROL/STATUS)

0x1002_2020 (SLCDCCONTROL/STATUS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	AUTOMODE		0	0	PROT 1	IRQEN	IRQ	UNDRFLOW	TEA	0	BUSY	ABORT	GO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-20. SLCDC Control/Status Register

Table 41-14. SLCDC Control/Status Register Field Description

Field	Description
31–13	Reserved.
12–11 AUTOMODE	Automatic Transfer Mode. These bits control how SLCDC transfers the data buffer to the LCD controller upon being triggered by the GO bit. DATA BASE ADDRESS bits designate the location of transferred data buffer in the system memory. Transfers triggered by GO bit can be configured to automatically insert page address and column address commands in the data stream to navigate through the LCD controller's RAM. This is done without CPU intervention. SLCDC can be also be configured to send the data buffer to the LCD controller without inserting any command strings. In this mode, the value of the LCD_RS signal can be configured to be either 1 or 0 during the transfer. 00 SLCDC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, LCD_RS signal is tied to 0 during the entire transfer. 01 SLCDC transfers the data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE without inserting control strings. With this setting, LCD_RS signal is tied to 1 during the entire transfer. 10 When transmitting data buffer defined by DATA BASE ADDRESS and DATA BUFFER SIZE, SLCDC inserts command strings to set the LCD controller's page and column addresses. These command strings are taken from the buffer defined by COMMAND BASE ADDRESS and COMMAND STRING SIZE. This insertion of command strings is done at beginning of each page of LCD controller RAM. 11 Reserved
10–9	Reserved.
8 PROT1	R-AHB Protection Code. This bit controls the HPROT[1] signal value sent to R-AHB each time a SLCDC access is done. This value controls the access type done on the bus (user or privileged). SLCDC_hprot[0] output of SLCDC is tied to 1 to indicate a data access. PROT1 bit resets to 0. 0 SLCDC_hprot[1:0] driven to 01 indicating SLCDC accesses will be user data accesses. 1 SLCDC_hprot[1:0] driven to 11 indicating SLCDC accesses will be privileged data accesses.
7 IRQEN	Interrupt Enable. This bit controls whether a SLCDC interrupt is generated upon completion of SLCDC transfer or not. SLCDC transfers can end in three ways: 1) When a transfer error occurs on the R-AHB. 2) When SLCDC completes the data transfer to the LCD controller. 3) When a SLCDC transfer is aborted by setting ABORT bit in SLCDC control/status register. SLCDC transfers are initiated by assertion of GO bit or a write to LCD WRITE DATA register. Clearing IRQEN bit prevents an interrupt from being generated when SLCDC completes a transfer. 0 SLCDC interrupt is masked. No interrupt is generated from SLCDC. 1 SLCDC interrupt is generated upon completion of a SLCDC transfer.
6 IRQ	SLCDC Interrupt Flag. This bit indicates that an interrupt condition occurred in SLCDC. Interrupt flag is set when a SLCDC transfer is completed or aborted. SLCDC transfers are initiated by assertion of GO bit or a write to LCD WRITE DATA register. This bit is cleared by writing a 1 to it. 0 SLCDC interrupt is not pending 1 Interrupt condition occurred in SLCDC.

Table 41-14. SLCDC Control/Status Register Field Description (continued)

Field	Description
5 UNDRFLOW	SLCDC FIFO Underflow. This bit indicates that SLCDC FIFO became empty during data transfer from system memory to LCDC. Although FIFO underflow does not cause an error in SLCDC transfer, it indicates the output portion of SLCDC had to wait for FIFO to be filled during transfer. This bit is cleared by writing a 1 to it. **Note:** Underflow flag will not set until some data has been read into the SLCDC FIFOs. It only sets when FIFOs does not refill fast enough for the SLCDC bit manipulator. In this case, SLCDC bit manipulator stalls while it waits for data to be read from system memory into the FIFOs. It is also not set in case when a SLCDC transfer is initiated, but SLCDC is never granted control of bus, and thus able to fill the FIFOs. FIFOs must go from a non-empty state to an empty state to set it. 0 No underflow occurred in SLCDC FIFO. 1 SLCDC FIFO became empty during data transfer to the LCD controller. This causes the output portion of SLCDC to wait, delaying data transfer to the LCD controller.
4 TEA	SLCDC DMA Transfer Error. This bit indicates that SLCDC DMA received a transfer error acknowledge from R-AHB bus while trying to read data from system memory. This is indicated by HRESP0=1 and HREADY=1 on the R-AHB bus. This is a fatal condition and causes SLCDC to terminate its transfer. This bit is cleared by writing a 1 to it. 0 No transfer error acknowledge error occurred. 1 A transfer error acknowledge was received by the SLCDC DMA from the R-AHB.
3	Reserved.
2 BUSY	SLCDC Busy. This bit indicates that SLCDC is in the process of transferring image buffer from system memory to LCD controller. Data transfer is initiated by setting the GO bit or by writing data to the LCD write data register. This bit is a read-only bit and clears after data transfer to the LCD controller is completed. 0 SLCDC idle. 1 SLCDC in process of transferring image buffer.
1 ABORT	Abort SLCDC Transfer. This bit causes SLCDC to abort the current data transfer. Termination occurs gracefully, that is, ongoing byte transfer to the LCD controller is completed before SLCDC goes to an idle state. This bit is cleared automatically when SLCDC has transitioned to the idle state. 0 Do not abort the current data transfer. 1 Abort the current SLCDC data transfer.
0 GO	Start SLCDC Transfer. This bits starts automatic transfer of image data from system memory to the external LCD controller. This bit always reads 1 until data transfer is complete. After data transfer is complete or has been aborted, GO bit will read 0. **Note:** Writing 1 to this bit has no effect if BUSY bit is set. 0 Do not start SLCDC transfer. 1 Start SLCDC image data transfer.

41.2.18 LCD Clock Configuration Register (LCDCLOCKCONFIG)

Address 0x1002_2024 (LCDCLOCKCONFIG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	DIVIDE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-21. LCD Clock Configuration Register

Table 41-15. LCD Clock Configuration Register Field Description

Field	Description
31–6	Reserved.
5–0 DIVIDE	LCD Clock Divide Value. This bit sets the divide ratio used to generate the LCD clock from HCLK_SLCDC clock. Setting DIVIDE to 0 disables the LCD_CLK by gating HCLK_SLCDC clock from the fractional divider. These bits must be set to some non-zero value before LCD transfers can occur. Frequency relations: 0 LCD_CLK off 1 LCD_CLK = HCLK_SLCDC clock*1/128 2 LCD_CLK = HCLK_SLCDC clock*2/128 … 63 LCD_CLK=HCLK_SLCDC clock*63/128

41.2.19 LCD Write Data Register (LCDWRITEDATA)

0x1002_2028 (LCD WRITE DATA) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LCDDAT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 41-22. LCD Write Data Register

Table 41-16. LCD Write Data Register Field Description

Field	Description
31–17	Reserved.
16 RS	LCD Register Select. Determines the value placed on LCD_RS pin during byte transfer from WRITE DATA register 0 LCD_RS signal is tied to 0 during transfer. 1 LCD_RS signal is set to 1 during transfer.
15–0 LCDDAT	LCD Controller Data. Data written is transferred to the external LCD controller. This register gives direct write access to the LCD controller. Data is determined to be command or display data using RS bit. Data written to this register is only transferred to the LCD controller if SLCDC is not in middle of another transfer (indicated by the BUSY bit of control/status register (Section 41.2.17, “SLCDC Control/Status Register (SLCDCCONTROL/STATUS)”). A read of these bits gives the last value written to this register (or the reset value), and not a value from the LCD controller. The amount of data that is transferred is determined by WORDDEFWRITE bit in the LCD transfer configuration register (Section 41.2.16, “LCD Transfer Configuration Register (LCDTRANSCONFIG)”). During 8-bit word transfers, LCDDAT[7:0] is transferred, and LCDDAT[15:8] is ignored.

41.3 LCD Controller Interface

SLCDC transfers data from the display memory buffer in system memory to the external display device. Transfers can be done using a 4-wire serial, 3-wire serial, 8-bit parallel, or a 16-bit parallel interface.

41.3.1 Serial Interface

In serial mode (XFRMODE bit in LCD Transfer Configuration Register is 0), data is transmitted to the display device using LCD_CS, LCD_DATA[7:6] and LCD_RS. Data is transmitted on LCD_DATA[7], serial clock toggles on LCD_DATA[6] and LCD_RS tells the display whether it is display data or command data. LCD_CS is used as a chip select signal. LCD_CS signal is asserted during all transfers. LCD_CS signal polarity is programmable using the CSPOL bit of LCD Transfer Configuration Register (Section 41.2.16, “LCD Transfer Configuration Register (LCDTRANSCONFIG)”).

Figure 41-23 shows timing associated with the transfer of one byte of data to the display. The polarity of the serial data clock on LCD_DATA[6] pin can be configured using SCKPOL bit of LCD Transfer Configuration Register (Section 41.2.16, “LCD Transfer Configuration Register (LCDTRANSCONFIG)”). This bit must be set in accordance with the external display device requirements.

If external device latches the serial data on rising edge of serial clock, SCKPOL must be set to 1. If external device latches the serial data on falling edge of serial clock, SCKPOL must be cleared. The command and data word definitions determines how long the data string is in serial mode.

Figure 41-23 shows 8-bit transfers. A serial transfer of 16-bits has same timing as 8-bit transfers, however 16-bits of data is transferred. It is possible to have a command defined as 8-bits and serial defined as 16-bits (or vice versa). In this case, when a command is being transferred, 8-bits of data are sent, and when data is being transferred, 16-bits of data is sent.

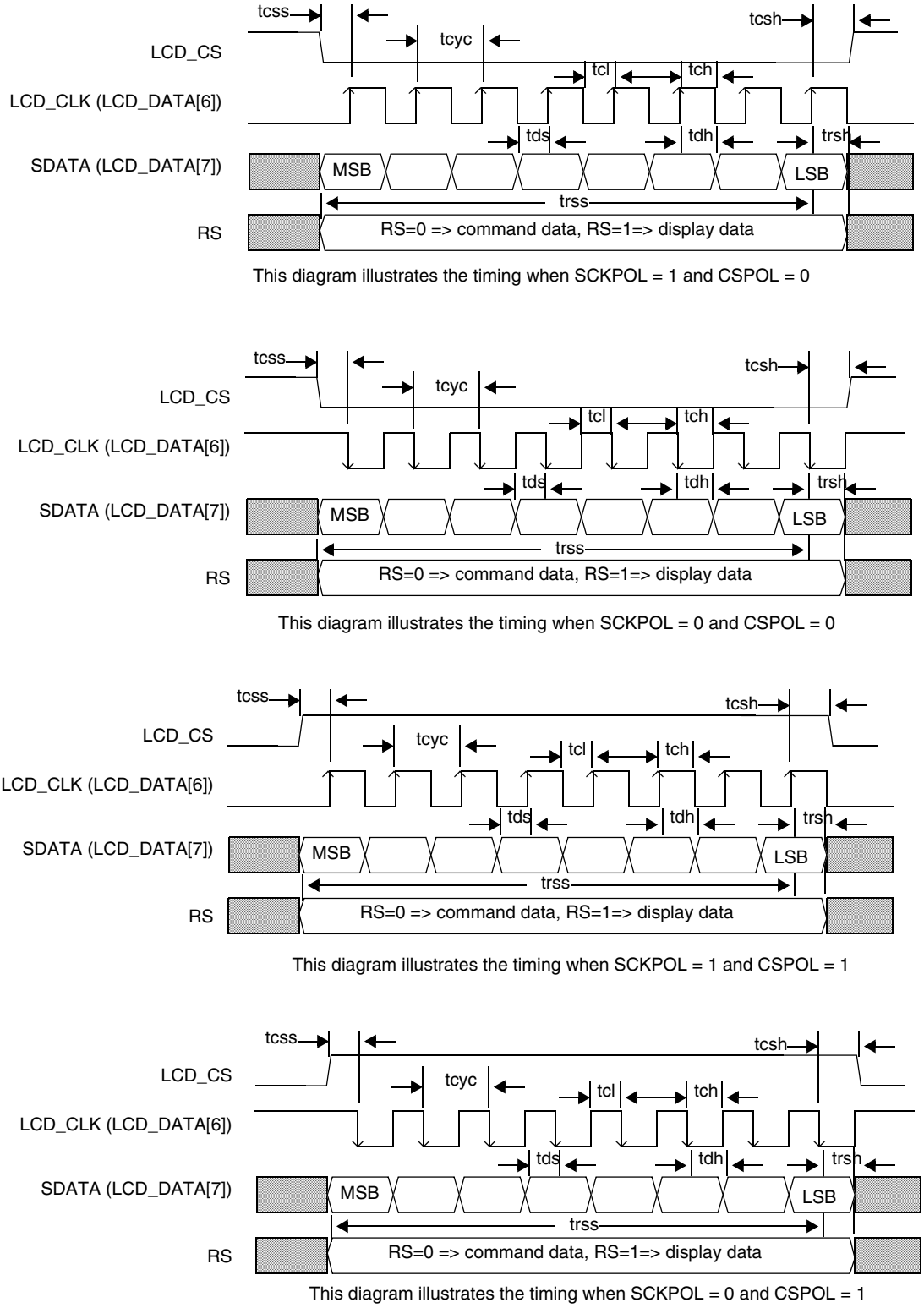


Figure 41-23. SLCDC Serial Transfers to LCD Device

Table 41-17. SLCDC Serial Interface Timing

Symbol	Parameter	Minimum (ns)	Typical (ns)	Maximum (ns)
t_{css}	Chip select setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{csh}	Chip select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{cyc}	Serial clock cycle time	39 (±) t_{prop}	—	2641
t_{cl}	Serial clock low pulse	18 (±) t_{prop}	—	—
t_{ch}	Serial clock high pulse	18 (±) t_{prop}	—	—
t_{ds}	Data setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{dh}	Data hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rss}	Register select setup time	(15 * t_{cyc} / 2) (±) t_{prop}	—	—
t_{rsh}	Register select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—

Note: t_{prop} is the propagation delay from SLCDC outputs to the pin outputs.

41.3.2 Parallel Interface

SLCDC module can be configured to execute parallel transfers of display data from the system memory to an external display device. Figure 41-24 shows the timing associated with an SLCDC parallel transfer to an external LCD device. The timing is based on the frequency of LCD_CLK, which is programmable using LCD Clock Configuration Register (Section 41.2.18, “LCD Clock Configuration Register (LCDCLOCKCONFIG)”).

In parallel mode, LCD_CS pin is used as a write strobe by the external LCD controller. LCD_CS latching edge occurs one LCD_CLK cycle after data is made available to the display on the LCD_DATA[15:0] pins. LCD_CS signal polarity is programmable using the CSPOL bit in the LCD transfer configuration register. Data transfers of 16- or 8-bit are determined by the word definition bits in the LCD transfer configuration register (Section 41.2.16, “LCD Transfer Configuration Register (LCDTRANSCONFIG)”). All 8-bit data transfer uses LCD_DATA[7:0]. LCD_DATA[15:8] shows 0’s. It is possible for there to be a 8-bit command data transfers and 16-bit display data transfers (and vice versa). In this case, when command data is transferred, only LCD_DATA[7:0] is used, and display data uses LCD_DATA[15:0].

Table 41-18. SLCDC Parallel Interface Timing

Symbol	Parameter	Minimum (ns)	Typical (ns)	Maximum (ns)
t_{cyc}	Parallel clock cycle time	78 (±) t_{prop}	—	4923
t_{ds}	Data setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{dh}	Data hold time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rss}	Register select setup time	(t_{cyc} / 2) (±) t_{prop}	—	—
t_{rsh}	Register select hold time	(t_{cyc} / 2) (±) t_{prop}	—	—

Note: t_{prop} is the propagation delay from the SLCDC outputs to the pin outputs.

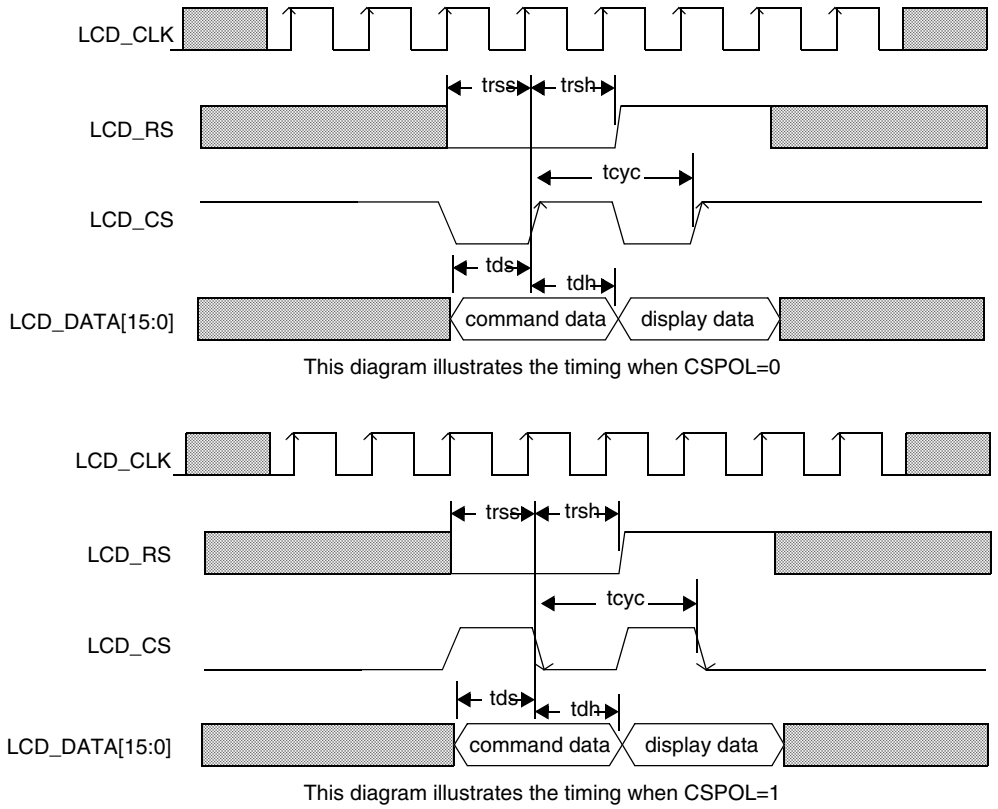


Figure 41-24. SLCDC Parallel Transfers to LCD Device

41.4 LCD Clock Configuration

SLCDC uses a fractional divider to generate the LCD data clock from HCLK_SLCDC clock signal. The fractional divider is programmed by setting a divide value using DIVIDE[5:0] bits of the LCD clock Configuration Register (Section 41.2.18, “LCD Clock Configuration Register (LCDCLOCKCONFIG)”). LCD_CLK frequency is calculated using Equation 41-1:

Eqn. 41-1

$$LCD\_CLK = \frac{HCLKx(DivideValue)}{128}$$

Divide value is taken directly from DIVIDE[5:0] bits of the LCD clock Configuration Register (Section 41.2.18, “LCD Clock Configuration Register (LCDCLOCKCONFIG)”). Setting DIVIDE[5:0] to 0, disables the LCD_CLK signal. These bits must be set to some non-zero value before LCD transfers can occur.

Table 41-19. LCD_CLK Frequency Range

HCLK_SLCDC Clock Frequency (MHz)	Minimum LCD_CLK Frequency (MHz)	Maximum LCD_CLK Frequency (MHz)	Resolution of Step (kHz)
52	0	25.59	406.25
26	0	12.8	203.13
16.8	0	8.27	131.25

41.5 R-AHB Interface and SLCDC FIFOs

SLCDC uses a DMA interface to access system memory without CPU intervention. When SLCDC needs data, DMA requests AHB control and reads 32-bit words from the system memory. The number of 32-bit words read during the burst is determined by the value stored in the BURST[2:0] bits of the FIFO Configuration Register (Section 41.2.14, “FIFO Configuration Register (FIFOCONFIG)”). AHB control is relinquished by DMA after the burst is complete. SLCDC generates idle cycles when data transfer is complete. The amount of R-AHB bandwidth required by SLCDC can be controlled by the system software. The frequency of LCD frame updates and the rate of LCD_CLK affect how often the SLCDC requests the use of R-AHB.

Data read by the DMA interface is stored in two 32-bit × 8 word FIFOs that are used to buffer data between the DMA and the SLCDC display interface. One FIFO is used to buffer data from the command buffer and the other is used to buffer data from the display data buffer. When one FIFO has enough room to store the data read during a DMA burst, it will request servicing. Because DMA burst length is programmable, the occupancy level of the FIFOs can vary when servicing is requested.

Chapter 42
Shared Peripheral Bus Arbiter (SPBA)

This chapter describes the shared peripheral bus arbiter module. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

42.1 Introduction

The shared peripheral bus arbiter (SPBA) is a three-to-one IP line interface (IP-Bus) arbiter, with a resource locking mechanism. Three masters arbitrate for access to up to 31 shared peripherals through the SPBA.

Figure 42-1 shows the SPBA block diagram and details of the out-of-band signals routing.

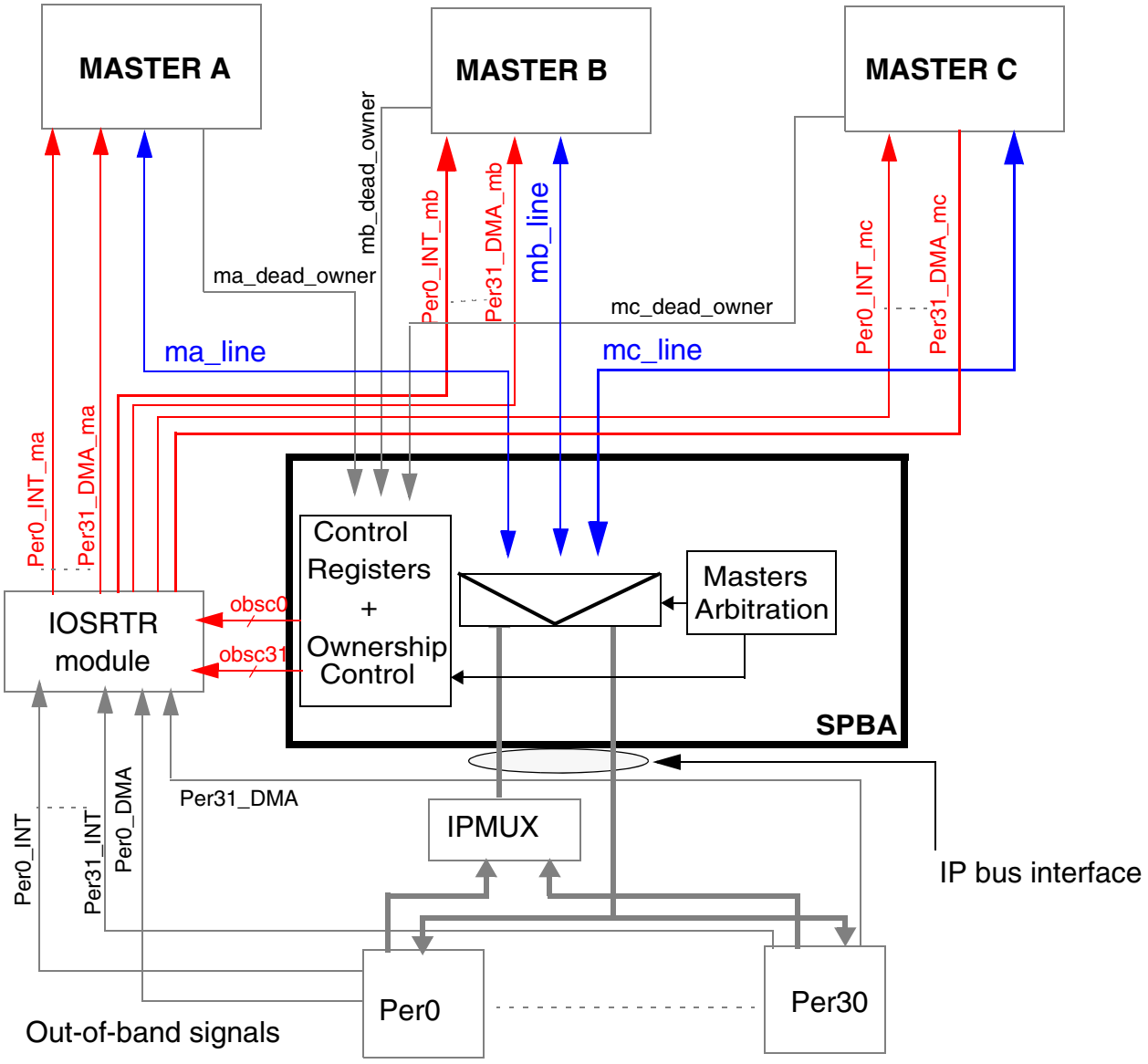


Figure 42-1. SPBA Block Diagram

42.2 Overview

As shown in Figure 42-1, the SPBA includes the following components:

- The control registers and ownership control, which includes a set of programmable registers that define resource ownership and access control for each peripheral
- The IP bus line, which switches each master to a single peripheral
- The masters arbiter, which arbitrates between the three masters to resolve concurrent access or restricted access to peripherals.

The SPBA also generates signals (obscn in Figure 42-1) which reflect control register values, and are used by the external steering logic (in-band out-band steering logic module, denoted IOSRTR in Figure 42-1) to route peripherals' interrupts and DMA signals to the appropriate master

42.3 Features

The SPBA includes the following features:

- Arbitrates between three IP bus masters (A, B and C)
- Supports DMA masters
- 32-bit width read and write data signals between master and peripheral
- Supports up to 31 shared peripherals, each occupying 16 Kbytes of address space,
- Treats the SPBA itself as the 32nd peripheral, used for resource ownership and access control mechanism to the 31 peripherals
- Provides 31 sets of out-of-band steering control (obsc*n*) signals to the off-module steering logic
- Operating frequency up to 67 MHz
- Uses clocks ipg_clk and ipg_clk_s

42.4 Modes of Operation

The SPBA acts transparently when accessing a peripheral. The following modes of operation can be distinguished:

- Reset/abort mode

  The SPBA has a hardware reset which initialize all registers, arbitration and peripherals rights registers (PRR). Additionally, an abort signal input is provided allowing each master to abort its current access and release ownership (used for instance in case of master reset sequence).
- Functional mode

  After a master request is granted, its IP bus signals are steered to the requested peripheral.
- Standby mode

  No clocks are needed in this mode. The SPBA requires clocks only during configuration, arbitration, and abort. The SPBA generates two clock enable signals which indicate when the clocks must be provided.
- Configuration mode

  This is the active mode when accessing the SPBA peripheral rights registers. The SPBA memory-mapped registers are treated as a shared peripheral.

42.4.1 Detailed Signal Descriptions

42.4.1.1 Out-of-Band Steering Control Signals (obsc*n*[4:0])

The SPBA provides 31 sets of out-of-band steering control signals to the in-band out-band steering logic module (IOSTR). These signals reflect the current ownership of the peripheral and the masters that have

access to the peripheral, which is information stored in the ROI and RAR fields of the peripheral rights register (PRR). Table 42-1 shows how ROI*n* and RAR*n* bits are available in the obsc*n*[4:0] bus.

Table 42-1. Out-Of Band Steering Control Signals obsc*n*[4:0]

obsc*n*[4]	obsc*n*[3]	obsc*n*[2]	obsc*n*[1]	obsc*n*[0]
ROI*n*[1]	ROI*n*[0]	RAR*n*[2]	RAR*n*[1]	RAR*n*[0]

These signals are handled at SoC level in the IOSTR in order to drive the interrupt back to the appropriate master(s) indicated by ROI*n* and RAR*n*.

42.5 Memory Map and Register Definition

The following section describes the memory maps and detailed descriptions of all registers which are accessible to the end user within and through SPBA.

42.5.1 Peripherals Memory Map

From the masters' side, the absolute address of a memory-mapped peripheral register is accessible by setting m*x*_ips_addr[24:0] as follows (*x* = A, B, or C):

- **m*x*_ips_addr[24:19]** is set to the SPBA master base address (for the SPBA master base address, see the system memory map).
- **m*x*_ips_addr[18:14]** is set to the start address of the appropriate peripheral, as shown in Table 42-2.
- **m*x*_ips_addr[13:0]** is set to the offset of the memory-mapped register from the peripheral's base address. Each peripheral is allocated 16 Kbytes for registers (m*x*_ips_addr[13:0] ranges from 0b00_0000_0000_0000 to 0b11_1111_1111). However, if the peripheral is the SPBA itself, then if m*x*_ips_addr[13:0] is greater than the last PRR location the SPBA returns a transfer error to the master.

Table 42-2. m*x*_ips_addr[18:14} Settings Corresponding to Each Peripheral

Peripheral	m*x*_ips_addr[18:14] Setting
Peripheral 0	0b0_0000
Peripheral 1	0b0_0001,
Peripheral 2	0b0_0010
Peripheral 3	0b0_0011
Peripheral 4	0b0_0100
Peripheral 5	0b0_0101
Peripheral 6	0b0_0110
Peripheral 7	0b0_0111

Table 42-2. m*x*_ips_addr[18:14} Settings Corresponding to Each Peripheral (continued)

Peripheral	m*x*_ips_addr[18:14] Setting
Peripheral 8	0b0_1000
Peripheral 9	0b0_1001
Peripheral 10	0b0_1010
Peripheral 11	0b0_1011
Peripheral 12	0b0_1100
Peripheral 13	0b0_1101
Peripheral 14	0b0_1110
Peripheral 15	0b0_1111
Peripheral 16	0b1_0000
Peripheral 17	0b1_0001
Peripheral 18	0b1_0010
Peripheral 19	0b1_0011
Peripheral 20	0b1_0100
Peripheral 21	0b1_0101
Peripheral 22	0b1_0110
Peripheral 23	0b1_0111
Peripheral 24	0b1_1000
Peripheral 25	0b1_1001
Peripheral 26	0b1_1010
Peripheral 27	0b1_1011
Peripheral 28	0b1_1100
Peripheral 29	0b1_1101
Peripheral 30	0b1_1110
Peripheral 31	0b1_1111

42.5.2 SPBA Registers Memory Map

Table 42-3 is the SPBA module memory map.The base address for the SPBA registers is the SPBA master base address (see the system memory map) plus 0b0111_1100_0000_0000_0000 (due to the fact that SPBA registers are treated as peripheral 31: see Table 42-2).

Table 42-3. SPBA Registers Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (PRR0) … 0x007C (PRR31)	Peripheral Rights Registers (PRR*n*, *n*=0…31)	R/W	0x0000_0007	42.5.4.1/42-7

42.5.3 SPBA Register Summary

The SPBA control registers (also denoted as peripheral rights registers) are mapped as a virtual shared peripheral.

SPBA can support up to 31 shared peripherals, each with its own peripheral right register (PRR) accessible within the SPBA memory mapped registers. The PRRs contain fields which specify requesting master owner (RMO), resource owner ID (ROI) and resource access rights (RAR).

Table 42-4. SPBA PRR Register Summary

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*n**0x0004 PRR*n* (*n*=0…31)	R	RMOn		0	0	0	0	0	0	0	0	0	0	0	0	ROIn	
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	RARn		
	W																

42.5.4 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 42-2 and Table 42-5 explain conventions used in register diagrams and tables.

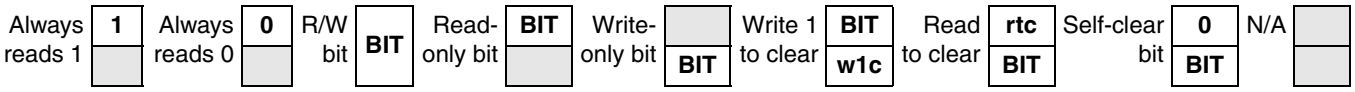


Figure 42-2. Register Field Conventions

Table 42-5. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).

Table 42-5. General Register Conventions (continued)

Convention	Description
rwm	A read/write bit that can be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

42.5.4.1 Peripheral Rights Register (PRR*n*)

Offset 0x0000 (PRR0)
0x0004 (PRR1)
0x0008 (PRR2)
…
0x007C (PRR31)

Access: Mixed

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RMO*n*		0	0	0	0	0	0	0	0	0	0	0	0	ROI*n*	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	RAR*n*		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 42-3. Peripheral Rights Register (PRR*n*)

Table 42-6. Peripheral Rights Register (PRR*n*) Field Descriptions

Field	Description
31—30 RMO*n*[1:0]	Requesting master owner. These read-only bits indicate whether the corresponding resource is owned by the requesting master. This register is reset to 0b00 if ROI*n* = 0b00. 00 Unowned resource 01 Reserved 10 Resource owned by another master 11 Resource owned by requesting master
29–18	Reserved.
17–16 ROI*n*[1:0]	Resource owner ID. These read-only bits indicate which master has access to the PRR for rights modification. No master with ID different from the value stored in ROI) is able to modify the RAR field. When a master performs a write access to an unowned PRR*n*, the ROI and RAR fields of the PRR*n* are automatically updated to reflect the master's ID and access rights.The master can then read back the RMO, RAR, ROI fields of PRR*n* to ensure their values. To release ownership of a peripheral, a master can assert its dead_owner signal, or write 0b000 in the RAR (which causes ROI[1:0] to reset to 0b00) ROI bits are cleared on reset. 00 Unowned resource 01 Resource owned by master A port 10 Resource owned by master B port 11 Resource owned by master C port
15–3	Reserved.
2–0 RAR*n*[2:0]	Resource access rights. This 3-bit field indicates which master can access the peripheral. Up to 3 masters can have permission to access a resource, but only one access at a time is granted by SPBA. RAR*n*[2] — Control and status bit for master C: 0 Master C access to peripheral is not allowed 1 Master C access to peripheral is granted RAR*n*[1] — Control and status bit for master B" 0 Master B access to peripheral is not allowed 1 Master B access to peripheral is granted RAR*n*[0] — Control and status bit for master A: 0 Master A access to peripheral is not allowed 1 Master A access to peripheral is granted **Note:** After reset all RAR*n* fields are set to 0b111, so that all masters have access to all shared peripherals. This is verified until one master modify this default value.

42.6 Functional Description

The following section provides a functional description of the SPBA module.

42.6.1 Masters Arbitration

The arbitration mechanism determines which port controls the master port based on a simple round-robin arbitration scheme. Different arbitration scenarios are described as follows:

- A single master requests access. In this case, the master is switched to the shared peripheral bus without arbitration. Figure 42-4 shows a master B request on the global module enable signal, served without wait state.

- If two masters simultaneously access the SPBA, then the most recently granted master is deferred using the m*x*_ips_xfr_wait output signal (default value is high). When the master is granted, sips_xfr_wait from the shared IPS peripheral is connected to m*x*_ips_xfr_wait outputs.
- If three masters simultaneously access to SPBA, then the two most recently granted masters are deferred, using their m*x*_ips_xfr_wait signals. Figure 42-5 shows the case when the last two accesses granted are masters A and B, and how the requests are used even if they are in the same cycle.
- After reset, at the first multiple access the priority is static: masters A, B, and C have highest to lowest priority, respectively.
- No master request. No more master switch to shared peripherals.

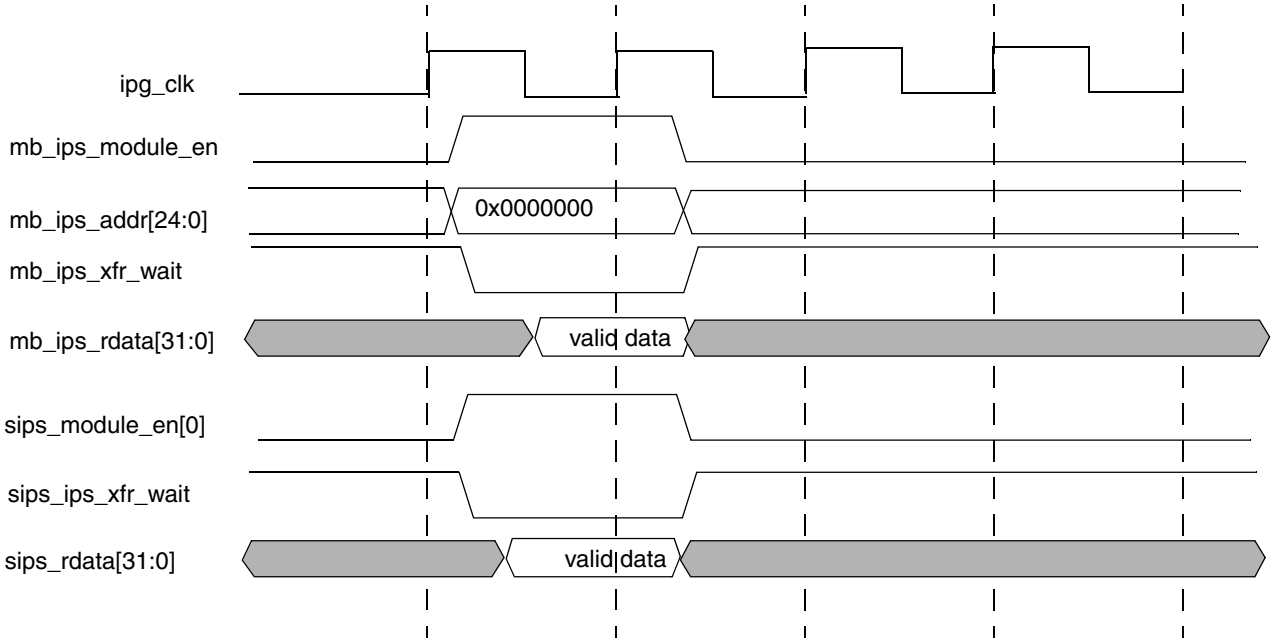


Figure 42-4. Example of One Master Request: No SPBA Arbitration;

Figure 42-5 assumes masters A and B respectively were granted the previous two transfers.

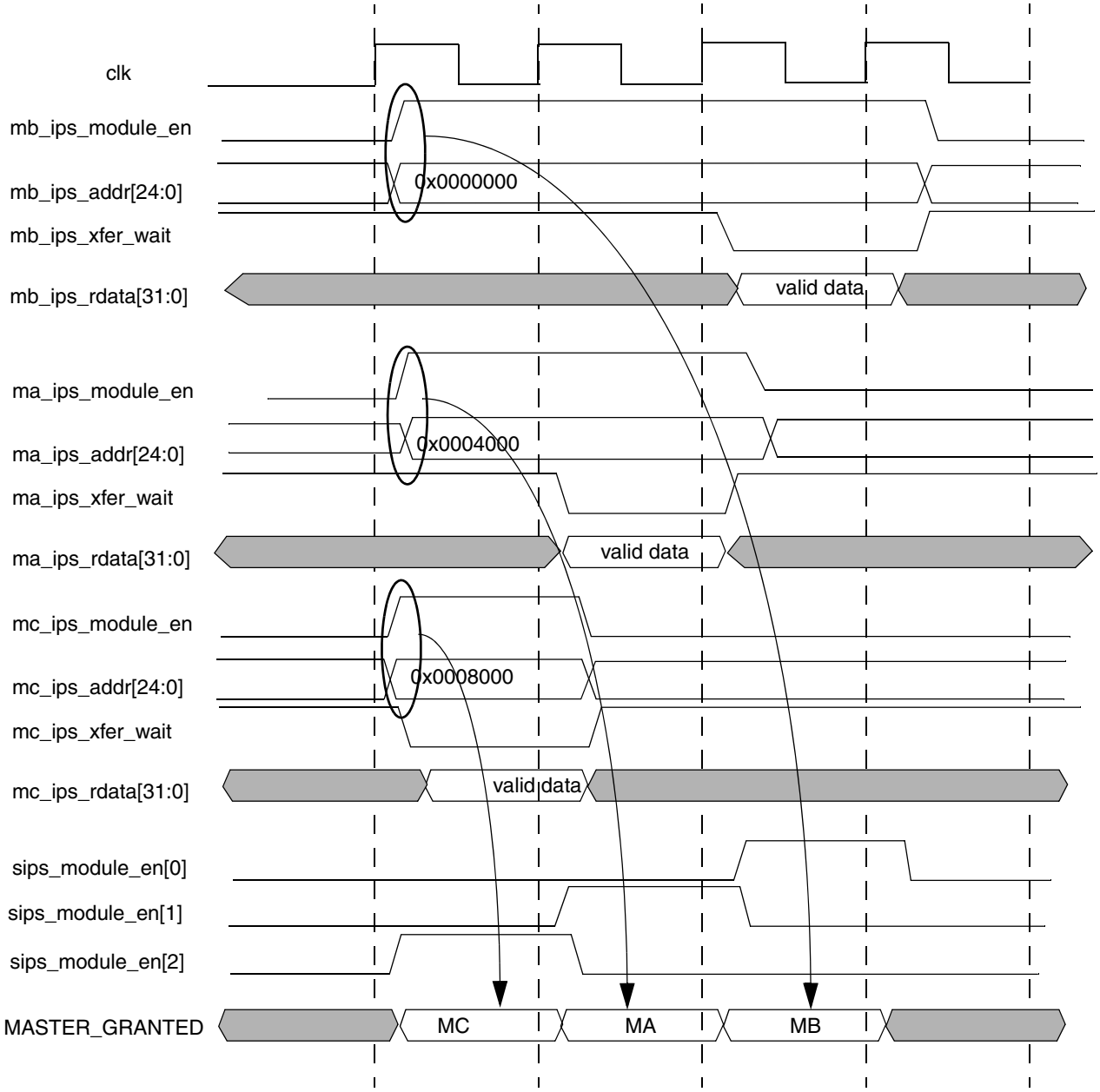


Figure 42-5. Example of Three Master Requests, Where Already-Granted Masters are "Waited";

42.6.2 Resource Ownership Control

The resource ownership control controls masters' accesses to the shared peripherals and determines steering of out-of-band signals.

42.6.3 Access Control

42.6.3.1 Peripheral Access

The peripheral access (also called resource access) of the requesting master is given by the corresponding RAR bit of the peripheral right register. Any attempted access to a resource by a requesting master *x* whose access privilege bit is not set in the RAR field is terminated with a bus error (m*x*_ips_xfr_err is asserted by the SPBA logic). The master that owns the resource can lock the peripheral for itself or grant other masters access to the peripheral by setting the appropriate bit(s) in the RAR field.

Figure 42-6 shows an example where master B takes ownership of peripheral 2 by writing 0b010 in the RAR field of the PRR2 register.Ownership can be verified by reading the corresponding ROI and RAR fields from the obsc2 output, since obsc2[4:0] = {ROI[1] ROI[0] RAR[2] RAR[1] RAR[0]}.

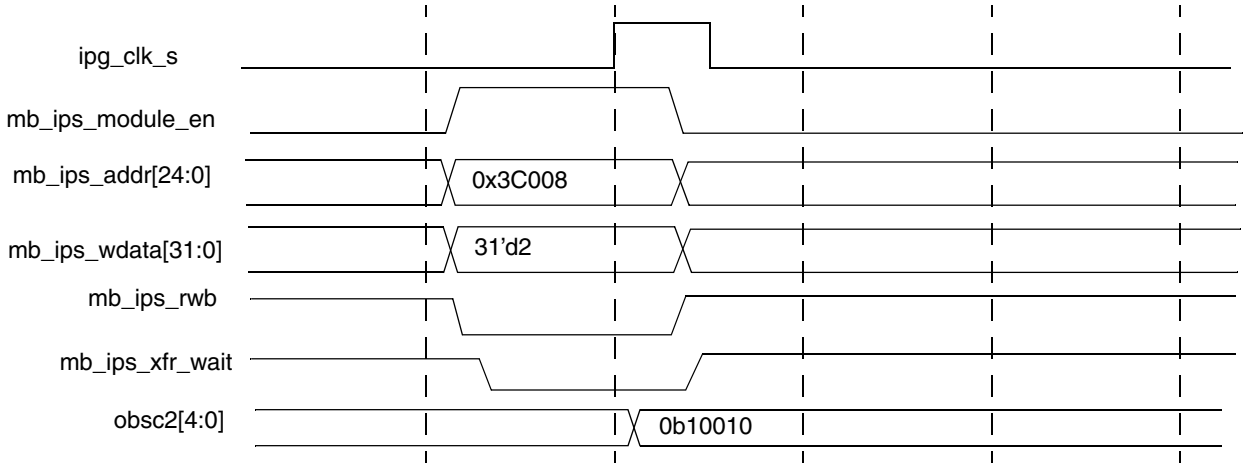


Figure 42-6. Example of Master B Gaining Ownership of Peripheral 2

42.6.3.2 Peripheral Right Register Access

The ROI bits of a peripheral right register (PRR*n*) determine which master is allowed write access to PRR*n*. The requesting master's ID is compared to the ROI bits of the PRR*n* to determine if the master has ownership of the corresponding register. Any attempted write access to a PRR*n* already owned by another master is ignored.

42.6.4 Owner Election

When the peripheral is unowned by any master (that is ROI= 0b00, which for example occurs when coming out of reset), the first master to successfully perform a write to the RAR bits of the PRR is granted ownership of the peripheral and its associated PRR.

After writing to the RAR field of the PRR, the master must read it back to make sure that it was actually granted ownership.

If the RMO field is 0b11, then the ownership claim is successful. If RMO is 0b10, then another master claimed ownership before this master was able to complete its write.

This resolves the case in which two or more masters attempt to write the PRR at the same time; only the first master is granted ownership. However all masters must read the PRR to determine if this case occurred, and if so, whether they were the first master which was actually granted ownership. See Section 42.5.3, "SPBA Register Summary," for more detail on the fields of the PRR.

NOTE

A master that has been granted ownership of PRR*n* does not automatically have access rights to peripheral *n*. The master must still set the corresponding RAR bit in PRR*n* to access the peripheral.

42.6.5 Termination of Ownership

A master that owns peripheral *n* can have its ownership ended in two different ways:

- By software, by writing 0b000 to RAR*n*. This causes the ROI*n* bits to clear automatically.
- By hardware, when a master is reset. The reset causes assertion of the m*x*_dead_owner signal, which in turn causes the ROI*n* bits to be cleared. All peripherals previously owned by the dead master are changed to the unowned state.

NOTE

It is the programmer's responsibility to make sure the peripherals are placed in an appropriate state before ending ownership.

42.6.6 The Unowned State

During the time when the peripheral is unowned (that is, the ROI field is 0b00), all masters have full access, and can modify the RAR bits. In such cases it is necessary for software to ensure any necessary coherency in the resource—there is no hardware protection.

Chapter 43
Synchronous Serial Interface (SSI)

This block guide presents the synchronous serial interface (SSI), and discusses the architecture, the programming model, the operating modes, and initialization of SSI.

See Figure 43-1 for illustration of the block diagram for the SSI. It consists of control registers to set up the port, status register, separate transmit and receive circuits with FIFO registers, and separate serial clock and frame sync generation for the transmit and receive sections. The second set of Tx and Rx FIFOs, replicates the logic used for the first set of FIFOs.

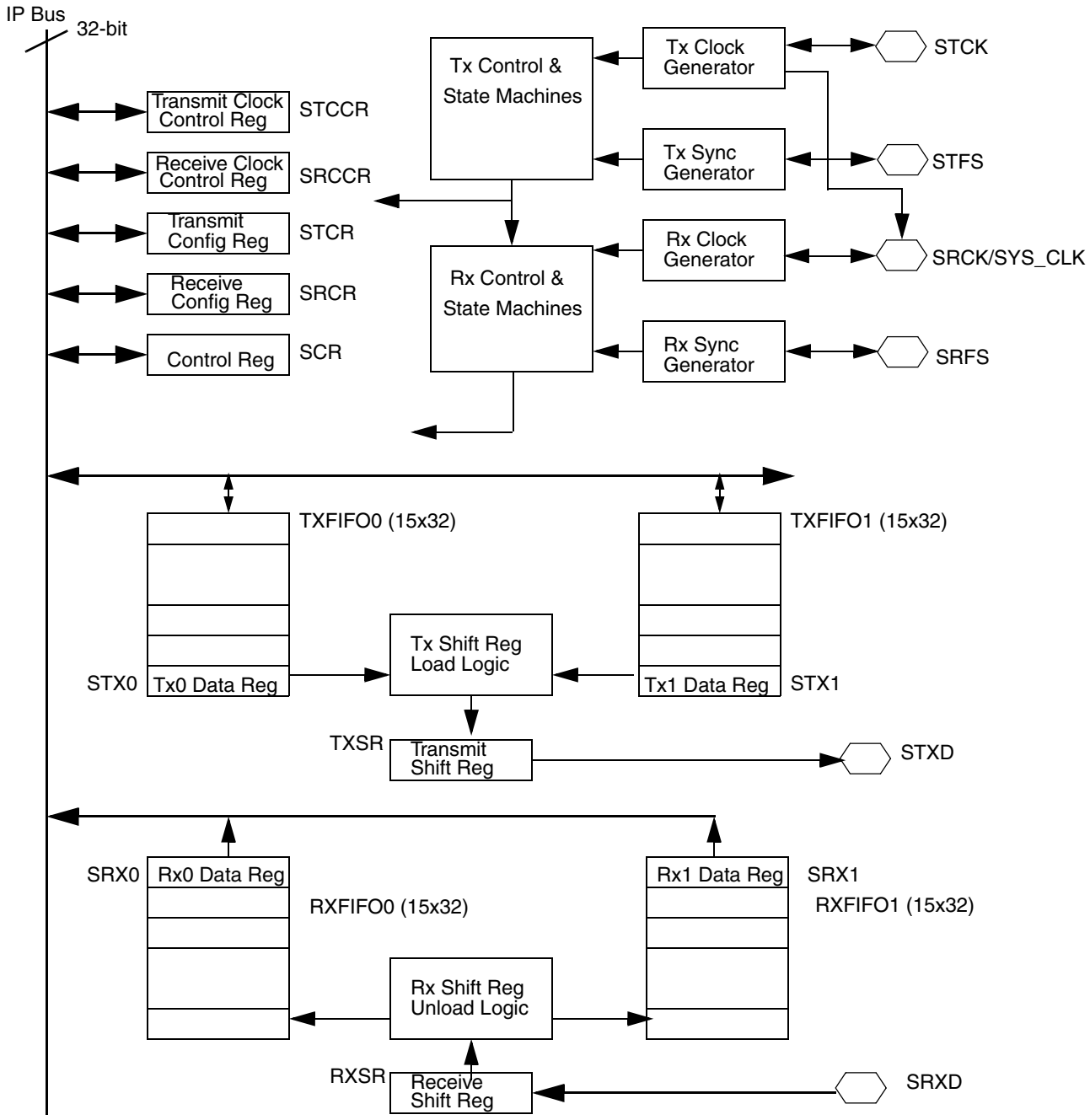


Figure 43-1. SSI Block Diagram

43.1 Overview

The SSI is a full-duplex, serial port that allows the chip to communicate with a variety of serial devices. These serial devices can be standard CODer-DECoder (CODECs), Digital Signal Processors (DSPs), microprocessors, peripherals, and popular industry audio CODECs that implement the inter-IC sound bus standard (I^2S) and Intel AC97 standard.

SSI is typically used to transfer samples in a periodic manner. The SSI consists of independent transmitter and receiver sections with independent clock generation and frame synchronization.

43.1.1 Features

The SSI includes the following features:

- Independent (asynchronous) or shared (synchronous) transmit and receive sections with separate or shared internal/external clocks and frame syncs, operating in Master or Slave mode.
- Normal mode operation using frame sync
- Network mode operation allowing multiple devices to share the port with as many as thirty-two time slots
- Gated Clock mode operation requiring no frame sync
- 2 sets of Transmit and Receive FIFOs. Each of the four FIFOs is 15x32 bits. The two sets of Tx/Rx FIFOs can be used in Network mode to provide 2 independent channels for transmission and reception
- Programmable data interface modes such like I2S, LSB, MSB aligned
- Programmable word length (8, 10, 12, 16, 18, 20, 22 or 24 bits)
- Program options for frame sync and clock generation
- Programmable I2S modes (Master, Slave or Normal). Oversampling clock, ccm_ssi_clk available as output from SRCK in I2S Master mode
- AC97 support
- Completely separate clock and frame sync selections for the receive and transmit sections. In AC97 standard, the clock is taken from an external source and frame sync is generated internally.
- External ccm_ssi_clk input for use in I2S Master mode. Programmable oversampling clock (ccm_ssi_clk) of the sampling frequency available as output in master mode at SRCK, when operated in sync mode.
- Programmable internal clock divider
- Time Slot Mask Registers for reduced CPU overhead (for Tx and Rx both)
- SSI power-down feature
- Programmable wait states for CPU accesses
- IP Interface for register accesses, compatible with SRS 3.0.2 standard

43.1.2 Modes of Operation

SSI has the following basic operating modes.

- Normal mode
  - — Asynchronous protocol
  - — Synchronous protocol
- Network mode
  - — Asynchronous protocol

  - Synchronous protocol
- Gated Clock mode
  - Synchronous protocol only

These modes can be programmed by several bits in the SSI control registers. See Table 43-1 for the list of SSI operating modes and some of the typical applications in which they can be used:

Table 43-1. SSI Operating Modes

TX, RX Sections	Serial Clock	Mode	Typical Application
Asynchronous	Continuous	Normal	Multiple synchronous CODECs
Asynchronous	Continuous	Network	TDM CODEC or DSP networks
Synchronous	Continuous	Normal	Multiple synchronous CODECs
Synchronous	Continuous	Network	TDM CODEC or DSP network
Synchronous	Gated	Normal	SPI-type devices; DSP to MCU

The transmit and receive sections of the SSI can be synchronous or asynchronous. In Synchronous mode, the transmitter and the receiver use a common clock and frame synchronization signal. Masking of slots for Transmit and Receive section can differ in synchronous mode. Also the shifting of data in independent and for receive section depends on RXBIT0 and RSHFD bits in SRCR register. In Asynchronous mode, the transmitter and receiver each has its own clock and frame synchronization signals.

Normal or Network mode can be selected. In Normal mode, the SSI functions with one data word of I/O per frame. In Network mode, any number from two to thirty-two data words of I/O per frame can be used. Network mode is typically used in star time division multiplex networks with other processors or CODECs, allowing interface to time division multiplexed networks without additional logic. Gated clock mode option can be selected in Normal synchronous mode only. During Gated-mode the clock is not-continuous and runs only during data-transmission. These distinctions result in the basic operating modes that allow the SSI to communicate with a wide variety of devices.

The SSI supports both Normal and Network modes, and these can be selected independently of whether the transmitter and receiver are synchronous or asynchronous. Typically these protocols are used in a periodic manner, where data is transferred at regular intervals, such as at the sampling rate of an external CODEC. Both modes use the concept of a frame. The beginning of the frame is marked with a frame sync when programmed with continuous clock. The frame sync occurs at a periodic interval. The length of the frame is determined by the DC[4:0] bits in either the SRCCR or STCCR register, depending on whether data is being transmitted or received. The number of words transferred per frame depends on the mode of the SSI.

In Normal mode, one data word is transferred per frame. In Network mode, the frame is divided into anywhere between two and thirty-two time slots, where in each time slot one data word can optionally be transferred.

Apart from the above basic modes of operation, SSI supports the following modes which require some specific programming.

- I2S mode

- AC97 mode
  - AC97 Fixed mode
  - AC97 Variable mode

In (non-I2S) slave modes (external frame sync), the programmed word length setting of the SSI should be equal to the word length setting of the master. In I2S slave mode, the programmed word length setting of the SSI can be lesser than or equal to the word length setting of the I2S master (external CODEC).

In slave modes, the programmed frame length setting (DC bits) of the SSI can be lesser than or equal to the frame length setting of the master (external CODEC).

The following sections provide detailed descriptions of the above modes.

43.1.2.1 Normal Mode

Normal mode is the simplest mode of the SSI. It is used to transfer data in one time slot per frame. A time slot is a unit of data and the WL[3:0] bits define the number of bits in a time slot. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. The length of the frame is determined by the following factors:

- The period of the Serial Bit Clock (DIV2, PSR, PM[7:0] bits for internal clock or the frequency of the external clock on the STCK port)
- The number of bits per time slot (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

If Normal mode is configured with more than one time slot per frame, data is transferred only in the first time slot. No data is transferred in subsequent time slots. In Normal mode, DC[4:0] values corresponding to more than a single time slot in a frame, only results in lengthening of the frame.

43.1.2.1.1 Normal Mode Transmit

The conditions for data transmission from the SSI in Normal mode are:

- SSI Enabled (SSIEN = 1)
- Write data to Transmit Data Register (STX)
- Transmitter Enabled (TE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

When the above conditions occur in Normal mode, the next data word is transferred into the Transmit Shift Register (TXSR) from the Transmit Data Register 0 (STX0), or from the Transmit FIFO 0 Register, if transmit FIFO 0 is enabled. The new data word is transmitted on arrival of frame-sync preceded by clocks in continuous clock mode. In gated external mode, data word in transmitted on arrival of frame-sync. In gated-internal mode, data word is transmitted whenever data is available in Tx-FIFO.

If Transmit FIFO 0 is not enabled and the transmit data register empty enable (TDE0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the word in SSI_STX0 is shifted to Transmit Shift (TXSR) register for shifting.

If Transmit FIFO 0 is enabled and the transmit fifo full enable (TFF0_EN) and transmit interrupt enable (TIE) bits are set, transmit interrupt occurs when the number of empty slots in Transmit Fifo 0 exceed or are equal to the selected threshold value i.e. Transmit Fifo 0 Watermark (TFWM0) value. If transmit FIFO 0 is enabled and filled with data, 15 data words can be transferred before the core must write new data to the STX0 register.

The STXD port is disabled except during the data transmission period. For a continuous clock, the optional frame sync output and clock outputs are not disabled, even if both receiver and transmitter are disabled.

43.1.2.1.2 Normal Mode Receive

The conditions for data reception from the SSI are:

- SSI enabled (SSIEN = 1)
- Receiver enabled (RE = 1)
- Frame sync active (for continuous clock case)
- Bit clock begins

With the above conditions in Normal mode with a continuous clock, each time the frame sync signal is generated (or detected) a data word is clocked in. With the above conditions and a gated clock, each time the clock begins, a data word is clocked in.

If Receive FIFO 0 is not enabled and Receive Interrupt enable (RIE) and Received Data 0 Ready enable (RDR0_EN) bits are set, receive interrupt occurs when received data word is transferred from the Receive Shift Register (RXSR) to the Receive Data Register 0 (SRX0) thus setting the Receive Data Ready 0 (RDR0) flag.

If Receive FIFO 0 is enabled, and Receive Interrupt enable (RIE) and Received Fifo 0 full enable (RDR0_EN) bits are set, receive interrupt occurs when the received data word is transferred to the Receive FIFO 0 and Receive FIFO 0 reaches the selected threshold and results in Receive FIFO Full 0 (RFF0) flag to get set.

The core program has to read the data from the Receive Data Register 0 (SRX0) (in case Receive FIFO0 is disabled) before a new data word is transferred from the Receive Shift Register (RXSR), otherwise the Receive Overrun Error 0 (ROE0) bit is set. If receive FIFO 0 is enabled, the Receive Overrun Error 0 (ROE0) bit is set when the Receive FIFO 0 data level reaches the selected threshold and a new data word is ready to be transferred to the Receive FIFO 0.

See Figure 43-2 for illustration of transmitter and receiver timing for an 8-bit word in the first time slot in Normal mode, continuous clock with a late word length frame sync. The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register.

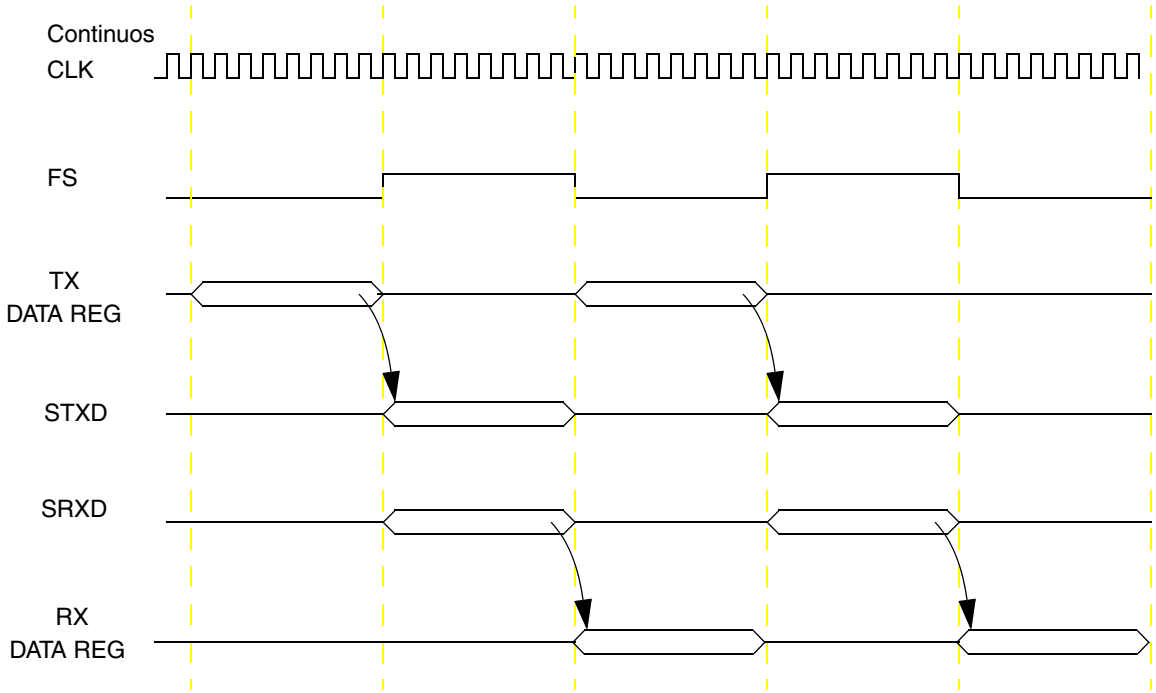


Figure 43-2. Normal Mode Timing - Continuous Clock

Figure 43-3 shows a similar case for internal (SSI generates clock) gated clock mode and Figure 43-4 shows a case for external (SSI receives clock) gated clock mode.

NOTE

A pull-down resistor is required in the gated clock case because the clock port is disabled between transmissions.

The Tx Data register is loaded with the data to be transmitted. On arrival of the clock, this data is transferred to the Transmit Shift Register which gets transmitted on arrival of the frame-sync on the STXD output. Simultaneously, the Receive Shift Register shifts in the received data available on the SRXD input and at the end of the time slot, this data is transferred to the Rx Data Register. In case of Internal Gated clock mode, the Tx Data line and clock output port are put in the high-impedance state at the end of transmission of the last bit (at the completion of the complete clock cycle), whereas, in External Gated clock mode, the Tx Data line is tri-stated at the last inactive edge of the incoming bit clock (during the last bit in a data word).

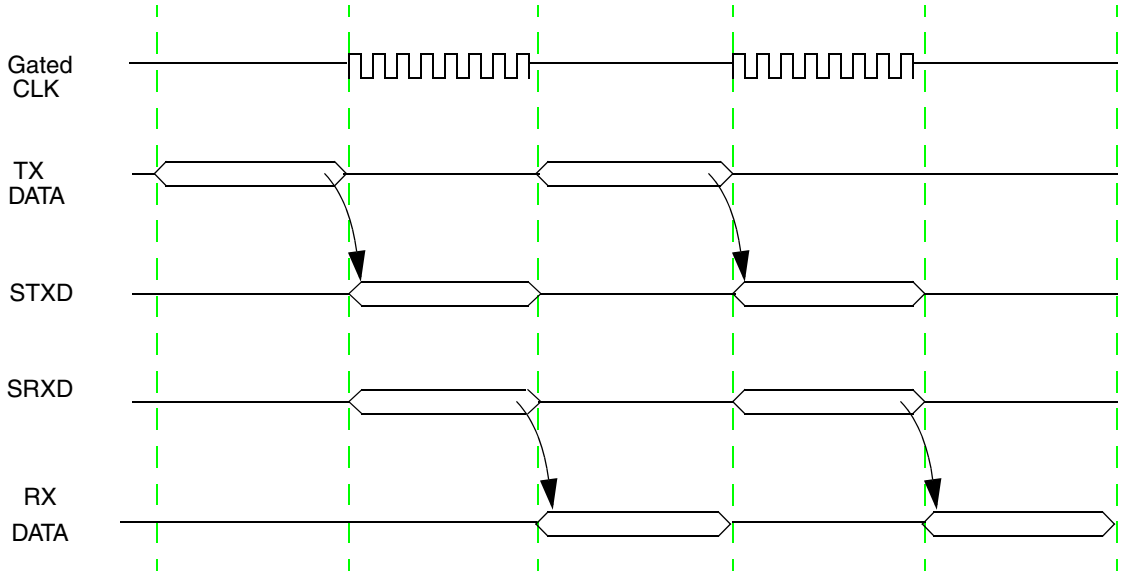


Figure 43-3. Normal Mode Timing—Internal Gated Clock

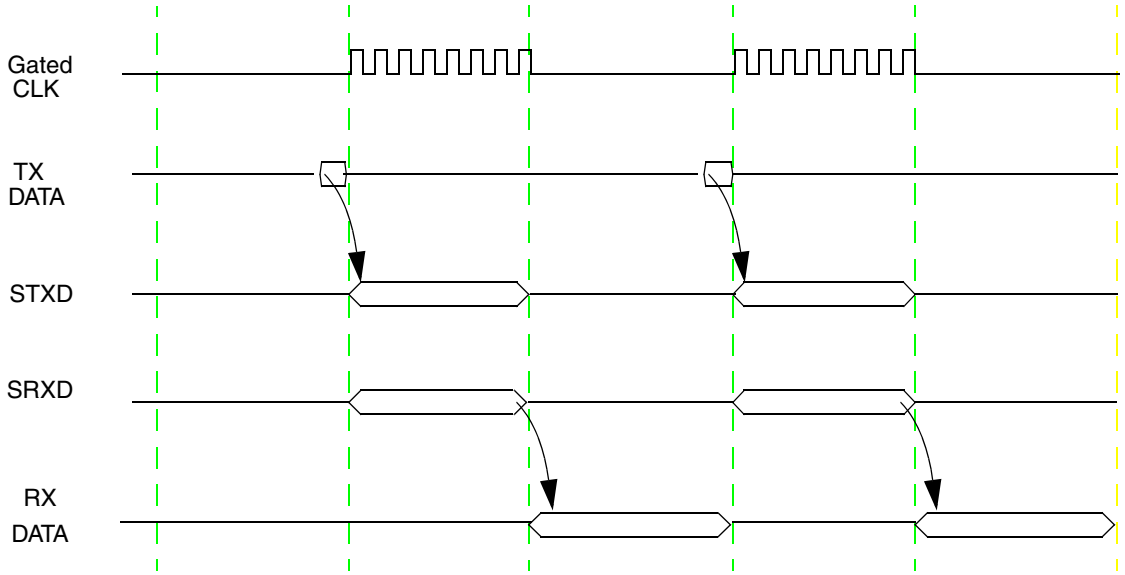


Figure 43-4. Normal Mode Timing—External Gated Clock

43.1.2.2 Network Mode

Network mode is used for creating a Time Division Multiplexed (TDM) network, such as a TDM CODEC network or a network of DSPs. In Continuous Clock mode, a frame sync occurs at the beginning of each frame. In this mode, the frame is divided into more than one time slot. During each time slot, one data word can be transferred. Each time slot is then assigned to an appropriate CODEC or DSP on the network. The DSP can be a master device that controls its own private network, or a slave device that is connected to an existing TDM network and occupies a few time slots.

The frame sync signal indicates the beginning of a new data frame. Each data frame is divided into time slots and transmission and/or reception of one data word can occur in each time slot (rather than in just the frame sync time slot as in Normal mode). The frame rate dividers, controlled by the DC[4:0] bits, select two to thirty-two time slots per frame. The length of the frame is determined by the following factors:

- The period of the serial bit clock (PSR, PM[7:0] bits for internal clock, or the frequency of the external clock on the STCK port)
- The number of bits per sample (WL[3:0] bits)
- The number of time slots per frame (DC[4:0] bits)

In Network mode, data can be transmitted in any time slot. The distinction of the Network mode is that each time slot is identified with respect to the frame sync (data word time). This time slot identification allows the option of transmitting data during the time slot by writing to the STX registers or ignoring the time slot as determined by STMSK register bits. The receiver is treated in the same manner and received data is only transferred to the receive data register/fifo if the corresponding time slot is enabled (through SRMSK).

By utilizing the STMSK and SRMSK registers, software only has to service the SSI during valid time slots. This eliminates any overhead associated with unused time slots. See Section 43.3.3.20, “SSI Transmit Time Slot Mask Register (STMSK),” and Section 43.3.3.21, “SSI Receive Time Slot Mask Register (SRMSK),” for more information on STMSK and SRMSK.

In the Two-Channel mode of operation, the second set of Transmit and Receive FIFOs and Data Registers are used to create two separate channels. These channels are completely independent, with a their own set of Core interrupts and DMA requests, which are identical to the ones available for the default channel. In this mode, data is transmitted/received in enabled time slots alternately from/to FIFO 0 and FIFO 1, starting from FIFO 0. The first data word is taken from FIFO 0 and transmitted in the first enabled time slot and subsequently, data is loaded from FIFO 1 and FIFO 0 alternately and transmitted. Similarly, the first received data is sent to FIFO 0 and subsequent data is sent to FIFO 1 and FIFO 0 alternately. Time slots can be selected through the Transmit and Receive Time Slot Mask registers (STMSK and SRMSK). For using this mode of operation, the TCH_EN bit (SCR[8]) needs to be set.

43.1.2.2.1 Network Mode Transmit

The transmit portion of SSI is enabled when the SSIEN and the TE bits in the SCR are both set. However, for continuous clock, when the TE bit is set, the transmitter is enabled only after detection of a new frame sync (transmission starts from the next frame boundary).

Normal start-up sequence for transmission is to perform the following:

1. Enable Network Mode.
2. Enable SSI
3. Write the data to be transmitted to the STX register. This clears the TDE flag
4. Set the TE bit to enable the transmitter on the next frame boundary (for continuous clock case).
5. Enable transmit interrupts.

(Alternatively, the programmer may decide not to transmit in a time slot by configuring the STMSK.) TDE flag is cleared as data is shifted from STX register to TXSR, but the STXD port remains disabled during

the time slots. When the next frame sync is detected or generated (continuous clock), the data word in TXSR and is shifted out (transmitted). When the STX register is empty, the TDE bit is set, which causes a transmitter interrupt (in case the FIFO is disabled) to be sent if the TIE bit is set. Software can poll the TDE bit or use interrupts to reload the STX register with new data for the next time slot. Failing to reload the STX register before the TXSR is finished shifting (empty) causes a transmitter underrun and the TUE error bit is set. In case the FIFO is enabled, the TFE flag is set in accordance with the watermark setting and this flag causes the transmitter interrupt to occur.

The operation of clearing the TE bit disables the transmitter after completion of transmission of the current frame. Setting the TE bit enables transmission from the next frame. During that time the STXD port is disabled. The TE bit should be cleared after the TDE bit is set to ensure that all pending data is transmitted.

To summarize, the Network mode transmitter generates interrupts every enabled time slot (when FIFO is disabled) and requires the core program to respond to each enabled time slot. These responses from the core are one of the following:

- Write data in data register to enable transmission in the next time slot.
- Configure the time slot register to disable transmission in the next time slot (unless time slot is already masked by STMSK register bit).
- Do nothing—transmit underrun occurs at the beginning of the next time slot and the previous data is re-transmitted.

In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected (TDE1 is low by default).

43.1.2.2.2 Network Mode Receive

The receiver portion of the SSI is enabled when both the SSIEN and the RE bits in the SCR are set. However, the receive enable only takes place during that time slot if RE is enabled before the second to last bit of the word. If the RE bit is cleared, the receiver is disabled at the end of the current frame. SSI is capable of finding the start of the next frame automatically. When the word is completely received, it is transferred to the SRX register, which sets the RDR bit (Receive Data Ready). Setting the RDR bit causes a receive interrupt to occur if the receiver interrupt is enabled (the RIE bit is set) and (Receive data ready enable) RDR_EN bit is set. The second data word (second time slot in the frame), begins shifting in immediately after the transfer of the first data word to the SRX register. The core program has to read the data from the Receive Data Register (which clears RDR) before the second data word is completely received (ready to transfer to RX data register) or a receive overrun error occurs (the ROE bit is set).

An interrupt can occur after the reception of each enabled data word or the programmer can poll the RDR flag. The core program response can be one of the following:

- Read RX and use the data.
- Read RX and ignore the data.
- Do nothing—the receiver overrun exception occurs at the end of the current time slot.

NOTE

For a continuous clock, the optional frame sync output and clock output signals are not affected, even if the transmitter or receiver is disabled. TE and RE do not disable the bit clock or the frame sync generation. To disable the bit clock and the frame sync generation, the SSIEN bit in the SCR can be cleared or TFR_CLK_DIS/RFR_CLK_DIS bits can be set, or the port control logic external to the SSI (for example, in the IOMUX) can be re configured.

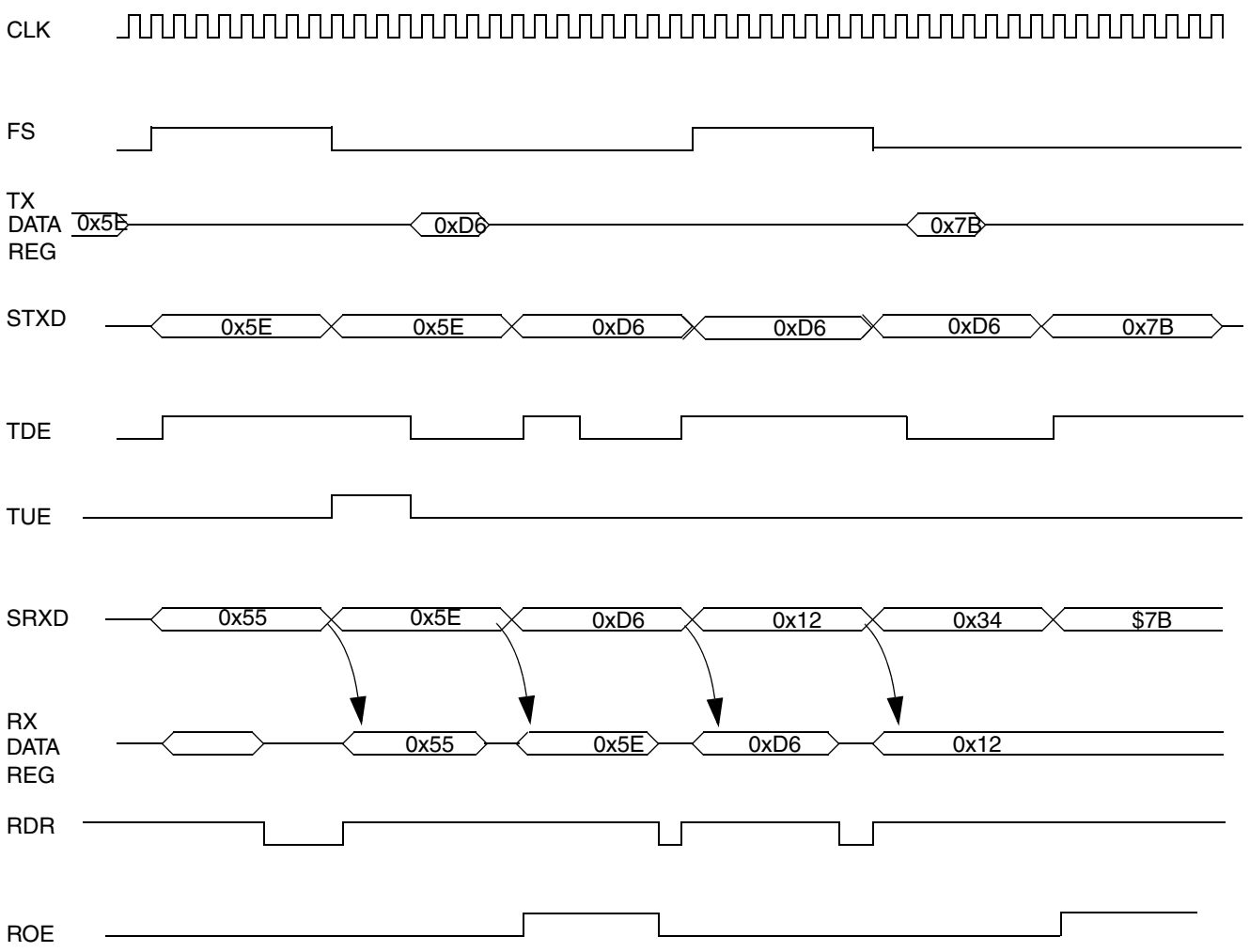
In the Two-Channel mode of operation, both the channels (Data Registers, FIFOs, Interrupts and DMA requests) operate in the same manner, as described above. The only difference in case of the second channel is that the Interrupts related to this channel are generated only in case this mode of operation is selected.

The transmitter and receiver timing for an 8-bit word with continuous clock, FIFO disabled, three words per frame sync in Network mode is shown in Figure 43-5 (“Network Mode Timing - Continuous Clock”).

NOTE

The transmitter repeats the value 0x5E because of an underrun condition

For the receive section, data received on the SRXD pin gets transferred to the Rx Data register at the end of each time slot. If the FIFO is disabled, the RDR flag gets set and causes a receiver interrupt if RE, RIE and RDR_EN bits are set. If the FIFO is enabled, then the RFF flag is used for interrupt generation (this flag is set in accordance with the watermark settings). Here all time slots are enabled. The receive data ready flag is set after reception of the first data (0x55). Since the flag is not cleared (Rx Data Register is not read by core), the Receive Overrun Error (ROE) flag is set on reception of the next data (0x5E). ROE flag is cleared on writing ‘1’ to the corresponding interrupt status bit in SSI Status Register.



Note: Processor must write '1' to the corresponding TUE/ROE Interrupt status bit in SISR to clear TUE/ROE Interrupt.

Figure 43-5. Network Mode Timing - Continuous Clock

43.1.2.3 Gated Clock Mode

Gated Clock mode is often used to hook up to SPI-type interfaces on Micro controller Units (MCUs) or external peripheral chips. In Gated Clock mode, the presence of the clock indicates that valid data is on the STXD or SRXD ports. For this reason, no frame sync is needed in this mode. Once transmission of data has completed, the clock is pulled to the inactive state. Gated clocks are allowed for both the transmit and receive sections with either internal or external clock in Normal mode. Gated clocks are not allowed in Network mode. See Table 43-5 for SSI configuration for gated-mode operation.

The clock runs when the TE bit and/or the RE bit are appropriately enabled. For the case of internally generated clock, all internal bit clocks, word clocks, and frame clocks continue to operate. When a valid time slot occurs (such as the first time slot in Normal mode), the internal bit clock is enabled onto the appropriate clock port. This allows data to be transferred out in periodic intervals in Gated Clock mode.

With an external clock, the SSI waits for a clock signal to be received. Once the clock begins, valid data is shifted in. Care should be taken to clear all DC bits (0x00000) when SSI is used in Gated mode. In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of AISR register are not generated.

For Gated clock operated in external clock mode, a proper clock signalling must be applied to the SSI STCK in order for it to function properly. When TSCKP is 0, CLK_IST value should be 1. When TSCKP is 1, CLK_IST value should be 0. If the SSI uses rising edge transition to clock data (TSCKP=0) and the falling edge transition to latch data (RSCKP=0), the clock must be in an active low state when idle. If the SSI uses falling edge transition to clock data (TSCKP=1) and the rising edge transition to latch data (RSCKP=1), the clock must be in a active high state when idle. Figure 43-6, through Figure 43-9 illustrate the different edge clocking/latching.

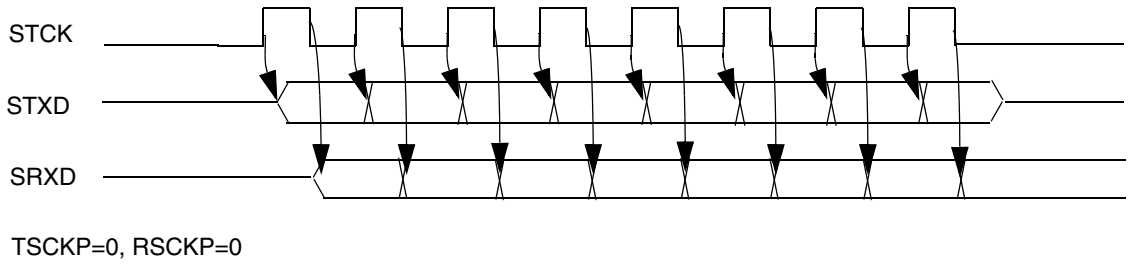


Figure 43-6. Internal Gated Mode Timing—Rising Edge Clocking / Falling Edge Latching

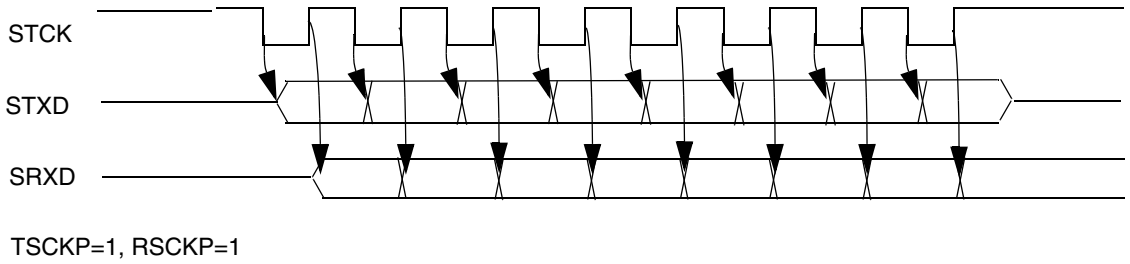


Figure 43-7. Internal Gated Mode Timing—Falling Edge Clocking / Rising Edge Latching

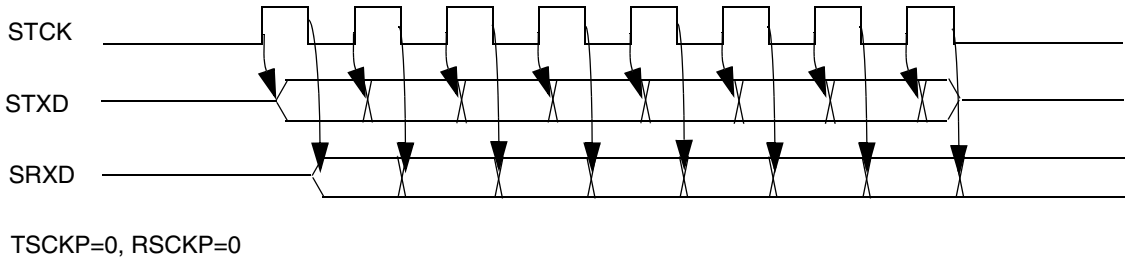


Figure 43-8. External Gated Mode Timing—Rising Edge Clocking / Falling Edge Latching

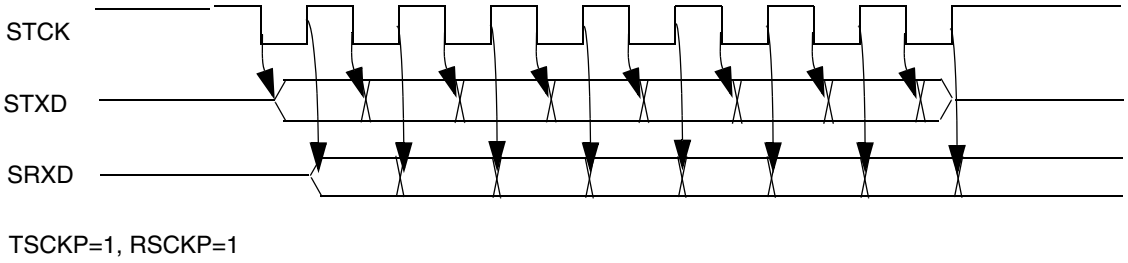


Figure 43-9. External Gated Mode Timing—Falling Edge clocking / Rising Edge Latching

NOTE

- The bit clock ports must be kept free of timing glitches. If a single glitch occurs, all ensuing transfers will be out of synchronization.
- In case of External Gated Mode, even though the Tx Data line is put in the high-impedance state at the last non-active edge of the bit clock, the round trip delay should be sufficient to take care of hold time requirements at the external receiver.

43.1.2.4 I²S Mode

The SSI is compatible with I²S bus specification from Philips Semiconductors (February 1986, Revised June 5, 1996). See Figure 43-10 on "I²S Mode Timing - Serial Clock, Frame Sync and Serial Data" for an illustration of the basic I2S protocol timing.

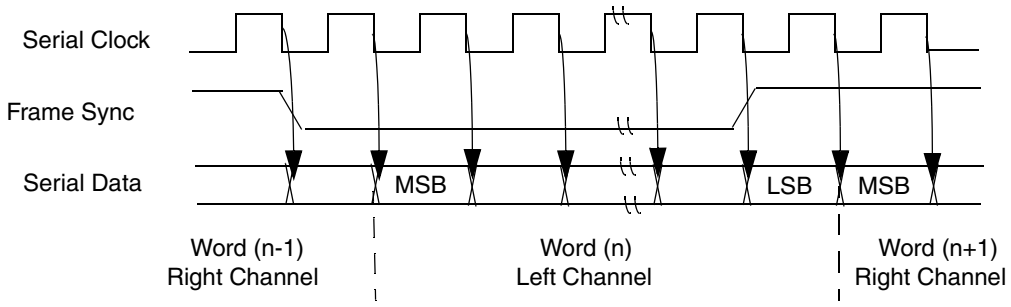


Figure 43-10. I2S Mode Timing—Serial Clock, Frame Sync and Serial Data

Select I²S mode using the options listed in Table 43-2.

Table 43-2. I²S Mode Selection

I²S_MODE[1]	I2S_MODE[0]	Mode Type
0	0	Normal mode
0	1	I2S master mode
1	0	I2S slave mode
1	1	Normal mode

In normal mode operation, no register bits are forced to any particular state internally and the user can program the SSI to work in any operating condition.

When I2S modes are entered (I2S master (01) or I2S slave (10)), the following settings are recommended:

- Sync mode (SCR[4] =1)
- Tx shift direction: MSB transmitted first (STCR[4]=0)
- Rx shift direction: MSB received first (SRCR[4]=0)
- Tx data clocked at falling edge of the clock (STCR[3]=1)
- Rx data latched at rising edge of the clock (SRCR[3]=1)
- Tx frame sync active low (STCR[2]=1)
- Rx frame sync active low (SRCR[2]=1)
- Tx frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SRCR[0]=1)
- TX Frame Rate should be 2 i.e. (STCCR[12:8] = 1)
- RX Frame Rate should be 2 i.e. (SRCCR[12:8] = 1)

In I^2S master mode(SCR[6:5]=01), the following additional settings are recommended:

- TXDIR bit (STCR[5]) set to 1 to select internal generated bit clock
- TFDIR bit (STCR[6]) set to 1 to select internal generated frame sync

In I^2S master mode(SCR[6:5]=01), the following settings are internally overridden by the hardware:

- Network mode is selected (SCR[3]=1)
- Tx frame sync length set to one-word-long-frame (STCR[1]=0)
- Rx frame sync length set to one-word-long-frame (SRCR[1]=0)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the bit clock and frame sync:

- PM (STCCR[7:0])
- PSR (STCCR[17])
- DIV2(STCCR[18]
- WL (STCCR[16:13]
- DC (STCCR[12:8])

The word length is fixed to 32 in I^2S Master mode and the WL bits determine the number of bits that will contain valid data (out of the 32 transmitted/received bits in each channel). The fixing of word duration as 32 simplifies the relation between oversampling clock (ccm_ssi_clk) and Frame Sync (ccm_ssi_clk becomes an integer multiple of Frame Sync).

In I^2S slave mode(SCR[6:5]=10), the following additional settings are recommended:

- TXDIR bit(STCR[5]) set to 0 to select external generated bit clock
- TFDIR bit(STCR[6]) set to 0 to select external generated frame sync

In I^2S slave mode(SCR[6:5]=10), the following settings are internally overridden by the hardware:

- Normal mode is selected (SCR[3]=0)
- Tx frame sync length set to one-bit-long-frame (STCR[1]=1)
- Rx frame sync length set to one-bit-long-frame (SRCR[1]=1)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)

The user needs to set the following control bits to configure the data transmission:

- WL (STCCR[16:13])
- DC (STCCR[12:8])

The word length is variable in I^2S slave mode and the WL bits determine the number of bits that will contain valid data. The actual word length is determined by the external CODEC. The external I^2S Master still sends frame sync according to the I^2S protocol (early, word wide and active low), the SSI internally operates so that each frame sync transition is the start of a new frame (the WL bits determine the number of bits to be transmitted/received). After one data word has been transferred, the SSI waits for the next frame sync transition to start operation in the next time slot. Transmit (STMSK) and receive (SRMSK) mask bits should not be used in I^2S Slave mode of operation. Masking is supported only for network mode of operation.

43.1.2.5 AC97 Mode

In AC97 mode of operation, the SSI transmits a 16-bit Tag Slot at the start of a frame and the rest of the slots (in that frame) are all 20-bits wide. The same sequence is followed while receiving data. See the AC97 specification for details regarding transmit and receive sequences and data formats.

Note that the SSI only has one RxDATA pin so the SSI can only support one codec. Secondary codecs are not supported.

When AC97 mode is enabled, the following settings are internally overridden by the hardware. The programmed register values are not changed by entering AC97 mode but they no longer apply to the module’s operation. Writing to the programmed register fields will update their values; these updates can be seen by reading back the register fields. However, these settings will not take effect until AC97 mode is turned off.

The register bits within the bracket are the equivalent settings:

- Sync mode is entered (SCR[4] =1)
- Network mode is selected (SCR[3]=1)
- Tx shift direction is MSB transmitted first (STCR[4]=0)
- Rx shift direction is MSB received first (SRCR[4]=0)
- Tx data is clocked at rising edge of the clock (STCR[3]=0)
- Rx data is latched at falling edge of the clock (SRCR[3]=0)
- Tx frame sync is active high (STCR[2]=0)
- Rx frame sync is active high (SRCR[2]=0)

- Tx frame sync length is one-word-long-frame (STCR[1]=0)
- Rx frame sync length is one-word-long-frame (SRCR[1]=0)
- Tx frame sync initiated one bit before data is transmitted (STCR[0]=1)
- Rx frame sync initiated one bit before data is received (SRCR[0]=1)
- Tx shifting w.r.t. bit 0 of TXSR (STCR[9]=1)
- Rx shifting w.r.t. bit 0 of RXSR (SRCR[9]=1)
- Tx FIFO is enabled (STCR[7]=1)
- Rx FIFO is enabled (SRCR[7]=1)
- TFDIR bit (STCR[6]) is forced to 1 internally to select internal generated frame sync
- TXDIR bit (STCR[5]) is forced to 0 internally to select external generated bit clock

Any alteration of these bits individually will not affect the operational conditions of the SSI unless AC97 mode is deselected.

Hence, the only control bits needed to be set by the user to configure the data transmission/reception are the WL (STCCR[16:13]) and DC (STCCR[12:8]) bits. In AC97 mode, the WL bits can only legally take the values corresponding to 16-bit (truncated data) or 20-bit time slots. In case WL bits are set to select 16-bit time slots, the SSI pads the transmit data (four least significant bits) with zeros and while receiving, stores only the most significant 16 bits in the Rx FIFO.

Follow the sequence for programming the SSI to work in AC97 mode:

1. Program the WL bits to a value corresponding to either 16 or 20 bits. The WL bit setting is only for the data portion of the AC97 frame (Slots #3 through #12). The Tag slot (Slot #0) is always 16 bits wide and the Command Address and Command Data slots (Slots #1 and #2) are always 20 bits wide.
2. Select the number of time slots by programming the DC bits. For AC97 operation, DC bits should be set to a value of ‘0xC’, resulting in 13 time slots per frame.
3. Write data to be transmitted, in Tx FIFO 0 (through Tx Data Register 0) and Tx FIFO 1 while using Two-Channel Mode (TCH_EN = 1).
4. Program the FV, TIF, RD, WR and FRDIV bits in SACNT register
5. Update the contents of SACADD, SACDAT and SATAG (for Fixed mode only) registers
6. Enable the AC97 mode of operation (AC97EN bit in SACNT register)

Once the SSI starts transmitting and receiving data (after being configured in AC97 mode), the programmer needs to service the interrupts, as and when they are raised (updates to command address/data or tag registers, reading of received data and writing more data for transmission). Further details regarding fixed and variable mode implementation are provided in the following sections.

While using AC97 in Two-Channel Mode (TCH_EN=1), it is recommended that the received tag is not stored in the Rx FIFO (TIF=0). In case the programmer needs to update the SATAG register and also issue a RD/WR command (in a single frame), it is recommended that the SATAG register be updated prior to issuing a RD/WR command.

43.1.2.5.1 AC97 Fixed Mode (SACNT[1]=0)

In fixed mode of operation, SSI transmits in accordance with the AC97 Frame Rate Divider bits (i.e. FRDIV in SACNT) which decides the number of frames for which the SSI should be idle, after operating for one frame.

In a valid frame, TAG Value (written by Core) will be transmitted in Slot #0, Command Address will be transmitted in Slot #1 in case of RD/WR Command, and Command Data will be transmitted in Slot #2 in case of a WR Command. The data from TX-FIFO is transmitted in Slot #3 - Slot #12 depending on the valid slots indicated by the TAG value.

While receiving, bit 15 of the TAG Value (Slot #0) is checked to see if the CODEC is ready. If this bit is set, the frame is received. The received TAG provides the information about Slots containing valid data. The the corresponding TAG bit is valid, the Command Address (Slot #1) and Command Data (Slot #2) values are stored in the corresponding registers. The received data (Slot #3 - Slot #12) is then stored in the Rx-FIFO (for valid slots).

43.1.2.5.2 AC97 Variable Mode (SACNT[1]=1)

In Variable Mode, the transmit slots which should contain data in the current frame are determined by SLOTREQ bits received in the previous frame. While receiving, if the CODEC is ready, the frame is received and the SLOTREQ bits (contained in Slot #1) are stored for scheduling transmission in the next frame.

The SACCST, SACCEN and SACCDIS registers helps in determining which transmit slots are active. This information is used to ensure that SSI does not transmit data for powered-down/inactive channels.

43.1.2.6 External Frame and Clock Operation

When applying external frame sync and clock signals to SSI, there should be at least 4-bit clock cycles between the enabling of the transmit or receive section and the rising edge of the corresponding frame sync signal. The transition of STFS or SRFS should be synchronized with the rising edge of external clock signal, STCK or SRCK.

43.1.2.7 Data Alignment Formats Supported

The SSI supports three data formats in order to provide flexibility with handling data. These formats dictate how data is written to (and read from) the data registers. Therefore, data can appear in different places in STX0/1 and SRX0/1 based on the data format and the number of bits per word. Independent data formats are supported for both the transmitter and receiver (that is, the transmitter and receiver can use different data formats).

The supported data formats are:

- MSB alignment
- LSB alignment
  - Zero-extended (receive data only)
  - Sign-extended (receive data only)

With MSB alignment, the most significant byte is bits 31 through 24 of the data register if the word length is larger than or equal to 16 bits. If the word length is less than 16 bits and MSB alignment is chosen, the most significant byte is bits 15 through 8. With LSB alignment, the least significant byte is bits 7 through 0. Data alignment is controlled by the TXBIT0 bit in the STCR and the RXBIT0 bit in the SRCR. See Table 43-3 for the bit assignment for all the data formats supported by the SSI.

Table 43-3. Data Alignment

Format	Bit Number																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-bit LSB Aligned																									7	6	5	4	3	2	1	0
8-bit MSB Aligned																	7	6	5	4	3	2	1	0								
10-bit LSB Aligned																							9	8	7	6	5	4	3	2	1	0
10-bit MSB Aligned																	9	8	7	6	5	4	3	2	1	0						
12-bit LSB Aligned																					11	10	9	8	7	6	5	4	3	2	1	0
12-bit MSB Aligned																	11	10	9	8	7	6	5	4	3	2	1	0				
16-bit LSB Aligned																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-bit MSB Aligned	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
18-bit LSB Aligned															17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18-bit MSB Aligned	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
20-bit LSB Aligned													19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20-bit MSB Aligned	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
22-bit LSB Aligned											21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
22-bit MSB Aligned	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
24-bit LSB Aligned									23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24-bit MSB Aligned	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								

In addition, receive data can either be zero-extended or sign-extended if LSB alignment is selected. With zero-extension, all bits above the most significant bit are 0's. This format is useful when data is stored in a pure integer format. With sign-extension, all bits above the most significant bit are equal to the most

significant bit. This format is useful when data is stored in a fixed-point integer format (which implies fractional values). Receive data extension is controlled by the RXEXT bit in the SRCR. Transmit data used with LSB alignment has no concept of sign/zero-extension. Unused bits above the most significant bit are simply ignored.

When configured in I2S or AC97 mode, the SSI forces the selection of LSB alignment. However, RXEXT still permits a choice between zero-extension and sign-extension.

See Section 43.3.3.10, "SSI Transmit Configuration Register (STCR)," and Section 43.3.3.11, "SSI Receive Configuration Register (SRCR)," for more details on the relevant bits in the STCR and SRCR registers.

43.2 External Signal Description

43.2.1 Overview

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio Mux (AUDMUX) module or directly as well. See the AUDMUX chapter for programming details of various multiplexing options.

Table 43-4. Signal Properties

Name	Port	I/O	Function	Reset State	Pull up
SRCK	—	I/O	Serial Receive Clock	0	Passive
SRFS	—	I/O	Serial Receive Frame Sync	0	Passive
SRXD	—	I	Serial Receive Data	—	—
STCK	—	I/O	Serial Transmit Clock	0	Passive
STFS	—	I/O	Serial Transmit Frame Sync	0	Passive
STXD	—	O	Serial Transmit Data	0	Passive

43.2.2 Detailed Signal Descriptions

43.2.2.1 SRCK—Serial Receive Clock

The SRCK port can be used as either an input or an output. This clock signal is used by the receiver in asynchronous mode and is always continuos. During synchronous mode, the STCK port is used instead for clocking in data. In SSI synchronous modes, this port can be used as an output port for the oversampling clock, (ccm_ssi_clk) e.g. In I^2S master mode, this port can be used to output ccm_ssi_clk to external CODEC.

43.2.2.2 SRFS—Serial Receive Frame Sync

The SRFS port can be used as either an input or an output. The frame sync is used by the receiver to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur

one bit before the transfer of data or right at the transfer of data. If SRFS is configured as input, the external device should drive SRFS during rising edge of STCK or SRCK.

43.2.2.3 SRXD—Serial Receive Data

The SRXD port is an input and is used to bring serial data into the Receive Data Shift Register.

43.2.2.4 STCK—Serial Transmit Clock

The STCK port can be used as either an input or an output. This clock signal is used by the transmitter and can be either continuous or gated. During Gated Clock mode, data on the STCK port is valid only during the transmission of data, otherwise it is pulled to the inactive state. In Synchronous mode, this port is used by both the transmit and receive sections.

43.2.2.5 STFS—Serial Transmit Frame Sync

The STFS port can be used as either an input or an output. The frame sync is used by the transmitter to synchronize the transfer of data. The frame sync signal can be one bit or one word in length and can occur one bit before the transfer of data or right at the transfer of data. In Synchronous mode, this port is used by both the transmit and receive sections. In Gated Clock mode, frame sync signals are not used. If STFS is configured as input, the external device should drive STFS during rising edge of STCK if TSCKP is +ve edge triggered. The external device should drive STFS during falling edge of STCK if TSCKP is -ve edge triggered.

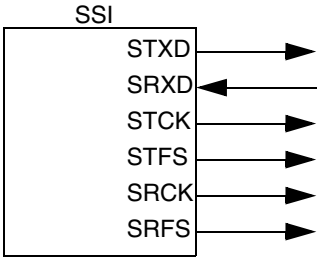
43.2.2.6 STXD—Serial Transmit Data

The STXD port is an output and transmits data from the Serial Transmit Shift Register. The STXD port is an output port when data is being transmitted and is disabled between data word transmissions and on the trailing edge of the bit clock after the last bit of a word is transmitted.

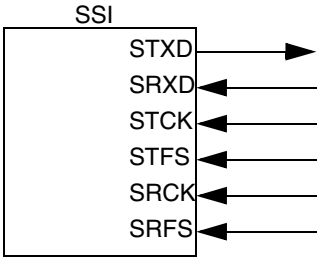
Figure 43-11 (“Asynchronous (SYN=0) SSI Configurations—Continuous Clock”) and Figure 43-12 (“Synchronous SSI Configurations—Continuous and Gated Clock”) show the main SSI configurations. These ports support all transmit and receive functions with continuous or gated clock as shown.

NOTE

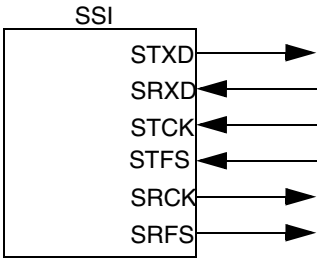
Gated clock implementations do not require the use of the frame sync ports (STFS and SRFS).



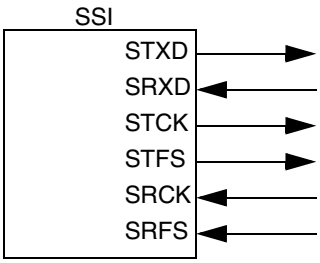
SSI Internal Continuous Clock for TX/RX (RXDIR=1,TXDIR=1,RFDIR=1,TFDIR=1,SYN=0)



SSI External Continuous Clock for TX/RX (RXDIR=0,TXDIR=0,RFDIR=0,TFDIR=0,SYN=0)

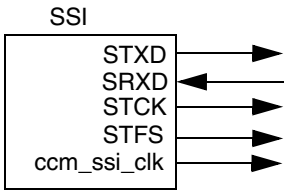


SSI Internal Continuous Clock for RX (RXDIR=1, TXDIR=0,RFDIR=1,TFDIR=0, SYN=0)
SSI External Continuous Clock for TX

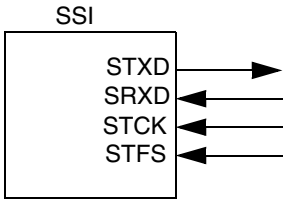


SSI Internal Continuous Clock for TX (RXDIR=0, TXDIR=1, RFDIR=0, TFDIR=1, SYN=0)
SSI EXternal Continuous Clock for RX

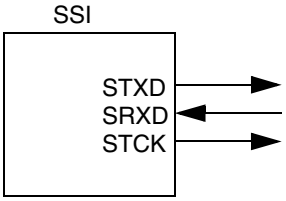
Figure 43-11. Asynchronous (SYN=0) SSI Configurations—Continuous Clock



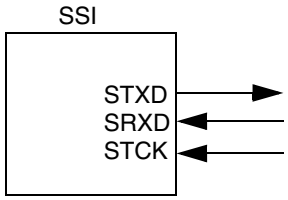
SSI Internal Continuous Clock (RXDIR=0, TXDIR=1, RFDIR=X, TFDIR=1, SYN=1, SYS_CLK_EN = 1)
SSI I2S Master Mode(I2S_Mode=01, SYS_CLK_EN)



SSI External Continuous Clock (RXDIR=0, TXDIR=0, RFDIR=X, TFDIR=0,SYN=1)
SSI I2S Slave Mode(I2S_Mode=10)



SSI Internal Gated Clock (RXDIR=1, TXDIR=1, SYN=1)



SSI External Gated Clock (RXDIR=1, TXDIR=0, SYN=1)

Figure 43-12. Synchronous SSI Configurations—Continuous and Gated Clock

See Figure 43-13 for an example of the port signals for an 8-bit data transfer. Continuous and gated clock signals are shown, as well as the bit-length frame sync signal and the word-length frame sync signal.

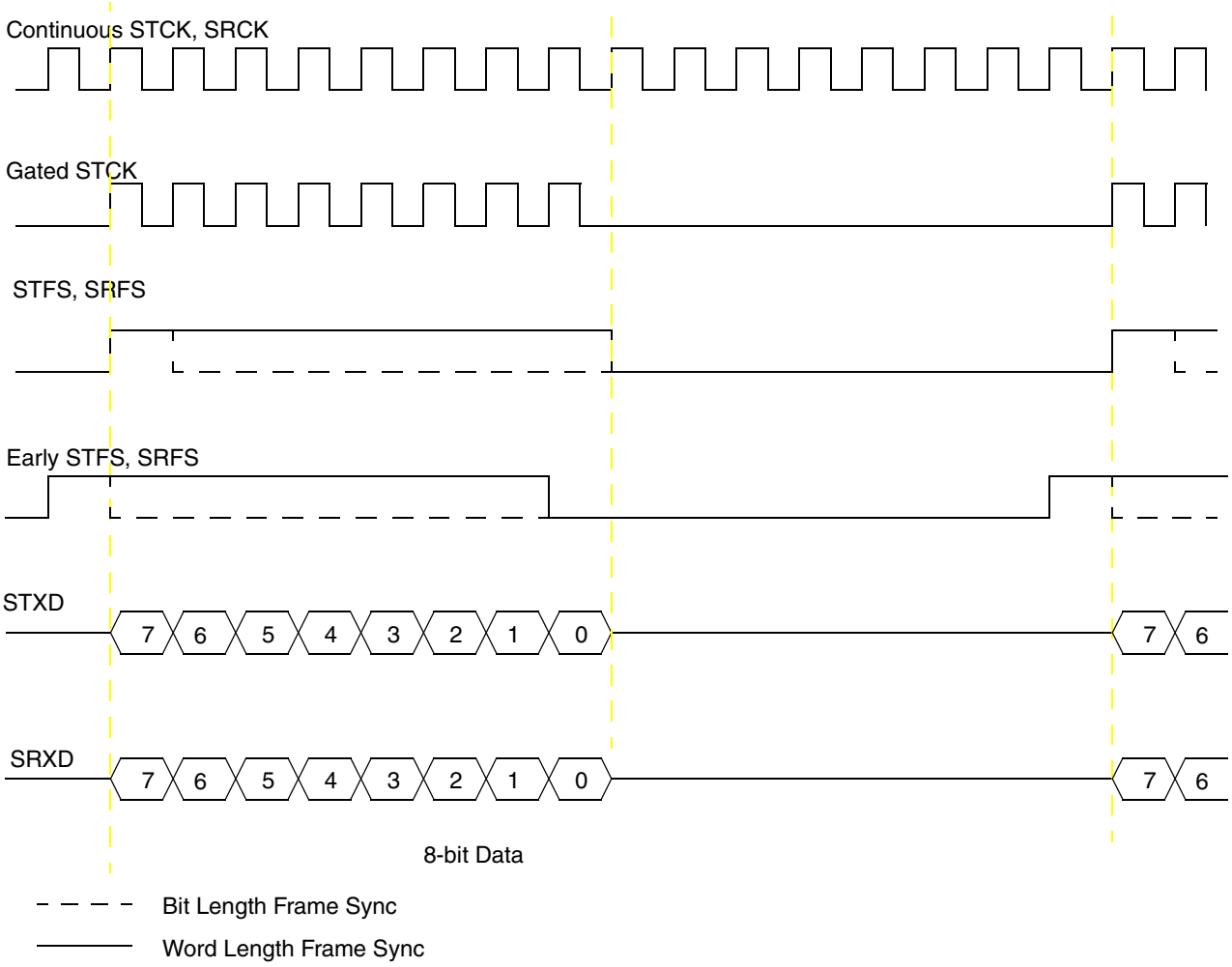


Figure 43-13. Serial Clock and Frame Sync Timing

See Table 43-5 for list of clock pin configurations.

Table 43-5. Clock Pin Configurations

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
Asynchronous Mode								
0	0	0	0	0	RCK in	TCK in	RFS in	TFS in
0	0	0	0	1	RCK in	TCK in	RFS in	TFS out
0	0	0	1	0	RCK in	TCK in	RFS out	TFS in
0	0	0	1	1	RCK in	TCK in	RFS out	TFS out
0	0	1	0	0	RCK in	TCK out	RFS in	TFS in
0	0	1	0	1	RCK in	TCK out	RFS in	TFS out

Table 43-5. Clock Pin Configurations (continued)

SYN	RXDIR	TXDIR	RFDIR	TFDIR	SRCK	STCK	SRFS	STFS
0	0	1	1	0	RCK in	TCK out	RFS out	TFS in
0	0	1	1	1	RCK in	TCK out	RFS out	TFS out
0	1	0	0	0	RCK out	TCK in	RFS in	TFS in
0	1	0	0	1	RCK out	TCK in	RFS in	TFS out
0	1	0	1	0	RCK out	TCK in	RFS out	TFS in
0	1	0	1	1	RCK out	TCK in	RFS out	TFS out
0	1	1	0	0	RCK out	TCK out	RFS in	TFS in
0	1	1	0	1	RCK out	TCK out	RFS in	TFS out
0	1	1	1	0	RCK out	TCK out	RFS out	TFS in
0	1	1	1	1	RCK out	TCK out	RFS out	TFS out
Synchronous Mode								
1	0	0	x	0	—	CK in	—	FS in
1	0	0	x	1	—	CK in	—	FS out
1	0	1	x	0	—	CK out	—	FS in
1	0	1	x	1	—	CK out	—	FS out
1	1	0	x	x	—	Gated in	—	—
1	1	1	x	x	—	Gated out	—	—

43.3 Memory Map and Register Definition

Section 43.3.3, "Register Descriptions," provides the detailed descriptions for all of the SSI registers.

43.3.1 SSI Memory Map

Table 43-6 shows the SSI memory map.

Table 43-6. SSI Memory Map

Address	Register	Access	Reset Value	Section/Page
0xBASE_00 (STX0_)	SSI Transmit Data Register	R/W	0x0000_0000	43.3.3.1/43-32
0xBASE_04 (STX1_)	SSI Transmit Data Register 1	R/W	0x0000_0000	
0xBASE_08 (SRX0_)	SSI Receive Data Register 0	Read-only	0x0000_0000	43.3.3.4/43-36
0xBASE_0C (SRX1_)	SSI Receive Data Register 1	Read-only	0x0000_0000	
0xBASE_10 (SCR)	SSI Control Register	R/W	0x0000_0000	43.3.3.7/43-39
0xBASE_14 (SISR)	SSI Interrupt Status Register	Read-only	0x0000_3003	43.3.3.8/43-41
0xBASE_18 (SIER)	SSI Interrupt Enable Register	R/W	0x0000_3003	43.3.3.9/43-46

Table 43-6. SSI Memory Map (continued)

Address	Register	Access	Reset Value	Section/Page
0xBASE_1C (STCR)	SSI Transmit Configuration Register	R/W	0x0000_0200	43.3.3.10/43-47
0xBASE_20 (SRCR)	SSI Receive Configuration Register	R/W	0x0000_0200	43.3.3.11/43-49
0xBASE_24 (STCCR)	SSI Transmit Clock Control Register	R/W	0x0004_0000	43.3.3.12/43-51
0xBASE_28 (SRCCR)	SSI Receive Clock Control Register	R/W	0x0004_0000	
0xBASE_2C (SFCSR)	SSI FIFO Control/Status Register	R/W	0x0081_0081	43.3.3.13/43-54
0xBASE_30 (STR)	SSI Test Register[1]	R/W	0x0000_1111	43.3.3.14/43-59
0xBASE_34 (SOR)	SSI Option Register[2]	R/W	0x0000_0000	43.3.3.15/43-61
0xBASE_38 (SACNT)	SSI AC97 Control Register	R/W	0x0000_0000	43.3.3.16/43-62
0xBASE_3C (SACADD)	SSI AC97 Command Address Register	R/W	0x0000_0000	43.3.3.17/43-63
0xBASE_40 (SACDAT)	SSI AC97 Command Data Register	R/W	0x0000_0000	43.3.3.18/43-64
0xBASE_44 (SATAG)	SSI AC97 Tag Register	R/W	0x0000_0000	43.3.3.19/43-65
0xBASE_48 (STMSK)	SSI Transmit Time Slot Mask Register	R/W	0x0000_0000	43.3.3.20/43-66
0xBASE_4C (SRMSK)	SSI Receive Time Slot Mask Register	R/W	0x0000_0000	43.3.3.21/43-67
0xBASE_50 (SACCST)	SSI AC97 Channel Status Register	R	0x0000_0000	43.3.3.22/43-68
0xBASE_54 (SACCEN)	SSI AC97 Channel Enable Register	W	0x0000_0000	43.3.3.23/43-69
0xBASE_58 (SACCDIS)	SSI AC97 Channel Disable Register	W	0x0000_0000	43.3.3.24/43-70
0xBASE_5C (PHCONFIG1)	SSI Phase Configuration Register	R/W	0x0000_0000	43.3.3.25/43-71
0xBASE_60 (FREQMEAS1)	SSI Frequency Measurement Register	R	0x0000_0000	43.3.3.26/43-72

1 SSI Test Register is intended for debugging purposes only and is not visible to the end user.

2 SSI Option Register intended for internal use only and is not visible to the end user.

43.3.2 Register Summary

Figure 43-14 shows the key to the register fields and Table 43-7 shows the register figure conventions.

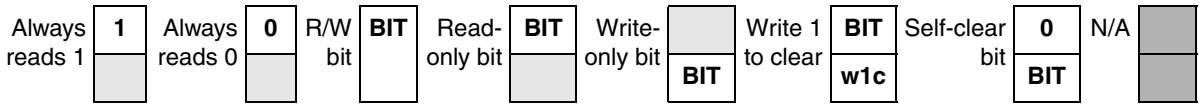


Figure 43-14. Key to Register Fields

Table 43-7. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	

Table 43-7. Register Figure Conventions (continued)

Convention	Description
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 43-8 shows the SSI register summary.

Table 43-8. SSI Register Summary

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_00 (STX0_)	R	STX0[31:16]															
	W																
	R	STX0[15:0]															
	W																
0xBASE_04 (STX1_)	R	STX1[31:16]															
	W																
	R	STX1[15:0]															
	W																
0xBASE_08 (SRX0_)	R	SRX0[31:16]															
	W																
	R	SRX0[15:0]															
	W																
0xBASE_0C (SRX1_)	R	SRX1[31:16]															
	W																
	R	SRX1[15:0]															
	W																
0xBASE_10 (SCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	SYNC_TX_FS	RFR_CLK_DIS	TFR_CLK_DIS	CLK_IST	TCH_EN	SYS_CLK_EN	I2S MODE[1:0]		SYN	NET	RE	TE	SSI EN
	W																
0xBASE_14 (SISR)	R	0	0	0	0	0	0	0	RFRC	TFRC	0	0	0	0	CMDAU	CMDDU	RXT
	W																
	R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
	W																

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_18 (SIER)	R	0	0	0	0	0	0	0	RFRC_EN	TFRC_EN	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
	W																
	R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
	W																
0xBASE_1C (STCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
	W																
0xBASE_20 (SRCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
	W																
0xBASE_24 (STCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0xBASE_28 (SRCCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
	W																
	R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
	W																
0xBASE_2C (SFCSR)	R	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
	W																
	R	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
	W																
0xBASE_30 (STR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	TEST	RCK2TCK	RFS2TFS	RXSTATE[4:0]					TXD2RXD	TCK2RCK	TFS2RFS	TXSTATE[4:0]				
	W																
0xBASE_34 (SOR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	CLKOFF	RX_CLR	TX_CLR	INIT	WAIT[1:0]		SYNRST
	W																

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_38 (SACNT)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97EN
	W																
0xBASE_3C (SACADD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
	W																
	R	SACADD[15:0]															
	W																
0xBASE_40 (SACDAT)	R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
	W																
	R	SACDAT[15:0]															
	W																
0xBASE_44 (SATAG)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	SATAG[15:0]															
	W																
0xBASE_48 (STMSK)	R	STMSK[31:0]															
	W																
	R																
	W																
0xBASE_4C (SRMSK)	R	SRMSK[31:0]															
	W																
	R																
	W																

Name		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBASE_50 (SACCST)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	SACCST[9:0]									
	W																
0xBASE_54 (SACCEN)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W							SACCEN[9:0]									
0xBASE_58 (SACCDIS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W							SACCDIS[9:0]									
0xBASE_5C (PHCONFIG1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	ClkSercSel[3:0]				LOCK	0	0	0	GainSel[2:0]		
	W																
0xBASE_60 (FREQMEAS1)	R	0	0	0	0	0	0	0	0	FREQMEAS[23:16]							
	W																
	R	FREQMEAS[15:0]															
	W																

43.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams in bit order.

43.3.3.1 SSI Transmit Data Registers 0 & 1 (STX0/1)

See Figure 43-15 for illustration of valid bits in the SSI0 Transmit Data Register and Table 43-9 for description of the bit fields in the register.

0xBASE_00 (STX0_) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STX0[31:16]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STX0[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-15. SSI0 Transmit Data Register

See Figure 43-16 for illustration of valid bits in SSI1 Transmit Data Register and Table 43-9 for description of the bit fields in the register.

0xBASE_04 (STX1_) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STX1[31:16]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STX1[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-16. SSI1 Transmit Data Register

Table 43-9. SSI Transmit Data Register Field Descriptions

Field	Description
31–0 STX0 STX1	SSI Transmit Data. These bits store the data to be transmitted by the SSI. These are implemented as the first word of their respective Tx FIFOs. Data written to these registers is transferred to the Transmit Shift Register (TXSR), when shifting of the previous data is complete. If both FIFOs are in use, data is alternately transferred from STX0 and STX1, to TXSR. Multiple writes to the STX registers will not result in the previous data being over-written by the subsequent data. STX1 can only be used in Two-Channel mode of operation. Protection from over-writing is present irrespective of whether the transmitter is enabled or not. Example 1: If Tx FIFO0 is in use and user writes Data1...Data16 to STX0, Data16 will not over-write Data1. Data1...Data15 are stored in the FIFO while Data16 is discarded. Example 2: If Tx FIFO0 is not in use and user writes Data1, Data2 to STX0, then Data2 will not over-write Data1 and will be discarded.

NOTE

Enable SSI (SSIEN=1) before writing to SSI Transmit Data Registers.

43.3.3.2 SSI Transmit FIFO 0 & 1 Registers

The SSI Transmit FIFO registers are 15x32-bit registers. These registers are not directly accessible by the end user. Transmit Shift Register (TXSR) receives its values from these FIFO registers. Transmitted data is first-in-first-out. When the Transmit Interrupt Enable (TIE) bit in the SIER and either of the Transmit FIFO Empty Enable (TFE0 or 1) bits in the SIER are set, the core is interrupted whenever the number of empty slots exceed or are equal to the selected threshold value of corresponding Tx-FIFO.

43.3.3.3 SSI Transmit Shift Register (TXSR)

The SSI Transmit Shift Register (TXSR) is a 24-bit shift register that contains the data being transmitted. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted out to the Serial Transmit Data (STXD) port by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted out to the STXD port by the selected (internal/external) gated clock. The Word Length control bits (WL[3:0]) in the STCCR (described in SSI Transmit and Receive Clock Control Registers) determine the number of bits to be shifted out of the TXSR before it is considered empty and can be written to again. This word length can be 8, 10, 12, 16, 18, 20, 22 or 24 bits. The data to be transmitted occupies the most significant portion of the shift register if TXBIT0 is ‘0’, otherwise it occupies the least significant portion. The unused portion of the register is ignored. Data is always shifted out of this register with the Most Significant Bit (MSB) first when the SHFD bit of the STCR is cleared. If this bit is set, the Least Significant Bit (LSB) is shifted out first. Figure through Figure 43-20 show the transmitter loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

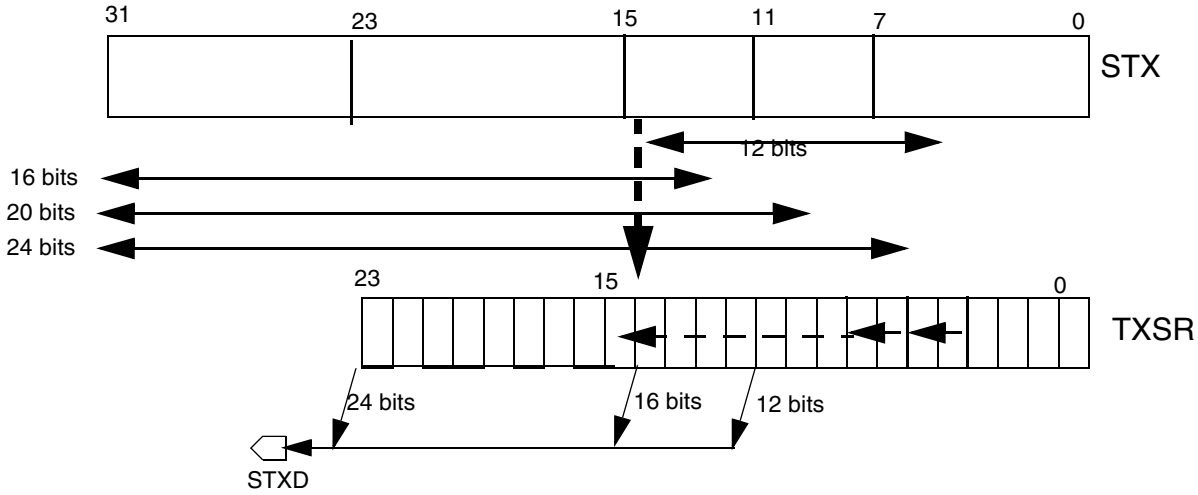


Figure 43-17. Transmit Data Path (TXBIT0=0, TSHFD=0) (MSB Alignment)

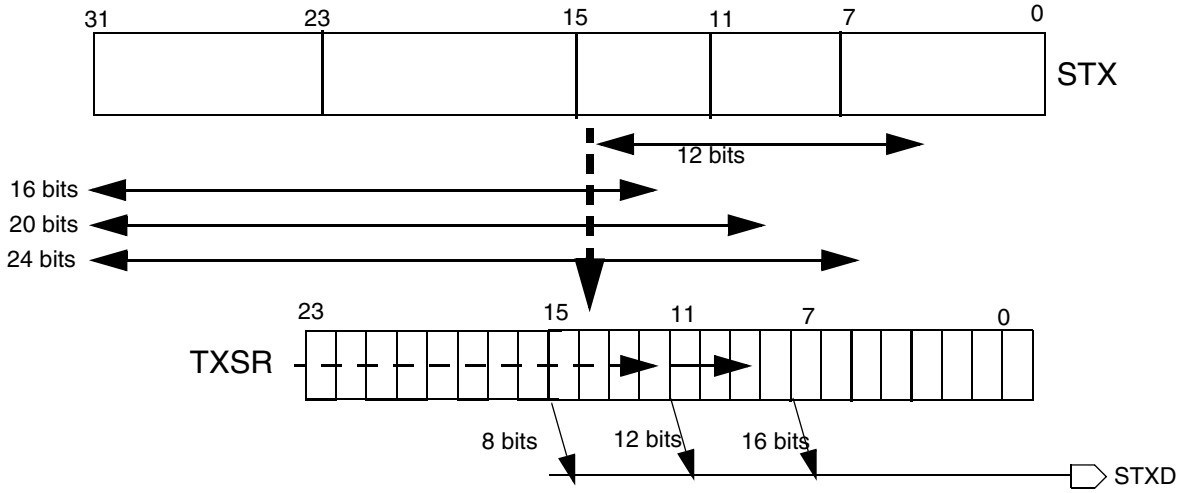


Figure 43-18. Transmit Data Path (TXBIT0=0, TSHFD=1) (MSB Alignment)

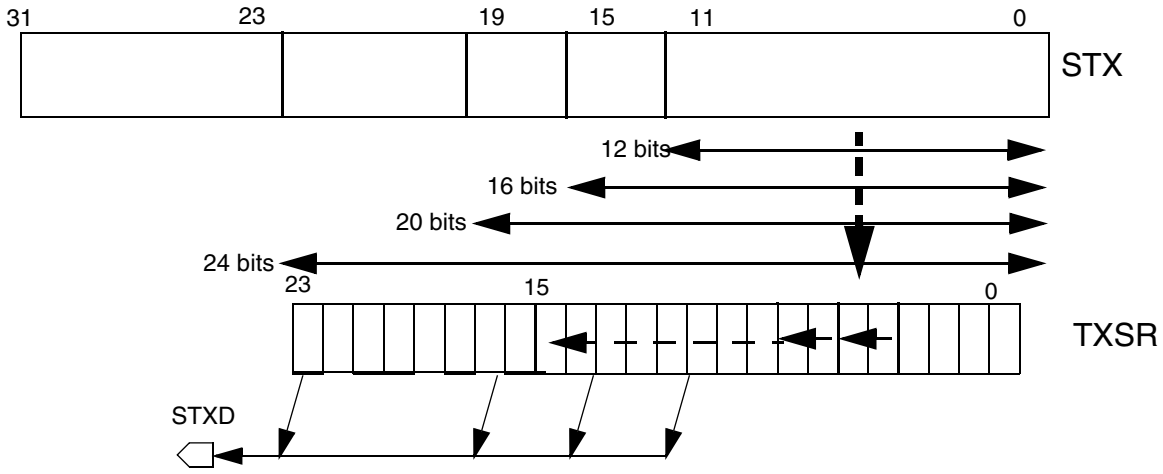


Figure 43-19. Transmit Data Path (TXBIT0=1, TSHFD=0) (LSB Alignment)

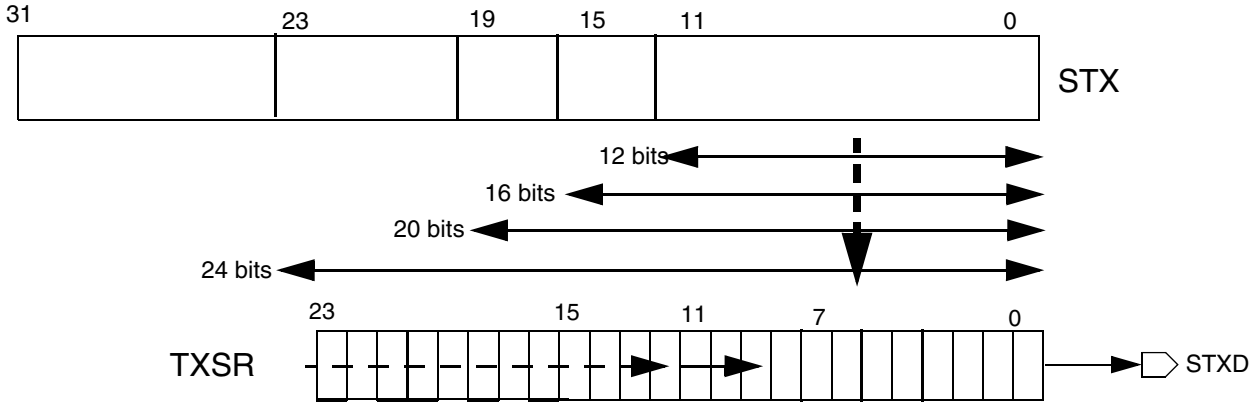


Figure 43-20. Transmit Data Path (TXBIT0=1, TSHFD=1) (LSB Alignment)

43.3.3.4 SSI Receive Data Registers 0 and 1 (SRX0/1)

See Figure 43-21 for illustration of valid bits in SSI0 Receive Data Register and Table 43-10 for description of the bit fields in the register.

0xBASE_08 (SRX0_) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX0[31:16]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX0[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-21. SSI0 Receive Data Register

See Figure 43-22 for illustration of valid bits in SSI1 Receive Data Register and Table 43-10 for description of the bit fields in the register.

0xBASE_0C (SRX1_) Access: User read-only
0xBASE_

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SRX1[31:16]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRX1[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-22. SSI1 Receive Data Register

Table 43-10. SSI_1 Receive Data Register Field Descriptions

Field	Description
31–0 SRX0 SRX1	SSI Receive Data. These bits store the data received by the SSI. These are implemented as the first word of their respective Rx FIFOs. These bits receive data from the RXSR depending on the mode of operation. In case both FIFOs are in use, data is transferred to each data register alternately. SRX1 can only be used in Two-Channel mode of operation.

43.3.3.5 SSI Receive FIFO 0 & 1 Registers

The SSI Receive FIFO Registers are 15x32-bit registers. These registers are not directly accessible by the end user (except in SSI test mode). They always accepts data from the Receive Shift Register (RXSR). The core is interrupted when the data level in either of the SSI Receive FIFOs reaches the selected threshold, if the associated interrupt is enabled.

43.3.3.6 SSI Receive Shift Register (RXSR)

The SSI Receive Shift Register (RXSR) is a 24-bit, shift register that receives incoming data from the serial receive data SRXD port. This register is not directly accessible by the end user. When a continuous clock is used, data is shifted in by the selected (internal/external) bit clock when the associated (internal/external) frame sync is asserted. When a gated clock is used, data is shifted in by the selected (internal/external) gated clock. Data is assumed to be received MSB first if the SHFD bit of the SRCR is cleared. If this bit is set, the data is received LSB first. Data is transferred to the appropriate SSI Receive Data Register (SRX0/1) or Receive FIFOs (if the receive FIFO is enabled and the corresponding SRX is full) after 8, 10, 12, 16, 18, 20, 22 or 24 bits have been shifted in depending on the WL[3:0] control bits. For receiving less than 24 bits of data, LSB bits are appended with zero. The following figures show the receiver loading and shifting operation. The figures show the working for some WL values, the same can be extended for other values.

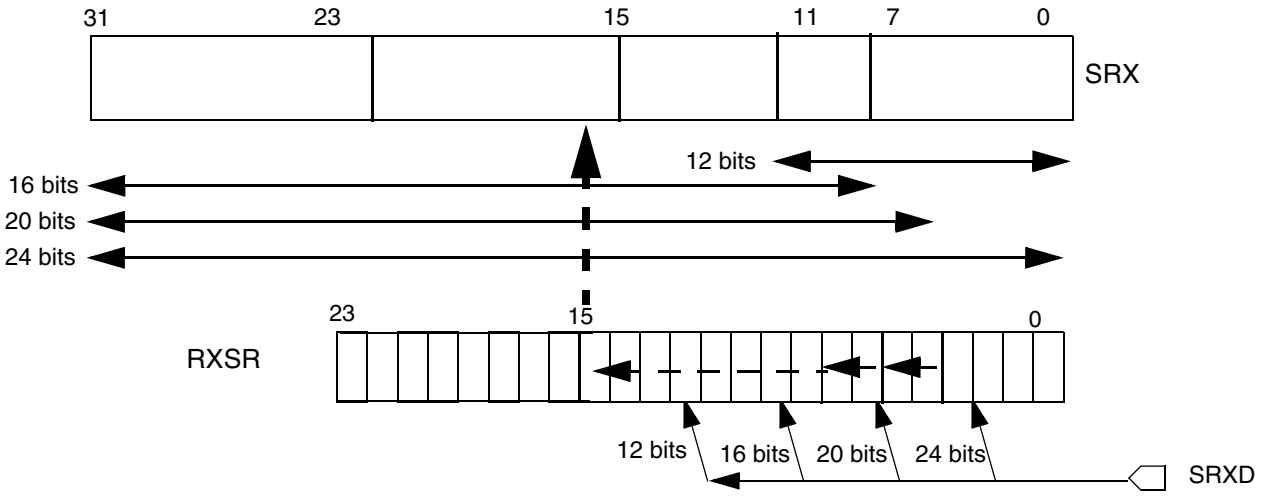


Figure 43-23. Receive Data Path (RXBIT0=0, RSHFD=0) (MSB Alignment)

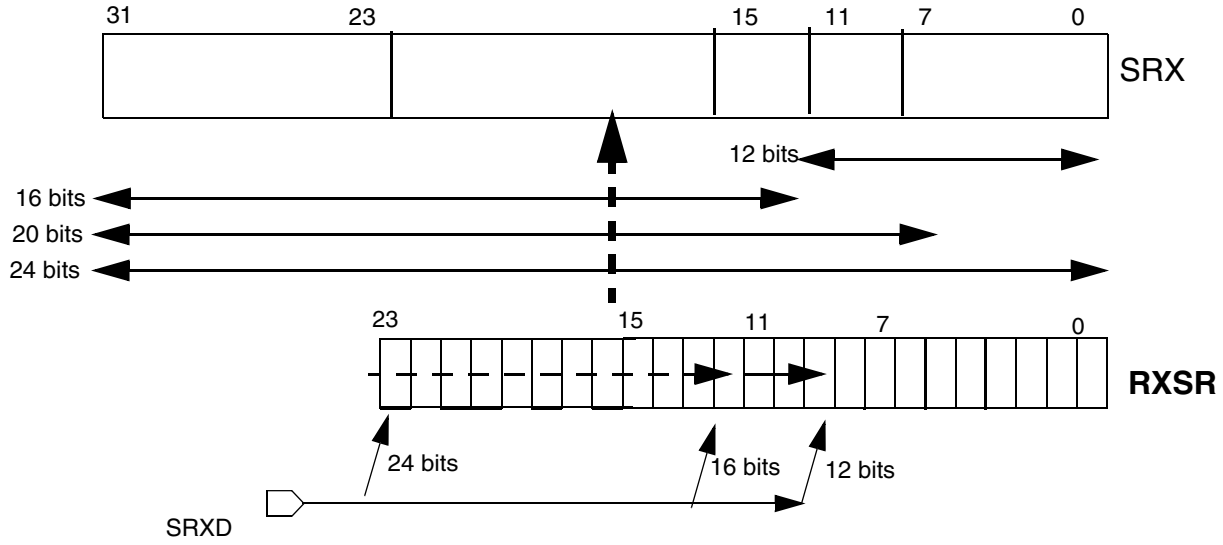


Figure 43-24. Receive Data Path (RXBIT0=0, RSHFD=1) (MSB Alignment)

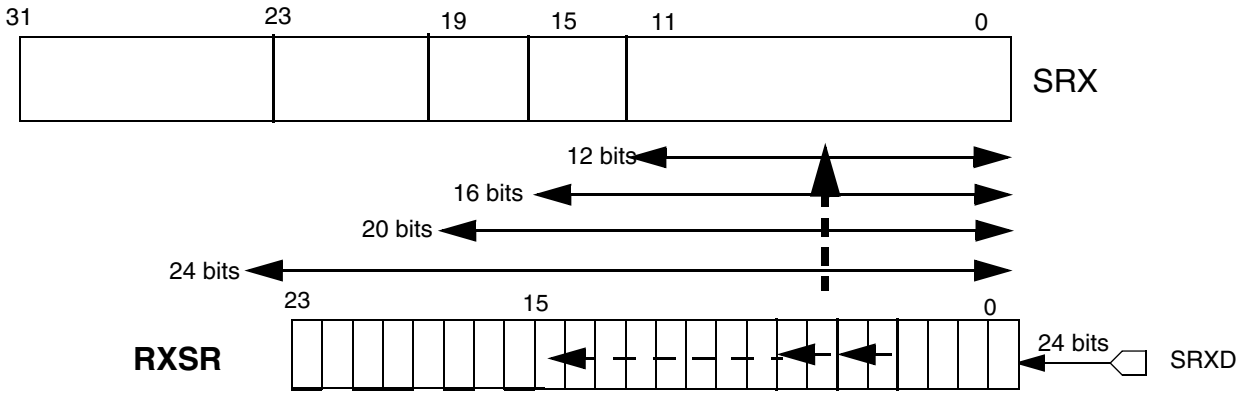


Figure 43-25. Receive Data Path (RXBIT0=1, RSHFD=0) (LSB Alignment)

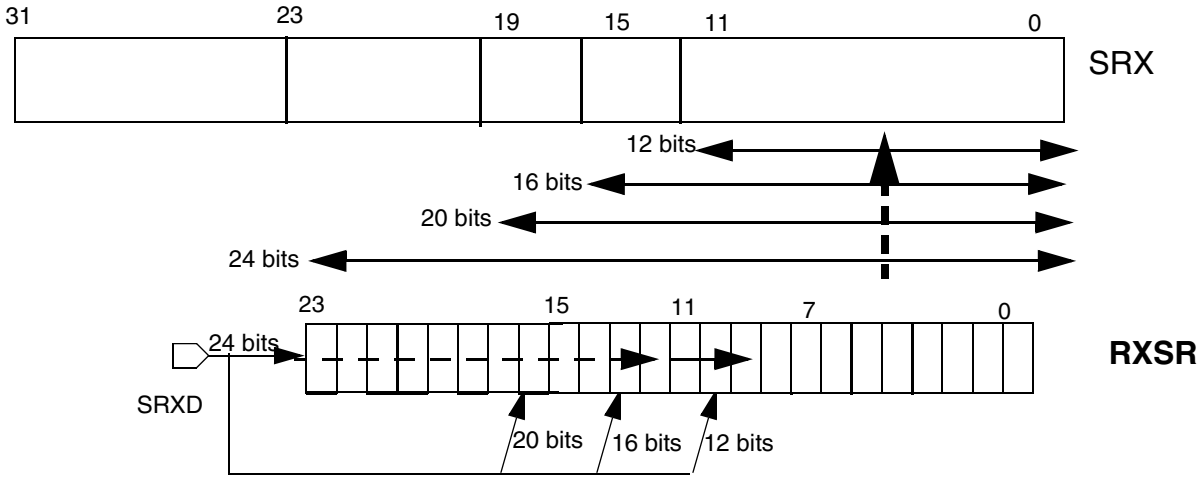


Figure 43-26. Receive Data Path (RXBIT0=1, RSHFD=1) (LSB Alignment)

43.3.3.7 SSI Control Register (SCR)

The SSI Control Register (SCR) is a 10-bit register used to set up the SSI. SSI reset is controlled by bit 0 in the SCR. SSI operating modes are also selected in this register (except AC97 mode which is selected in SACNT register).

See Figure 43-27 for illustration of valid bits in SSI Control Register and Table 43-11 for description of the bit fields in the register.

0xBASE_10 (SCR)
0xBASE_

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	SYNC_TX_FS	RFR_CLK_DIS	TFR_CLK_DIS	CLK_IST	TCH_EN	SYS_CLK_EN	I2S MODE[1:0]		SYN	NET	RE	TE	SS-IEN
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-27. SSI Control Register

Table 43-11. SSI Control Register Field Descriptions

Field	Description
31-13	Reserved
12 SYNC_TX_FS	SYNC_FS_TX bit provides a safe window for TE to be visible to the internal circuit which is just after FS occurrence. When SYNC_TX_FS is set, TE(SCR[1]) gets latched on FS occurrence & latched TE is used to enable/disable SSI transmitter. TE needs setup of 2 bit-clock cycles before occurrence of FS. If TE is changed within 2 bit-clock cycles of FS occurrence, there is high probability that TE will be latched on next FS. **Note:** With the TFR_CLK_DIS feature on, TE is used directly to enable transmitter in following cases (i) Sync mode & Rx disabled (ii) Async Mode. Latched-TE is used to disable the transmitter. This bit has no relevance in gated mode and AC97 mode. 0 TE not latched with FS occurrence & used directly for transmitter enable/disable. 1 TE latched with FS occurrence & latched-TE used for transmitter enable/disable.
11 RFR_CLK_DIS	Receive Frame Clock Disable. This bit provide option to keep the Frame-sync and Clock enabled or disabled after current receive frame, in which receiver is disabled by clearing RE bit. Writing to this bit has effect only when RE is disabled. 0 Continue Frame-sync/Clock generation after current frame during which RE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where receiver is already disabled in current or previous frames.

Table 43-11. SSI Control Register Field Descriptions (continued)

Field	Description
10 TFR_CLK_DIS	Transmit Frame Clock Disable. This bit provide option to keep the Frame-sync and Clock enabled or disabled after current transmit frame, in which transmitter is disabled by clearing TE bit. Writing to this bit has effect only when SSI is enabled TE is disabled. 0 Continue Frame-sync/Clock generation after current frame during which TE is cleared. This may be required when Frame-sync and Clocks are required from SSI, even when no data is to be received. 1 Stop Frame-sync/Clock generation at next frame boundary. This will be effective also in case where transmitter is already disabled in current or previous frames.
9 CLK_IST	Clock Idle State. This bit controls the idle state of the transmit clock port during SSI internal gated mode. Note: When Clock idle state is '1' the clock polarity should always be negedge triggered and when Clock idle = '0' the clock polarity should always be positive edge triggered. 0 Clock idle state is '0'. 1 Clock idle state is '1'.
8 TCH_EN	Two-Channel Operation Enable. This bit allows SSI to operate in the two-channel mode.In this mode while receiving, the RXSR transfers data to SRX0 and SRX1 alternately and while transmitting, data is alternately transferred from STX0 and STX1 to TXSR. For an even number of slots, Two-Channel Operation can be enabled to optimize usage of both FIFOs or disabled as in the case of odd number of active slots. This feature is especially useful in I2S mode, where data for Left Speaker can be placed in Tx-FIFO0 and for Right speaker in Tx-FIFO1. 0 Two-channel mode disabled. 1 Two-channel mode enabled.
7 SYS_CLK_EN	System Clock (Oversampling Clock) Enable. When set, this bit allows the SSI to output the (ccm_ssi_clk) at the SRCK port, provided that synchronous mode, and transmit internal clock mode are set. The relationship between bit clock and ccm_ssi_clk is determined by DIV2, PSR, and PM bits. This feature is especially useful in I2S Master mode to output oversampling clock on SRCK port. 0 ccm_ssi_clk not output on SRCK port. 1 ccm_ssi_clk output on SRCK port.
6–5 I2S MODE[1:0]	I2S Mode Select. These bits allow the SSI to operate in Normal, I2S Master or I2S Slave mode. See Section 43.1.2.4 for a detailed description of I2S Mode of operation. See Table 43-2 ("Mode Selection") for details regarding settings.
4 SYN	Synchronous Mode. This bit controls whether SSI is in synchronous mode or not. In synchronous mode, the transmit and receive sections of SSI share a common clock port (STCK) and frame sync port (STFS). 0 Asynchronous mode selected. 1 Synchronous mode selected.
3 NET	Network Mode. This bit controls whether SSI is in network mode or not. 0 Network mode not selected. 1 Network mode selected.
2 RE	Receive Enable. This control bit enables the receive section of the SSI. When this bit is enabled, data reception starts with the arrival of the next frame sync. If data is being received when this bit is cleared, data reception continues until the end of the current frame and then stops. If this bit is set again before the second to last bit of the last time slot in the current frame, then reception continues without interruption. RE should not be toggled in the same frame. 0 Receive section disabled. 1 Receive section enabled.

Table 43-11. SSI Control Register Field Descriptions (continued)

Field	Description
1 TE	Transmit Enable. This control bit enables the transmit section of the SSI. It enables the transfer of the contents of the STX registers to the TXSR and also enables the internal transmit clock. The transmit section is enabled when this bit is set and a frame boundary is detected. When this bit is cleared, the transmitter continues to send data until the end of the current frame and then stops. Data can be written to the STX registers with the TE bit cleared (the corresponding TDE bit will be cleared). If the TE bit is cleared and then set again before the second to last bit of the last time slot in the current frame, data transmission continues without interruption. The normal transmit enable sequence is to write data to the STX register(s) and then set the TE bit. The normal disable sequence is to clear the TE and TIE bits after the TDE bit is set. In gated clock mode, clearing the TE bit results in the clock stopping after the data currently in TXSR has shifted out. When the TE bit is set, the clock starts immediately (for internal gated clock mode). TE should not be toggled in the same frame. After enabling/disabling transmission, SSI expects 4 setup clock cycles before arrival of frame-sync for frame-sync to be accepted by SSI. In case of fewer clock cycles, there is high probability of the frame-sync to get missed. Note: If continuos clock is not provided, SSI expects 6 clock cycles before arrival of frame-sync for frame-sync to be accepted by SSI. 0 Transmit section disabled. 1 Transmit section enabled.
0 SSIEN	SSIEN — SSI Enable This bit is used to enable/disable the SSI. When disabled, all SSI status bits are preset to the same state produced by the power-on reset, all control bits are unaffected, the contents of Tx and Rx FIFOs are cleared. When SSI is disabled, all internal clocks are disabled (except register access clock). 0 SSI is disabled. 1 SSI is enabled.

43.3.3.8 SSI Interrupt Status Register (SISR)

The SSI Interrupt Status Register (SISR) is used to monitor the SSI. This register is used by the core to interrogate the status of the SSI. In gated mode of operation the TFS, RFS, TLS, RLS, TFRC and RFRC bits of AISR register are not generated. The status bits are described in the following table.

NOTE

- SSI Status flags are valid when SSI is enabled.
- See Section 43.4.3, “Receive Interrupt Enable Bit Description,” and Section 43.4.4, “Transmit Interrupt Enable Bit Description,” for interrupt source mapping.
- All the flags in the SISR are updated after the first bit of the next SSI word has completed transmission or reception. Certain status bits (ROE0/1 and TUE0/1) are cleared by writing 1 to the corresponding interrupt status bit in SISR.

See Figure 43-28 for illustration of valid bits in SSI Interrupt Register and Table 43-12 for description of the bit fields in the register.

0xBASE_14 (SISR) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	RFRC	TRFC	0	0	0	0	CMDAU	CMDDU	RXT
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1	RDR0	TDE1	TDE0	ROE1	ROE0	TUE1	TUE0	TFS	RFS	TLS	RLS	RFF1	RFF0	TFE1	TFE0
W																
RESET	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 43-28. SSI Interrupt Status Register

Table 43-12. SSI Interrupt Status Register Field Descriptions

Field	Description
31–25	Reserved
24 RFRC	Receive Frame Complete. This flag is set at the end of the frame during which Receiver is disabled. If Receive Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Receive Frame & Clock are disabled. See description of RFR_CLK_DIS (add cross reference) bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled. 0 End of Frame not reached 1 End of frame reached after disabling RE or disabling RFR_CLK_DIS, when receiver is already disabled.
23 TFRC	Transmit Frame Complete. This flag is set at the end of the frame during which Transmitter is disabled. If Transmit Frame & Clock are not disabled in the same frame, this flag is also set at the end of the frame in which Transmit Frame & Clock are disabled. See description of TFR_CLK_DIS (add cross reference) bit for more details on how to disable Receiver Frame & Clock or keep them enabled after receiver is disabled. 0 End of Frame not reached 1 End of frame reached after disabling TE or disabling TFR_CLK_DIS, when transmitter is already disabled.
22–19	Reserved.
18 CMDAU	Command Address Register Updated. This bit causes the Command Address Updated interrupt (when CMDAU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Address. This bit is cleared on reading the SACADD register. 0 No change in SACADD register. 1 SACADD register updated with different value.

Table 43-12. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
17 CMDDU	Command Data Register Updated. This bit causes the Command Data Updated interrupt (when CMDDU_EN bit is set). This status bit is set each time there is a difference in the previous and current value of the received Command Data. This bit is cleared on reading the SACDAT register. 0 No change in SACDAT register. 1 SACDAT register updated with different value.
16 RXT	Receive Tag Updated. This status bit is set each time there is a difference in the previous and current value of the received tag. It causes the Receive Tag Interrupt (if RXT_EN bit is set). This bit is cleared on reading the SATAG register. 0 No change in SATAG register. 1 SATAG register updated with different value.
15 RDR1	Receive Data Ready 1. This flag bit is set when SRX1 or Rx FIFO 1 is loaded with a new value and Two-Channel mode is selected. RDR1 is cleared when the Core reads the SRX1 register. If Rx FIFO 1 is enabled, RDR1 is cleared when the FIFO is empty. If RIE and RDR1_EN are set, a Receive Data 1 interrupt request is issued on setting of RDR1 bit in case Rx FIFO1 is disabled, if the FIFO is enabled, the interrupt is issued on RFF1 assertion. The RDR1 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
14 RDR0	Receive Data Ready 0. This flag bit is set when SRX0 or Rx FIFO 0 is loaded with a new value. RDR0 is cleared when the Core reads the SRX0 register. If Rx FIFO 0 is enabled, RDR0 is cleared when the FIFO is empty. If RIE and RDR0_EN are set, a Receive Data 0 interrupt request is issued on setting of RDR0 bit in case Rx FIFO0 is disabled, if the FIFO is enabled, the interrupt is issued on RFF0 assertion. The RDR0 bit is cleared by POR and SSI reset. 0 No new data for Core to read. 1 New data for Core to read.
13 TDE1	Transmit Data Register Empty 1. This flag is set whenever data is transferred to TXSR from STX1 register and Two-Channel mode is selected. If Tx FIFO1 is enabled, this occurs when there is at least one empty slot in STX1 or Tx FIFO1. If Tx FIFO1 is not enabled, this occurs when the contents of STX1 are transferred to TXSR. The TDE1 bit is cleared when the Core writes to STX1. If TIE and TDE1_EN are set, an SSI Transmit Data 1 interrupt request is issued on setting of TDE1 bit. The TDE1 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.
12 TDE0	Transmit Data Register Empty 0. This flag is set whenever data is transferred to TXSR from STX0 register. If Tx FIFO 0 is enabled, this occurs when there is at least one empty slot in STX0 or Tx FIFO 0. If Tx FIFO 0 is not enabled, this occurs when the contents of STX0 are transferred to TXSR. The TDE0 bit is cleared when the Core writes to STX0. If TIE and TDE0_EN are set, an SSI Transmit Data 0 interrupt request is issued on setting of TDE0 bit. The TDE0 bit is cleared by POR and SSI reset. 0 Data available for transmission. 1 Data needs to be written by the Core for transmission.

Table 43-12. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
11 ROE1	Receiver Overrun Error 1. This flag is set when the RXSR is filled and ready to transfer to SRX1 register or to Rx FIFO 1 (when enabled) and these are already full and Two-Channel mode is selected. If Rx FIFO 1 is enabled, this is indicated by RFF1 flag, else this is indicated by the RDR1 flag. The RXSR is not transferred in this case. The ROE1 flag causes an interrupt if RIE and ROE1_EN are set. The ROE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE1 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
10 ROE0	Receiver Overrun Error 0. This flag is set when the RXSR is filled and ready to transfer to SRX0 register or to Rx FIFO 0 (when enabled) and these are already full. If Rx FIFO 0 is enabled, this is indicated by RFF0 flag, else this is indicated by the RDR0 flag. The RXSR is not transferred in this case. The ROE0 flag causes an interrupt if RIE and ROE0_EN are set. The ROE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. Clearing the RE bit does not affect the ROE0 bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
9 TUE1	Transmitter Underrun Error 1. This flag is set when the TXSR is empty (no data to be transmitted), the TDE1 flag is set, a transmit time slot occurs and the SSI is in Two-Channel mode. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE1 flag causes an interrupt if TIE and TUE1_EN are set. The TUE1 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
8 TUE0	Transmitter Underrun Error 0. This flag is set when the TXSR is empty (no data to be transmitted), the TDE0 flag is set and a transmit time slot occurs. When a transmit underrun error occurs, the previous data is retransmitted. In Network mode, each time slot requires data transmission (unless masked through STMSK register), when the transmitter is enabled (TE is set). The TUE0 flag causes an interrupt if TIE and TUE0_EN are set. The TUE0 bit is cleared by POR and SSI reset. It is also cleared by writing '1' to this bit. 0 Default interrupt issued to the Core. 1 Exception interrupt issued to the Core.
7 TFS	Transmit Frame Sync. This flag indicates the occurrence of transmit frame sync. Data written to the STX registers during the time slot when the TFS flag is set, is sent during the second time slot (in Network mode) or in the next first time slot (in Normal mode). In Network mode, the TFS bit is set during transmission of the first time slot of the frame and is then cleared when starting transmission of the next time slot. In Normal mode, this bit is high for the first time slot. This flag causes an interrupt if TIE and TFS_EN are set. The TFS bit is cleared by POR and SSI reset. 0 No Occurrence of Transmit frame sync. 1 Transmit frame sync occurred during transmission of last word written to STX registers.
6 RFS	Receive Frame Sync. This flag indicates the occurrence of receive frame sync. In Network mode, the RFS bit is set when the first slot of the frame is being received. It is cleared when the next slot begins to be received. In Normal mode, this bit is always high. This flag causes an interrupt if RIE and RFS_EN are set. The RFS bit is cleared by POR and SSI reset. 0 No Occurrence of Receive frame sync. 1 Receive frame sync occurred during reception of next word in SRX registers.

Table 43-12. SSI Interrupt Status Register Field Descriptions (continued)

Field	Description
5 TLS	Transmit Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last time slot of the frame. TLS is set at the start of the last transmit time slot and causes the SSI to issue an interrupt (if TIE and TLS_EN are set). TLS is not generated when frame rate is 1 in normal mode of operation. TLS is cleared when the SISR is read with this bit set. The TLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last transmit time slot of frame.
4 RLS	Receive Last Time Slot. This flag indicates the last time slot in a frame. When set, it indicates that the current time slot is the last receive time slot of the frame. RLS is set at the end of the last time slot and causes the SSI to issue an interrupt (if RIE and RLS_EN are set). RLS is cleared when the SISR is read with this bit set. The RLS bit is cleared by POR and SSI reset. 0 Current time slot is not last time slot of frame. 1 Current time slot is the last receive time slot of frame.
3 RFF1	Receive FIFO Full 1. This flag is set when Rx FIFO1 is enabled, the data level in Rx FIFO1 reaches the selected Rx FIFO WaterMark 1 (RFWM1) threshold and the SSI is in Two-Channel mode. The setting of RFF1 only causes an interrupt when RIE and RFF1_EN are set, Rx FIFO1 is enabled and the Two-Channel mode is selected. RFF1 is automatically cleared when the amount of data in Rx FIFO1 falls below the threshold. The RFF1 bit is cleared by POR and SSI reset. When Rx FIFO1 contains 15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space available in Receive FIFO1. 1 Receive FIFO1 is full.
2 RFF0	Receive FIFO Full 0. This flag is set when Rx FIFO0 is enabled and the data level in Rx FIFO0 reaches the selected Rx FIFO WaterMark 0 (RFWM0) threshold. The setting of RFF0 only causes an interrupt when RIE and RFF0_EN are set and Rx FIFO0 is enabled. RFF0 is automatically cleared when the amount of data in Rx FIFO0 falls below the threshold. The RFF0 bit is cleared by POR and SSI reset. When Rx FIFO0 contains 15 words, the maximum it can hold, all further data received (for storage in this FIFO) is ignored until the FIFO contents are read. 0 Space available in Receive FIFO0. 1 Receive FIFO0 is full.
1 TFE1	Transmit FIFO Empty 1. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 1 (TFWM1) threshold and the Two-Channel mode is selected. The setting of TFE1 only causes an interrupt when TIE and TFE1_EN are set, Tx FIFO1 is enabled and Two-Channel mode is selected. The TFE1 bit is automatically cleared when the data level in Tx FIFO1 becomes more than the amount specified by the watermark bits. The TFE1 bit is set by POR and SSI reset. 0 Transmit FIFO1 has data for transmission. 1 Transmit FIFO1 is empty.
0 TFE0	Transmit FIFO Empty 0. This flag is set when the empty slots in Tx FIFO exceed or are equal to the selected Tx FIFO WaterMark 0 (TFWM0) threshold. The setting of TFE0 only causes an interrupt when TIE and TFE0_EN are set and Tx FIFO0 is enabled. The TFE0 bit is automatically cleared when the data level in Tx FIFO0 becomes more than the amount specified by the watermark bits. The TFE0 bit is set by POR and SSI reset. 0 Transmit FIFO0 has data for transmission. 1 Transmit FIFO0 is empty.

43.3.3.9 SSI Interrupt Enable Register (SIER)

The SSI Interrupt Enable Register (SIER) is a 25-bit register used to set up the SSI interrupts and DMA requests. See Figure 43-29 for illustration of valid bits in SSI Interrupt Enable Register and Table 43-13 for description of the bit fields in the register.

0xBASE_18 (SIER) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	RFRC_EN	TFRC_EN	RDMAE	RIE	TDMAE	TIE	CMDAU_EN	CMDDU_EN	RXT_EN
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDR1_EN	RDR0_EN	TDE1_EN	TDE0_EN	ROE1_EN	ROE0_EN	TUE1_EN	TUE0_EN	TFS_EN	RFS_EN	TLS_EN	RLS_EN	RFF1_EN	RFF0_EN	TFE1_EN	TFE0_EN
W																
RESET	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1

Figure 43-29. SSI Interrupt Enable Register

Table 43-13. SSI Interrupt Enable Register Field Descriptions

Field	Description
31–25	Reserved
24–23 Enable Bits	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.
22 RDMAE	Receive DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the RFF0/1 bits in the SISR are set and if the corresponding RFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding RDR bit is set. 0 SSI Receiver DMA requests disabled. 1 SSI Receiver DMA requests enabled.
21 RIE	Receive Interrupt Enable. This control bit allows the SSI to issue receiver related interrupts to the Core. See Section 43.4.3 for a detailed description of this bit. 0 SSI Receiver Interrupt requests disabled. 1 SSI Receiver Interrupt requests enabled.
20 TDMAE	Transmit DMA Enable. This bit allows SSI to request for DMA transfers. When enabled, DMA requests are generated when any of the TFE0/1 bits in the SISR are set and if the corresponding TFEN bit is also set. If the corresponding FIFO is disabled, a DMA request is generated when the corresponding TDE bit is set. 0 SSI Transmitter DMA requests disabled. 1 SSI Transmitter DMA requests enabled.

Table 43-13. SSI Interrupt Enable Register Field Descriptions (continued)

Field	Description
19 TIE	Transmit Interrupt Enable. This control bit allows the SSI to issue transmitter data related interrupts to the Core. See Section 43.4.4 for a detailed description of this bit. 0 SSI Transmitter Interrupt requests disabled. 1 SSI Transmitter Interrupt requests enabled.
18–0 Enable Bits	Enable Bit. Each bit controls whether the corresponding status bit in SISR can issue an interrupt to the Core or not. 0 Corresponding status bit cannot issue interrupt. 1 Corresponding status bit can issue interrupt.

43.3.3.10 SSI Transmit Configuration Register (STCR)

The SSI Transmit Configuration Register (STCR) is a read/write control registers used to direct the transmit operation of the SSI. STCR controls the direction of the bit clock and frame sync ports, STCK and STFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all STCR bits. However, SSI reset does not affect the STCR bits. The STCR bits are described in the following paragraphs. See Table 43-6 for the programming model of the SSI. The SSI Control Register (SCR) must first be set to enable interrupts. Next, the SSI interrupt bit in the Interrupt Enable Register (SIER) must be set to enable the interrupt. Finally, the interrupt can be enabled from within the SSI.

See Figure 43-30 for illustration of valid bits in SSI Transmit Configuration Register and Table 43-14 for description of the bit fields in the register.

0xBASE_1C (STCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	TXBIT0	TFEN1	TFEN0	TFDIR	TXDIR	TSHFD	TSCKP	TFSI	TFSL	TEFS
W																
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 43-30. SSI Transmit Configuration Register

Table 43-14. SSI Transmit Configuration Register Field Descriptions

Field	Description
31–10	Reserved
9 TXBIT0	Transmit Bit 0. This control bit allows SSI to transmit the data word from bit position 0 or 15/31 in the transmit shift register. The shifting data direction can be MSB or LSB first, controlled by the TSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of transmit shift register (MSB aligned). 1 Shifting with respect to bit 0 of transmit shift register (LSB aligned).
8 TFEN1	Transmit FIFO Enable 1. This bit enables transmit FIFO 1. When enabled, the FIFO allows 15 samples to be transmitted by the SSI (per channel) (a 9th sample can be shifting out) before TDE1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 1 disabled. 1 Transmit FIFO 1 enabled.
7 TFEN0	Transmit FIFO Enable 0. This bit enables transmit FIFO 0. When enabled, the FIFO allows 15 samples to be transmitted by the SSI per channel (a 9th sample can be shifting out) before TDE0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is transferred to the transmit shift register (provided the interrupt is enabled). 0 Transmit FIFO 0 disabled. 1 Transmit FIFO 0 enabled.
6 TFDIR	Transmit Frame Direction. This bit controls the direction and source of the transmit frame sync signal. Internally generated frame sync signal is sent out through the STFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 TXDIR	Transmit Clock Direction. This bit controls the direction and source of the clock signal used to clock the TXSR. Internally generated clock is output through the STCK port. External clock is taken from this port. See Table 43-5 for details of clock pin configurations 0 Transmit Clock is external. 1 Transmit Clock generated internally.
4 TSHFD	Transmit Shift Direction. This bit controls whether the MSB or LSB will be transmitted first in a sample. **Note:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (TSHFD cleared). 0 Data transmitted MSB first. 1 Data transmitted LSB first.
3 TSCKP	Transmit Clock Polarity. This bit controls which bit clock edge is used to clock out data for the transmit section. 0 Data clocked out on rising edge of bit clock. 1 Data clocked out on falling edge of bit clock. Note: TSCKP is 0 CLK_IST = 0; TSCKP is 1CLK_IST = 1
2 TFSI	Transmit Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the transmit section of SSI. 0 Transmit frame sync is active high. 1 Transmit frame sync is active low.

Table 43-14. SSI Transmit Configuration Register Field Descriptions (continued)

Field	Description
1 TFSL	Transmit Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the transmit section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Transmit frame sync is one-word long. 1 Transmit frame sync is one-clock-bit long.
0 TEFS	Transmit Early Frame Sync. This bit controls when the frame sync is initiated for the transmit section. The frame sync signal is deasserted after one bit-for-bit length frame sync and after one word-for-word length frame sync. In case of synchronous operation, the frame sync can also be initiated on receiving the first bit of data. 0 Transmit frame sync initiated as the first bit of data is transmitted. 1 Transmit frame sync is initiated one bit before the data is transmitted.

43.3.3.11 SSI Receive Configuration Register (SRCR)

The SSI Receive Configuration Register (SRCR) is a read/write control registers used to direct the receive operation of the SSI. SRCR controls the direction of the bit clock and frame sync ports, SRCK and SRFS. Interrupt enable bit for the transmit sections is provided in this control register. The Power-on reset clears all SRCR bits. However, SSI reset does not affect the SRCR bits. See Figure 43-31 for illustration of valid bits in SSI Receive Configuration Register and Table 43-15 for description of the bit fields in the register.

0xBASE_20 (SRCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	RXEXT	RXBIT0	RFEN1	RFEN0	RFDIR	RXDIR	RSHFD	RSCKP	RFSI	RFSL	REFS
W																
RESET	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 43-31. SSI Receive Configuration Register

Table 43-15. SSI Receive Configuration Register Field Descriptions

Field	Description
31–11	Reserved
10 RXEXT	Receive Data Extension. This control bit allows SSI to store the received data word in sign extended form. This bit affects data storage only in case received data is LSB aligned (SRCR[9]=1) 0 Sign extension turned off. 1 Sign extension turned on.

Table 43-15. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
9 RXBIT0	Receive Bit 0. This control bit allows SSI to receive the data word at bit position 0 or 15/31 in the receive shift register. The shifting data direction can be MSB or LSB first, controlled by the RSHFD bit. 0 Shifting with respect to bit 31 (if word length = 16, 18, 20, 22 or 24) or bit 15 (if word length = 8, 10 or 12) of receive shift register (MSB aligned). 1 Shifting with respect to bit 0 of receive shift register (LSB aligned).
8 RFEN1	Receive FIFO Enable 1. This bit enables receive FIFO 1. When enabled, the FIFO allows 15 samples to be received by the SSI per channel (a 16th sample can be shifting in) before RDR1 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 1 disabled. 1 Receive FIFO 1 enabled.
7 RFEN0	Receive FIFO Enable 0. This bit enables receive FIFO 0. When enabled, the FIFO allows 15 samples to be received by the SSI (per channel) (a 16th sample can be shifting in) before RDR0 bit is set. When the FIFO is disabled, an interrupt is generated when a single sample is received by the SSI (provided the interrupt is enabled). 0 Receive FIFO 0 disabled. 1 Receive FIFO 0 enabled.
6 RFDIR	Receive Frame Direction. This bit controls the direction and source of the receive frame sync signal. Internally generated frame sync signal is sent out through the SRFS port and external frame sync is taken from the same port. 0 Frame Sync is external. 1 Frame Sync generated internally.
5 RXDIR	Receive Clock Direction. This bit controls the direction and source of the clock signal used to clock the RXSR. Internally generated clock is output through the SRCK port. External clock is taken from this port. See Table 43-5 for details on clock pin configurations. 0 Receive Clock is external. 1 Receive Clock generated internally.
4 RSHFD	Receive Shift Direction. This bit controls whether the MSB or LSB will be received first in a sample. **Note:** The CODEC device labels the MSB as bit 0, whereas the Core labels the LSB as bit 0. Therefore, when using a standard CODEC, Core MSB (CODEC LSB) is shifted in first (RSHFD cleared). 0 Data received MSB first. 1 Data received LSB first.
3 RSCKP	Receive Clock Polarity. This bit controls which bit clock edge is used to latch in data for the receive section. 0 Data latched on rising edge of bit clock. 1 Data latched on falling edge of bit clock.
2 RFSI	Receive Frame Sync Invert. This bit controls the active state of the frame sync I/O signal for the receive section of SSI. 0 Receive frame sync is active high. 1 Receive frame sync is active low.

Table 43-15. SSI Receive Configuration Register Field Descriptions (continued)

Field	Description
1 RFSL	Receive Frame Sync Length. This bit controls the length of the frame sync signal to be generated or recognized for the receive section. The length of a word-long frame sync is same as the length of the data word selected by WL[3:0]. 0 Receive frame sync is one-word long. 1 Receive frame sync is one-clock-bit long.
0 REFS	Receive Early Frame Sync. This bit controls when the frame sync is initiated for the receive section. The frame sync is disabled after one bit-for-bit length frame sync and after one word-for-word length frame sync. 0 Receive frame sync initiated as the first bit of data is received. 1 Receive frame sync is initiated one bit before the data is received.

43.3.3.12 SSI Transmit and Receive Clock Control Registers (STCCR & SRCCR)

The SSI Transmit and Receive Control (STCCR and SRCCR) registers are 19-bit, read/write control registers used to direct the operation of the SSI. The Clock and Reset Module (CRM) can source the SSI clock (ccm_ssi_clk) from multiple sources and perform fractional division to support commonly used audio bit rates. The CRM can maintain the ccm_ssi_clk frequency at a constant rate even in cases where the ipg_clk frequency changes. These registers control the SSI clock generator, bit and frame sync rates, word length, and number of words per frame for the serial data. The STCCR register is dedicated to the transmit section, and the SRCCR register is dedicated to the receive section except in Synchronous mode, in which the STCCR register controls both the receive and transmit sections. Power-on reset clears all STCCR and SRCCR bits. SSI reset does not affect the STCCR and SRCCR bits. The control bits are described in the following paragraphs. Although the bit patterns of the STCCR and SRCCR registers are the same, the contents of these two registers can be programmed differently.

See Figure 43-32 for illustration of valid bits in SSI Transmit Clock Control Register and Table 43-16 for description of the bit fields for both SSI Transmit and Receive Clock Control registers.

0xBASE_24 (STCCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-32. SSI Transmit Clock Control Register

See Figure 43-33 for illustration of valid bits in SSI Receive Clock Control Register and Table 43-16 for description of the bit fields for both SSI Transmit and Receive Clock Control registers.

0xBASE_28 (SRCCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV2	PSR	WL3
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WL2	WL1	WL0	DC4	DC3	DC2	DC1	DC0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-33. SSI Receive Clock Control Register

Table 43-16. SSI Transmit and Receive Clock Control Register Field Descriptions

Field	Description
31–19	Reserved
18 DIV2	Divide By 2. This bit controls a divide-by-two divider in series with the rest of the prescalers. 0 Divider bypassed. 1 Divider used to divide clock by 2.
17 PSR	Prescaler Range. This bit controls a fixed divide-by-eight prescaler in series with the variable prescaler. It extends the range of the prescaler for those cases where a slower bit clock is required. 0 Prescaler bypassed. 1 Prescaler used to divide clock by 8.
16–13 WL3–WL0	Word Length Control. These bits are used to control the length of the data words being transferred by the SSI. These bits control the Word Length Divider in the Clock Generator. They also control the frame sync pulse length when the FSL bit is cleared. In I2S Master mode, the SSI works with a fixed word length of 32, and the WL bits are used to control the amount of valid data in those 32 bits. See Table 43-17 for details of data word lengths supported by SSI. In AC97 Mode of operation, if word length is set to any value other than 16 bits, it will result in a word length of 20 bits.

Table 43-16. SSI Transmit and Receive Clock Control Register Field Descriptions (continued)

Field	Description
12–8 DC4–DC0	Frame Rate Divider Control. These bits are used to control the divide ratio for the programmable frame rate dividers. The divide ratio works on the word clock. In Normal mode, this ratio determines the word transfer rate. In Network mode, this ratio sets the number of words per frame. The divide ratio ranges from 1 to 32 in Normal mode and from 2 to 32 in Network mode. In Normal mode, a divide ratio of 1 (DC=00000) provides continuous periodic data word transfer. A bit-length frame sync must be used in this case. These bits can be programmed with values ranging from “00000” to “11111” to control the number of words in a frame.
7–0 PM7–PM0	Prescaler Modulus Select. These bits control the prescale divider in the clock generator. This prescaler is used only in Internal Clock mode to divide the internal clock (ccm_ssi_clk). The bit clock output is available at the clock port. A divide ratio from 1 to 256 (PM[7:0] = 0x00 to 0xFF) can be selected. See Section 43.4.2.2 for details regarding settings.

Table 43-17. SSI Data Length

WL3	WL2	WL1	WL0	Number of Bits/Word	Supported in Implementation
0	0	0	0	2	No
0	0	0	1	4	No
0	0	1	0	6	No
0	0	1	1	8	Yes
0	1	0	0	10	Yes
0	1	0	1	12	Yes
0	1	1	0	14	No
0	1	1	1	16	Yes
1	0	0	0	18	Yes
1	0	0	1	20	Yes
1	0	1	0	22	Yes
1	0	1	1	24	Yes
1	1	0	0	26	No
1	1	0	1	28	No
1	1	1	0	30	No
1	1	1	1	32	No

43.3.3.13 SSI FIFO Control/Status Register (SFCSR)

See Figure 43-34 for illustration of valid bits in SSI FIFO Control/Status Register and Table 43-18 for description of the bit fields in the register.

0xBASE_2C (SFCSR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RFCNT1[3:0]				TFCNT1[3:0]				RFWM1[3:0]				TFWM1[3:0]			
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RFCNT0[3:0]				TFCNT0[3:0]				RFWM0[3:0]				TFWM0[3:0]			
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Figure 43-34. SSI FIFO Control/Status Register

Table 43-18. SSI FIFO Control/Status Register Field Descriptions

Field	Description
31–28 RFCNT1[3:0]	Receive FIFO Counter1. These bits indicate the number of data words in Receive FIFO 1. See Table 43-19 for details regarding settings for receive FIFO counter bits.
27–24 TFCNT1[3:0]	Transmit FIFO Counter1. These bits indicate the number of data words in Transmit FIFO. See Table 43-20 for details regarding settings for transmit FIFO counter bits.
23–20 RFWM1[3:0]	Receive FIFO Full WaterMark 1. These bits control the threshold at which the RFF1 flag will be set. The RFF1 flag is set whenever the data level in Rx FIFO 1 reaches the selected threshold. See Table 43-21 for details regarding settings for receive FIFO watermark bits.
19–16 TFWM1[3:0]	Transmit FIFO Empty WaterMark 1. These bits control the threshold at which the TFE1 flag will be set. The TFE1 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. See Table 43-22 for details regarding settings for transmit FIFO watermark bits.
15–12 RFCNT0[3:0]	Receive FIFO Counter 0. These bits indicate the number of data words in Receive FIFO 0. See Table 43-19 for details regarding settings for receive FIFO counter bits.
11–8 TFCNT0[3:0]	Transmit FIFO Counter 0. These bits indicate the number of data words in Transmit FIFO 0. See Table 43-20 for details regarding settings for transmit FIFO counter bits.

Table 43-18. SSI FIFO Control/Status Register Field Descriptions (continued)

Field	Description
7–4 RFWM0[3:0]	Receive FIFO Full WaterMark 0. These bits control the threshold at which the RFF0 flag will be set. The RFF0 flag is set whenever the data level in Rx FIFO 0 reaches the selected threshold. See Table 43-21 for details regarding settings for receive FIFO watermark bits.
3–0 TFWM0[3:0]	Transmit FIFO Empty WaterMark 0. These bits control the threshold at which the TFE0 flag will be set. The TFE0 flag is set whenever the empty slots in Tx FIFO exceed or are equal to the selected threshold. See Table 43-22 for details regarding settings for transmit FIFO watermark bits.

Table 43-19. Receive FIFO Counter Bit Description

Bits	Description
0000	0 data word in receive FIFO
0001	1 data word in receive FIFO
0010	2 data word in receive FIFO
0011	3 data word in receive FIFO
0100	4 data word in receive FIFO
0101	5 data word in receive FIFO
0110	6 data word in receive FIFO
0111	7 data word in receive FIFO
1000	8 data word in receive FIFO
1001	9 data word in receive FIFO
1010	10 data word in receive FIFO
1011	11 data word in receive FIFO
1100	12 data word in receive FIFO
1101	13 data word in receive FIFO
1110	14 data word in receive FIFO
1111	15 data word in receive FIFO

Table 43-20. Transmit FIFO Counter Bit Description

Bits	Description
0000	0 data word in transmit FIFO
0001	1 data word in transmit FIFO
0010	2 data word in transmit FIFO
0011	3 data word in transmit FIFO
0100	4 data word in transmit FIFO
0101	5 data word in transmit FIFO
0110	6 data word in transmit FIFO
0111	7 data word in transmit FIFO
1000	8 data word in transmit FIFO
1001	9 data word in transmit FIFO
1010	10 data word in transmit FIFO
1011	11 data word in transmit FIFO
1100	12 data word in transmit FIFO
1101	13 data word in transmit FIFO
1110	14 data word in transmit FIFO
1111	15 data word in transmit FIFO

Table 43-21. Receive FIFO WaterMark Bit Description

Bits	Description
0000	Reserved
0001	RFF set when at least one data word have been written to the Receive FIFO Set when RxFIFO = 1,2.....15 data words
0010	RFF set when more than or equal to 2 data word have been written to the Receive FIFO. Set when RxFIFO = 2,3.....15 data words
0011	RFF set when more than or equal to 3 data word have been written to the Receive FIFO. Set when RxFIFO = 3,4.....15 data words
0100	RFF set when more than or equal to 4 data word have been written to the Receive FIFO. Set when RxFIFO = 4,5.....15 data words
0101	RFF set when more than or equal to 5 data word have been written to the Receive FIFO. Set when RxFIFO = 5,6.....15 data words
0110	RFF set when more than or equal to 6 data word have been written to the Receive. Set when RxFIFO = 6,7.....15 data words
0111	RFF set when more than or equal to 7 data word have been written to the Receive FIFO. Set when RxFIFO = 7,8.....15 data words
1000	RFF set when more than or equal to 8 data word have been written to the Receive FIFO. Set when RxFIFO = 8,9.....15 data words
1001	RFF set when more than or equal to 9 data word have been written to the Receive FIFO. Set when RxFIFO = 9,10.....15 data words
1010	RFF set when more than or equal to 10 data word have been written to the Receive FIFO. Set when RxFIFO = 10,11.....15 data words
1011	RFF set when more than or equal to 11 data word have been written to the Receive FIFO. Set when RxFIFO = 11,12.....15 data words
1100	RFF set when more than or equal to 12 data word have been written to the Receive FIFO. Set when RxFIFO = 12,13.....15 data words
1101	RFF set when more than or equal to 13 data word have been written to the Receive FIFO. Set when RxFIFO = 13,14,15data words
1110	RFF set when more than or equal to 14 data word have been written to the Receive FIFO. Set when RxFIFO = 14,15 data words
1111	RFF set when 15 data word have been written to the Receive FIFO (default). Set when RxFIFO = 15 data words

Table 43-22. Transmit FIFO WaterMark Bit Description

Bits	Description
0000	Reserved
0001	TFE set when there are more than or equal to 1 empty slots in Transmit FIFO. (default) Transmit FIFO empty is set when TxFIFO <= 14 data.
0010	TFE set when there are more than or equal to 2 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 13 data.
0011	TFE set when there are more than or equal to 3 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 12 data.
0100	TFE set when there are more than or equal to 4 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 11 data.
0101	TFE set when there are more than or equal to 5 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 10 data.
0110	TFE set when there are more than or equal to 6 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 9 data.
0111	TFE set when there are more than or equal to 7 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 8 data.
1000	TFE set when there are more than or equal to 8 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 7 data.
1001	TFE set when there are more than or equal to 9 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 6 data.
1010	TFE set when there are more than or equal to 10 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 5 data.
1011	TFE set when there are more than or equal to 11 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 4 data.
1100	TFE set when there are more than or equal to 12 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 3 data.
1101	TFE set when there are more than or equal to 13 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 2 data.
1110	TFE set when there are more than or equal to 14 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO <= 1 data.
1111	TFE set when there are 15 empty slots in Transmit FIFO. Transmit FIFO empty is set when TxFIFO = 0 data.

Table 43-23 indicates the status of the Transmit FIFO Empty flag, with different settings of the Transmit FIFO WaterMark bits and varying amounts of data in the Tx FIFO.

Table 43-23. Status of Transmit FIFO Empty Flag

Transmit FIFO Watermark (TFWM)	Number of data in Tx-Fifo														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

43.3.3.14 SSI Test Register (STR)

NOTE

SSI Test Register is designed for debugging purpose only and is not intended for general user access.

See Figure 43-35 for illustration of valid bits in SSI Test Register and Table 43-24 for description of the bit fields for the register.

0xBASE_30 (STR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST	RCK2 TCK	RFS2 TFS	RXSTATE[4:0]					TXD2 RXD	TCK2 RCK	TFS2 RFS	TXSTATE[4:0]				
RESET	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 43-35. SSI Test Register

Table 43-24. SSI Test Register Field Descriptions

Field	Description
31–16	Reserved
15 TEST	Test Mode. This bit enables the test features of SSI. When set, the RXSTATE and TXSTATE bit values get transferred to the state machine. When in test mode, the user can read the contents of a specific FIFO by writing an appropriate value to the RFCNT0/1 or TFCNT 0/1 bits (in SFCSR). 0 No effect on SSI operation. 1 SSI in Test Mode.
14 RCK2TCK	Receive Clock to Transmit Clock Loop Back. This bit connects SRCK (used as output) to STCK (used as input). 0 No effect on SSI operation. 1 SRCK to STCK loop back enabled.
13 RFS2TFS	Receive Frame to Transmit Frame Loop Back. This bit connects SRFS (used as output) to STFS (used as input). 0 No effect on SSI operation. 1 SRFS to STFS loop back enabled.
12–8 RXSTATE[4:0]	Receiver State Machine Status. These bits indicate the current status of the Receive State Machine.
7 TXD2RXD	Transmit Data to Receive Data Loop Back. This bit connects STXD to SRXD. 0 No effect on SSI operation. 1 STXD to SRXD loop back enabled.
6 TCK2RCK	Transmit Clock to Receive Clock Loop Back. This bit connects STCK (used as output) to SRCK (used as input). 0 No effect on SSI operation. 1 STCK to SRCK loop back enabled.

Table 43-24. SSI Test Register Field Descriptions (continued)

Field	Description
5 TFS2RFS	Transmit Frame to Receive Frame Loop Back. This bit connects STFS (used as output) to SRFS (used as input). 0 No effect on SSI operation. 1 STFS to SRFS loop back enabled.
4–0 TXSTATE	Transmitter State Machine Status. These bits indicate the current status of the Transmit State Machine.

43.3.3.15 SSI Option Register (SOR)

NOTE

SSI Option Register is designed for debugging purpose only and is not intended for general user access.

See Figure 43-36 for illustration of valid bits in SSI Option Register and Table 43-25 for description of the bit fields for the register.

0xBASE_34 (SOR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	CLKOFF	RX_CLR	TX_CLR	INIT	WAIT[1:0]		SYNRST
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-36. SSI Option Register

Table 43-25. SSI Option Register Field Descriptions

Field	Description
31–7	Reserved
6 CLKOFF	Clock Off. This bit is used to turn off the ipg_clk to further reduce power consumption. 0 No effect on SSI operation. 1 Turn off ipg_clk.
5 RX_CLR	Receiver Clear. This bit flushes the contents of Rx FIFOs. It is always read as cleared (‘0’). 0 No effect on SSI operation. 1 Flush Rx FIFOs.

Table 43-25. SSI Option Register Field Descriptions (continued)

Field	Description
4 TX_CLR	Transmitter Clear This bit flushes the contents of Tx FIFOs. It is always read as cleared ('0'). 0 No effect on SSI operation. 1 Flush Tx FIFOs. **It is recommended that we flush Tx-FIFOs after re-enabling Transmission.
3 INIT	Initialize. The setting of this bit causes the SSI state machine to reset. 0 No effect on SSI operation. 1 Initialize SSI state machine.
2–1 WAIT[1:0]	Wait. These bits control the number wait states to be added to all transactions between the Core and SSI. The value of these bits ranges from '00' (no wait states) to '11' (three wait states).
0 SYNRST	Frame Sync Reset. This bit automatically resets the accumulation of data in Receive Data Registers (SRX0/1) and Receive FIFOs (RXFIFO 0/1) on the next frame synchronization. 0 Data accumulation not affected. 1 Reset data accumulation on Frame Synchronization.

43.3.3.16 SSI AC97 Control Register (SACNT)

See Figure 43-37 for illustration of valid bits in SSI AC97 Control Register and Table 43-26 for description of the bit fields for the register.

0xBASE_38 (SACNT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FRDIV[5:0]						WR	RD	TIF	FV	AC97EN
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-37. SSI AC97 Control Register

Table 43-26. SSI AC97 Control Register Field Descriptions

Field	Description
31–11	Reserved
10–5 FRDIV[5:0]	Frame Rate Divider. These bits control the frequency of AC97 data transmission/reception. They are programmed with the number of frames for which the SSI should be idle, after operating in one frame. Through these bits, AC97 frequency of operation, from 48 KHz (000000) to 1 KHz (101111) can be achieved. Sample Value: 001010 (10 Decimal) = SSI will operate once every 11 frames.

Table 43-26. SSI AC97 Control Register Field Descriptions (continued)

Field	Description
4 WR	Write Command. This bit specifies whether the next frame will carry an AC97 Write Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bits (corresponding to Command Address and Command Data slots of the next Tx frame) are automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Write Command. 1 Next frame will have a Write Command.
3 RD	Read Command. This bit specifies whether the next frame will carry an AC97 Read Command or not. The programmer should take care that only one of the bits (WR or RD) is set at a time. When this bit is set, the corresponding tag bit (corresponding to Command Address slot of the next Tx frame) is automatically set. This bit is automatically cleared by the SSI after completing transmission of a frame. 0 Next frame will not have a Read Command. 1 Next frame will have a Read Command.
2 TIF	Tag in FIFO. This bit controls the destination of the information received in AC97 tag slot (Slot #0). 0 Tag info stored in SATAG register. 1 Tag info stored in Rx FIFO 0.
1 FV	Fixed/Variable Operation. This bit selects whether the SSI is in AC97 Fixed mode or AC97 Variable mode. 0 AC97 Fixed Mode. 1 AC97 Variable Mode.
0 AC97EN	AC97 Mode Enable. This bit is used to enable SSI AC97 operation. See Section 43.1.2.5 for details of AC97 operation. 0 AC97 mode disabled. 1 SSI in AC97 mode.

43.3.3.17 SSI AC97 Command Address Register (SACADD)

See Figure 43-38 for illustration of valid bits in SSI AC97 Command Address Register and Table 43-27 for description of the bit fields for the register.

0xBASE_3C (SACADD) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SACADD[18:16]		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACADD[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-38. SSI AC97 Command Address Register

Table 43-27. SSI AC97 Command Address Register Field Descriptions

Field	Description
31–19	Reserved
18–0 SACADD	AC97 Command Address. These bits store the Command Address Slot information (bit 19 of the slot is sent in accordance with the Read and Write Command bits in SACNT register). These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Address Slot. If the contents of these bits change due to an update, the CMDAU bit in SISR is set.

43.3.3.18 SSI AC97 Command Data Register (SACDAT)

See Figure 43-39 for illustration of valid bits in SSI AC97 Command Data Register and Table 43-28 for description of the bit fields for the register.

0xBASE_40 (SACDAT) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	SACDAT[19:16]			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SACDAT[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-39. SSI AC97 Command Data Register

Table 43-28. SSI AC97 Command Data Register

Field	Description
31–20	Reserved
19–0 SACDAT	AC97 Command Data. The outgoing Command Data Slot carries the information contained in these bits. These bits can be updated by a direct write from the Core. They are also updated with the information received in the incoming Command Data Slot. If the contents of these bits change due to an update, the CMDDU bit in SISR is set. These bits are transmitted only during AC97 Write Command. During AC97 Read Command, 0x00000 is transmitted in time slot #2.

43.3.3.19 SSI AC97 Tag Register (SATAG)

See Figure 43-40 for illustration of valid bits in SSI AC97 Tag Register and Table 43-29 for description of the bit fields for the register.

0xBASE_44 (SATAG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SATAG[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-40. SSI AC97 Tag Register

Table 43-29. SSI AC97 Tag Register Field Descriptions

Field	Description
31–16	Reserved
15–0 SATAG	AC97 Tag Value. Writing to this register (by the Core) sets the value of the Tx-Tag in AC97 fixed mode of operation. On a read, the Core gets the Rx-Tag Value received (in the last frame) from the Codec. If TIF bit in SACNT register is set, the TAG value is also stored in Rx-FIFO in addition to SATAG register. When the received Tag value changes, the RXT bit in SISR register is set. Bits SATAG[1:0] convey the Codec -ID. In current implementation only Primary Codecs are supported. Thus writing value 2'b00 to this field is mandatory.

43.3.3.20 SSI Transmit Time Slot Mask Register (STMSK)

See Figure 43-41 for illustration of valid bits in SSI Transmit Time Slot Register and Table 43-30 for description of the bit fields for the register.

0xBASE_48 (STMSK) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STMSK[31:16]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STMSK[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-41. SSI Transmit Time Slot Mask Register

Table 43-30. SSI Transmit Time Slot Mask Register Field Descriptions

Field	Description
31–0 STMSK	Transmit Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI transmits data. Each bit has info corresponding to the respective time slot in the frame. Transmit mask bits should not be used in I2S Slave mode of operation. STMSK register value must be set before enabling Transmission. 0 Valid Time Slot. 1 Time Slot masked (no data transmitted in this time slot).

43.3.3.21 SSI Receive Time Slot Mask Register (SRMSK)

See Figure 43-42 for illustration of valid bits in SSI Receive Time Slot Mask Register and Table 43-31 for description of the bit fields for the register.

0xBASE_4C (SRMSK) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SRMSK[31:16]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SRMSK[15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-42. SSI Receive Time Slot Mask Register

Table 43-31. SSI Receive Time Slot Mask Register Field Descriptions

Field	Description
31–0 SRMSK	Receive Mask. These bits indicate which slot has been masked in the current frame. The Core can write to this register to control the time slots in which the SSI receives data. Each bit has info corresponding to the respective time slot in the frame. SRMSK register value must be set before enabling Receiver.Receive mask bits should not be used in I2S Slave mode of operation. 0 Valid Time Slot. 1 Time Slot masked (no data received in this time slot).

43.3.3.22 SSI AC97 Channel Status Register (SACCST)

See Figure 43-43 for illustration of valid bits in SSI AC97 Channel Status Register and Table 43-32 for description of the bit fields for the register.

0xBASE_50 (SACCST) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SACCST[9:0]									
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-43. SSI AC97 Channel Status Register

Table 43-32. SSI AC97 Channel Status Register Field Descriptions

Field	Description
31–10	Reserved.
9–0 SACCST	AC97 Channel Status. These bits indicate which data slot has been enabled in AC97 variable mode operation. This register is updated in case the core enables/disables a channel through a write to SACCEN/SACCDIS register or the external codec enables a channel by sending a ‘1’ in the corresponding SLOTREQ bit. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). The contents of this register only have relevance while the SSI is operating in AC97 variable mode. Writes to this register result in an error response on the IP interface. 0 Data channel disabled. 1 Data channel enabled.

43.3.3.23 SSI AC97 Channel Enable Register (SACCEN)

See Figure 43-44 for illustration of valid bits in SSI AC97 Channel Enable Register and Table 43-33 for description of the bit fields for the register.

0xBASE_54 (SACCEN) Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W							SACCEN[9:0]									
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-44. SSI AC97 Channel Enable Register

Table 43-33. SSI AC97 Channel Enable Register Field Descriptions

Field	Description
31–10	Reserved.
9–0 SACCEN	AC97 Channel Enable. The Core writes a ‘1’ to these bits to enable an AC97 data channel. Writing a ‘0’ has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as ‘0’ by the Core. 0 Write Has no effect. 1 Write Enables the corresponding data channel.

43.3.3.24 SSI AC97 Channel Disable Register (SACCDIS)

See Figure 43-45 for illustration of valid bits in SSI AC97 Channel Disable Register and Table 43-34 for description of the bit fields for the register.

0xBASE_58 (SACCDIS) Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W							SACCDIS[9:0]									
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-45. SSI AC97 Channel Disable Register

Table 43-34. SSI AC97 Channel Disable Register Field Descriptions

Field	Description
31–10	Reserved.
9–0 SACCDIS	AC97 Channel Disable. The Core writes a ‘1’ to these bits to disable an AC97 data channel. Writing a ‘0’ has no effect. Bit [0] corresponds to the first data slot in an AC97 frame (Slot #3) and Bit [9] corresponds to the tenth data slot (slot #12). Writes to these bits only have effect in the AC97 Variable mode of operation. These bits are always read as ‘0’ by the Core. 0 Write Has no effect. 1 Write Disables the corresponding data channel.

43.3.3.25 SSI Phase Configuration Register (PHCONFIG)

See Figure 43-46 for illustration of valid bits in SSI Phase Configuration Register and Table 43-35 for description of the bit fields for the register.

0xBASE_5C (PHCONFIG1) Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ClkSrcSel[3:0]				LOCK	0	0	0	GainSel[2:0]		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-46. SSI Phase Configuration Register

Table 43-35. SSI Phase Configuration Register

Field	Description
31-11	Reserved
10-7 ClkSrcSel	ClkSrc_Sel Clock source selection: 0000: if (DPLL Locked) advance_pulse else EXTAL 0001: if (DPLL Locked) advance_pulse else HCKT 0010: if (DPLL Locked) advance_pulse else HCKT1 0011: if (DPLL Locked) advance_pulse else HCKT2 0100: if (DPLL Locked) advance_pulse else HCKT3 0101: EXTAL 0110: HCKT 0111: HCKT1 1000: HCKT2 1001: HCKT3 Others: Reserved
6 LOCK	LOCK bit to show that the internal DPLL is locked, read only
5-3	Reserved
2-0 GainSel	Gain selection: 000: 24*2**10 001: 16*2**10 010: 12*2**10 011: 8*2**10 100: 6*2**10 101: 4*2**10 110: 3*2**10

43.3.3.26 SSI Frequency Measurement Register (FREQMEAS)

See Figure 43-47 for illustration of valid bits in SSI Frequency Measurement Register and Table 43-36 for description of the bit fields for the register.

0xBASE_60 (FREQMEAS1) Access: Read only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	FREQMEAS[23:16]							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FREQMEAS[15:0]															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-47. SSI Frequency Measurement Register

Table 43-36. SSI Frequency Measurement Register

Field	Description
31-24	Reserved
23:0 FreqMeas	Frequency measurement value.

43.4 Functional Description

43.4.1 SSI Architecture

The Synchronous Serial Interface (SSI) is connected to chip pads through the Digital Audio Mux (AUDMUX) module or directly as well. The AUDMUX can be configured to connect the SSI module to the chip pads in various ways. See Figure 43-1 for a block diagram of the SSI.

43.4.2 SSI Clocking

The SSI uses the following clocks:

- Bit clock — Used to serially clock the data bits in and out of the SSI port. This clock is either generated internally (from ccm_ssi_clk) or taken from external clock source (through the Tx/Rx clock ports).
- Word clock — Used to count the number of data bits per word (8, 10, 12, 16, 18, 20, 22 or 24 bits). This clock is generated internally from the bit clock.

- Frame clock (Frame Sync) — Used to count the number of words in a frame. This signal can be generated internally from the bit clock, or taken from external source (from the Tx/Rx frame sync ports).
- Sys clock —In master mode, this is an integer multiple of frame clock. This is ccm_ssi_clk. It is used in cases when SSI has to provide the clock.

Care should be taken to ensure that the bit clock frequency (either internally generated by dividing the ccm_ssi_clk or sourced from external device through Tx/Rx clock ports) is never greater than 1/5 of the ipg_clk frequency.

In Normal mode (SCR[6:5]=00), the bit clock, used to serially clock the data, is visible on the Serial Transmit Clock (STCK) and Serial Receive Clock (SRCK) ports. The word clock is an internal clock used to determine when transmission of an 8, 10, 12, 16, 18, 20, 22 or 24 bit word has completed. The word clock in turn then clocks the frame clock, which counts the number of words in the frame. The frame clock can be viewed on the STFS and SRFS frame sync ports, because a frame sync is generated after the correct number of words in the frame have passed. In master and synchronous mode, the unused port SRCK is used as Serial Oversampling Clock (ccm_ssi_clk) enabled by the SCR register bit 15, SYS_CLK_EN. This Serial System Clock is an oversampling clock of the frame sync clock (STFS). In this mode, the word length (WL), Prescaler Range (PSR), Prescaler Modulus (PM) and Frame rate (DC) selects the ratio of ccm_ssi_clk to sampling clock STFS. In case of I2S mode, the oversampling clock ccm_ssi_clk can be made available on this port if the SYS_CLK_EN bit is set. The relationship between the clocks and the dividers is shown in Figure 43-48 (“SSI Clocking”). The bit clock can be received from an SSI clock port or can be generated from the ccm_ssi_clk through a divider, as shown in Figure 43-49 (“SSI Transmit Clock Generator Block Diagram”).

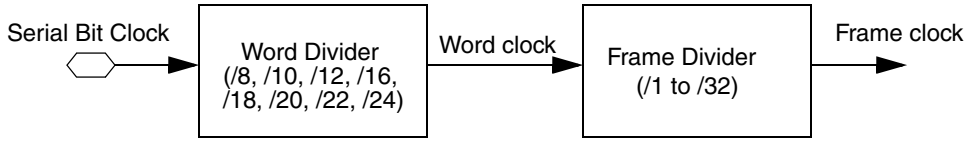


Figure 43-48. SSI Clocking

43.4.2.1 SSI Clock and Frame Sync Generation

Data clock and frame sync signals can be generated internally, or can be obtained from external sources. If internally generated, the SSI clock generator is used to derive bit clock and frame sync signals from the ccm_ssi_clk clock. The SSI clock generator consists of a selectable, fixed prescaler and a programmable prescaler for bit rate clock generation. In Gated Clock mode, the data clock is valid only when data is being transmitted. Otherwise the clock port is pulled to the inactive state. A programmable frame rate divider and a word length divider are used for frame rate sync signal generation.

Figure 43-49 shows a block diagram of the clock generator for the transmit section. The serial bit clock can be internal or external, depending on the Transmit Direction (TXDIR) bit in the SSI Transmit Configuration Register (STCR). The receive section contains an equivalent clock generator circuit.

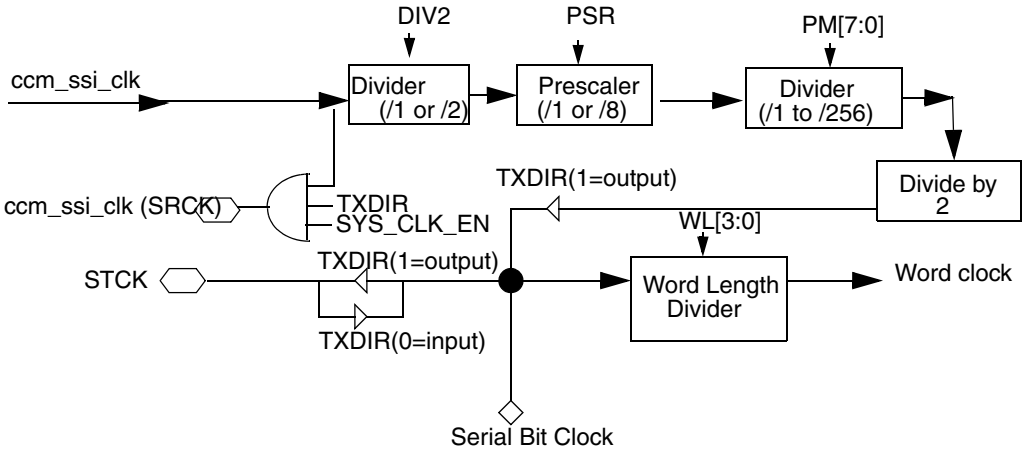


Figure 43-49. SSI Transmit Clock Generator Block Diagram

See Figure 43-50 shows the Frame Sync Generator block for the transmit section. When internally generated, both receive and transmit frame sync are generated from the word clock and are defined by the Frame Rate Divider (DC[4:0]) bits and the Word Length (WL[3:0]) bits of the SSI Transmit Clock Control Register (STCCR). The receive section contains an equivalent circuit for the Frame Sync Generator.

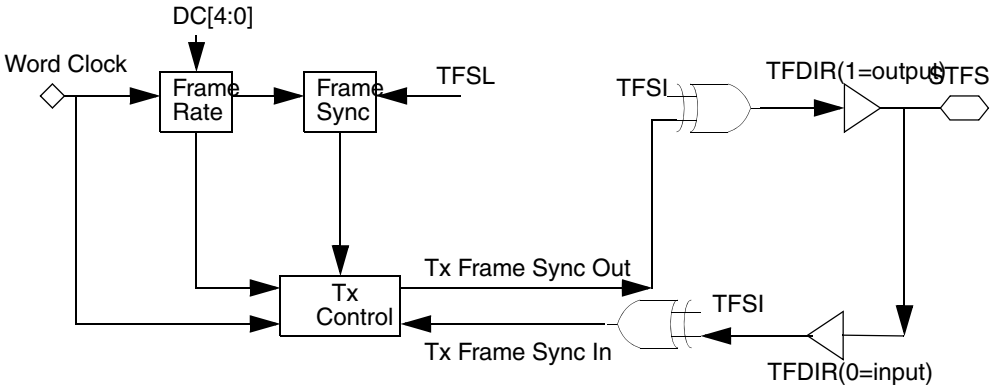


Figure 43-50. SSI Transmit Frame Sync Generator Block Diagram

43.4.2.2 DIV2, PSR and PM Bit Description

The bit clock frequency can be calculated from the SSI Serial System Clock (ccm_ssi_clk) using the equation in Figure 43-51.

NOTE

You must ensure that the bit-clock frequency must be 5 times the ipg_clk frequency. The oversampling clock frequency can go up to ipg_clk frequency. Bits DIV2, PSR and PM should not be all set to zero at the same time.

$$f_{INT\_BIT\_CLK} = f_{ccm\_ssi\_clk} / [(DIV2 + 1) \times (7 \times PSR + 1) \times (PM + 1) \times 2]$$

where PM=PM[7:0]

$$f_{FRAME\_SYN\_CLK} = (f_{INT\_BIT\_CLK}) / [(DC + 1) \times WL]$$

where DC = DC[4:0] and WL = 8, 10, 12, 16, 18, 20, 22, 24

Figure 43-51. SSI Bit Clock Equation

For example, if the SSI oversampling clock (ccm_ssi_clk) is 12.288, in 8-bit word Normal mode with DC[4:0] set to 1 (00001), PM[7:0] set to 47 (0010 1111), the PSR bit cleared, DIV2 bit set to 1, a bit clock rate of 12.288 Mhz / [1 x 4 x 48] = 64 kHz is generated. Since the 8-bit word rate is equal to one (i.e. normal mode), the sampling rate (FS rate) would then be 64 kHz / [1 * 8] = 8 kHz.

In next example, the oversampling clock (ccm_ssi_clk) clock is 11.2896 Mhz. A 16-bit word Network mode with DC[4:0] set to 1 (00001), PM[7:0] set to 3 (0000 0011), the PSR bit is set to 0, DIV2 bit set to 0, and a 11.2896 MHz oversampling clock, a bit clock rate of 11.2896 Mhz / [1 x 2 x 4] = 1.4112 MHz is generated. Since the 16-bit word rate is equal to two, the sampling rate (FS rate) would be 1.4112 MHz / [2 * 16] = 44.1 kHz.

Table 43-37 shows programming examples for the clock dividers in the CRM and the SSI to support various bit clock (STCK) frequencies.

Table 43-37. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words/ Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (Mhz)	DIV 2	PS R	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Target Bit_Clk Freq (kHz) STCK	Error (Hz)
16	1	8	294.912	48	12.288	0	0	47	7	0	128	128	0
16	2	8	294.912	48	12.288	0	0	23	7	1	256	256	0
16	4	8	294.912	48	12.288	0	0	11	7	3	512	512	0
16	1	12	294.912	48	12.288	0	0	31	7	0	192	192	0
16	2	12	294.912	48	12.288	0	0	15	7	1	384	384	0

Table 43-37. SSI Bit Clock and Frame Rate as a Function of PSR, PM, and DIV2

Bits/ Word	Words/ Frame	Ideal Frame Rate (kHz)	PLL Freq (Mhz)	SSIDIV (in CRM)	MCLK/ ccm_ssi_clk Freq (Mhz)	DIV 2	PS R	PM	WL	DC	Actual Bit_Clk Freq (kHz) STCK	Target Bit_Clk Freq (kHz) STCK	Error (Hz)
16	4	12	294.912	48	12.288	0	0	7	7	3	768	768	0
16	1	16	294.912	48	12.288	0	0	23	7	0	256	256	0
16	2	16	294.912	48	12.288	0	0	11	7	1	512	512	0
16	4	16	294.912	48	12.288	0	0	5	7	3	1024	1024	0
16	1	24	294.912	48	12.288	0	0	15	7	0	384	384	0
16	2	24	294.912	48	12.288	0	0	7	7	1	768	768	0
16	4	24	294.912	48	12.288	0	0	3	7	3	1536	1536	0
16	1	32	294.912	48	12.288	0	0	11	7	0	512	512	0
16	2	32	294.912	48	12.288	0	0	5	7	1	1024	1024	0
16	4	32	294.912	48	12.288	0	0	2	7	3	2048	2048	0
16	1	48	294.912	48	12.288	0	0	15	7	0	768	768	0
16	2	48	294.912	48	12.288	0	0	3	7	1	1536	1536	0
16	4	48	294.912	48	12.288	0	0	1	7	3	3072	3072	0
16	1	11.025	270.9504	48	11.2896	0	0	31	7	0	176.4	176.4	0
16	2	11.025	270.9504	48	11.2896	0	0	15	7	1	352.8	352.8	0
16	4	11.025	270.9504	48	11.2896	0	0	7	7	3	705.6	705.6	0
16	1	22.05	270.9504	48	11.2896	0	0	15	7	0	352.8	352.8	0
16	2	22.05	270.9504	48	11.2896	0	0	7	7	1	705.6	705.6	0
16	4	22.05	270.9504	48	11.2896	0	0	3	7	3	1411.2	1411.2	0
16	1	44.1	270.9504	48	11.2896	0	0	7	7	0	705.6	705.6	0
16	2	44.1	270.9504	48	11.2896	0	0	3	7	1	1411.2	1411.2	0
16	4	44.1	270.9504	48	11.2896	0	0	1	7	3	2822.4	2822.4	0

NOTE

Table 43-37 describes how various frame rates can be achieved with the PLL0/1 supplying a frequency of 294.912 MHz and 270.9504 MHz (with WL and DC settings as shown). Using PLL2 requires that these input frequencies be lowered by a factor of 2 (and the dividers be changed accordingly). Using the MRCG allows the input frequency to be lowered by a factor of 4 (provided the dividers are changed accordingly). These clocks are recommended as convenient starting points but the system allows for other input clock frequencies as well.

Table 43-37 above shows programming of the CRM and SSI dividers in order to generate the appropriate oversampling clock and BIT_CLK frequencies for various sampling rates. In these examples, the master mode is selected either by setting I2S master bit (SCR[6:5]=01) or individually programming the SSI in network, synchronous, transmit internal mode (the table specifically illustrates the I2S mode frequencies/sample rates). The oversampling clock is ccm_ssi_clk.

Note that the I2S master mode requires that a word length of 32 bits be used (regardless of the actual data type). Consequently, the fixed I2S frame rate of 64 bits per frame (word length (WL) can be any value) and DC of 1 are assumed.

43.4.3 Receive Interrupt Enable Bit Description

When the RIE and RE bit are set, the processor is interrupted when either of the SSI Receive FIFO Full (RFF0/1) bits in SISR is set (if the corresponding Receive FIFO is enabled). If the Receive FIFO is not enabled, the interrupt is generated when the corresponding SSI Receive Data Ready (RDR0/1) bit in the SISR is set. When the receive FIFO is enabled, a maximum of 15 values are available to be read (15 values per channel in Two-Channel mode). If not enabled, then one value can be read from the SRX register (one each in case of Two-Channel mode). If the RIE bit is cleared, these interrupts are disabled. However, the RFF0/1 and RDR0/1 bits still indicate the receive data register full condition. Reading the SRX registers clears the RDR bits, thus clearing the pending interrupt. Two receive data interrupts (two per channel in case of Two-Channel mode) are available: receive data with exception status and receive data without exception. Table 43-38 and Table 43-39 show the conditions under which these interrupts are generated.

Table 43-38. SSI Receive Data 1 Interrupts

Interrupt	RIE	ROE0	RFF0/RDR0
Receive Data 1(with Exception Status)	1	1	1
Receive Data 1(without exception)	1	0	1

Table 43-39. SSI Receive Data 0 Interrupts

Interrupt	RIE	ROE1	RFF1/RDR1
Receive Data 0 (with Exception Status)	1	1	1
Receive Data 0 (without exception)	1	0	1

43.4.4 Transmit Interrupt Enable Bit Description

The SSI Transmit Interrupt Enable (TIE) control bit determines whether the processor is interrupted when the SSI transmitter needs to be serviced. When the TIE and TE bits are set, the program controller is interrupted when either of the SSI Transmit FIFO Empty (TFE0/1) flags in SISR are set (if corresponding Transmit FIFO is enabled). If the corresponding Transmit FIFO is not enabled, an interrupt is generated when the corresponding SSI Transmit Data Register Empty (TDE0/1) flag in the SISR is set and Transmit Enable (TE) bit is set.

When Transmit FIFO 0 is enabled, a maximum of 15 values can be written to the SSI (15 per channel in case of Two-Channel mode, using Tx FIFO 1). If not enabled, then one value can be written to the STX0

register (one per channel in case of Two-Channel mode using STX1). When the TIE bit is cleared, all transmit interrupts are disabled. However, the TDE0/1 bits always indicate the corresponding STX register empty condition, even when the transmitter is disabled by the Transmit Enable (TE) bit (in the SCR). Writing data to the STX clears the corresponding TDE bit, thus clearing the interrupt. Two transmit data interrupts are available (four in case of Two-Channel mode, two per channel): transmit data with exception status and transmit data without exceptions. Table 43-40 and Table 43-41 show the conditions under which these interrupts are generated.

Table 43-40. SSI Transmit Data 1 Interrupts

Interrupt	TIE	TUE1	TFE1/TDE1
Transmit Data 1 (with Exception Status)	1	1	1
Transmit Data 1 (without exception)	1	0	1

Table 43-41. SSI Transmit Data 0 Interrupts

Interrupt	TIE	TUE0	TFE0/TDE0
Transmit Data 0 (with Exception Status)	1	1	1
Transmit Data 0 (without exception)	1	0	1

43.4.5 Internal Frame and Clock Shutdown

During transmit/receive operation, disabling TE/RE will ensure that data transmission/reception stops after current frame ends following which TFRC/RFRC Status bits will get set to indicate the Frame Completion State. If TE is disabled 4 clock cycles before the next frame, extra frame generated are invalid frames. TFR_CLK_DIS/RFR_CLK_DIS bit is set in the current or any of the previous frames, SSI will stop driving the STFS/SRFS and STCK/SRCK signals after the current frame ends.

If TFR_CLK_DIS/RFR_CLK_DIS bit is not set, SSI will continue generating STFS/SRFS and STCK/SRCK signals (in case direction is from SSI), which then can be disabled by writing ‘1’ to TFR_CLK_DIS/RFR_CLK_DIS bit. SSI will then stop driving these signals after end of frame is reached following which TFRC/RFRC status bits will get set to indicate the Frame Completion State.

Figure 43-52 is an illustration of transmission case where TXDIR and TFDIR are both set to ‘1’. In this case TE is disabled with TFR_CLK_DIS bit set in current or any of the previous frames.

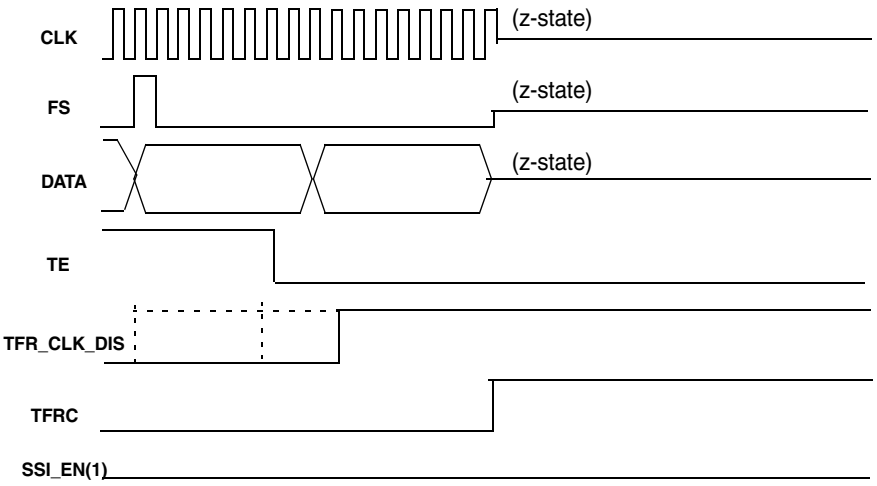


Figure 43-52. TFR_CLK_DIS assertion in current or previous frame as TE disable

Figure 43-53 is an illustration of transmission case where TXDIR and TFDIR are both set to ‘1’. In this case TFR_CLK_DIS bit is set after few frames of disabling TE. TFRC (Transmit Frame Complete) is set at frame boundary after TE is cleared. Once software services this interrupt and sets TFR_CLK_DIS bit later, TFRC bit is again set at next frame boundary.

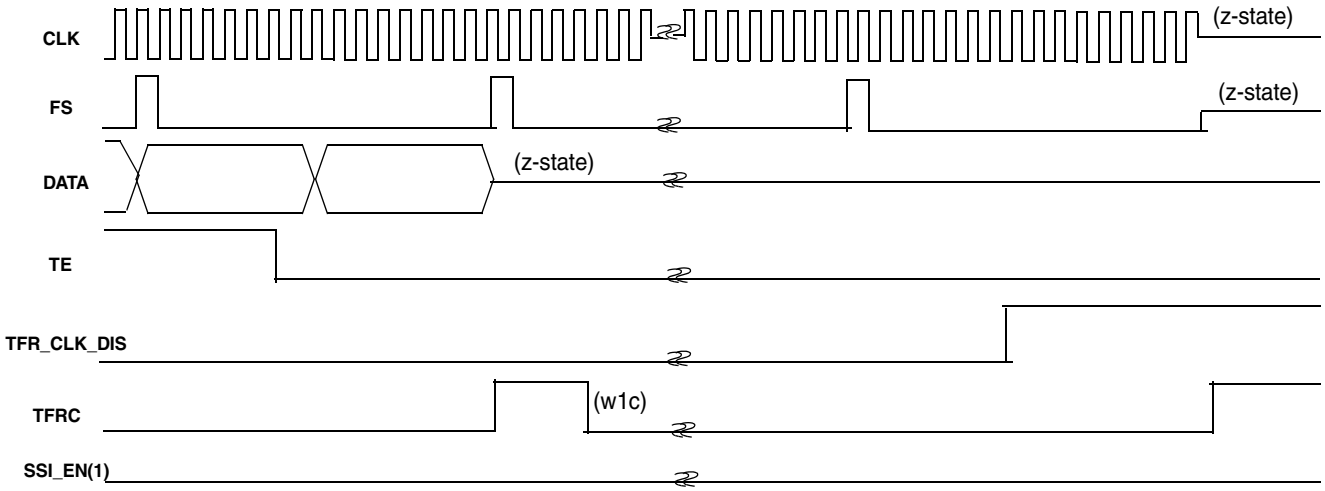


Figure 43-53. TFR_CLK_DIS assertion in subsequent frame after disabling TE

43.4.6 Frequency Measurement Block

The internal DPLL can extract the bit clock (advanced pulse) from the input bitstream. It is necessary, however, to measure the frequency of the incoming signal in relationship with the clock SYSTEM_CLK. The circuit is shown in Table 43-54.

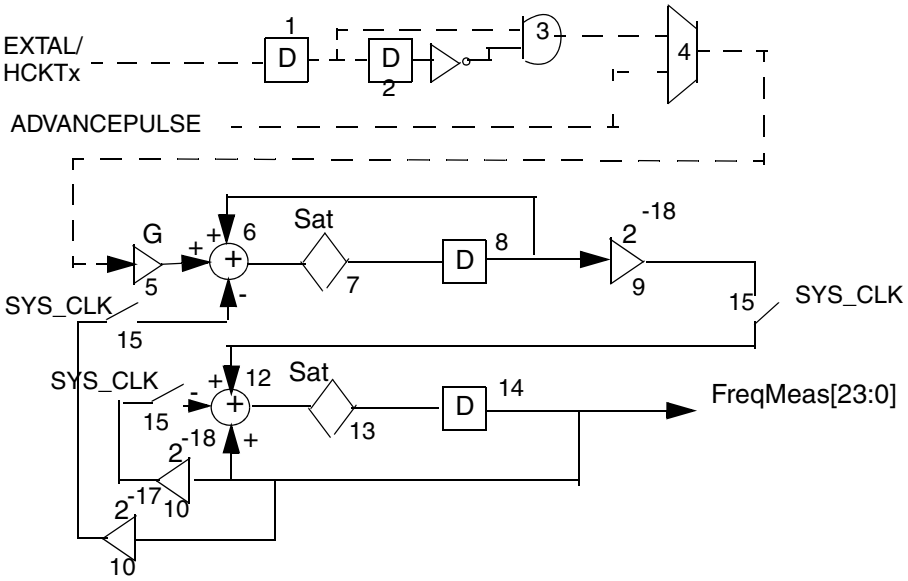


Figure 43-54. FreqMeas Circuit

Associated with it, are 2 registers, PHCONFIG and FREQMEAS. The circuit will measure the frequency of the incoming clock as a function of the SYSTEM_CLK. The circuit is a second-order filter. The output is a value represented by an unsigned number stored in the 24 bit FreqMeas register, giving the frequency of the source as a function of the SYSTEM_CLK.

FreqMeas[23:0] = FreqMeas_CLK / SYSTEM_CLK * 2**10 * GAIN.

For example, if the GAIN is selected as 8*2**10 (PhaseConfig[2:0] = 3’b011), the actual result FreqMeas_CLK / SYSTEM_CLK is equal to FreqMeas[23:0] / 2**23.

43.4.7 IP Bus Interface

The SSI has an IP Bus interface compatible with SRS 3.0.2 in order to provide a control and data interface. This interface is used by both the processor and DMA controller.

43.4.7.1 Transfer Lengths Supported

The IP Bus interface of the SSI only supports 32-bit transfers with all SSI registers other than STX0, STX1, SRX0, and SRX1 (that is, the data registers). With the exception of the data registers, using 8-bit and 16-bit transactions could result in undesired behavior but will not result in a transfer bus error. The data registers (STX0, STX1, SRX0, and SRX1) support 8-bit, 16-bit, and 32-bit transfer lengths without restrictions.

43.4.7.2 Transfer Bus Errors

Transfer bus errors are generated upon response to the following:

- Write transfer to a read-only register.
- Read or write access to a register space beyond the last populated register of the SSI in its memory map (up until the end of the allocated memory address range of the SSI).

43.4.7.3 Clock Rate

The IP Bus clock frequency must be at least five times the serial bit clock frequency.

43.5 Initialization/Application Information

The SSI is affected by the following types of reset:

- Power-on Reset—The Power-on reset is generated by asserting the RESET port. The Power-on reset clears the SSIEN bit in SCR, which disables the SSI. All other status and control bits in the SSI are affected as described in SSI Programming Model in Section 43.3.
- SSI Reset—The SSI reset is generated when the SSIEN bit in the SCR is cleared. The SSI status bits are preset to the same state produced by the Power-on reset. The SSI control bits are unaffected. The control bits in the SCR are also unaffected. The SSI reset is useful for selective reset of the SSI without changing the present SSI control bits and without affecting the other peripherals.

The correct sequence to initialize the SSI is as follows:

1. Issue a Power-on or SSI reset (SCR[SSIEN]=0).
2. Disable SSI clocks (ipg_clk, ccm_ssi_clk).
3. Set all control bits for configuring the SSI (see Table 43-42 for the list of “SSI Control Bits Requiring SSI to be Disabled Before Change”).
4. Enable appropriate interrupts/DMA requests through SIER.
5. Set the SCR[SSIEN] bit (=1) to enable the SSI.
6. Enable SSI clocks (ipg_clk, ccm_ssi_clk), as required.
7. In case of AC97 mode, set the SACNT[AC97EN] bit after programming the SATAG register (if needed, for AC97 Fixed mode).
8. In case of AC97 fixed mode, do not program the slot request bits without programming the frame valid bits in SATAG register.
9. In case of gated mode of operation refer Table 43-5.
10. Set SCR[TE/RE] bits.
11. To ensure proper operation of the SSI, use the “Power-on or SSI reset before changing any of the SSI Control” bits listed in Table 43-42).

NOTE

These control bits should not be changed when SSI is enabled

Table 43-42. SSI Control Bits Requiring SSI to be Disabled Before Change

Control Register	Bit
SCR	[9]=CLK_IST [8]=TCH_EN [7]=SYS_CLK_EN [6:5]=I2S_MODE [4]=SYN [3]=NET
SIER	[22]=RDMAE [20]=TDMAE
SRCR STCR	[9]=RXBIT0 [9]=TXBIT0 [8]=RFEN1 [8]=TFEN1 [7]=RFEN0 [7]=TFEN0 [6]=RFDIR [6]=TFDIR [5]=RXDIR [5]=TXDIR [4]=RSHFD [4]=TSHFD [3]=RSCKP [3]=TSCKP [2]=RFSI [2]=TFSI [1]=RFSL [1]=TFSL [0]=REFS [0]=TEFS
SRCCR STCCR	[16]=WL3 [15]=WL2 [14]=WL1 [13]=WL0
SACNT	[1]=FV [10:5]=FRDIV
PHCONFIG	[10:7]=CLKSRCSEL [0:2]=GAINSEL

Chapter 44
Touch Screen Controller (TSC) and Analog-to-Digital Converter (ADC)

44.1 Introduction

Together the touch screen controller (TSC) and associated analog-to-digital converter (ADC) provide a resistive touch screen solution for low cost PDAs, Cell Phones, ePOS devices, and multimedia players. The module implements simultaneous touch screen control and auxiliary ADC operation for temperature, voltage and other measurement functions. It includes the driver switches for controlling the screen and an input multiplexer to allow support for five screen purpose inputs: XP(UL), XN(UR), YP(LL), YN(LR), WIPER, and three additional inputs: INAUX0, INAUX1, INAUX2. The ADC reference voltage can be configured in differential and single-ended modes. The controller supports both polled and interrupt-driven measurement, including TSC pen-down event interrupts.

44.1.1 Overview

Figure 44-1 shows a system level overview of the i.MX25 Touch Screen Controller and Analog to Digital converter components, connected to an external touch screen. Touch screen position and auxiliary input signals are shown, as well as external reference voltage, ADC analog power supply, ADC analog ground, digital power supply and digital ground.

NOTE

All ADC inputs, including the touch screen inputs, may be used to convert analog input signals even if the TSC function is not used.

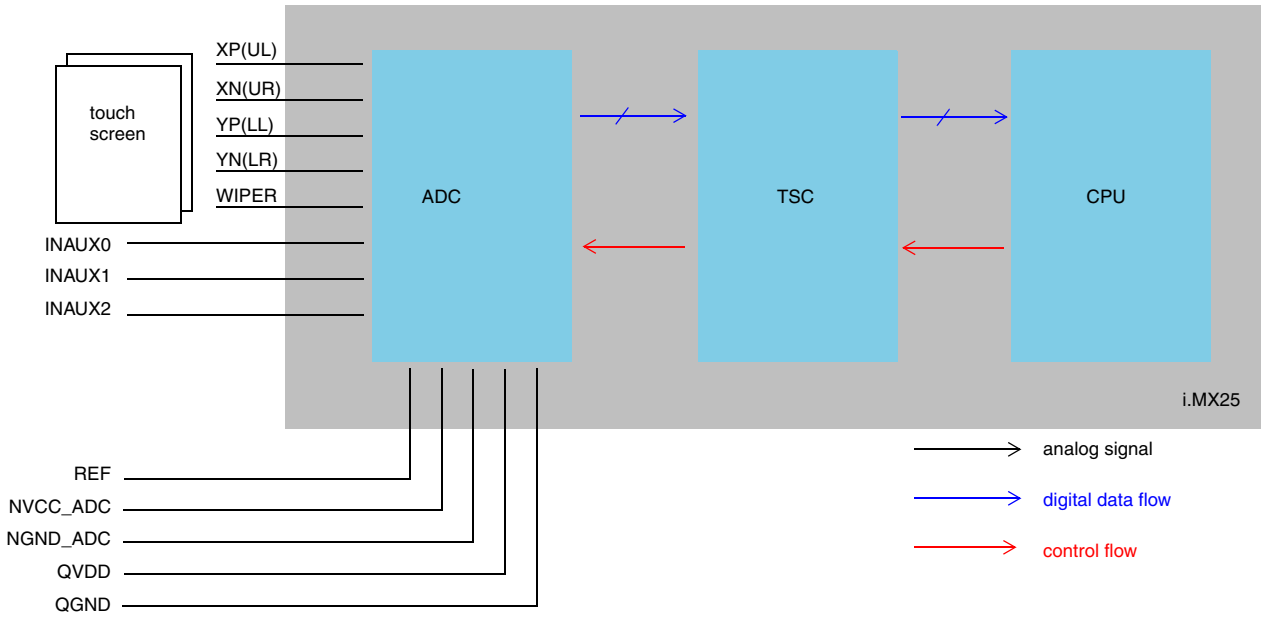


Figure 44-1. i.MX25 Touch Screen Solution

44.1.2 Features

- Integrated 12-bit, 125-kHz ADC
- Supports ratiometric measurement drivers
- Configurable in single-ended or differential (ratiometric) topologies
- Configurable to built-in voltage reference generator or external reference voltage
- Supports 4- and 5-wire touch screens with five inputs channels for touch screen measurements ($x+$, $x-$, $y+$, $y-$, w)
- Supports general-purpose analog measurements (such as temperature and voltage) with three input channels (aux0, aux1, aux2), or 8 input channels if touch interface is not used
- Two independent measurement queues: TCQ for touch screen and GCQ for general-purpose measurements
- Two independent FIFOs, each with 16 entries×16 bits, for storing TCQ and GCQ conversion results
- Supports pen touch screen detection interrupt to awaken the system from sleep mode
- Supports three power modes: always-off, power-saving, always-on
- Configurable pen down debounce logic
- Configurable LCD noise-reduction logic
- Configurable settling time before each measurement
- Configurable multisampling for each measurement
- Configurable data discarding for each measurement

44.2 Application Information

44.2.1 Introduction to Resistive Touch Screens

A touch screen is a sensor to measure the physical location of a touch, by pen or finger, at a given point in a rectangular area, normally above a LCD Screen. Resistive touch screens have a top layer and a bottom layer separated by insulating dots, with the inside surface of each layer coated with a transparent conductive coating. All resistive touch screens use essentially the same voltage-driven operating principles. Voltage applied to the resistive layer produces a gradient across the layer. Pressing the flexible top sheet creates electrical contact between the layers, essentially closing a switch in the circuit.

44.2.1.1 4-Wire Touch Screen

4-wire touch screen technology and electronics are simple, making 4-wire the cheapest touch screen technology. First, the distance along the x-axis at the point of touch is measured by creating a horizontal voltage gradient on the top sheet, with the bottom acting as the return layer. Second, a vertical voltage gradient is created on the bottom layer, to measure the y-axis.

Since the voltage gradient is needed on both the layers, any damage to either layer causes the touchscreen to stop functioning. Four-wire touch screens are prone to damage with heavy use, since both layers are often plastic. This lack of durability means that 4-wire technology should not be used for applications such as public access kiosks, industrial locations or on displays larger than 12 inches.

Figure 44-2 shows an example of a 4-wire touch screen. It consists of two transparent and flexible resistive layers: X layer and Y layer.

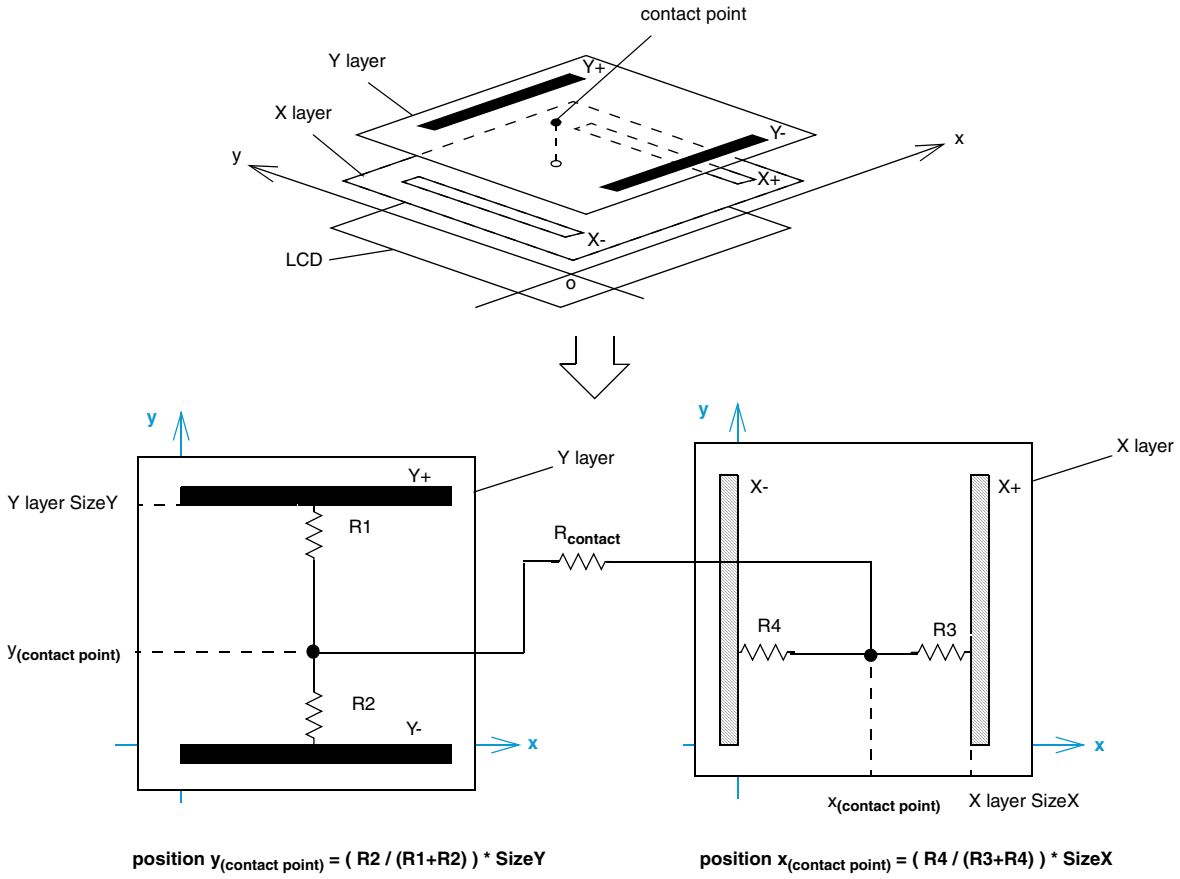


Figure 44-2. 4-Wire Touch Screen Model

Provided that the X and Y resistive layers are of uniform resistivity, the resistance value at any contact point between the two electrodes (X+/X- in X layer or Y+/Y- in Y layer) is proportional to its position in each layer. The contact point physical location $\{x_{(\text{contact point})}, y_{(\text{contact point})}\}$ in two coordinate pair dimensions can be measured by the resistance ratios in X layer and Y layer, when the screen is touched and these two layers make contact with each other.

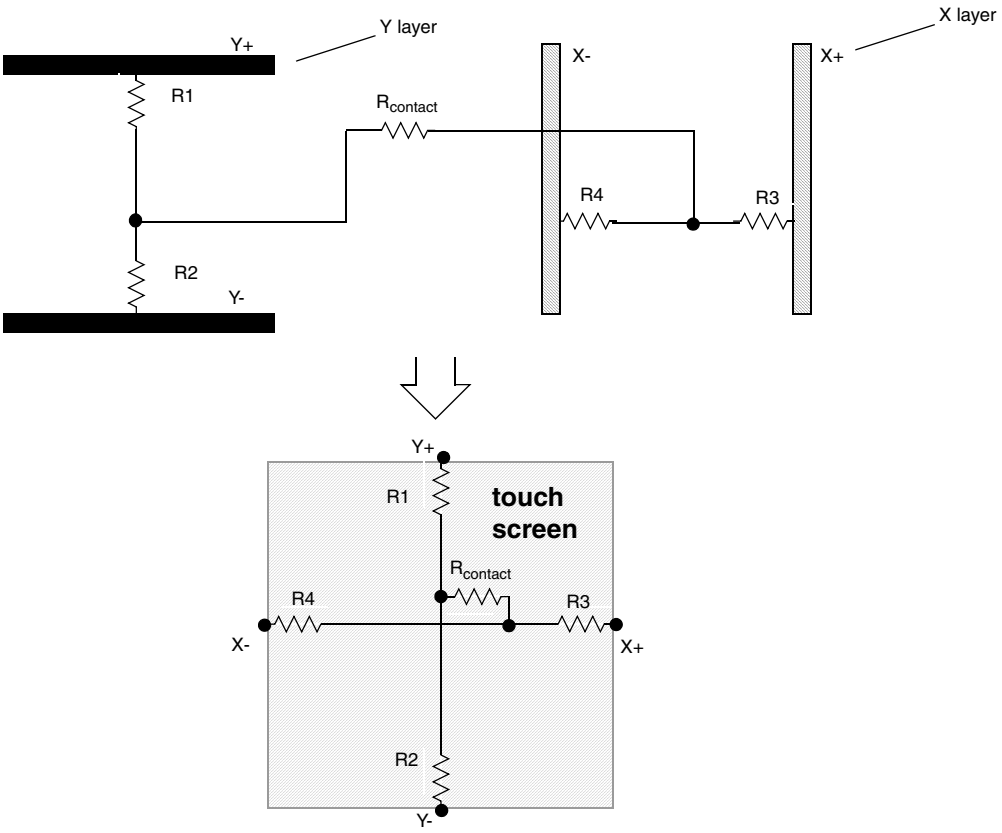


Figure 44-3. 4-Wire Touch Screen Simplified Model

44.2.1.2 5-Wire Touch Screen

5-Wire touch screen consists of a resistive layer and a conductive layer. The conductive layer has a contact bar (WIPER), usually along one edge. The resistive layer has a contact point at each corner (UL at upper left, UR at upper right, LL at lower left and LR at lower right.

To measure along the x-axis, a uniform voltage is applied to the upper left corner and lower left corner and the upper right corner and lower right corner are connected to ground. Because the left and right corners are at the same voltage, the effect is the same as attaching electrodes along the left and right edges, similar to the method used with the 4-wire touch screen.

To measure along the y-axis, a uniform voltage is applied to the upper left corner and upper right corner and the lower left corner and lower right corner are connected to ground. Because upper and lower corners are at the same voltage, the effect is about the same as attaching electrodes along the top and bottom edges, similar to the method used with the 4-wire touch screen.

The technology and electronics used in 5-wire touch screens makes them more expensive than 4-wire devices. However, improved accuracy makes it possible to use 5-wire for sizes up to 22 inches, larger than with 4-wire. Also, since the voltage measurement is on the stable bottom layer, 5-wire touchscreens can keep working despite damage to one portion of the top layer. This gives 5-wire touch screens excellent long term durability. Figure 44-4 shows an example of a 5-wire touch screen.

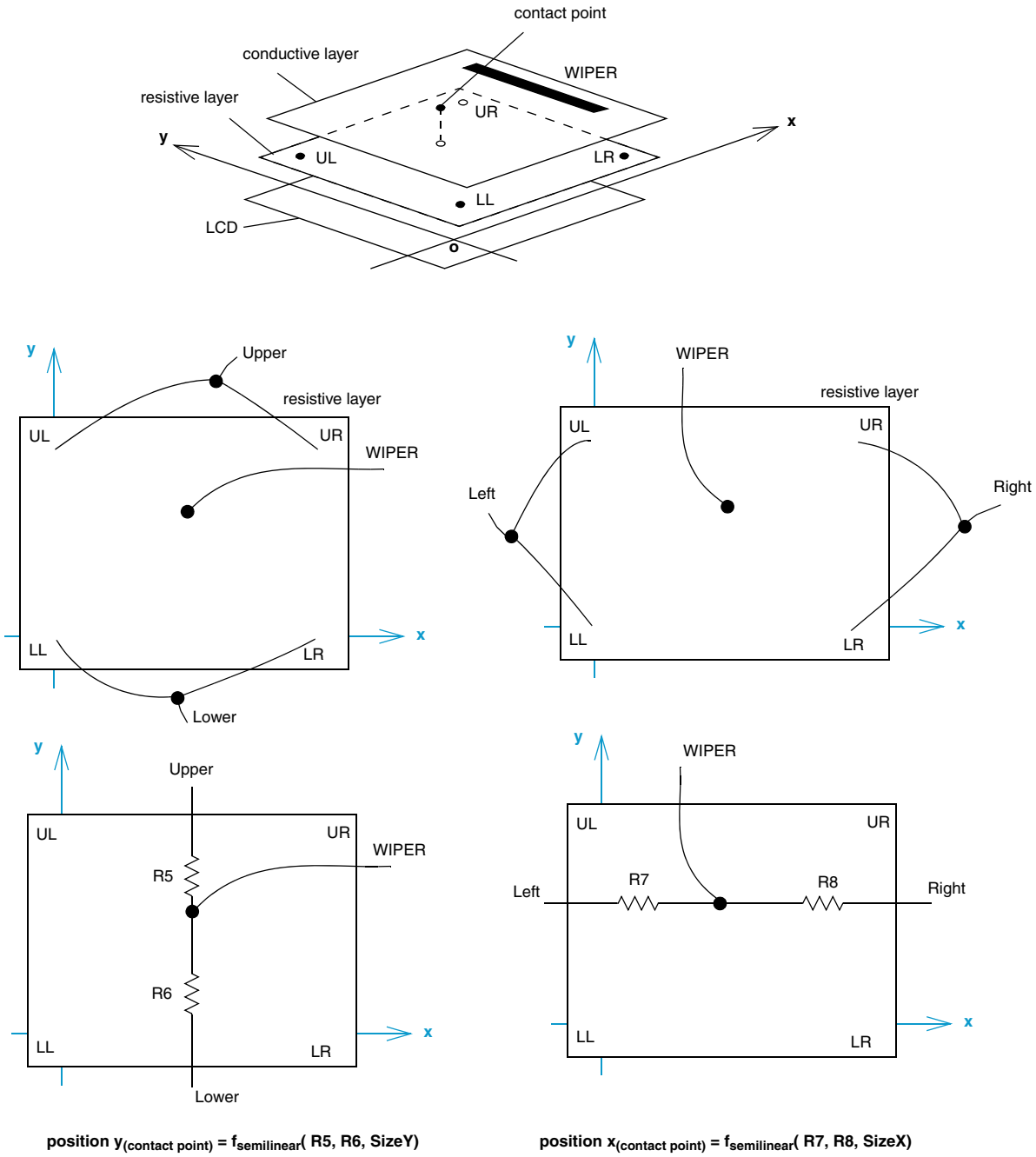
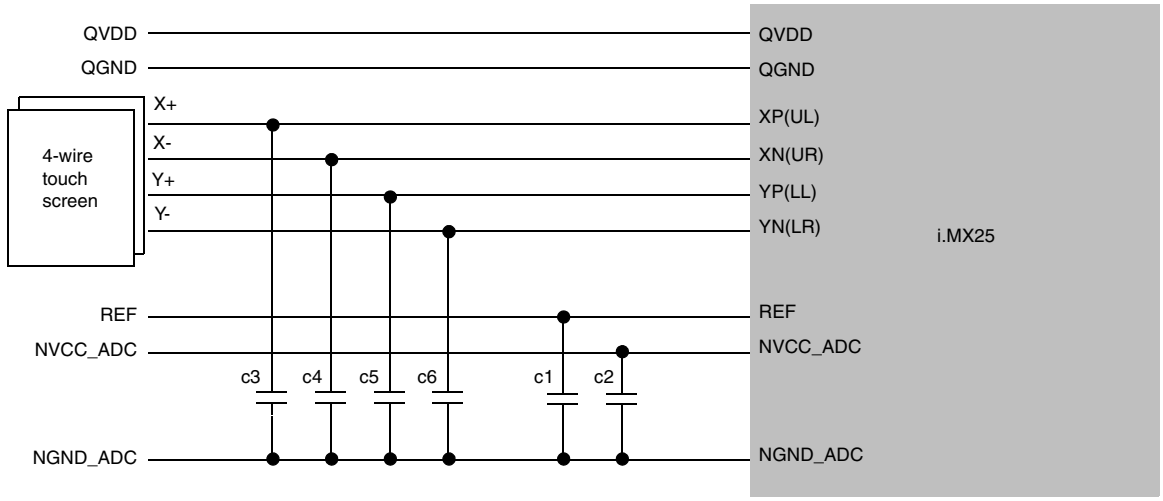


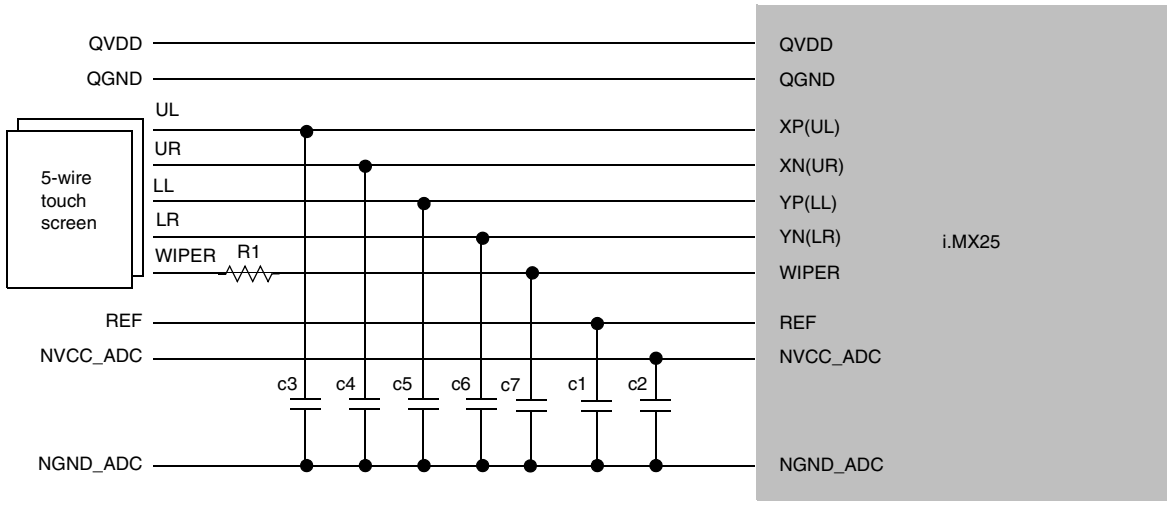
Figure 44-4. 5-Wire Touch Screen Model

44.2.2 Touch Screen Connection

The TSC is compatible with 4-, and 5-wire touch-screens. Figure 44-5 shows how to connect the controller to these touch screens.



c1 - decoupling capacitors for REF
c2 - decoupling capacitors for NVCC_ADC
c3, c4, c5, c6 - optional additional capacitor added for noise filtering



c1 - decoupling capacitors for REF
c2 - decoupling capacitors for NVCC_ADC
c3, c4, c5, c6, c7 - optional additional capacitor added for noise filtering
R1 - optional additional resistor added for noise filtering

Figure 44-5. Connection for 4-wire and 5-wire Touch Screen

The simplest way to minimize the effects of a noisy environment on the measurements is to add capacitors and resistors, as needed, on the connections to the touch screen to perform a low pass operation. The values of the optional decoupling capacitors and resistors depend on the number of measurements per second the

system has to perform. The main restriction is the required settling time for the touch screen nodes being driven. This time depends on the following factors:

- R_{plate}: Plate resistor for each specific touchscreen (its effective value changes with the point being pressed by the user at each measurement, so, worst case should be considered)
- R_{opt}: Optional additional noise filtering resistor
- C_{opt}: Optional additional capacitor added for noise filtering
- C_{par}: Total parasitic capacitance on driven node (typical value around 15 pF)
- R_{par}: Additional total parasitic resistance on driven node (typical value around 10 Ω)

For N bits precision on the settling of the driven nodes, the required settling time is roughly given by the following first order equation:

$$t_{settling}=(2R_{plate} + R_{opt} + R_{par})(C_{opt} + C_{par})\ln(2^N) \quad \textit{Eqn. 44-1}$$

$t_{settling}$ is the theoretical required time for the driven nodes to settle. Additional margin should be given due to the uncertainty of the parasitics. A minimum $2t_{settling}$ should be considered to be given before applying an ADC measurement.

For a typical plate resistance of R_{plate}=150 Ω (each plate), C_{opt}=10 nF, R_{opt}=0 Ω, R_{par}=10 Ω, C_{par}=15 pF and a system resolution of 12 bits, $t_{settling}$ is around 26 us. Given the $2t_{settling}$ safe margin and including the ADC conversion time of 8 us working at 125 KHz, the system should be able to perform one measurement each 60 us, thus performing around 16,666 measurements per second. This performance value is extremely dependent on system level options (R_{plate}, R_{opt}, C_{opt}, N) so these factors should be carefully considered.

Power supply decoupling should be done according to Figure 44-5. The 10 nF capacitors should be ceramic (good quality), and must be placed as close as possible to the device.

44.2.3 Touch Screen Measurement

For each touch screen reading, it is advisable to follow the sequence below:

1. Pre-charge
2. Touch screen detection
3. X-measurement (several measurements are advisable)
4. Y-measurement (several measurements are advisable)
5. Pre-charge
6. Touch screen detection

It should be assumed that, if a touch was detected at the beginning and end of the measurement (in both step 2 and step 6), then the measurement is valid. Four different functions are defined:

- X-measurement
- Y-measurement
- Pre-charge
- Touch screen touch detection

44.2.3.1 Pen IRQ Operation

The pen interrupt feature is implemented separately for the resistive and conductive plate of the touch screen. This is a fully asynchronous process and it operates even without a clock signal present.

44.2.3.2 Pressure Measurement

For a 4-wire touch screen, the pressure applied to the touch screen can be determined by measuring the contact resistance ($R_{contact}$) between the X and Y plates. $R_{contact}$ decreases as the pressure increases. To calculate the contact resistance, the X plate resistance must be known and three measurements must be performed:

- X measurement
- Yn measurement
- Xp measurement

See Table 44-1 for the configuration of the TSC corresponding to these three measurements for a 4-wire touch screen. The contact resistance is then calculated using the following equation:

$$R_{contact}=R_{x\text{-}plate} * (X_{measurement} / 4096) * ((Yn_{measurement} - Xp_{measurement}) / Xp_{measurement}) \quad \textit{Eqn. 44-2}$$

44.2.4 ADC Acquisition

i.MX25 includes a 12-bit ADC macro-cell. This macro-cell is flexible, compact and power efficient. It includes a 12-bit Successive-Approximation (SAR) analog-digital-converter and the driver switches to control several kinds of touch screens. Both single-ended and differential (ratiometric) position measurements are supported. This cell also features pen touching screen detection capabilities and screen pressure measurements. Auxiliary inputs are provided for general purpose measurements, such as temperature and battery voltage. When these inputs are used, the ADC reference voltage can either be the one internally generated from a bandgap reference, or an external voltage supplied to the cell. All biasing and reference circuits are included. A power down mode is included with less than 10 μA of current consumption.

The ADC startup sequence and ADC conversion operations are transparent to the software programmer and automatically implemented by TSC hardware. The ADC startup sequence includes a dummy conversion cycle at the end of the sequence to let the voltages of the internal analog nodes settle to the right values. The result from this conversion should be disregarded.

44.2.5 Example Programs

Some typical example programs are given below.

```
1. initial tsc
        *P_CCM_CGR2 &= ~(0x1<<13); // disable tsc ipgclk in CCM
        *P_TSC_TGCR &= ~(0x1<<0);  // disable tsc ipgclken bit
        *P_CCM_CCTL |= 0x1<<15;    // set ipgclk control by tsc ipgclken bit
        *P_CCM_CGR2 |= 0x1<<13;    // enable tsc ipgclk in CCM
        *P_TSC_TGCR |= 0x1<<0;     // enable tsc ipgclken bit
        *P_TSC_TGCR |= 0x1<<1;     // self-reset tsc
        while (*P_TSC_TGCR & (0x1<<1));// wait until self-reset done
```

```
2. config tcq as 4-wire touch-screen measurement
        #define TSC_4WIRE_PRECHARGE        0x158C
        #define TSC_4WIRE_TOUCH_DETECT     0x578E
        #define TSC_4WIRE_X_MEASUMENT      0x1C90
        #define TSC_4WIRE_Y_MEASUMENT      0x4604

        *P_TCC0 = TSC_4WIRE_PRECHARGE;
        *P_TCC0 |= 0x1<<20; //ignore the ADC data during precharge;
        *P_TCC1 = TSC_4WIRE_TOUCH_DETECT;
        *P_TCC2 = TSC_4WIRE_X_MEASUMENT;
        *P_TCC2 |= 0x3<<16; //4 samples for X measurement;
        *P_TCC3 = TSC_4WIRE_Y_MEASUMENT;
        *P_TCC3 |= 0x3<<16; //4 samples for Y measurement;

        // item 0 = 4-wire precharge
        // item 1 = 4-wire detect
        // item 2 = 4-wire x
        // item 3 = 4-wire y
        // item 4 = 4-wire precharge
        // item 5 = 4-wire detect
        *P_TSC_TCQITEM_7_0= 0x00103210;
        lastitemid = 5;
        *P_TSC_TCQCR =(*P_TSC_TCQCR & ~(0xf<<4)) | lastitemid<<4;

3. config tcq as 5-wire touch-screen measurement
        #define TSC_5WIRE_PRECHARGE        0x158C
        #define TSC_5WIRE_TOUCH_DETECT     0x978E
        #define TSC_5WIRE_X_MEASUMENT      0x4CC4
        #define TSC_5WIRE_Y_MEASUMENT      0x70C4

        *P_TCC4 = TSC_5WIRE_PRECHARGE;
        *P_TCC4 |= 0x1<<20; //ignore the ADC data during precharge;
        *P_TCC5 = TSC_5WIRE_TOUCH_DETECT;
        *P_TCC6 = TSC_5WIRE_X_MEASUMENT;
        *P_TCC6 |= 0x3<<16; //4 samples for X measurement;
        *P_TCC7 = TSC_5WIRE_Y_MEASUMENT;
        *P_TCC7 |= 0x3<<16; //4 samples for Y measurement;

        // item 0 = 5-wire precharge
        // item 1 = 5-wire detect
        // item 2 = 5-wire x
        // item 3 = 5-wire y
        // item 4 = 5-wire precharge
        // item 5 = 5-wire detect
        *P_TSC_TCQITEM_7_0= 0x00547654;
        lastitemid = 5;
        *P_TSC_TCQCR = (*P_TSC_TCQCR & ~(0xf<<4)) | lastitemid<<4;

4. config tcq as software invoke and software polling mode
        *P_TSC_TCQCR = (*P_TSC_TCQCR & ~0x3) | 0x2; // set qsm as fqs invoke
        *P_TSC_TCQCR |= (0x1<<2);                  // set fqs
        while (!(*P_TSC_TCQSR & 0x1<<1));          // wait until end of queue
        *P_TSC_TCQCR &= ~0x3;                      // set qsm as stop
        *P_TSC_TCQCR &= ~(0x1<<2);                 // clear fqs
        *P_TSC_TCQSR |= 0x1<<1;                    // clear eoq status bit
        // then check the data in FIFO ...
```

```
5. config tcq as pen down invoke and software interrupt mode
        *P_TICR = TSC_4WIRE_PRECHARGE; // or *P_TICR = TSC_5WIRE_PRECHARGE;
        for(i=0;i<precharge_delay;i++);// add some delay for pre-charge
        *P_TICR = TSC_4WIRE_TOUCH_DETECT; // or *P_TICR = TSC_5WIRE_TOUCH_DETECT;
        *P_TSC_TCQCR = (*P_TSC_TCQCR & ~0x3) | 0x1; // set qsm as pen down invoke

        *P_TSC_TCQMR &= ~(0x1<<0);                  // enable tcq pen down interrupt
        *P_TSC_TCQCR &= ~(0x1<<18);                 // tcq pen down unmasked;
        *P_TSC_TCQCR = (*P_TSC_TCQCR & ~0x3) | 0x1; // set qsm as pen down
        dbtime=3;
        *P_TSC_TGCR =
        (*P_TSC_TGCR & ~(0x7f<<25)) | (dbtime<<25); // set debound time
        *P_TSC_TGCR |= (0x1<<24);                   // enable debounce
        *P_TSC_TGCR |= (0x1<<23);                   // pen down enable
```

44.2.6 Quick Reference for ADC Convert Configuration Programming

Table 44-1 provides a quick reference for programming ADC convert configurations, including the switches setting, reference voltage selection, input voltage selection and peniack setting, in different application cases, such as General ADC, 4-wire position measurement, pressure measurement and 5-wire position measurement. See Section 44.2.5, “Example Programs,” for detailed example programs and the corresponding convert configurations.

Table 44-1. Quick Reference for Convert Configuration

TICR, TCC0–7,GCC0–7															
bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	bit 0
WIPE RSW	YNL RSW	YPLLSW [1:0]		XNURSW [1:0]		XPUL SW	SEL REFP [1:0]		SEL IN [2:0]			SEL REFN [1:0]		PEN IACK	0
RESET VALUE: 0x1600															
0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0
off	off	off		off		off	YP(LL)		XP(UL)			XN(UR)		low	
General ADC from AUX0 with internal reference voltage: 0x17DC															
0	0	0	1	0	1	1	1	1	1	0	1	1	1	0	0
off	off	off		off		off	Int		INAUX0			NGND_ADC		low	
General ADC from AUX2 with external reference voltage: 0x177C															
0	0	0	1	0	1	1	1	0	1	1	1	1	1	0	0
off	off	off		off		off	Ext		INAUX2			NGND_ADC		low	
4-wire PRECHARGE: 0x158C															
0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0
off	off	off		off		high	Int		XP(UL)			NGND_ADC		low	
4-wire TOUCH-SCREEN DETECT: 0x578E															
0	1	0	1	0	1	1	1	1	0	0	0	1	1	1	0
off	low	off		off		off	Int		XP(UL)			NGND_ADC		high	
4-wire X MEASUREMENT: 0x1C90															
0	0	0	1	1	1	0	0	1	0	0	1	0	0	0	0
off	off	off		low		high	XP(UL)		YP(LL)			XN(UR)		low	
4-wire Y MEASUREMENT: 0x4604															
0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0
off	low	high		off		off	YP(LL)		XP(UL)			YN(LR)		low	
4-wire Yn MEASUREMENT: 0x0FBC															

Table 44-1. Quick Reference for Convert Configuration (continued)

0	0	0	0	1	1	1	1	1	0	1	1	1	1	0	0
off	off	high		low		off	Int		YN(LR)			NGND_ADC		low	
4-wire Xp MEASUREMENT: 0x0F8C															
0	0	0	0	1	1	1	1	1	0	0	0	1	1	0	0
off	off	high		low		off	Int		XP(UL)			NGND_ADC		low	
5-wire PRECHARGE: 0x158C															
0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0
off	off	off		off		high	Int		XP(UL)			NGND_ADC		low	
5-wire TOUCH-SCREEN DETECT: 0x978E															
1	0	0	1	0	1	1	1	1	0	0	0	1	1	1	0
low	off	off		off		off	Int		XP(UL)			NGND_ADC		high	
5-wire X MEASUREMENT: 0x4CC4															
0	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0
off	low	high		low		high	XP(UL)		WIPER			YN(LR)		low	
5-wire Y MEASUREMENT: 0x70C4															
0	1	1	1	0	0	0	0	1	1	0	0	0	1	0	0
off	low	low		high		high	XP(UL)		WIPER			YN(LR)		low	

44.3 Functional Description

44.3.1 Block Diagram

Figure 44-6 shows the block diagram of the TSC.

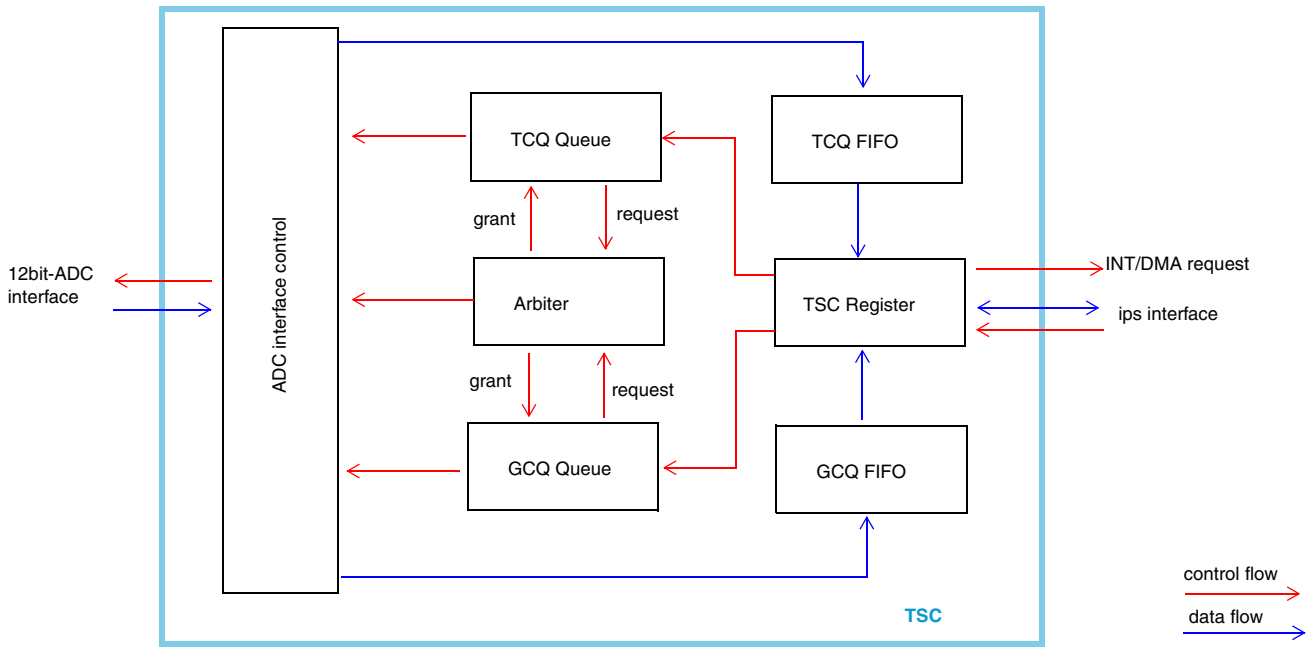


Figure 44-6. TSC Block Diagram

44.3.2 Clocks

The TSC module has one root clock, namely the ipg_clk from CCM. This eliminates the complexity of a multi-clock domain.

44.3.2.1 ipg_clk

Normally, the ipg_clk is 66.67 MHz. In low power mode, ipg_clk can be reduced to 33.33 MHz or 16.66 MHz. The user can gate on/off the TSC input ipg_clk by programming the IPG CLKEN bit (bit 0 in TGCR).

44.3.2.2 ADC Clock Generation

The TSC provides a 1.6667 MHz clock (period of 600 ns) to the ADC, which implements about 119K samples (12 bits per sample) per second. The ADC clock frequency should not be greater than 1.75 MHz. The ADC clock is sourced and divided from ipg_clk.

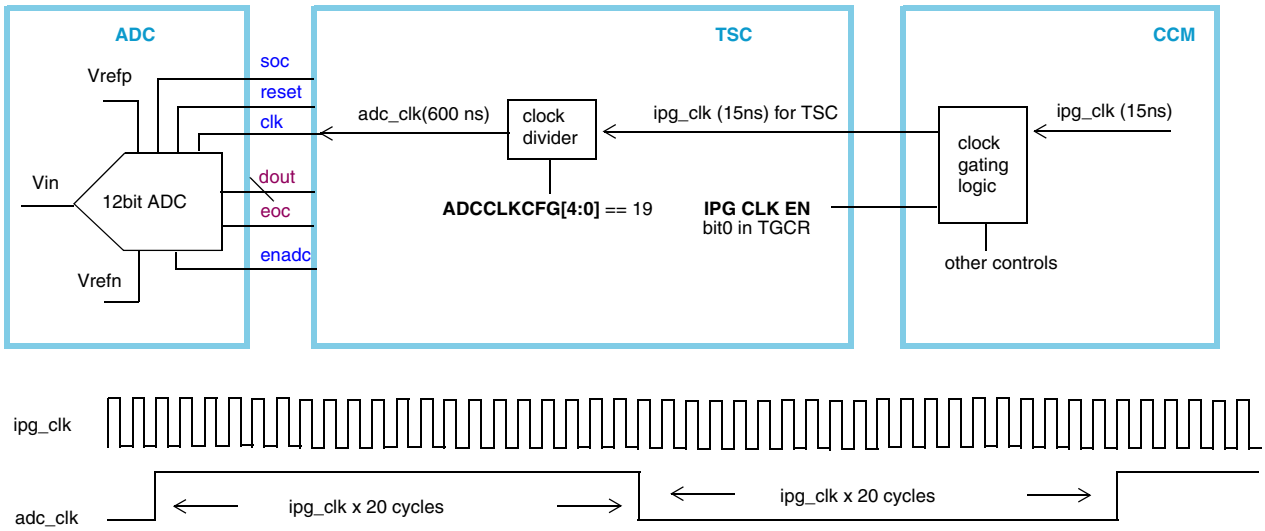


Figure 44-7. ADC Clock Generation

Table 44-2 provides configuration examples for different ipg_clk frequencies. The ADCCLKCFG bits are described in Section 44.4.3.1, "TSC General Configuration Register (TGCR)."

Table 44-2. Clock Examples

ipg_clk		ADCCLKCFG	ADC Clock	
66.67 MHz	15 ns	19	1.667 MHz	600 ns
33.33 MHz	30 ns	9	1.667 MHz	600 ns
16.67 MHz	60 ns	4	1.667 MHz	600 ns

44.3.3 Reset

There are several hardware and software resets inside TSC module. The hardware resets include the following:

- Global hardware reset signal from CCM. When this reset is asserted, all of the TSC is reset.
- Internally generated hardware self-reset signal by setting the TSCRST bit in TGCR. When this reset is asserted, all of the TSC is reset.
- Internally generated hardware self-reset signal by setting the FUNCRST bit in TGCR. When this reset is asserted, all parts, excluding the programmable registers, of the TSC is reset.

The software resets include the following:

- Internally generated software reset signal by setting the QRST bit in TCQCR. When this bit is asserted in tcq_queue, queue_state is reset to IDLE state, item_counter is reset to 0, and the TCQ request to arbiter is negated.
- Internally generated software reset signal by setting the FRST bit in TCQCR. When this reset is asserted, the write-pointer and read-pointer inside TCQ_FIFO is reset to 0.

- Internally generated software reset signal by setting the QRST bit in GCQCR. When this bit is asserted in gcq_queue, queue_state is reset to IDLE state, item_counter is reset to 0, and the GCQ request to arbiter is negated.
- Internally generated software reset signal by setting the FRST bit in TCQCR. When this reset is asserted, the write-pointer and read-pointer inside GCQ_FIFO is reset to 0.

44.3.4 ADC Power

There are three power modes for the ADC:

- Always-off mode—ADC main powered down continuously (default)
- Power saving mode—ADC main powered down when ADC is not converting.
- Always-on mode—ADC main powered up continuously

See POWERMODE[1:0] in Table 44-6 for more information. When the ADC is in always-on mode, it is in normal mode and uses approximately 2.6 mA. In always-off mode, the ADC uses approximately 10uA. Putting ADC in power-off mode while the ADC is IDLE reduces power usage. However, the power-up sequence must be repeated every time ADC power on is re-asserted, which introduces additional time for the dummy conversion (14 ADC clock cycles) before the normal conversion can begin. If the internal reference voltage is not used, turning off the internal reference voltage generator also helps reduce power usage.

44.3.5 Pen Down Detect

Pen down detection is hardware logic to determine if a pen down event is asserted. Such an event could be used to start the convert queue. This event is also used to invoke the Pen Down IRQ. Pen down detection is a different from Touch Screen detection. Touch Screen detection is a Touch Measurement step to measure the XP(UL) voltage with definitive weak pull-up and possible strong pull-down, and then judge if the panel has been touched or not.

Once the pen down event is recognized, it propagates into the TCQ/GCQ, configurable as either edge-sensitive or level-sensitive, and is captured as the PD status bit in TCQSR/GCQCR, bit0.

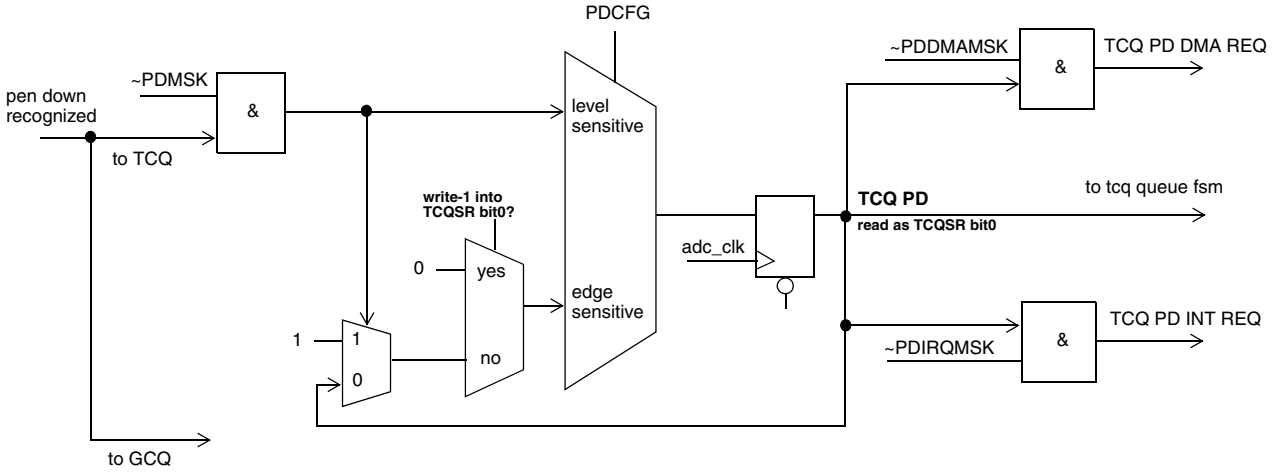


Figure 44-8. Pen Down Status, bit0 in TCQSR/GCQCR

The PD status bit, bit0 in TCQSR/GCQSR, starts the convert queue. If the PDIRQMSK bit in TCQMR/GCQMR is cleared, the interrupt request is asserted to the CPU. Similarly, if the PDDMAMSK bit is cleared, a DMA request is asserted from TSC to the SDMA.

44.3.6 Wake up from Deep Sleep Mode

Before entering into DSM, the TICR should be configured to Pre-Charge, and then to Touch-Screen Detection after some delay. After Pre-Charge and Touch-Screen-Detection in TICR, the SLPC and PDEN bits in TGCR should be asserted to enable the pen down DSM wake up logic. The ADC should be in the power-off state to reduce the power consumption.

When the i.MX25 is in DSM, the ADC clock is also shut down since ipg_clk is shut down. The 32K clock is still continuous in DSM. The PMIC (power management IC) must confirm that NVCC_ADC is still active in DSM. Once the panel/screen is touched, a TSC SLPC IRQ wakes up the device from DSM.

After the device has awakened from DSM, the corresponding interrupt service routine must clear the SLPC bit in TGCR to disable the pen down DSM wake up logic and clear the SLPC IRQ.

44.3.7 Interrupt Request & DMA Request Generation

Figure 44-9 shows how the interrupt requests and DMA requests are generated in TSC. See Section 44.4.3.5, “Queue Status Registers (TCQSR, GCQSR),” and Section 44.4.3.6, “Queue Mask Registers (TCQMR, GCQMR),” for details on the status and configuration bits.

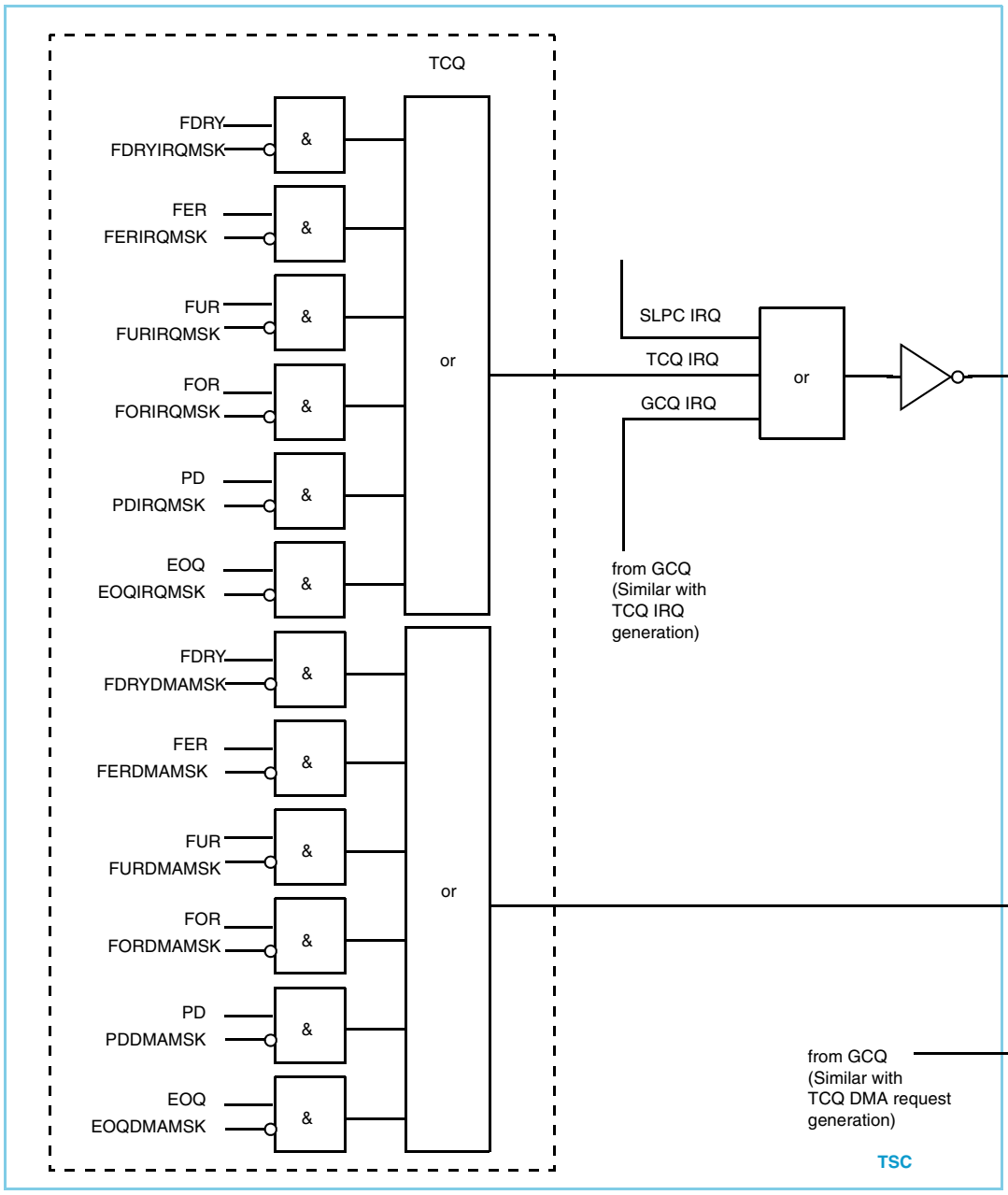


Figure 44-9. Interrupt and DMA Requests

44.3.8 LCD Noise Reduction

Touch screens are composed of two resistive layers (4-wire case), or of one conductive layer with one resistive layer (5-wire case). Because the resistive layers are usually placed close to the LCD screen, noise can be coupled from the LCD screen to the resistive layers causing errors in the touch screen measurements.

In the TSC module there is an optional to start the ADC conversion only when LCD Hsync signal is asserted. When this signal is asserted, the LCD DATA signal is normally in unchanged state. At that time, the noise coupled from LCD screen is relatively low. To enable this function, assert the HSYNCEN bit in TGCR and adjust the HSYNCPOL setting according to LCD hsync polarity. See Table 44-6.

Enabling this function reduces the sample rate, since conversion is held until LCDC Hsync is asserted. Ensure that the LCDC is enabled when this noise reduce function is enabled.

44.3.9 Queues

Touch Screen position measurement requires a special sequence, which in the TSC module, is implemented by convert queues. The convert queues apply different functions, such as Pre-charge, touch detection, X-measurement, and Y-measurement, to the ADC, see Figure 44-10.

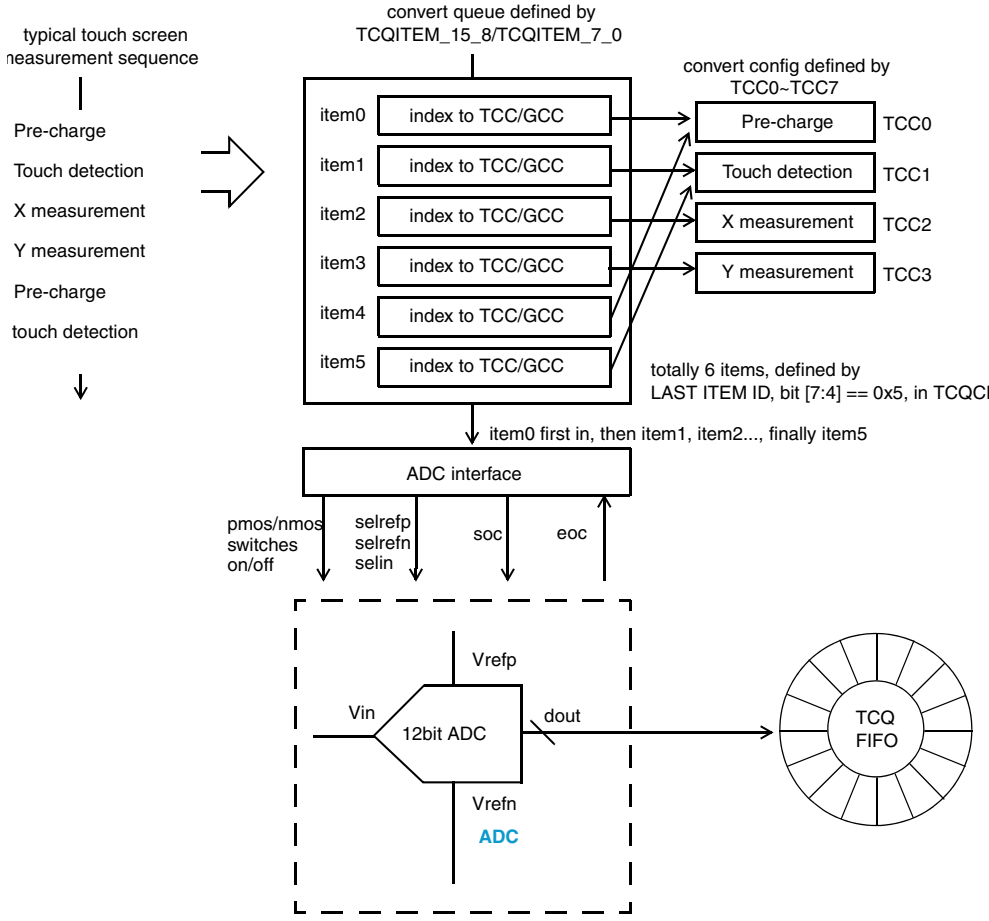


Figure 44-10. Example for Touch Screen Measurement Queue

In the TSC there are two individual queues:

- TouchScreen Convert Queue (TCQ) for 4-wire/5-wire touch screen measurement purpose
- General ADC Convert Queue (GCQ) for general measurement purpose, such as temperature, pressure, voltage, and so on.

Each queue contains 1 to 16 measurement items. The number of the items contained in a queue is defined by the LAST_ITEM_ID bit in Section 44.4.3.4, “Queue Control Registers (TCQCR, GCQCR).” An item is the element for a queue. Each item is a pointer which links to a set of convert configurations, ranged from TCC0–TCC7, and GCC0–GCC7. Each queue applies the linked configurations to the ADC one by one and stores the sample results into the corresponding FIFO.

TCQ items are defined in Section 44.4.3.7, “TCQ ITEM Registers (TCQITEM_7_0, TCQITEM_15_8),” and GCQ items are defined in Section 44.4.3.8, “GCQ ITEM Registers (GCQITEM_7_0, GCQITEM_15_8).”

TCC0–TCC7 and GCC0–GCC7 are 16 configurable settings for the ADC that controls the behavior of ADC to implement each function item.

Queue data structure is shown in Figure 44-11.

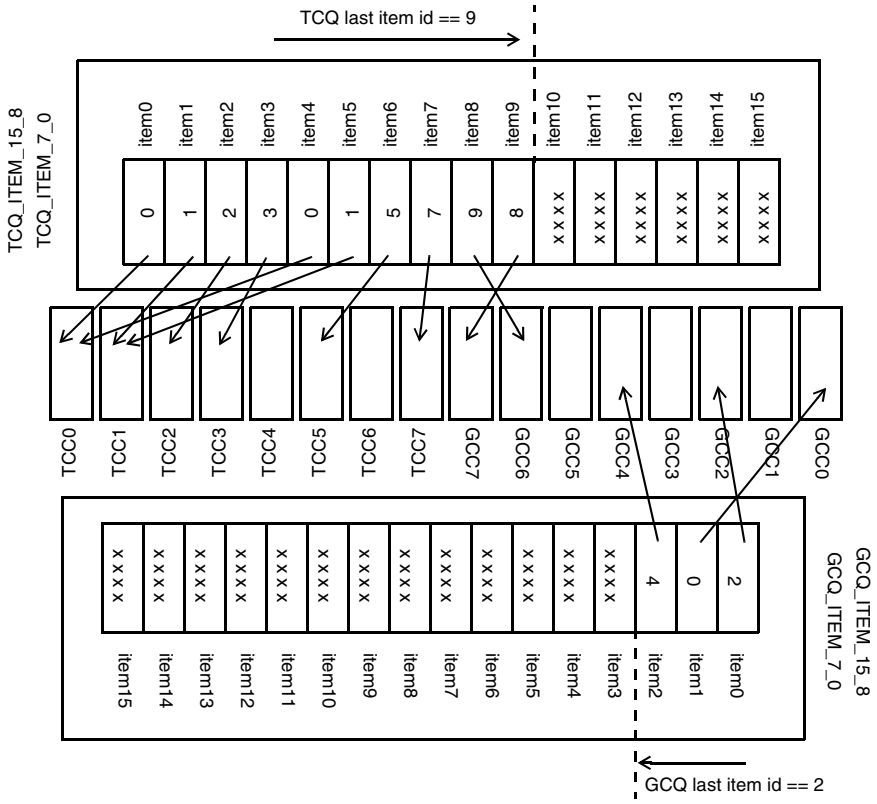


Figure 44-11. Convert Configuration Look Up

Figure 44-12 shows a timing diagram of the convert queue sequence.

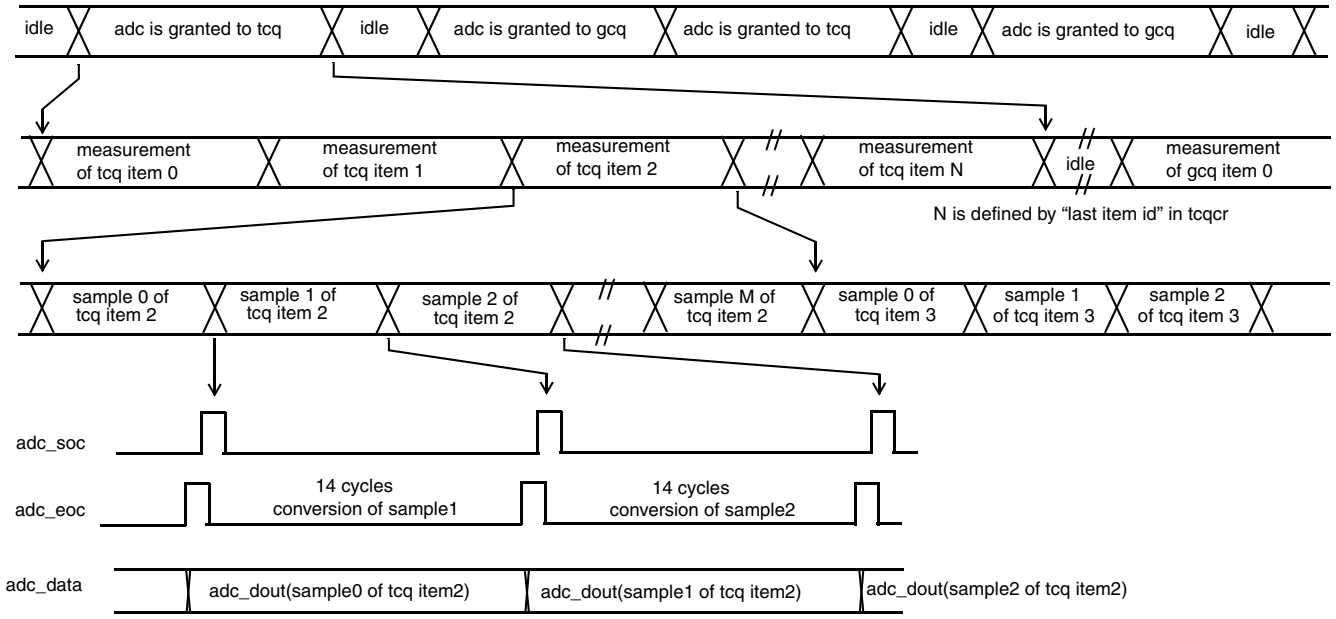


Figure 44-12. Convert Queue Timing Diagram

44.3.9.1 Queue Finite State Machine

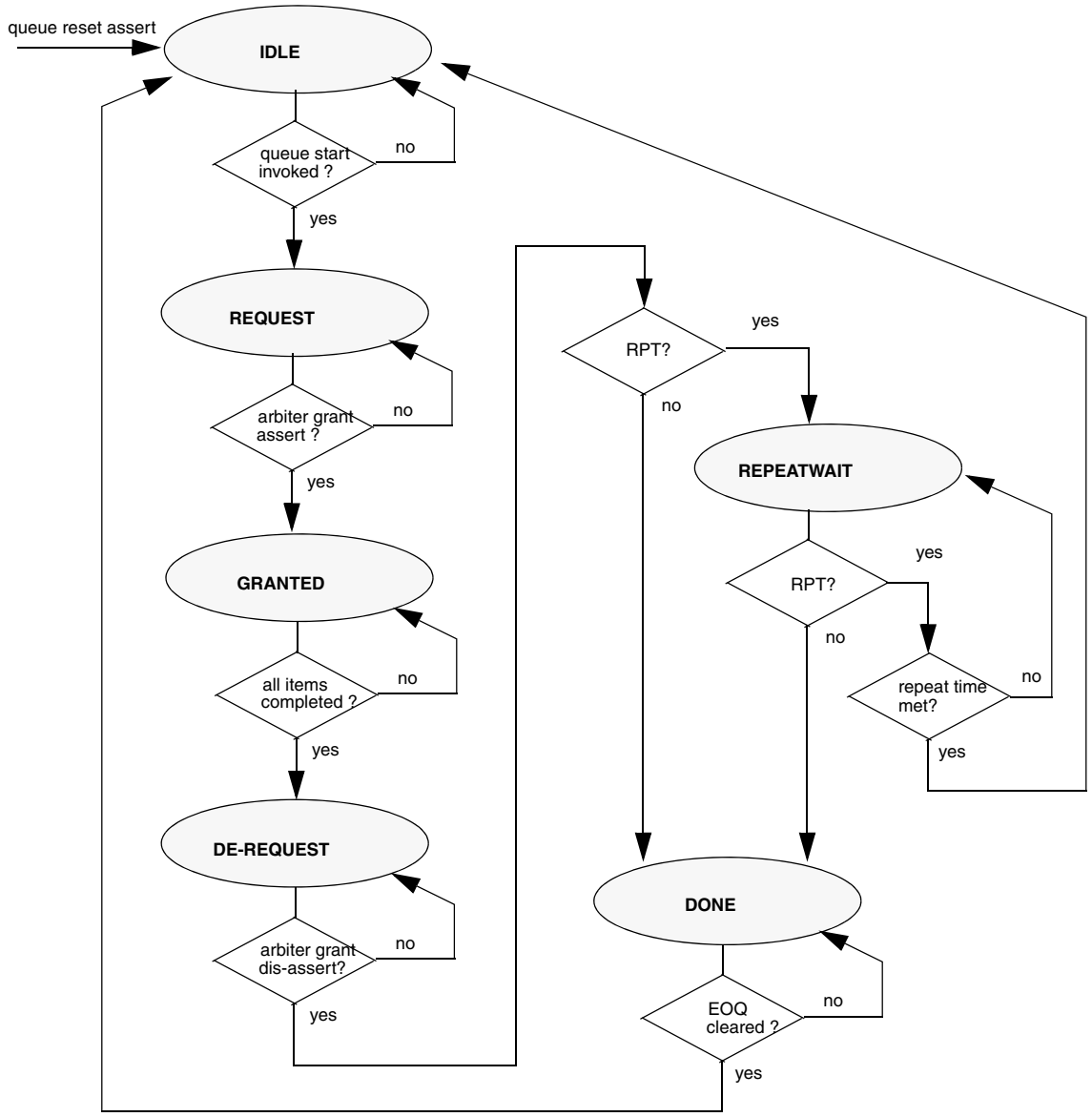


Figure 44-13. Finite State Machine Diagram

44.3.9.2 Idle State

Idle state is the default state when the queue is released from hardware or software reset. In this state, the queue does not request mastership of ADC until the queue start is invoked.

There are three ways to invoke the queue start:

- Forcing the FQS bit
- Pen down event
- Both

For more information See the descriptions of FQS and QSM bits in the queue control register.

44.3.9.3 Request State

The queue switches from the Idle state to the Request state once the queue is invoked to start. In this state, the request from queue to arbiter is asserted. The queue loops in the Request state until the grant signal from arbiter to queue is asserted.

44.3.9.4 Grant State

The queue switches from Request state to Grant state once the grant signal from arbiter to queue is asserted. In this state, the request from queue to arbiter is still asserted. An item counter starts to count from 0 to LAST_ITEM_ID. After all the items in queue are converted in ADC, queue switches from Grant state to De-Request state.

44.3.9.5 De-Request State

In this state, the request from queue to arbiter is negated. The queue loops in De-Request state until the grant signal from arbiter to queue is negated. When the grant signal is negated, the queue switches to Repeat wait state, if the RPT bit is set for queue repeat mode; else the queue switches to the Done state.

44.3.9.6 Repeatwait State

This state is for the repeat feature of the queue. The repeat feature can be used for periodic measurement application case, such as temperature monitor, voltage monitor, etc., and reduces the CPU interaction for the queue control.

44.3.9.7 Done State

In this state, the queue asserts the EOQ (end of queue) signal. The queue remains in the Done state until the EOQ bit in queue state register is cleared by writing “1” to make queue switch to Idle state.

44.3.10 Arbiter

The ADC is shared by both the TCQ and GCQ. Therefore, it is necessary for an arbiter to manage which queue, TCQ or GCQ, is granted mastership of the ADC. The ADC arbiter uses a round-robin algorithm and only changes the mastership state when the ADC is in the power-off or idle state.

44.3.10.1 Arbiter Finite State Machine

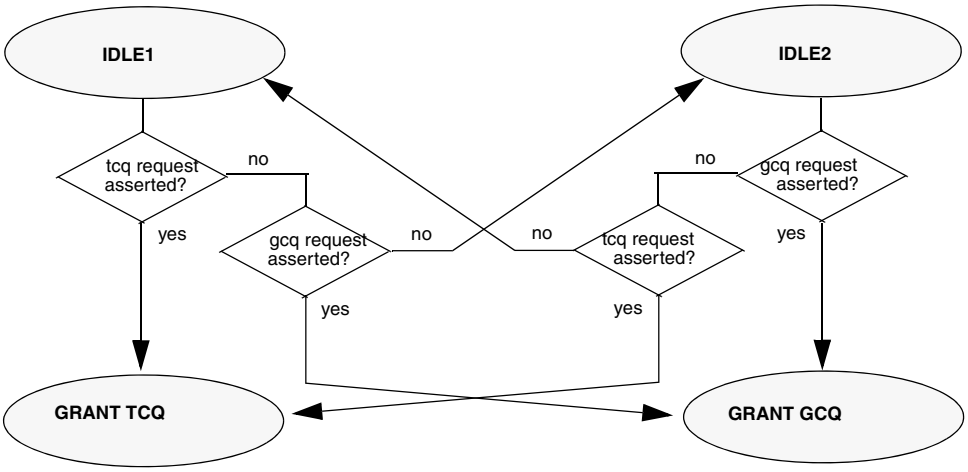


Figure 44-14. Arbiter Finite State Machine

44.3.11 FIFOs

There are two internal, independent FIFOs, each with sixteen 16-bit entries, used for storing TCQ and GCQ conversion results. These FIFOs help to pass sample data from the ADC to the CPU data bus. Both FIFOs have the same structure. Therefore all the description below applies to both the TCQ and GCQ FIFO. The FIFO data is stored in the TCQFIFO register for the TCQ and the GCQFIFO register for the GCQ as described in Section 44.4.3.3, "Queue FIFO Registers (TCQFIFO, GCQFIFO)."

Each FIFO contains 16 entries and 2 pointers. Each entry is 16 bits for storing 12-bits of data and a 4-bit item ID. Each pointer is 6 bits. One pointer is for write purposes and the other pointer is for read purposes.

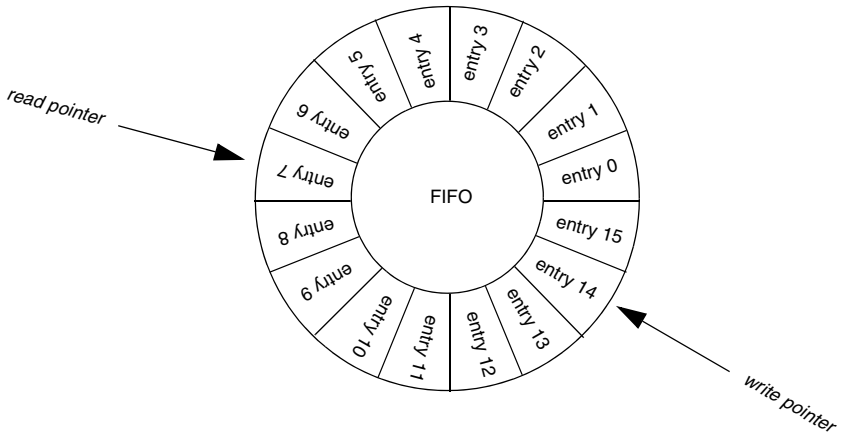
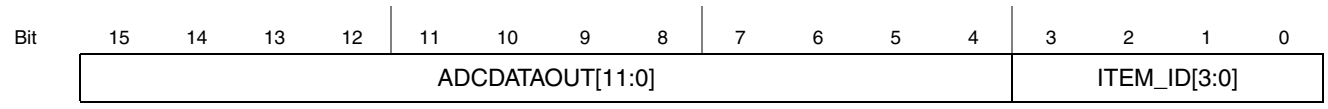


Figure 44-15. FIFOs

The 16 bit FIFO entry format is shown below.



The write pointer always points to the next entry to be written. On a FIFO write operation, if the FIFO is not full, the FIFO entry location pointed to by the write pointer is written, and then the write pointer is incremented to point to the next location to be written. This process is known as write then increase, write before increase or increase after write. On reset, write pointer is set to zero.

The read pointer always points to the current FIFO entry to be read. On a FIFO read operation, if the FIFO is not empty, the FIFO word location pointed to by the read pointer is read, and then the read pointer is incremented to point to the next location to be read. This process is known as read then increase, read before increase or increase after read. Similarly, on reset, the read pointer is reset to zero.

44.4 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers. For the base address of the TSC module, see the system memory map.

44.4.1 Memory Map

Table 44-3 shows the TSC memory map.

Table 44-3. TSC Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (TGCR)	TSC General Config Register	R/W	0x0013_0000	44.4.3.1/44-30
0x0004 (TGSR)	TSC General Status Register	RO	0x0000_0000	44.4.3.2/44-32
0x0008 (TICR)	TSC IDLE Config Register	R/W	0x0000_1600	44.4.3.9/44-41
0x0400 (TCQFIFO)	Touch Screen Convert Queue FIFO Register	RO	0x0000_*nnn*0	44.4.3.3/44-33
0x0404 (TCQCR)	Touch Screen Convert Queue Control Register	R/W	0x0004_0700	44.4.3.4/44-33
0x0408 (TCQSR)	Touch Screen Convert Queue Status Register	RO	0x0000_2000	44.4.3.5/44-35
0x040C (TCQMR)	Touch Screen Convert Queue Mask Register	R/W	0xFFFF_FFFF	44.4.3.6/44-36
0x0420 (TCQITEM_7_0)	Touch Screen Convert Queue ITEM 7~0	R/W	0x0000_0000	44.4.3.7/44-38
0x0424 (TCQITEM_15_8)	Touch Screen Convert Queue ITEM 15~8	R/W	0x0000_0000	44.4.3.7/44-38
0x0440–0x045C (TCC0–7)	Touch Screen Convert Config 0–7	R/W	0x0000_1600	44.4.3.9/44-41
0x0800 (GCQFIFO)	General ADC Convert Queue FIFO Register	RO	0x0000_*nnn*0	44.4.3.3/44-33
0x0804 (GCQCR)	General ADC Convert Queue Control Register	R/W	0x0004_0700	44.4.3.4/44-33
0x0808 (GCQSR)	General ADC Convert Queue Status Register	RO	0x0000_2000	44.4.3.5/44-35
0x080C (GCQMR)	General ADC Convert Queue Mask Register	R/W	0xFFFF_FFFF	44.4.3.6/44-36

Table 44-3. TSC Memory Map (continued)

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0820 (GCQITEM_7_0)	General ADC Convert Queue ITEM 7~0	R/W	0x0000_0000	44.4.3.8/44-39
0x0824 (GCQITEM_15_8)	General ADC Convert Queue ITEM 15~8	R/W	0x0000_0000	44.4.3.8/44-39
0x0840–0x085C (GCC0–7)	General ADC Convert Config 0–7	R/W	0x0000_1600	44.4.3.9/44-41

44.4.2 Register Summary

Figure 44-16 shows the key to register tables and Table 44-4 shows the register figure conventions.

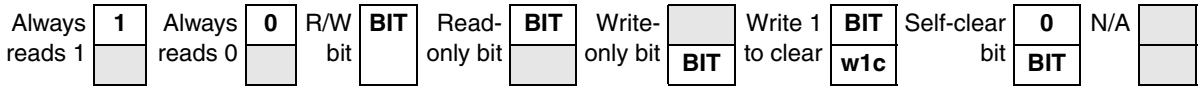


Figure 44-16. Key to Register Fields

Table 44-4. Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 44-5 shows the TSC register summary table.

Table 44-5. TSC Register Summary

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 TGCR	R	PDBTIME[6:0]							PDB EN	PD EN	0	0	ADCCLKCFG[4:0]				
	W																
	R	0	0	0	0	0	INT REF EN	POWER MODE[1:0]		HSY NCP OL	HSY NCE N	STL C	SLP C	0	0	0	IPG CLK EN
	W														FUN C RST	TSC RST	
0x004 TGSR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GCQ DMA	TCQ DMA
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLP INT	GCQ INT	TCQ INT
	W																
0x008 TICR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WIP ERSW	YNL RSW	YPLLSW[1:0]		XNURSW[1:0]		XPU LSW	SEL REFP[1:0]		SELIN[2:0]			SEL REFN[1:0]		PEN IACK	0
	W																
0x400 TCQFIFO	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ADC_DATAOUT[11:0]												ITEM_ID[3:0]			
	W																
0x404 TCQCR	R	0	0	0	0	0	0	0	0	0	0	0	0	PD CFG	PD MSK	FRS T	QRS T
	W																
	R	REPEATWAIT[3:0]				FIFOWATERMARK[3:0]				LAST_ITEM_ID[3:0]				RPT	FQS	QSM[1:0]	
	W																
0x408 TCQSR	R	0	0	0	0	0	0	FWPTR[4:0]					FRPTR[4:0]				
	W																
	R	FRD Y	FUL L	EMP T	FDN[4:0]					0	FER	FUR	FOR	0	0	EOQ	PD
	W										w1c	w1c	w1c			w1c	w1c

Table 44-5. TSC Register Summary (continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40C TCQMR	R	FDRY DMAMSK	0	0	0	0	0	0	0	0	FER DMA MSK	FUR DMA MSK	FOR DMA MSK	0	0	EOQ DMA MSK	PD DMA MSK
	W																
	R	FDRY IRQMSK	0	0	0	0	0	0	0	0	FER IRQ MSK	FUR IRQ MSK	FOR IRQ MSK	0	0	EOQ IRQ MSK	PD IRQ MSK
	W																
0x420 TCQ_ITEM_7_0	R	ITEM7				ITEM6				ITEM5				ITEM4			
	W																
	R	ITEM3				ITEM2				ITEM1				ITEM0			
	W																
0x424 TCQ_ITEM_15_8	R	ITEM15				ITEM14				ITEM13				ITEM12			
	W																
	R	ITEM11				ITEM10				ITEM9				ITEM8			
	W																
0x440 TCC0 ~ 0x45C TCC7	R	SETTLING_TIME[7:0]								0	0	0	IGS	NOS[3:0]			
	W																
	R	WIPERSW	YNLRSW	YPLLSW[1:0]		XNURSW[1:0]		XPULSW	SELREFP[1:0]		SELIN[2:0]			SELREFN[1:0]		PENIACK	0
	W																
0x800 GCQFIFO	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ADC_DATAOUT[11:0]												ITEM_ID[3:0]			
	W																
0x804 GCQCR	R	0	0	0	0	0	0	0	0	0	0	0	0	PDCFG	PDMSK	FRST	QRST
	W																
	R	REPEATWAIT[3:0]				FIFOWATERMARK[3:0]				LAST_ITEM_ID[3:0]				RPT	FQS	QSM[1:0]	
	W																

Table 44-5. TSC Register Summary (continued)

Base Address Offset (Register Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x808 GCQSR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	FDRY	FULL	EMPT	FDN[4:0]					0	FER	FUR	FOR	0	0	EOQ	PD
	W										w1c	w1c	w1c			w1c	w1c
0x80C GCQMR	R	FDRY DMAMSK	0	0	0	0	0	0	0	0	FOR DMA MSK	FUR DMA MSK	FOR DMA MSK	0	0	EOQ DMA MSK	PD DMA MSK
	W																
	R	FDRY IRQMSK	0	0	0	0	0	0	0	0	FER IRQ MSK	FUR IRQ MSK	FOR IRQ MSK	0	0	EOQ IRQ MSK	PD IRQ MSK
	W																
0x820 GCQ_ITEM_7_0	R	ITEM7				ITEM6				ITEM5				ITEM4			
	W																
	R	ITEM3				ITEM2				ITEM1				ITEM0			
	W																
0x820 GCQ_ITEM_15_8	R	ITEM15				ITEM14				ITEM13				ITEM12			
	W																
	R	ITEM11				ITEM10				ITEM9				ITEM8			
	W																
0x840 GCC0 ~ 0x85C GCC7	R	SETTLING_TIME[7:0]								0	0	0	IGS	NOS[3:0]			
	W																
	R	WIPERSW	YNLRSW	YPLLSW[1:0]		XNURSW[1:0]		XPULSW	SEL REFP[1:0]		SELIN[2:0]			SEL REFN[1:0]		PENIACK	0
	W																

44.4.3 Register Descriptions

This section contains the register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams in bit order.

44.4.3.1 TSC General Configuration Register (TGCR)

Offset 0x0000 (TGCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PDBTIME[6:0]							PDB EN	PD EN	0	0	ADCCLKCFG[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	INTR EFEN	POWERMO DE[1:0]		HSYN CPOL	HSYN CEN	STLC	SLPC	0	0	0	IPG CLK EN
W														FUNC RST	TSC RST	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-17. TSC General Configuration Register

Table 44-6. TSC General Configuration Register Field Descriptions

Field	Description
31–25 PDBTIME	Pen debounce time. See Section 44.3.5, “Pen Down Detect”. 0 Pen debounce delay is 1 x 8 cycles of ADC clock 1 Pen debounce delay is 2 x 8 cycles of ADC clock … 127 Pen debounce delay is 128 x 8 cycles of ADC clock
24 PDBEN	Pen debounce enable. See Section 44.3.5, “Pen Down Detect.” 0 Pen debounce disabled 1 Pen debounce enabled
23 PDEN	Pen down detect enable. See Section 44.3.5, “Pen Down Detect.” 0 Pen Down detect disabled 1 Pen Down detect enabled
22–21	Reserved
20–16 ADCCLKCFG	ADC clock configuration. Program ADC clock based on ipg_clk from CCM. See Section 44.3.2.2, “ADC Clock Generation.” 0 ADC clock = ipg_clk / 10 1 ADC clock = ipg_clk / 10 2 ADC clock = ipg_clk / 10 3 ADC clock = ipg_clk / 10 4 ADC clock = ipg_clk / (2*4+2) 5 ADC clock = ipg_clk / (2*5+2) ... 31 ADC clock = ipg_clk / (2*31+2) **Note:** The ADC clock frequency should not be greater than 1.75 MHz.
15–11	Reserved.
10 INTREFEN	Internal reference enable. Enables the internal 2.5V reference so that the ADC can use the on-chip reference as the positive reference. 0 Internal 2.5V reference always off 1 Internal 2.5V reference on/off as same as ADC main power, according to POWERMODE[1:0]

Table 44-6. TSC General Configuration Register Field Descriptions (continued)

Field	Description
9–8 POWERMODE	Power mode. See Section 44.3.4, “ADC Power.” 00 Always-off mode—ADC main powered down continuously (default) 01 Power saving mode—ADC main powered down when ADC is not converting. 1x Always-on mode—ADC main powered up continuously
7 HSYNCPOL	LCD hsync polarity. This bit adjusts the signal polarity of LCD Hsync. See Section 44.3.8, “LCD Noise Reduction.” 0 “1/high” means “LCD Hsync asserted”; and “0/low” means “LCD Hsync negated”. 1 “0/low” means “LCD Hsync asserted”; and “1/high” means “LCD Hsync negated”.
6 HSYNCEN	LCD hsync enable. To reduce the effects of noise from LCD, this bit prevents the ADC Vin sampling by negating adc_soc until LCD Hsync is asserted. See Section 44.3.8, “LCD Noise Reduction.” 0 adc_soc is asserted regardless to LCD Hsync 1 adc_soc is not asserted until LCD Hsync is asserted
5 STLC	Settling configuration. This bit increases the settling time (about 12 cycles of the ADC clock) by switching the item to the next one, once the Vin (ADC input voltage to be measured) is latched internally by the ADC. **Note:** This configuration should not be asserted unless all the conversions’ refp (positive reference voltage) and refn (negative reference voltage) are all based on stable reference source, such as external reference, internal reference or agnd. Any conversions with dynamic reference source, such as differential conversion whose refp from YP(LL)/XP(UL) or refn from XN(UR)/YN(LR), results in incorrect measurement results if this bit is asserted. 0 The ADC input voltage source is not be switched to next item until the conversion is completed 1 The ADC input voltage source is switched to next item, once the Vin was latched by the ADC internally
4 SLPC	Sleep configuration. This bit enables a SLPC interrupt (invoked by pen down) when device in sleep mode. See Section 44.3.6, “Wake up from Deep Sleep Mode.” 0 TSC interrupt logic operates in normal mode 1 TSC interrupt logic operates in sleep mode. A pen touch leads to an interrupt to propagate to CCM/ASIC to wake up the device from sleep mode
3	Reserved
2 FUNC RST	TSC function reset (self-clearing bit). Writing “1” to this bit forces the TSC module into reset state, except the programmable registers are not changed. The TSC is released from reset state when this bit self-clears to “0”. See Section 44.3.3, “Reset.”
1 TSC RST	TSC module reset (self-clearing bit). Writing “1” into this bit forces entire TSC module into reset state, except the IPG CLK EN bit is not changed. The TSC module is released from reset state when this bit self-clears to “0”. See Section 44.3.3, “Reset.”
0 IPG CLK EN	TSC IPG clock enable. Clearing this bit gates off the ipg_clk into the TSC. See Section 44.3.2, “Clocks.” **Note:** Programming this bit informs the CCM that the TSC ipg clock is to be gated-off or turn-on. The ipg clock gating logic is implemented and controlled inside the CCM. There are other register bits inside CCM that control the TSC ipg clock gating behavior together with this TSC bit. 0 ipg_clk for TSC is to be gated off 1 ipg_clk for TSC is to be turn on

44.4.3.2 TSC General Status Register (TGSR)

Offset 0x0004 (TGSR) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GCQ DMA	TCQ DMA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	SLP INT	GCQ INT	TCQ INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-18. TSC General Status Register

Table 44-7. TSC General Status Register Field Descriptions

Field	Description
31–18	Reserved.
17 GCQ DMA	General ADC-convert-queue DMA request status. 0 GCQ DMA request asserted 1 GCQ DMA request negated
16 TCQ DMA	Touchscreen-convert-queue DMA request status. 0 TCQ DMA request asserted 1 TCQ DMA request negated
15-3	Reserved.
2 SLP INT	Sleep interrupt status. 0 Sleep interrupt asserted 1 Sleep interrupt negated
1 GCQ INT	General ADC convert queue interrupt status. 0 GCQ interrupt asserted 1 GCQ interrupt negated
0 TCQ INT	Touch screen convert queue interrupt status. 0 TCQ interrupt asserted 1 TCQ interrupt negated

44.4.3.3 Queue FIFO Registers (TCQFIFO, GCQFIFO)

Offset 0x0400 (TCQFIFO)
0x0800 (GCQFIFO)

Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADCDATAOUT[11:0]												ITEM_ID[3:0]			
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	0	0	0	0

Figure 44-19. Queue FIFO Registers

Table 44-8. Queue FIFO Registers Field Descriptions

Field	Description
31–16	Reserved.
15–4 ADCDATAOUT	ADC dataout. Contains the 12-bit ADC dataout.
3–0 ITEM_ID	Item ID. Specifies the Item ID (Item0, Item1, Item2, Item3... Item15). The ID ranges from 0 to 15, corresponding to the conversion associated with the bit result.

44.4.3.4 Queue Control Registers (TCQCR, GCQCR)

Offset 0x0404 (TCQCR)
0x0804 (GCQCR)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	PD CFG	PD MSK	FRST	QRST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REPEATWAIT[3:0]				FIFOWATERMARK[3:0]				LAST_ITEM_ID[3:0]				RPT	FQS	QSM[1:0]	
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 44-20. Queue Control Registers

Table 44-9. Queue Control Registers Field Descriptions

Field	Description
31–20	Reserved
19 PDCFG	Pen down config. This bit sets the pen down event as level or edge sensitive. 1 Level sensitive 0 Edge sensitive
18 PDMSK	Pen down mask. This bit masks the pen down event, so that if a pen down occurs, the QSM does not start the queue in relevant mode. 1 Pen down masked 0 Pen down unmasked
17 FRST	FIFO reset. Setting this bit causes a software reset of the queue FIFO.
16 QRST	Queue reset. Setting this bit causes a software reset of all the queue logic, excluding queue FIFO.
15–12 REPEATWAIT	Wait time between repeated queues in repeat mode. This parameter is only useful when QSM[1:0] is set to 2'b11 (repeat mode). 0 Repeat wait latency is equal to 1 ADC clock 1 Repeat wait latency is equal to 2 ADC clocks 2 Repeat wait latency is equal to 4 ADC clocks 3 Repeat wait latency is equal to 8 ADC clocks 4 Repeat wait latency is equal to 16 ADC clocks 5 Repeat wait latency is equal to 32 ADC clocks ... 15 Repeat wait latency is equal to 32,768 ADC clocks
11–8 FIFOWATERMARK	FIFO Watermark. Programmed to values between 0 and 15. This value corresponds to watermark levels between 1 and 16, respectively. When the FIFO fills to this level, the FIFO watermark flag (FDRY bit in Queue Status Register) is asserted. This flag generates an interrupt and/or a DMA request. 0 FDRY is asserted while FDN is equal to or greater than 1 1 FDRY is asserted while FDN is equal to or greater than 2 2 FDRY is asserted while FDN is equal to or greater than 3 3 FDRY is asserted while FDN is equal to or greater than 4 ... 14 FDRY is asserted while FDN is equal to or greater than 15 15 FDRY is asserted while FDN is equal to 16
7–4 LAST_ITEM_ID	Last queue item ID. There are 16 Items (Item0, Item1,... Item15) for convert queue. Last Item ID specifies which Items are used. For example, if the Last Item ID is 9, the convert queue is "Item0, Item1, Item2... Item9" (totally 10 items) and the rest items after Item9 are ignored.
3 RPT	Repeat. 0 The conversion queue does not repeat 1 The conversion queue repeats
2 FQS	Force queue start. 0 FQS does not start the conversion queue 1 FQS starts the conversion queue, if QSM = 10 or 11.
1–0 QSM	Queue start mode. 00 Queue stop 01 Pen-Down invokes new conversion queue 10 New conversion queue is started by FQS setting 11 FQS setting or Pen-Down starts new conversion queue

44.4.3.5 Queue Status Registers (TCQSR, GCQSR)

Offset 0x0408 (TCQSR)
0x0808 (GCQSR)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	FWPTR[4:0]					FRPTR[4:0]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FDRY	FULL	EMPT	FDN[4:0]					0	FER	FUR	FOR	0	0	EOQ	PD
W										w1c	w1c	w1c			w1c	w1c
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-21. Queue Status Registers

Table 44-10. Queue Status Registers Field Descriptions

Field	Description
31–26	Reserved.
25–21 FRPTR	FIFO write pointer.
20–16 FRPTR	FIFO read pointer.
15 FDRY	FIFO data ready. See Queue Control Registers, “FIFOWATERMARK”, on page 44-34 and Queue Status Registers, “FDN”, on page 44-35” and Queue Status Registers 0 FDN < FIFOWATERMARK+1. The number of data in the FIFO has not reach the FIFO watermark. 1 FDN ≥ FIFOWATERMARK+1. The number of data in the FIFO has reached the FIFO watermark.
14 FULL	FIFO full. This is a redundant flag used for software polling and is set if FDN is equal to 16. 0 FIFO is not full 1 FIFO is full
13 EMPT	FIFO empty. This is a redundant flag used for software polling and is set if FDN is equal to 0. 0 FIFO is not empty 1 FIFO is empty
12–8 FDN	FIFO data number. FDR indicates amount of data in the FDN FIFO. 00000 0 x 16-bit data in FIFO 000011 x 16-bit data in FIFO … 0111115 x 16-bit data in FIFO 1000016 x 16-bits data in FIFO 10001 –11111Reserved.
7	Reserved

Table 44-10. Queue Status Registers Field Descriptions (continued)

Field	Description
6 FRR	FIFO error. Indicates that FDN is in error state. This is a redundant flag, and is equal to (FDN[4:0] > 16). Write 1 to clear this bit. 0 FDN[4:0] ≤ 16 1 FDN[4:0] > 16
5 FUR	FIFO underrun. This bit is set if a read of the FIFO is attempted when the FIFO is empty. Write 1 to clear this bit. 0 FIFO underrun has not occurred 1 FIFO underrun has occurred
4 FOR	FIFO overrun. This bit is set if a write to the FIFO is attempted when the FIFO is full. Write 1 to clear this bit. 0 FIFO overrun has not occurred 1 FIFO overrun has occurred
3–2	Reserved.
1 EOQ	End of queue. 0 Convert queue is not complete 1 Convert queue is completed
0 PD	Pen down 0 Pen down is not asserted 1 Pen down is asserted

44.4.3.6 Queue Mask Registers (TCQMR, GCQMR)

Offset 0x040C (TCQMR)
0x080C (GCQMR)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FDRY DMA MSK	1	1	1	1	1	1	1	1	FER DMA MSK	FUR DMA MSK	FOR DMA MSK	1	1	EOQ DMA MSK	PD DMA MSK
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FDRY IRQM SK	1	1	1	1	1	1	1	1	FER IRQ MSK	FUR IRQ MSK	FOR IRQ MSK	1	1	EOQ IRQM SK	PD IRQ MSK
W																
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 44-22. Queue Mask Register

Table 44-11. Queue Mask Register Field Descriptions

Field	Description
31 FDRY DMA MASK	FIFO data ready (FDRY) DMA mask. 0 FDRY DMA is not masked 1 FDRY DMA is masked
30–23	Reserved
22 FER DMA MASK	FIFO error (FER) DMA mask 0 FER DMA is not masked 1 FER DMA is masked
21 FUR DMA MASK	FIFO underrun DMA mask 0 FIFO underrun DMA is not masked 1 FIFO underrun DMA is masked
20 FOR DMA MASK	FIFO overrun DMA mask 0 FIFO overrun DMA is not masked 1 FIFO overrun DMA is masked
19–18	Reserved
17 EOQ DMA MASK	End-of-queue DMA mask 0 End-of-queue DMA is not masked 1 End-of-queue DMA is masked
16 PD DMA MASK	Pen down DMA mask 0 Pen down DMA is not masked 1 Pen down DMA is masked
15 FDRY IRQ MASK	FIFO data ready (FDRY) IRQ mask 0 FDRY IRQ is not masked 1 FDRY IRQ is masked
14–7	Reserved
6 FER IRQ MASK	FIFO error (FER) IRQ mask 0 FER IRQ is not masked 1 FER IRQ is masked
5 FUR IRQ MASK	FIFO underrun IRQ mask 0 FIFO underrun IRQ is not masked 1 FIFO underrun IRQ is masked
4 FOR IRQ MASK	FIFO overrun IRQ mask 0 FIFO overrun IRQ is not masked 1 FIFO overrun IRQ is masked
3–2	Reserved
1 EOQ IRQ MASK	End-of-queue IRQ mask 0 End-of-queue IRQ is not masked 1 End-of-queue IRQ is masked
0 PD IRQ MASK	Pen down IRQ mask 0 Pen down IRQ is not masked 1 Pen down IRQ is masked

44.4.3.7 TCQ ITEM Registers (TCQITEM_7_0, TCQITEM_15_8)

Offset 0x0420 (TCQITEM_7_0) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ITEM7				ITEM6				ITEM5				ITEM4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ITEM3				ITEM2				ITEM1				ITEM0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-23. TCQ Item_7_0 Register

Offset 0x0424 (TCQITEM_15_8) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ITEM15				ITEM14				ITEM13				ITEM12			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ITEM11				ITEM10				ITEM9				ITEM8			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-24. TCQ Item_15_8 Register

TCQITEM_7_0 and TCQITEM_15_8 contain 16 4-bit fields as show in Figure 44-23 and Figure 44-24. Each four-bit field value indicates the convert configuration register to be used for one of the 16 items (ITEM15–0).

NOTE

TCQITEMs 4-bit field pointer decoding is different with GCQITEMs.

Table 44-12. Four-bit Field Pointer Definitions for TCQ Items

Four-bit Field Pointer	Convert Config
0000	TCC0
0001	TCC1
0010	TCC2

Table 44-12. Four-bit Field Pointer Definitions for TCQ Items (continued)

Four-bit Field Pointer	Convert Config
0011	TCC3
0100	TCC4
0101	TCC5
0110	TCC6
0111	TCC7
1000	GCC7
1001	GCC6
1010	GCC5
1 011	GCC4
1100	GCC3
1101	GCC2
1110	GCC1
1111	GCC0

44.4.3.8 GCQ ITEM Registers (GCQITEM_7_0, GCQITEM_15_8)

Offset 0x0820 (GCQITEM_7_0) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ITEM7				ITEM6				ITEM5				ITEM4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ITEM3				ITEM2				ITEM1				ITEM0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-25. GCQ Item_7_0 Register

Offset 0x0824 (GCQITEM_15_8) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ITEM15				ITEM14				ITEM13				ITEM12			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ITEM11				ITEM10				ITEM9				ITEM8			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 44-26. GCQ Item_15_8 Register

GCQITEM_7_0 and GCQITEM_15_8 contain 16 4-bit fields as show in Figure 44-23 and Figure 44-24. Each field value indicates the convert configuration register used for one of the 16 items (ITEM15–0).

NOTE

GCQITEMs 4-bit field pointer decoding is different with TCQITEMs.

Table 44-13. four-bit field pointer define for GCQ Item

Four-bit Field Pointer	Convert Config
0000	GCC0
0001	GCC1
0010	GCC2
0011	GCC3
0100	GCC4
0101	GCC5
0110	GCC6
0111	GCC7
1000	TCC7
1001	TCC6
1010	TCC5
1 011	TCC4
1100	TCC3
1101	TCC2
1110	TCC1
1111	TCC0

44.4.3.9 Convert Configuration Registers (TICR, TCC0–7, GCC0–7)

TICR, TCC0–TCC7, and GCC0–GCC7 are programmable registers for convert configuration lookup. TICR defines all the setting of switches, positive reference voltage, negative reference voltage, ADC input voltage source and PENIACK when the ADC is in idle state. TCC0–TCC7 are used to store different convert configuration options for touch screen usage case, such as 4-wire pre-charge, 4-wire touch detection, 4-wire X measurement, 4-wire Y measurement, 5-wire pre-charge, etc. GCC0–GCC7 are used to store different convert configuration options for general ADC usage cases, such as General ADC measurement using INAUX0, INAUX1, INAUX2, etc.

Offset 0x0008 (TICR)
0x0440–0x045C (TCC0–7)
0x0840–0x085C (GCC0–7)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SETTLING_TIME[7:0]								0	0	0	IGS	NOS[3:0]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WIPE RSW	YNLR SW	YPLLSW[1:0]		XNURSW[1:0]		XPUL SW	SEL REFP[1:0]		SELIN[2:0]			SEL REFN[1:0]		PEN IACK	0
W																
Reset	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0

Figure 44-27. Convert Configuration Registers

Table 44-14. TSQ Configure Register Field Descriptions

Field	Description
31–24 SETTLING_TIME	Settling time. Specifies the number of ADC clock cycles allocated for the input signal to settle to within required accuracy before conversion starts. The exact number of clock cycles is (SETTLING_TIME[7:0] x 8 +1).
23–21	Reserved
20 IGS	Ignore sample data. Indicates if the sample data of the current convert item is to be put into the FIFO or not. 0 Sample data of current item is saved in queue FIFO 1 Sample data of current item is not saved in queue FIFO
19–16 NOS	Number of samples. Indicates the number (NOS+1) of samples which are continuously converted/sampled. 0 1 sample is converted/sampled 1 2 samples are continuously converted/sampled 2 3 samples are continuously converted/sampled 3 4 samples are continuously converted/sampled … 15 16 samples are continuously converted/sampled
15 WIPERSW	Wiper switch control. 0 WIPER off 1 WIPER low.

Table 44-14. TSQ Configure Register Field Descriptions (continued)

Field	Description
14 YNLRSW	YNLR switch control. 0 YN(LR) off 1 YN(LR) low
13–12 YPLLSW	YPLL switch control. 00 YP(LL) high 01 YP(LL) off 10 Reserved 11 YP(LL) low
11–10 XNURSW	XNUR switch control. 00 XN(UR) high 01 XN(UR) off 10 Reserved 11 XN(UR) low
9 XPULSW	XPUL switch control. 0 XP(UL) high 1 XP(UL) off
8–7 SEL REFP	Positive reference selection. Selects positive reference voltage to the ADC. 00 YP(LL) 01 XP(UL) 10 External reference 11 Internal reference
6–4 SEL IN	Channel selection. Selects the ADC input voltage source. 000 XP(UL) 001 YP(LL) 010 XN(UR) 011 YN(LR) 100 WIPER 101 INAUX0 110 INAUX1 111 INAUX2
3–2 SEL REFN	Negative reference select. Selects the negative reference voltage to the ADC. 00 XN(UR) 01 YN(LR) 10 NGND_ADC 11 NGND_ADC
1 PENIACK	Pen-down acknowledge signal for XP(UL). 0 Pen-down acknowledge negated for XP(UL) 1 Pen-down acknowledge asserted for XP(UL)
0	Reserved

Chapter 45
UTMI-USB-PHY

45.1 Reference Documentation

[1] USB Specs. Revision 2.0 Apr27, 2000

[2] USB 2.0 Transceiver Macrocell Interface (UTMI) Specification Version 1.05 Mar29, 2001

[3] On-The-Go Supplement to the USB2.0 specification Revision 1.0a June24, 03

[4] USB Engineering Change Notice for Pull-up/Pull-down Resistors

45.2 Acronyms, and Abbreviations

Table 45-1. Acronyms and Abbreviations

Acronym	Meaning
USB XCVR <=> transceiver	Universal Serial Bus
UTMI	Universal Transceiver Macrocell Interface
ULPI	UTMI+ Low Pin Interface
OTG	On The Go
HS	High Speed
FS	Full Speed
LS	Low Speed
PHY	Physical Layer

45.3 Overview

The Universal Serial Bus specification defines the transmitter and receiver characteristics of the transceiver. The High Speed USB Phy (Physical Layer) is designed for use in SoC applications. Main features of the High Speed USB Transceiver are as follows:

- Compliance with Universal Serial Bus Specification Rev. 2.0
- UTMI interface with Link Controller
- Supports HS/FS/LS modes of operation
- Supports OTG signalling

Figure 45-1 shows the top-level interface for the USB system. The USB-PHY connects the USB cable through a standard 5-pin interface [1] and communicates to the Link Controller through the standard UTMI interface [2].

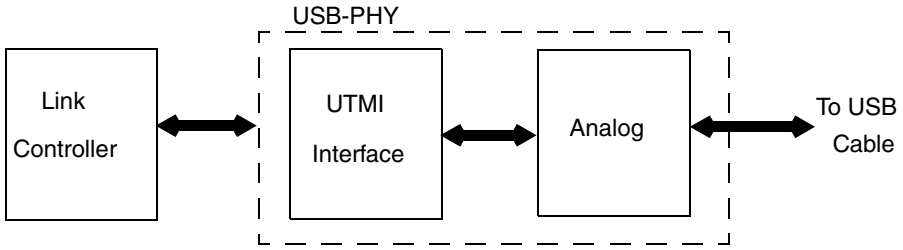


Figure 45-1. USB System

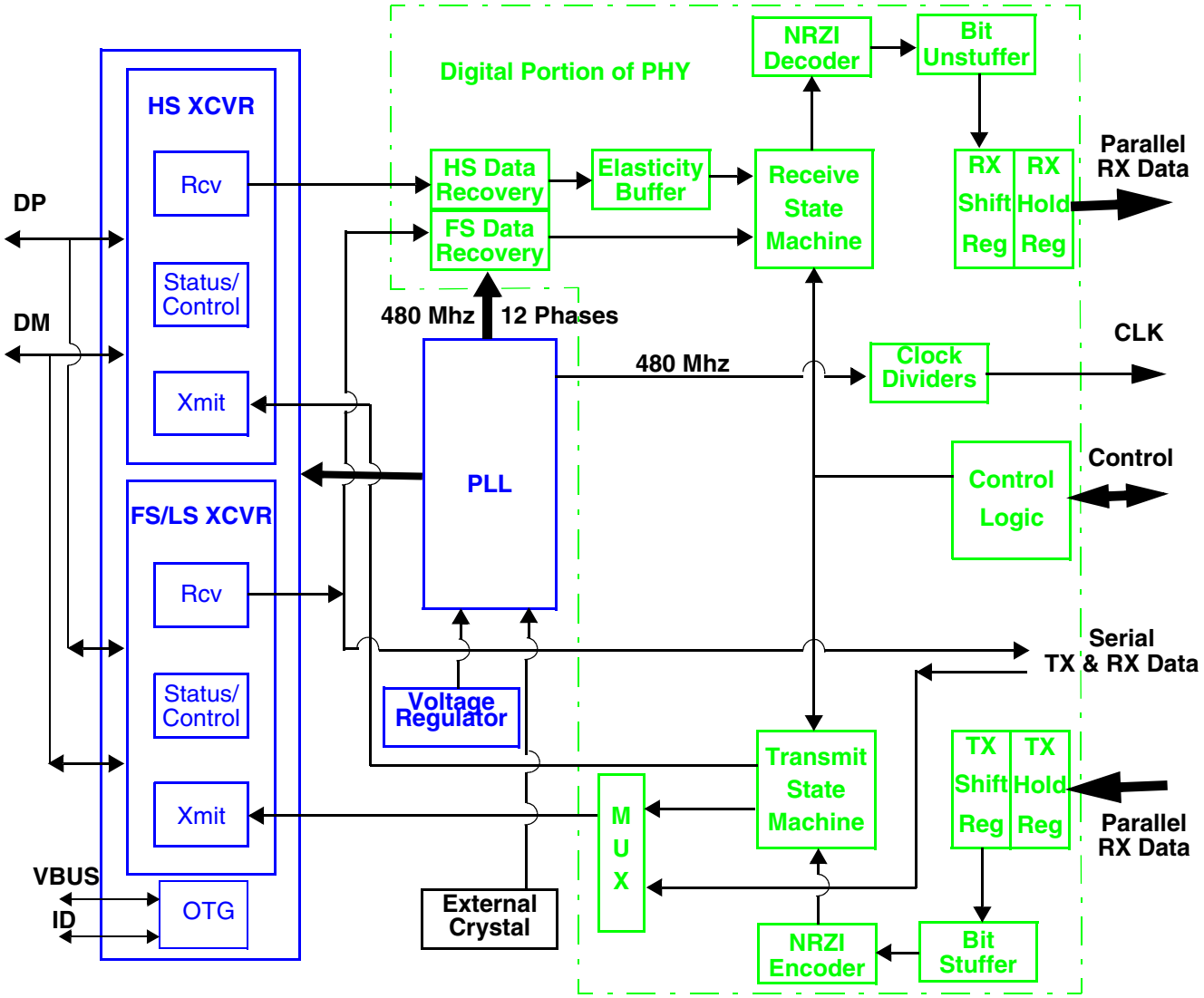


Figure 45-2. Implementation

Figure 45-2 shows the internal block diagram implementation for the UTMI-USB-PHY. The blocks in Blue comprise of all analog circuitry. The green portion shows the digital blocks. These are explained in more detail in Section 45.5, “Functional Description

45.4 External Signal Descriptions

Refer Table 45-2 for detail signal list for UTMI-USB-PHY.

Table 45-2. Detail signal List for UTMI-USB-PHY

Signal	I/O	Active State	Description
Analog Interface Signals			
RREF	I	1	Used by Analog block for generating accurate bias current. A 10 Kbyte ± 1% precision resistor is connected on board to ground.
DP	I/O	NA	USB 2.0 D+ line from cable
DM	I/O	NA	USB 2.0 D- line from cable
VBUS	I/O	1	USB 2.0 VBUS line
UID	I	1	Pin used to distinguish between A & B device. Connected from cable

45.5 Functional Description

This section provides a detailed description of USB PHY. USB transceiver is used for driving transmission line loads and consists of transmit and receive sections which are connected to USB port through USB data lines. The control signals for transceiver come from the USB link controller. The PHY communicates with the controller through the standard UTMI protocol.

45.5.1 USB PHY sub-blocks

A USB2.0-compatible transceiver requires the following blocks.

45.5.1.1 Analog Modules

45.5.1.1.1 Transmitter

- High Speed Current Driver
  The high-speed current driver is used for high-speed data transmission.
- FS/LS Transmitter
  Used to transmit data at Full-speed (12Mbps) and Low-Speed (1.5Mbps) data rates.

45.5.1.1.2 Receiver

- HS Receiver
  HS Receiver consists of the following three blocks:

- Differential Receiver
  It receives the HS data on DP/DM lines at 480Mbps. It is required to receive data compatible with the eye diagram as shown in Figure 7-16. Template 4, Pg136 USB2.0 specs.
- Transmission envelope Detector
  This envelope detector is used to indicate that data is invalid when the amplitude of the differential signal at a receiver's inputs falls below the squelch threshold (VHSSQ). It must indicate Squelch when the signal drops below 100 mV differential amplitude, and it must indicate that the line is not in the Squelch state when the signal exceeds 150 mV differential amplitude.
- Disconnection Envelope Detector
  This envelope detector is required in downstream facing ports to detect the high-speed Disconnect state on the line (VHSDSC). Disconnection must be indicated when the amplitude of the differential signal at the downstream facing drivers connector is more than 625 mV, and it must not be indicated when the signal amplitude is below 525 mV. The output of this detector is sampled at a specific time during the transmission of the high-speed SOF EOP,

• LS/FS Receiver
  Used to receive data at 12Mbps and 1.5Mbs data rates

45.5.1.1.3 PLL

PLL is a clock generating module for USB application. It has a fixed frequency Analog-PLL to produce 480 MHz ± 500 ppm for different input frequencies. The clock out represents the phases of the PLL used by the transceiver to transmit and receive data. It requires an input reference of 24 MHz which comes from the 24 MHz oscillator. It is required that the external clock/crystal source jitter be less than 2 ps-rms (in the frequency range100 KHz to10 MHz).

45.5.1.1.4 Bias

The Bias block generates bias currents to be used in different parts of the XCVR. A off-chip precision resistor of 10 Kbyte ± 1% is connected at RREF for this purpose.

45.5.1.1.5 Voltage Regulator

The PHY contains a Voltage regulator to generate 1.575 V +/– 4% supply voltage for the PLL.

45.5.1.1.6 Termination Calibration

The FS/HS 45 Ω terminations are calibrated by generating precision current from a off-chip precision (10 Kbyte ± 1%) resistor on the pin RREF. All DP/DM terminations are internal to the PHY.

45.5.1.1.7 Vbus and ID detection

This module detects USB device/host attachments and removals. It contains comparators for OTG signalling. The PHY cannot source current in VBUS and an external charge-pump is required. Also, the ID detector comparator is used to know the whether the PHY is configured as an A-device or B-device.

The PHY is configurable to accept direct VBUS from cable or a divided down version of VBUS. When ExtVbus_div_option is 1, an off-chip divided down version is provided to the PHY. In this scenario, ChargeVbus functionality is also expected to be performed off-chip. An easy solution is to OR both ChrgVbus and DrvVbus signals and provide to the charge-pump enable. When ExtVbus_div_option is 0, the PHY will do the ChrgVbus function using a pull-up resistor to UXCVR.

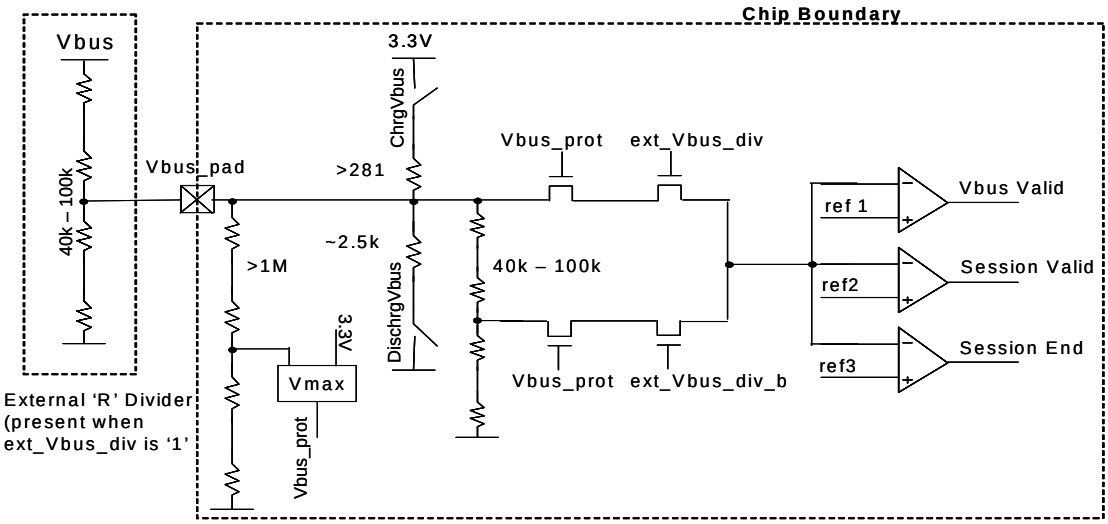


Figure 45-3. OTG implementation

Table 45-3 shows the VBUS and ID threshold specs.

Table 45-3. VBUS and ID Thresholds

Function	VBUS Comp Thresholds (V)	ID Comparator Threshold (V)
A-device Valid	4.4–5.25	—
B-device Valid	0.8–2.0	—
Vbus Valid	0.8–4.0	—
Session end	0.2–0.8	—
A/B device detection	—	1.0–1.5

45.5.1.1.8 Short Circuit Protection (SCP)

SCP protects the transceiver module during fault cases. Fault cases catered to are defined in Section 7.1.1 Pg.124 USB2.0 Specs. The SCP circuit does not trigger in AC stress conditions. For DC shorts to Vbus, SCP trigger threshold is from 4.0V to 4.6V.

Chapter 46
Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

46.1 Overview

This section briefly introduces the module. The full description of the module is in Section 46.4, “Functional Description.”

This section includes a top level diagram that shows the functional organization of the module, including all off-chip signals. Figure 46-1 is block diagram.

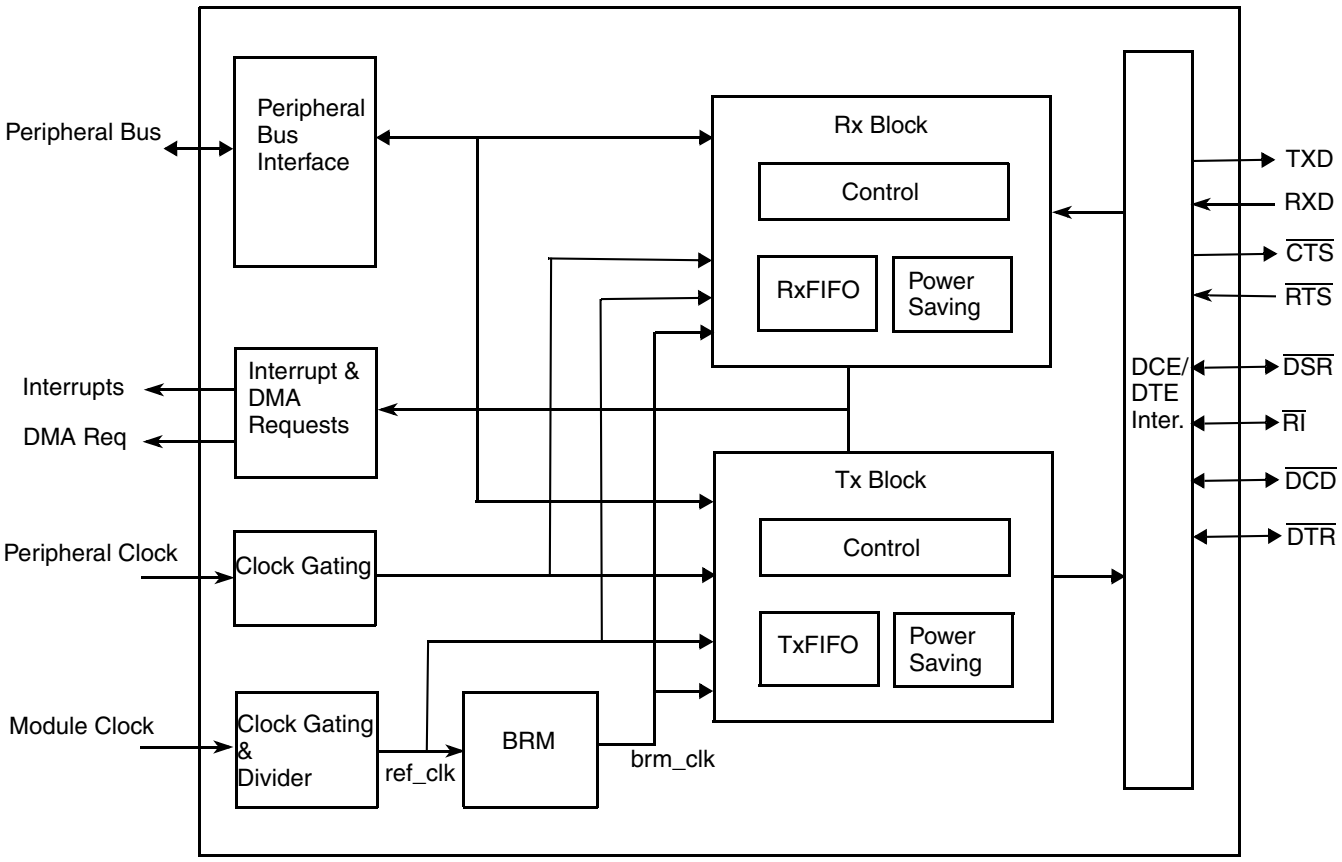


Figure 46-1. UART Block Diagram

46.1.1 Features

The UART includes the following features:

- High-speed TIA/EIA-232-F compatible
- 7 or 8 data bits
- 1 or 2 stop bits
- Programmable parity (even, odd, and no parity)
- Hardware flow control support for request to send ($\overline{RTS}$) and clear to send ($\overline{CTS}$) signals
- Edge-selectable $\overline{RTS}$ and edge-detect interrupts
- Status flags for various flow control and FIFO states
- Serial IR interface low-speed, IrDA-compatible (up to 115.2 Kbit/s).
- Voting logic for improved noise immunity (16x oversampling)
- Transmitter FIFO empty interrupt suppression

- UART internal clocks enable/disable
- Auto baud rate detection (up to 115.2 Kbit/s)
- Receiver and transmitter enable/disable for power saving
- DCE/DTE capability
- $\overline{\text{RTS}}$, IrDA asynchronous wake (AIRINT), receive asynchronous wake (AWAKE), $\overline{\text{RI}}$ (DTE only), $\overline{\text{DCD}}$ (DTE only), $\overline{\text{DTR}}$ (DCE only) and $\overline{\text{DSR}}$ (DTE only) interrupts wake the processor from STOP mode
- Maskable interrupts
- Two DMA Requests (TxFIFO DMA Request and RxFIFO DMA Request)
- Escape character sequence detection
- Software reset ($\overline{\text{SRST}}$)
- Two independent, 32-entry FIFOs for transmit and receive
- The peripheral clock can be totally asynchronous with the module clock. The module clock determines baud rate. This allows frequency scaling on peripheral clock (such as during DVFS mode) while remaining the module clock frequency and baud rate.

46.1.2 Modes of Operation

- Serial RS-232 NRZ format.
- IrDA

46.2 External Signals

Conventions: Table 46-1 lists conventions for representing signals.

Table 46-1. Module Signal Conventions

Category	Convention	Example(s)
Off-chip signal	Uppercase (all capital letters)	TXD
Internal signal[1]	Lowercase italics	*core_int*
Active low signal	_B (_b) suffix or overbar	RESET_EN_B or $\overline{\text{RESET\_EN}}$
Range of bussed or commonly named signals	Beginning and end points of the range are: • Separated by a colon. • Surrounded by square brackets.	ADDR[31:0] CSE_B[7:0] or $\overline{\text{CSE}}$[7:0]
Individual signal in a range of bussed or commonly named signals	Individual number in the range appears without a colon or square brackets	ADDR31 CSE0_B or $\overline{\text{CSE0}}$

1 Internal signals are for reference only in descriptions of internal module or SoC functionality.

Table 46-2 describes all UART signals that connect off-chip.

46.2.1 Detailed Signal Descriptions

Table 46-2. Off-Chip Module Signals

Signal name	I/O	Active state	Description	Reset state
Serial / IrDA Signals				
RXD	I		Serial / infrared data receive	—
TXD	O		Serial/infrared data transmit	High
Modem Control Signals				
$\overline{\text{CTS}}$	O	Low	Clear to send	High
$\overline{\text{RTS}}$	I	Low	Request to send	—
$\overline{\text{DSR}}$	I/O	Low	Data set ready	High
$\overline{\text{DCD}}$	I/O	Low	Data carrier detected	High
$\overline{\text{DTR}}$	I/O	Low	Data terminal ready	—
$\overline{\text{RI}}$	I/O	Low	Ring indicator	High
Interrupts				
interrupt_uart	O	Low	UART interrupt	High
DMA Requests				
dma_req_rx	O	Low	Receiver DMA request	High
dma_req_tx	O	Low	Transmitter DMA request	High
Clocks				
peripheral_clock	I		Peripheral clock	—
module_clock	I		Clock source for the module's logic	—
Special Signals				
stop_req	I	High	Module stop mode	—
doze_req	I	High	Module doze mode	—
debug_req	I	High	Module debug	—

46.2.1.1 Serial / IrDA Signals

46.2.1.1.1 RXD — Data Receive

Input asynchronous data receive in Serial and IrDA modes.

46.2.1.1.2 TXD — Data Transmit

Output asynchronous data transmit in Serial and IrDA modes.

46.2.1.2 Modem Control Signals

46.2.1.2.1 $\overline{\text{CTS}}$ — Clear To Send

Output in DCE and DTE mode. This signal informs the remote modem that UART is ready to receive data.

46.2.1.2.2 $\overline{\text{RTS}}$ — Request To Send

Input in DCE and DTE mode. This signal informs UART that remote modem is ready to receive data.

46.2.1.2.3 $\overline{\text{DSR}}$ — Data Set Ready

Input in DTE mode. Indicates to UART that remote modem is operational.

Output in DCE mode. Indicates to remote modem that UART is operational.

46.2.1.2.4 $\overline{\text{DCD}}$ — Data Carrier Detected

Input in DTE mode. Indicates to UART that a good carrier is being received from the remote modem.

Output in DCE mode. Indicates to remote device that a good carrier is being received from the UART.

46.2.1.2.5 $\overline{\text{DTR}}$ — Data Terminal Ready

Input in DCE mode. Indicates to UART (in DCE mode) that remote device (in DTE mode) is operational.

Output in DTE mode. Indicates to remote modem (in DCE mode) that UART (in DTE mode) is operational.

46.2.1.2.6 $\overline{\text{RI}}$ — Ring Indicator

Input in DTE mode. Indicates to UART that remote modem is detecting a ringing tone.

Output in DCE mode. Indicates to remote device that UART is detecting a ringing tone.

46.2.1.3 Interrupt Signals

46.2.1.3.1 $\overline{\textit{interrupt\_uart}}$ — UART Interrupt

Output interrupt request.

46.2.1.4 DMA Request Signals

46.2.1.4.1 $\overline{\textit{dma\_req\_rx}}$ — Receiver DMA Request

Output DMA Request signal for receiver interface.

46.2.1.4.2 $\overline{\textit{dma\_req\_tx}}$ — Transmitter DMA Request

Output DMA Request signal for transmitter interface. Set at 0 when TXDMAEN (UCR1[3]) is at 1 and TRDY (USR1[13]) is also at 1.

46.2.1.5 Clock Signals

46.2.1.5.1 *peripheral_clock* — Peripheral Clock

See Section 46.4.2, “Clocks,” for more information about *peripheral_clock*.

46.2.1.5.2 *module_clock* — Module Clock

See Section 46.4.2, “Clocks,” for more information about *module_clock*.

46.2.1.6 Special Signals

46.2.1.6.1 *stop_req* — Stop Mode

Input stop mode. Indicates UART that MCU is going to enter in Stop Mode and clocks are going to stop running. See Section 46.4.8, “Low Power Modes,” for more information about Stop Mode.

46.2.1.6.2 *doze_req* — Doze Mode

Input doze mode. MCU requests UART to switch in doze mode (power saving mode). See Section 46.4.8, “Low Power Modes,” for more information about Doze Mode.

46.2.1.6.3 *debug_req* — Debug Mode

Input debug mode. Indicates UART it has to enter in debug mode. See Section 46.4.9, “UART Operation in System Debug State,” for more information about Debug Mode.

46.3 Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers.

46.3.1 Memory Map

Table 46-3 is the UART memory map.

Table 46-3. UART Memory Map

Offset Address (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (URXD)	UART Receiver Register	R	00– –	46.3.3.1/46-11
0x0004–0x003c	Reserved	—	—	—
0x0040 (UTXD)	UART Transmitter Register	W	00– –	46.3.3.2/46-13
0x0044–0x007c	Reserved	—	—	—
0x0080 (UCR1)	UART Control Register 1	R/W	0000	46.3.3.3/46-14
0x0084 (UCR2)	UART Control Register 2	R/W	0001	46.3.3.4/46-16
0x0088 (UCR3)	UART Control Register 3	R/W	0700	46.3.3.5/46-18

Table 46-3. UART Memory Map (continued)

Offset Address (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x008c (UCR4)	UART Control Register 4	R/W	8000	46.3.3.6/46-20
0x0090 (UFCR)	UART FIFO Control Register	R/W	0801	46.3.3.7/46-22
0x0094 (USR1)	UART Status Register 1	R/W	2040	46.3.3.8/46-23
0x0098 (USR2)	UART Status Register 2	R/W	4028	46.3.3.9/46-25
0x009c (UESC)	UART Escape Character Register	R/W	002b	46.3.3.10/46-28
0x00a0 (UTIM)	UART Escape Timer Register	R/W	0000	46.3.3.11/46-28
0x00a4 (UBIR)	UART BRM Incremental Register	R/W	0000	46.3.3.12/46-29
0x00a8 (UBMR)	UART BRM Modulator Register	R/W	0000	46.3.3.13/46-30
0x00ac (UBRC)	UART Baud Rate Count Register	R	0004	46.3.3.14/46-31
0x00b0 (ONEMS)	UART One Millisecond Register	R/W	000000	46.3.3.15/46-32
0x00b4 (UTS)	UART Test Register	R/W	0060	46.3.3.16/46-33

46.3.2 Register Summary

Table 46-5 is the register summary table.

The UART supports 8-bit, 16-bit and 32-bit accesses to 32-bit memory-mapped addresses. Any access to unmapped memory location will yield a transfer error.

All the memory mapped registers are 32 bits wide, however as the 16 MSB are not used for all of the registers except the ONEMS register:

- For 32-bits write access, the 16 MSB will not be taken into account for all of the registers except the ONEMS.
- For 32-bits read access the 16 MSB will be read as 0 for all of the registers except the ONEMS.

The ONEMS register is expanded from 16 bits to 24 bits in order to support the high frequency of the BRM internal clock *ref_clk* (*module_clock* after divider). The ONEMS register can be accessed in the way of 8-bits, 16-bits or 32-bits.

- For 32-bits write access, the 8 MSB of the ONEMS will not be taken into account.
- For 32-bits read access, the 8 MSB of the ONEMS will be read as 0.

All registers except the ONEMS described in this section are for 16 LSB. The ONEMS register is for 24 LSB.

Figure 46-2 shows the key to the register fields and Table 46-4 shows the register figure conventions.

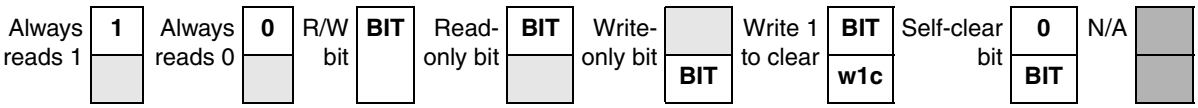


Figure 46-2. Key to Register Fields

Table 46-4. Register Figure Convention

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
r	Read only. Writing this bit has no effect.
w	Write only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

Table 46-5 shows the UART register summary.

Table 46-5. UART Register Summary

Offset Address (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (URXD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CHARRDY	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA							
	W																
0x0040 (UTXD)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0								
	W									TX_DATA							

Table 46-5. UART Register Summary (continued)

Offset Address (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0080 (UCR1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	ADEN	ADBR	TRDYEN	IDEN	ICD		RRDYEN	RXDMAEN	IREN	TXMPTYEN	RTSDEN	SNDBRK	TXDMAEN	ATDMAEN	DOZE	UARTEN
0x0084 (UCR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
0x0088 (UCR3)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
0x008c (UCR4)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	BKEN	BKEN	OREN	DREN
0x0090 (UFCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	TXTL						RFDIV			DCEDTE	RXTL					
0x0094 (USR1)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PARITYERR	RTSS	TRDY	RTSD	ESCF	FRAMERR	RRDY	AGTIM	DTRD	RXDS	AIRINT	AWAKE	0	0	0	0
	W	w1c			w1c	w1c	w1c		w1c	w1c		w1c	w1c				

Table 46-5. UART Register Summary (continued)

Offset Address (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0098 (USR2)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	ADET	TXFE	DTRF	IDLE	ACST	RIDELT	RIIN	IRINT	WAKE	DCDDELT	DCDIN	RTSF	TXDC	BRCD	ORE	RDR
	W	w1c		w1c	w1c	w1c	w1c		w1c	w1c	w1c		w1c		w1c	w1c	
0x009c (UESC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	ESC_CHAR							
	W																
0x00a0 (UTIM)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	TIM											
	W																
0x00a4 (UBIR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	INC															
	W																
0x00a8 (UBMR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	MOD															
	W																
0x00ac (UBRC)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	BCNT															
	W																
0x00b0 (ONEMS)	R	0	0	0	0	0	0	0	0								
	W																
	R	ONEMS															
	W																

Table 46-5. UART Register Summary (continued)

Offset Address (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00b4 (UTS)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	FRCPERR	LOOP	DBGEN	LOOPIR	RXDBG	0	0	TXEMPTY	RXEMPTY	TXFULL	RXFULL	0	0	SOFTRST
	W																

46.3.3 Register Descriptions

This section provides detailed descriptions of the UART registers.

46.3.3.1 UART Receiver Register (URXD)

See Figure 46-3 for illustration of valid bits in the UART Receiver Register and Table 46-6 for description of the bit fields.

0x0000 (URXD) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHARRDY	ERR	OVRRUN	FRMERR	BRK	PRERR	0	0	RX_DATA							
W																
RESET	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 46-3. UART Receiver Register

Table 46-6. UART Receiver Register Field Descriptions

Field	Description
31–16	Reserved
15 CHARRDY	Character Ready. This read-only bit indicates an invalid read when the FIFO becomes empty and software tries to read the same old data. This bit should not be used for polling for data written to the RX FIFO. 0 Character in RX_DATA field and associated flags are invalid. 1 Character in RX_DATA field and associated flags valid and ready for reading.

Table 46-6. UART Receiver Register Field Descriptions (continued)

Field	Description
14 ERR	Error Detect. Indicates whether the character present in the RX_DATA field has an error (OVRRUN, FRMERR, BRK or PRERR) status. The ERR bit is updated and valid for each received character. 0 No error status was detected 1 An error status was detected
13 OVRRUN	Receiver Overrun. This read-only bit, when HIGH, indicates that the corresponding character was stored in the last position (32nd) of the Rx FIFO. Even if a 33rd character has not been detected, this bit will be set to '1' for the 32nd character. 0 No RxFIFO overrun was detected 1 A RxFIFO overrun was detected
12 FRMERR	Frame Error. Indicates whether the current character had a framing error (a missing stop bit) and is possibly corrupted. FRMERR is updated for each character read from the RxFIFO. 0 The current character has no framing error 1 The current character has a framing error
11 BRK	BREAK Detect. Indicates whether the current character was detected as a BREAK character. The data bits and the stop bit are all 0. The FRMERR bit is set when BRK is set. When odd parity is selected, PRERR is also set when BRK is set. BRK is valid for each character read from the RxFIFO. 0 The current character is not a BREAK character 1 The current character is a BREAK character
10 PRERR	Parity Error. Indicates whether the current character was detected with a parity error and is possibly corrupted. PRERR is updated for each character read from the RxFIFO. When parity is disabled, PRERR always reads as 0. 0 = No parity error was detected for data in the RX_DATA field 1 = A parity error was detected for data in the RX_DATA field
9–8	Reserved
7–0 RX_DATA	Received Data. Holds the received character. In 7-bit mode, the most significant bit (MSB) is forced to 0. In 8-bit mode, all bits are active.

NOTE

The UART will yield a transfer error on the peripheral bus when core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0).

46.3.3.2 UART Transmitter Register (UTXD)

See Figure 46-4 for illustration of valid bits in the UART Transmitter Register and Table 46-7 for description of the bit fields.

0x0040 (UTXD) Access: User write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0								
W									TX_DATA							
RESET	0	0	0	0	0	0	0	0	—	—	—	—	—	—	—	—

Figure 46-4. UART Transmitter Register

Table 46-7. UART Transmitter Register Field Descriptions

Field	Description
31–8	Reserved
7–0 TX_DATA	Transmit Data. Holds the parallel transmit data inputs. In 7-bit mode, D7 is ignored. In 8-bit mode, all bits are used. Data is transmitted least significant bit (LSB) first. A new character is transmitted when the TX_DATA field is written. The TX_DATA field must be written only when the TRDY bit is high to ensure that corrupted data is not sent.

NOTE

The UART will yield a transfer error on the peripheral bus when core is writing into URXD register with transmit interface disabled (TXEN=0 or UARTEN=0).

Memory space between URXD and UTXD registers is reserved. Any read or write access to this space will be considered as an invalid access and yield a transfer error.

46.3.3.3 UART Control Register 1 (UCR1)

See Figure 46-5 for illustration of valid bits in the UART Control Register 1 and Table 46-8 for description of the bit fields.

0x0080 (UCR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADEN	ADBR	TRDY EN	IDEN	ICD		RRDY EN	RXD MAE N	IREN	TXMP TYEN	RTSD EN	SNDB RK	TXD MAE N	ATDM AEN	DOZE	UAR TEN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-5. UART Control Register 1

Table 46-8. UART Control Register 1 Field Descriptions

Field	Description
31–16	Reserved
15 ADEN	Automatic Baud Rate Detection Interrupt Enable. Enables/Disables the automatic baud rate detect complete (ADET) bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the automatic baud rate detection interrupt 1 Enable the automatic baud rate detection interrupt
14 ADBR	Automatic Detection of Baud Rate. Enables/Disables automatic baud rate detection. When the ADBR bit is set and the ADET bit is cleared, the receiver detects the incoming baud rate automatically. The ADET flag is set when the receiver verifies that the incoming baud rate is detected properly by detecting an ASCII character “A” or “a” (0x61 or 0x41). 0 Disable automatic detection of baud rate 1 Enable automatic detection of baud rate
13 TRDYEN	Transmitter Ready Interrupt Enable. Enables/Disables the transmitter Ready Interrupt (TRDY) when the transmitter has one or more slots available in the TxFIFO. The fill level in the TXFIFO at which an interrupt is generated is controlled by TxTL bits. When TRDYEN is negated, the transmitter ready interrupt is disabled. **Note:** An interrupt will be issued as long as TRDYEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TRDY interrupt. 0 Disable the transmitter ready interrupt 1 Enable the transmitter ready interrupt
12 IDEN	Idle Condition Detected Interrupt Enable. Enables/Disables the IDLE bit to generate an interrupt (*interrupt_uart* = 0). 0 Disable the IDLE interrupt 1 Enable the IDLE interrupt

Table 46-8. UART Control Register 1 Field Descriptions (continued)

Field	Description
11–10 ICD	Idle Condition Detect. Controls the number of frames RXD is allowed to be idle before an idle condition is reported. 00 Idle for more than 4 frames 01 Idle for more than 8 frames 10 Idle for more than 16 frames 11 Idle for more than 32 frames
9 RRDYEN	Receiver Ready Interrupt Enable. Enables/Disables the RRDY interrupt when the RxFIFO contains data. The fill level in the RxFIFO at which an interrupt is generated is controlled by the RXTL bits. When RRDYEN is negated, the receiver ready interrupt is disabled. 0 Disables the RRDY interrupt 1 Enables the RRDY interrupt
8 RXDMAEN	Receive Ready DMA Enable. Enables/Disables the receive DMA request $\overline{dma\_req\_rx}$ when the receiver has data in the RxFIFO. The fill level in the RxFIFO at which a DMA request is generated is controlled by the RXTL bits. When negated, the receive DMA request is disabled. 0 Disable DMA request 1 Enable DMA request
7 IREN	Infrared Interface Enable. Enables/Disables the IR interface. See the IR interface description in Section 46.4.7, "Infrared Interface," for more information. 0 Disable the IR interface 1 Enable the IR interface
6 TXMPTYEN	Transmitter Empty Interrupt Enable. Enables/Disables the transmitter FIFO empty (TXFE) interrupt. $\overline{interrupt\_uart}$. When negated, the TXFE interrupt is disabled. **Note:** An interrupt will be issued as long as TXMPTYEN and TXFE are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXFE interrupt. 0 Disable the transmitter FIFO empty interrupt 1 Enable the transmitter FIFO empty interrupt
5 RTSDEN	RTS Delta Interrupt Enable. Enables/Disables the RTSD interrupt. The current status of the $\overline{RTS}$ pin is read in the RTSS bit. 0 Disable RTSD interrupt 1 Enable RTSD interrupt
4 SNDBRK	Send BREAK. Forces the transmitter to send a BREAK character. The transmitter finishes sending the character in progress (if any) and sends BREAK characters until SNDBRK is reset. Because the transmitter samples SNDBRK after every bit is transmitted, it is important that SNDBRK is asserted high for a sufficient period of time to generate a valid BREAK. After the BREAK transmission completes, the UART transmits 2 mark bits. The user can continue to fill the TxFIFO and any characters remaining are transmitted when the BREAK is terminated. 0 Do not send a BREAK character 1 Send a BREAK character (continuous 0s)
3 TXDMAEN	Transmitter Ready DMA Enable. Enables/Disables the transmit DMA request $\overline{dma\_req\_tx}$ when the transmitter has one or more slots available in the TxFIFO. The fill level in the TxFIFO that generates the $\overline{dma\_req\_tx}$ is controlled by the TXTL bits. **Note:** A DMA request will be issued as long as TXDMAEN and TRDY are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the transmit DMA request. 0 Disable transmit DMA request 1 Enable transmit DMA request
2 ATDMAEN	Aging DMA Timer Enable. Enables/Disables the receive DMA request $\overline{dma\_req\_rx}$ for the aging timer interrupt (triggered with AGTIM flag in USR1[8]). 0 Disable AGTIM DMA request 1 Enable AGTIM DMA request

Table 46-8. UART Control Register 1 Field Descriptions (continued)

Field	Description
1 DOZE	DOZE. Determines the UART enable condition in the DOZE state. When *doze_req* input pin is at '1', (the CPU executes a doze instruction and the system is placed in the Doze State), the DOZE bit affects operation of the UART. While in the Doze State, if this bit is asserted, the UART is disabled. See the description in Section 46.4.8, "Low Power Modes." 0 The UART is enabled when in DOZE state 1 The UART is disabled when in DOZE state
0 UARTEN	UART Enable. Enables/Disables the UART. If UARTEN is negated in the middle of a transmission, the transmitter stops and pulls the TXD line to a logic 1. UARTEN must be set to 1 before any access to UTXD and URXD registers, otherwise a transfer error is returned. This bit can be set to 1 along with other bits in this register. There is no restriction to the sequence of programing this bit and other control registers. 0 Disable the UART 1 Enable the UART

46.3.3.4 UART Control Register 2 (UCR2)

See Figure 46-6 for illustration of valid bits in the UART Control Register 2 and Table 46-9 for description of the bit fields.

0x0084 (UCR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ESCI	IRTS	CTSC	CTS	ESCEN	RTEC		PREN	PROE	STPB	WS	RTSEN	ATEN	TXEN	RXEN	SRST
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 46-6. UART Control Register 2

Table 46-9. UART Control Register 2 Field Descriptions

Name	Description
31–16	Reserved
15 ESCI	Escape Sequence Interrupt Enable. Enables/Disables the ESCF bit to generate an interrupt. 0 Disable the escape sequence interrupt 1 Enable the escape sequence interrupt
14 IRTS	Ignore RTS Pin. Forces the RTS input signal presented to the transmitter to always be asserted (set to low), effectively ignoring the external pin. When in this mode, the RTS pin serves as a general purpose input. 0 Transmit only when the RTS pin is asserted 1 Ignore the RTS pin

Table 46-9. UART Control Register 2 Field Descriptions (continued)

Name	Description
13 CTSC	CTS Pin Control. Controls the operation of the $\overline{CTS}$ output pin. When CTSC is asserted, the $\overline{CTS}$ output pin is controlled by the receiver. When the RxFIFO is filled to the level of the programmed trigger level and the start bit of the overflowing character (TRIGGER LEVEL + 1) is validated, the $\overline{CTS}$ output pin is negated to indicate to the far-end transmitter to stop transmitting. When the trigger level is programmed for less than 32, the receiver continues to receive data until the RxFIFO is full. When the CTSC bit is negated, the $\overline{CTS}$ output pin is controlled by the CTS bit. On reset, because CTSC is cleared to 0, the $\overline{CTS}$ pin is controlled by the CTS bit, which again is cleared to 0 on reset. This means that on reset the $\overline{CTS}$ signal is negated. 0 The $\overline{CTS}$ pin is controlled by the CTS bit 1 The $\overline{CTS}$ pin is controlled by the receiver
12 CTS	Clear to Send. Controls the $\overline{CTS}$ pin when the CTSC bit is negated. CTS has no function when CTSC is asserted. 0 The $\overline{CTS}$ pin is high (inactive) 1 The $\overline{CTS}$ pin is low (active)
11 ESCEN	Escape Enable. Enables/Disables the escape sequence detection logic. 0 Disable escape sequence detection 1 Enable escape sequence detection
10-9 RTEC	Request to Send Edge Control. Selects the edge that triggers the RTS interrupt. This has no effect on the RTS delta interrupt. RTEC has an effect only when RTSEN = 1 (see Table 46-23). 00 Trigger interrupt on a rising edge 01 Trigger interrupt on a falling edge 1X Trigger interrupt on any edge
8 PREN	Parity Enable. Enables/Disables the parity generator in the transmitter and parity checker in the receiver. When PREN is asserted, the parity generator and checker are enabled, and disabled when PREN is negated. 0 Disable parity generator and checker 1 Enable parity generator and checker
7 PROE	Parity Odd/Even. Controls the sense of the parity generator and checker. When PROE is high, odd parity is generated and expected. When PROE is low, even parity is generated and expected. PROE has no function if PREN is low. 0 Even parity 1 Odd parity
6 STPB	Stop. Controls the number of stop bits after a character. When STPB is low, 1 stop bit is sent. When STPB is high, 2 stop bits are sent. STPB also affects the receiver. 0 The transmitter sends 1 stop bit. The receiver expects 1 or more stop bits. 1 The transmitter sends 2 stop bits. The receiver expects 2 or more stop bits.
5 WS	Word Size. Controls the character length. When WS is high, the transmitter and receiver are in 8-bit mode. When WS is low, they are in 7-bit mode. The transmitter ignores bit 7 and the receiver sets bit 7 to 0. WS can be changed in-between transmission (reception) of characters, however not when a transmission (reception) is in progress, in which case the length of the current character being transmitted (received) is unpredictable. 0 7-bit transmit and receive character length (not including START, STOP or PARITY bits) 1 8-bit transmit and receive character length (not including START, STOP or PARITY bits)
4 RTSEN	Request to Send Interrupt Enable. Controls the RTS edge sensitive interrupt. When RTSEN is asserted and the programmed edge is detected on the $\overline{RTS}$ pin (the RTSF bit is asserted), an interrupt will be generated on the *interrupt_uart* pin. (See Table 46-23.) 0 Disable request to send interrupt 1 Enable request to send interrupt

Table 46-9. UART Control Register 2 Field Descriptions (continued)

Name	Description
3 ATEN	Aging Timer Enable. This bit is used to enable the aging timer interrupt (triggered with AGTIM) 0 AGTIM interrupt disabled 1 AGTIM interrupt enabled
2 TXEN	Transmitter Enable. Enables/Disables the transmitter. When TXEN is negated the transmitter is disabled and idle. When the UARTEN and TXEN bits are set the transmitter is enabled. If TXEN is negated in the middle of a transmission, the UART disables the transmitter immediately, and starts marking 1s. The transmitter FIFO cannot be written when this bit is cleared. 0 Disable the transmitter 1 Enable the transmitter
1 RXEN	Receiver Enable. Enables/Disables the receiver. When the receiver is enabled, if the RXD input is already low, the receiver does not recognize BREAK characters, because it requires a valid 1-to-0 transition before it can accept any character. 0 Disable the receiver 1 Enable the receiver
0 $\overline{\text{SRST}}$	Software Reset. Once the software writes 0 to $\overline{\text{SRST}}$, the software reset remains active for 4 *module_clock* cycles before the hardware negates $\overline{\text{SRST}}$. The software can only write 0 to $\overline{\text{SRST}}$. Writing 1 to $\overline{\text{SRST}}$ is ignored. 0 Reset the transmit and receive state machines, all FIFOs and register USR1, USR2, UBIR, UBMR, UBRC, URXD, UTXD and UTS[6-3]. 1 No reset

46.3.3.5 UART Control Register 3 (UCR3)

See Figure 46-7 for illustration of valid bits in the UART Control Register 3 and Table 46-10 for description of the bit fields.

0x0088 (UCR3) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DPEC		DTREN	PARERREN	FRAERREN	DSR	DCD	RI	ADNIMP	RXDSEN	AIRINTEN	AWAKEN	DTRDEN	RXDMUXSEL	INVT	ACIEN
RESET	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

Figure 46-7. UART Control Register 3

Table 46-10. UART Control Register 3 Field Descriptions

Name	Description
31–16	Reserved
15–14 DPEC	DTR/DSR Interrupt Edge Control. These bits control the edge of $\overline{\text{DTR}}$ (DCE) or $\overline{\text{DSR}}$ (DTE) on which an interrupt will be generated. An interrupt will only be generated if the DTREN bit is set. 00 interrupt generated on rising edge 01 interrupt generated on falling edge 1X interrupt generated on either edge
13 DTREN	Data Terminal Ready Interrupt Enable. When this bit is set, it will enable the status bit DTRF (USR2 [13]) (DTR/DSR edge sensitive interrupt) to cause an interrupt. 0 Data Terminal Ready Interrupt Disabled 1 Data Terminal Ready Interrupt Enabled
12 PARERREN	Parity Error Interrupt Enable. Enables/Disables the interrupt. When asserted, PARERREN causes the PARITYERR bit to generate an interrupt. 0 Disable the parity error interrupt 1 Enable the parity error interrupt
11 FRAERREN	Frame Error Interrupt Enable. Enables/Disables the interrupt. When asserted, FRAERREN causes the FRAMERR bit to generate an interrupt. 0 Disable the frame error interrupt 1 Enable the frame error interrupt
10 DSR	Data Set Ready. This bit is used by software to control the DSR/DTR output pin for the modem interface. In DCE mode it applies to $\overline{\text{DSR}}$ and in DTE mode it applies to $\overline{\text{DTR}}$. 0 DSR/ DTR pin is logic zero 1 DSR/ DTR pin is logic one
9 DCD	Data Carrier Detect. In DCE mode this bit is used by software to control the $\overline{\text{DCD}}$ output pin for the modem interface. In DTE mode, when this bit is set, it will enable the status bit DCDDELT (USR2 (6)) to cause an interrupt. 0 $\overline{\text{DCD}}$ pin is logic zero (DCE mode) 1 $\overline{\text{DCD}}$ pin is logic one (DCE mode) 0 DCDDELT interrupt disabled (DTE mode) 1 DCDDELT interrupt enabled (DTE mode)
8 RI	Ring Indicator. In DCE mode this bit is used by software to control the $\overline{\text{RI}}$ output pin for the modem interface. In DTE mode, when this bit is set, it will enable the status bit RIDELT (USR2 (10)) to cause an interrupt. 0 $\overline{\text{RI}}$ pin is logic zero (DCE mode) 1 $\overline{\text{RI}}$ pin is logic one (DCE mode) 0 RIDELT interrupt disabled (DTE mode) 1 RIDELT interrupt enabled (DTE mode)
7 ADNIMP	Autobaud Detection Not Improved—. Disables new features of autobaud detection (see Section 46.4.5.6.2, "Baud Rate Automatic Detection Protocol Improved," for more details). 0 Autobaud detection new features selected 1 Keep old autobaud detection mechanism
6 RXDSEN	Receive Status Interrupt Enable. Controls the receive status interrupt ($\overline{\textit{interrupt\_uart}}$). When this bit is enabled and RXDS status bit is set, the interrupt $\overline{\textit{interrupt\_uart}}$ will be generated. 0 Disable the RXDS interrupt 1 Enable the RXDS interrupt
5 AIRINTEN	Asynchronous IR WAKE Interrupt Enable. Controls the asynchronous IR WAKE interrupt. An interrupt is generated when AIRINTEN is asserted and a pulse is detected on the RXD pin. 0 Disable the AIRINT interrupt 1 Enable the AIRINT interrupt

Table 46-10. UART Control Register 3 Field Descriptions (continued)

Name	Description
4 AWAKEN	Asynchronous WAKE Interrupt Enable. Controls the asynchronous WAKE interrupt. An interrupt is generated when AWAKEN is asserted and a falling edge is detected on the RXD pin. 0 Disable the AWAKE interrupt 1 Enable the AWAKE interrupt
3 DTRDEN	Data Terminal Ready Delta Enable. Enables / Disables the asynchronous DTRD interrupt. When DTRDEN is asserted and an edge (rising or falling) is detected on $\overline{\text{DTR}}$ (in DCE mode) or on $\overline{\text{DSR}}$ (in DTE mode), then an interrupt is generated. 0 Disable DTRD interrupt 1 Enable DTRD interrupt
2 RXDMUXSEL	RXD Muxed Input Selected. Selects proper input pins for serial and Infrared input signal. **Note:** In this chip, UARTs are used in MUXED mode, so that this bit should always be set. 0 Input pin RXD is not used for serial and IR interfaces 1 Input pin RXD is used for serial and IR interfaces
1 INVT	Inverted Infrared Transmission. Sets the active level for the transmission. When INVT is cleared, the infrared logic block transmits a positive IR 3/16 pulse for all 0s and 0s are transmitted for 1s. When INVT is set (INVT = 1), the infrared logic block transmits an active low or negative infrared 3/16 pulse for all 0s and 1s are transmitted for 1s. 0 Active low transmission 1 Active high transmission.
0 ACIEN	Autobaud Counter Interrupt Enable. This bit is used to enable the autobaud counter stopped interrupt (triggered with ACST (USR2[11]). 0 ACST interrupt disabled 1 ACST interrupt enabled

46.3.3.6 UART Control Register 4 (UCR4)

See Figure 46-8 for illustration of valid bits in the UART Control Register 4 and Table 46-11 for description of the bit fields.

0x008c (UCR4) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CTSTL						INVR	ENIRI	WKEN	IDDMAEN	IRSC	LPBYP	TCEN	BKEN	OREN	DREN
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-8. UART Control Register 4

Table 46-11. UART Control Register 4 Field Descriptions

Name	Description
31–16	Reserved
15–10 CTSTL	CTS Trigger Level. Controls the threshold at which the $\overline{CTS}$ pin is deasserted by the RxFIFO. After the trigger level is reached and the $\overline{CTS}$ pin is deasserted, the RxFIFO continues to receive data until it is full. The CTSTL bits are encoded as shown in the Settings column. 000000 0 characters received 000001 1 characters in the RxFIFO ... ... 100000 32 characters in the RxFIFO (maximum) All Other Settings Reserved
9 INVR	Inverted Infrared Reception. Determines the logic level for the detection. When cleared, the infrared logic block expects an active low or negative IR 3/16 pulse for 0s and 1s are expected for 1s. When INVR is set (INVR1), the infrared logic block expects an active high or positive IR 3/16 pulse for 0s and 0s are expected for 1s. 0 Active low detection 1 Active high detection
8 ENIRI	Serial Infrared Interrupt Enable. Enables/Disables the serial infrared interrupt. 0 Serial infrared Interrupt disabled 1 Serial infrared Interrupt enabled
7 WKEN	WAKE Interrupt Enable. Enables/Disables the WAKE bit to generate an interrupt. The WAKE bit is set at the detection of a start bit by the receiver. 0 Disable the WAKE interrupt 1 Enable the WAKE interrupt
6 IDDMAEN	DMA IDLE Condition Detected Interrupt Enable Enables/Disables the receive DMA request $\overline{dma\_req\_rx}$ for the IDLE interrupt (triggered with IDLE flag in USR2[12]). 0 DMA IDLE interrupt disabled 1 DMA IDLE interrupt enabled
5 IRSC	IR Special Case. Selects the clock for the vote logic. When set, IRSC switches the vote logic clock from the sampling clock to the UART reference clock. The IR pulses are counted a predetermined amount of time depending on the reference frequency. See Section 46.4.7.3, “InfraRed Special Case (IRSC) Bit. 0 The vote logic uses the sampling clock (16x baud rate) for normal operation 1 The vote logic uses the UART reference clock
4 LPBYP	Low Power Bypass. Allows to bypass the low power new features in UART. To use during debug phase. 0 Low power features enabled 1 Low power features disabled
3 TCEN	Transmit Complete Interrupt Enable. Enables/Disables the TXDC bit to generate an interrupt ($\overline{interrupt\_uart}$ = 0) **Note:** An interrupt will be issued as long as TCEN and TXDC are high even if the transmitter is not enabled. In general, user should enable the transmitter before enabling the TXDC interrupt. 0 Disable TXDC interrupt 1 Enable TXDC interrupt
2 BKEN	BREAK Condition Detected Interrupt Enable. Enables/Disables the BRCD bit to generate an interrupt. 0 Disable the BRCD interrupt 1 Enable the BRCD interrupt

Table 46-11. UART Control Register 4 Field Descriptions (continued)

Name	Description
1 OREN	Receiver Overrun Interrupt Enable. Enables/Disables the ORE bit to generate an interrupt. 0 Disable ORE interrupt 1 Enable ORE interrupt
0 DREN	Receive Data Ready Interrupt Enable. Enables/Disables the RDR bit to generate an interrupt. 0 Disable RDR interrupt 1 Enable RDR interrupt

46.3.3.7 UART FIFO Control Register (UFCR)

See Figure 46-9 for illustration of valid bits in the UART FIFO Control Register and Table 46-12 for description of the bit fields.

0x0090 (UFCR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXTL						RFDIV			DCE DTE	RXTL					
RESET	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Figure 46-9. UART FIFO Control Register

Table 46-12. UART FIFO Control Register Field Descriptions

Name	Description
31–16	Reserved
15–10 TXTL	Transmitter Trigger Level. Controls the threshold at which a maskable interrupt is generated by the TxFIFO. A maskable interrupt is generated whenever the data level in the TxFIFO falls below the selected threshold. The bits are encoded as shown in the Settings column. 000000 Reserved 000001 Reserved 000010 TxFIFO has 2 or fewer characters ... ... 011111 TxFIFO has 31 or fewer characters 100000 TxFIFO has 32 characters (maximum) All Other Settings Reserved

Table 46-12. UART FIFO Control Register Field Descriptions (continued)

Name	Description
9–7 RFDIV	Reference Frequency Divider. Controls the divide ratio for the reference clock. The input clock is *module_clock*. The output from the divider is *ref_clk* which is used by BRM to create the 16x baud rate oversampling clock (*brm_clk*). 000 Divide input clock by 6 001 Divide input clock by 5 010 Divide input clock by 4 011 Divide input clock by 3 100 Divide input clock by 2 101 Divide input clock by 1 110 Divide input clock by 7 111 Reserved
6 DCEDTE	DCE/DTE mode select. Selects data communication equipment (DCE) or data terminal equipment (DTE) mode. This bit controls the pin direction of the bi-directional modem pins DSR, DCD, DTR and RI. 0 DCE mode selected 1 DTE mode selected
5–0 RXTL	Receiver Trigger Level. Controls the threshold at which a maskable interrupt is generated by the RxFIFO. A maskable interrupt is generated whenever the data level in the RxFIFO reaches the selected threshold. The RXTL bits are encoded as shown in the Settings column. 000000 0 characters received 000001 RxFIFO has 1 character ... ... 011111 RxFIFO has 31 characters 100000 RxFIFO has 32 characters (maximum) All Other Settings Reserved

46.3.3.8 UART Status Register 1 (USR1)

See Figure 46-10 for illustration of valid bits in the UART Status Register 1 and Table 46-13 for description of the bit fields.

0x0094 (USR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PARITYERR	RTSS	TRDY	RTSD	ESCF	FRAMERR	RRDY	AGTIM	DTRD	RXDS	AIRINT	AWAKE	0	0	0	0
W	w1c			w1c	w1c	w1c		w1c	w1c		w1c	w1c				
RESET	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 46-10. UART Status Register 1

Table 46-13. UART Status Register 1 Field Descriptions

Name	Description
31–16	Reserved
15 PARITYERR	Parity Error Interrupt Flag. Indicates a parity error is detected. PARITYERR is cleared by writing 1 to it. Writing 0 to PARITYERR has no effect. When parity is disabled, PARITYERR always reads 0. At reset, PARITYERR is set to 0. 0 No parity error detected 1 Parity error detected (write 1 to clear)
14 RTSS	$\overline{\text{RTS}}$ Pin Status. Indicates the current status of the $\overline{\text{RTS}}$ pin. A “snapshot” of the pin is taken immediately before RTSS is presented to the data bus. RTSS cannot be cleared because all writes to RTSS are ignored. At reset, RTSS is set to 0. 0 The $\overline{\text{RTS}}$ pin is high (inactive) 1 The $\overline{\text{RTS}}$ pin is low (active)
13 TRDY	Transmitter Ready Interrupt / DMA Flag. Indicates that the TxFIFO emptied below its target threshold and requires data. TRDY is automatically cleared when the data level in the TxFIFO exceeds the threshold set by TXTL bits. At reset, TRDY is set to 1. 0 The transmitter does not require data 1 The transmitter requires data (interrupt posted)
12 RTSD	RTS Delta. Indicates whether the $\overline{\text{RTS}}$ pin changed state. It (RTSD) generates a maskable interrupt. When in STOP mode, RTS assertion sets RTSD and can be used to wake the processor. The current state of the $\overline{\text{RTS}}$ pin is available on the RTSS bit. Clear RTSD by writing 1 to it. Writing 0 to RTSD has no effect. At reset, RTSD is set to 0. 0 $\overline{\text{RTS}}$ pin did not change state since last cleared 1 $\overline{\text{RTS}}$ pin changed state (write 1 to clear)
11 ESCF	Escape Sequence Interrupt Flag. Indicates if an escape sequence was detected. ESCF is asserted when the ESCEN bit is set and an escape sequence is detected in the RxFIFO. Clear ESCF by writing 1 to it. Writing 0 to ESCF has no effect. 0 No escape sequence detected 1 Escape sequence detected (write 1 to clear).
10 FRAMERR	Frame Error Interrupt Flag. Indicates that a frame error is detected. The $\overline{\textit{interrupt\_uart}}$ interrupt will be generated if a frame error is detected and the interrupt is enabled. Clear FRAMERR by writing 1 to it. Writing 0 to FRAMERR has no effect. 0 No frame error detected 1 Frame error detected (write 1 to clear)
9 RRDY	Receiver Ready Interrupt / DMA Flag. Indicates that the RxFIFO data level is above the threshold set by the RXTL bits. (See the RXTL bits description in Table 46-12 for setting the interrupt threshold.) When asserted, RRDY generates a maskable interrupt or DMA request. RRDY is automatically cleared when data level in the RxFIFO goes below the set threshold level. At reset, RRDY is set to 0. 0 No character ready 1 Character(s) ready (interrupt posted)
8 AGTIM	Ageing Timer Interrupt Flag. Indicates that data in the RxFIFO has been idle for a time of 8 character lengths (where a character length consists of 7 or 8 bits, depending on the setting of the WS bit in UCR2, with the bit time corresponding to the baud rate setting) and FIFO data level is less than RxFIFO threshold level (RXTL in the UFCR). Clear by writing a 1 to it. 0 AGTIM is not active 1 AGTIM is active (write 1 to clear)

Table 46-13. UART Status Register 1 Field Descriptions (continued)

Name	Description
7 DTRD	DTR Delta. Indicates whether $\overline{DTR}$ (in DCE mode) or $\overline{DSR}$ (in DTE mode) pins changed state. DTRD generates a maskable interrupt if DTRDEN (UCR3[3]) is set. Clear DTRD by writing 1 to it. Writing 0 to DTRD has no effect. 0 $\overline{DTR}$ (DCE) or $\overline{DSR}$ (DTE) pin did not change state since last cleared 1 $\overline{DTR}$ (DCE) or $\overline{DSR}$ (DTE) pin changed state (write 1 to clear)
6 RXDS	Receiver IDLE Interrupt Flag. Indicates that the receiver state machine is in an IDLE state, the next state is IDLE, and the receive pin is high. RXDS is automatically cleared when a character is received. RXDS is active only when the receiver is enabled. 0 Receive in progress 1 Receiver is IDLE
5 AIRINT	Asynchronous IR WAKE Interrupt Flag. Indicates that the IR WAKE pulse was detected on the RXD pin. Clear AIRINT by writing 1 to it. Writing 0 to AIRINT has no effect. 0 No pulse was detected on the RXD IrDA pin 1 A pulse was detected on the RXD IrDA pin
4 AWAKE	Asynchronous WAKE Interrupt Flag. Indicates that a falling edge was detected on the RXD pin. Clear AWAKE by writing 1 to it. Writing 0 to AWAKE has no effect. 0 No falling edge was detected on the RXD Serial pin 1 A falling edge was detected on the RXD Serial pin
3–0	Reserved

46.3.3.9 UART Status Register 2 (USR2)

See Figure 46-11 for illustration of valid bits in the UART Status Register 2 and Table 46-14 for description of the bit fields.

0x0098 (USR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADET	TXFE	DTRF	IDLE	ACST	RIDELT	RIIN	IRINT	WAKE	DCDDELT	DCDIN	RTSF	TXDC	BRCD	ORE	RDR
W	w1c		w1c	w1c	w1c	w1c		w1c	w1c	w1c		w1c		w1c	w1c	
RESET	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0

Figure 46-11. UART Status Register 2

Table 46-14. UART Status Register 2 Field Descriptions

Name	Description
31–16	Reserved
15 ADET	Automatic Baud Rate Detect Complete. Indicates that an “A” or “a” was received and that the receiver detected and verified the incoming baud rate. Clear ADET by writing 1 to it. Writing 0 to ADET has no effect. 0 ASCII “A” or “a” was not received 1 ASCII “A” or “a” was received (write 1 to clear)
14 TXFE	Transmit Buffer FIFO Empty. Indicates that the transmit buffer (TxFIFO) is empty. TXFE is cleared automatically when data is written to the TxFIFO. Even though TXFE is high, the transmission might still be in progress. 0 The transmit buffer (TxFIFO) is not empty 1 The transmit buffer (TxFIFO) is empty
13 DTRF	DTR edge triggered interrupt flag. This bit is asserted, when the programmed edge is detected on the $\overline{\text{DTR}}$ pin (DCE mode). or on $\overline{\text{DSR}}$ (DTE mode). This flag can cause an interrupt if DTREN (UCR3[13]) is enabled. 0 Programmed edge not detected on DTR/DSR 1 Programmed edge detected on DTR/DSR (write 1 to clear)
12 IDLE	Idle Condition. Indicates that an idle condition has existed for more than a programmed amount frame (see Section 46.4.5.1, “Idle Line Detect”). An interrupt can be generated by this IDLE bit if IDEN (UCR1[12]) is enabled. IDLE is cleared by writing 1 to it. Writing 0 to IDLE has no effect. 0 No idle condition detected 1 Idle condition detected (write 1 to clear)
11 ACST	Autobaud Counter Stopped. In autobaud detection (ADBR=1), indicates the counter which determines the baud rate was running and is now stopped. This means either START bit is finished (if ADNIMP=1), or Bit 0 is finished (if ADNIMP=0). See Section , “New Autobaud Counter Stopped bit and Interrupt,” for more details. An interrupt can be flagged on $\overline{\textit{interrupt\_uart}}$ if ACIEN=1. 0 Measurement of bit length not finished (in autobaud) 1 Measurement of bit length finished (in autobaud). (write 1 to clear)
10 RIDELT	Ring Indicator Delta. This bit is used in DTE mode to indicate that the Ring Indicator input ($\overline{\text{RI}}$) has changed state. This flag can generate an interrupt if RI (UCR3[8]) is enabled. RIDELT is cleared by writing 1 to it. Writing 0 to RIDELT has no effect. 0 Ring Indicator input has not changed state 1 Ring Indicator input has changed state (write 1 to clear)
9 RIIN	Ring Indicator Input. This bit is used in DTE mode to reflect the status if the Ring Indicator input ($\overline{\text{RI}}$). The Ring Indicator input is used to indicate that a ring has occurred. In DCE mode this bit is always zero. 0 Ring Detected 1 No Ring Detected
8 IRINT	Serial Infrared Interrupt Flag. When an edge is detected on the RXD pin during SIR Mode, this flag will be asserted. This flag can cause an interrupt which can be masked using the control bit ENIRI: UCR4 [8]. 0 no edge detected 1 valid edge detected (write 1 to clear)
7 WAKE	Wake. Indicates the start bit is detected. WAKE can generate an interrupt that can be masked using the WKEN bit. Clear WAKE by writing 1 to it. Writing 0 to WAKE has no effect. 0 start bit not detected 1 start bit detected (write 1 to clear)
6 DCDDELT	Data Carrier Detect Delta. This bit is used in DTE mode to indicate that the Data Carrier Detect input ($\overline{\text{DCD}}$) has changed state. This flag can cause an interrupt if DCD (UCR3[9]) is enabled. When in STOP mode, this bit can be used to wake the processor. In DCE mode this bit is always zero. 0 Data Carrier Detect input has not changed state 1 Data Carrier Detect input has changed state (write 1 to clear)

Table 46-14. UART Status Register 2 Field Descriptions (continued)

Name	Description
5 DCDIN	Data Carrier Detect Input. This bit is used in DTE mode reflect the status of the Data Carrier Detect input ($\overline{DCD}$). The Data Carrier Detect input is used to indicate that a carrier signal has been detected. In DCE mode this bit is always zero. 0 Carrier signal Detected 1 No Carrier signal Detected
4 RTSF	RTS Edge Triggered Interrupt Flag. Indicates if a programmed edge is detected on the $\overline{RTS}$ pin. The RTEC bits select the edge that generates an interrupt (see Table 46-23). RTSF can generate an interrupt that can be masked using the RTSEN bit. Clear RTSF by writing 1 to it. Writing 0 to RTSF has no effect. 0 Programmed edge not detected on $\overline{RTS}$ 1 Programmed edge detected on $\overline{RTS}$ (write 1 to clear)
3 TXDC	Transmitter Complete. Indicates that the transmit buffer (TxFIFO) and Shift Register is empty; therefore the transmission is complete. TXDC is cleared automatically when data is written to the TxFIFO. 0 Transmit is incomplete 1 Transmit is complete
2 BRCD	BREAK Condition Detected. Indicates that a BREAK condition was detected by the receiver. Clear BRCD by writing 1 to it. Writing 0 to BRCD has no effect. 0 No BREAK condition was detected 1 A BREAK condition was detected (write 1 to clear)
1 ORE	Overrun Error. When set to 1, ORE indicates that the receive buffer (RxFIFO) was full (32 chars inside), and a 33rd character has been fully received. This 33rd character has been discarded. Clear ORE by writing 1 to it. Writing 0 to ORE has no effect. 0 No overrun error 1 Overrun error (write 1 to clear)
0 RDR	Receive Data Ready—Indicates that at least 1 character is received and written to the RxFIFO. If the URXD register is read and there is only 1 character in the RxFIFO, RDR is automatically cleared. 0 No receive data ready 1 Receive data ready

46.3.3.10 UART Escape Character Register (UESC)

See Figure 46-12 for illustration of valid bits in the UART Escape Character Register and Table 46-15 for description of the bit fields.

0x009c (UESC) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	ESC_CHAR							
W																
RESET	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1

Figure 46-12. UART Escape Character Register

Table 46-15. UART Escape Character Register Field Descriptions

Name	Description
31–8	Reserved
7–0 ESC_CHAR	UART Escape Character. Holds the selected escape character that all received characters are compared against to detect an escape sequence.

46.3.3.11 UART Escape Timer Register (UTIM)

See Figure 46-13 for illustration of valid bits in the UART Escape Timer Register and Table 46-16 for description of the bit fields.

0x00a0 (UTIM) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	TIM											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-13. UART Escape Timer Register

Table 46-16. UART Escape Timer Register Field Descriptions

Name	Description
31–12	Reserved
11–0 TIM	UART Escape Timer. Holds the maximum time interval (in ms) allowed between escape characters. The escape timer register is programmable in intervals of 2 ms. See Section 46.4.5.7, “Escape Sequence Detection,” and Table 46-28 for more information on the UART escape sequence detection. Reset value 0x000 = 2 ms up to 0xFFF = 8.192 s.

46.3.3.12 UART BRM Incremental Register (UBIR)

See Figure 46-14 for illustration of valid bits in the UART BRM Incremental Register and Table 46-17 for description of the bit fields.

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write 0x000F value into the UBIR after finishing detecting baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[1].

Note that software reset resets the register to its reset value.

0x00a4 (UBIR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INC															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-14. UART BRM Incremental Register

Table 46-17. UART BRM Incremental Register Field Descriptions

Name	Description
31–16	Reserved
15–0 INC	Incremental Numerator. Holds the numerator value minus one of the BRM ratio (see Section 46.4.6, “Binary Rate Multiplier (BRM)”). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this field using byte accesses is not recommended and is undefined.

1. The write priority in the new design is not same as the original UART. In the original design, software has higher priority than hardware when writing this register at the same time.

46.3.3.13 UART BRM Modulator Register (UBMR)

See Figure 46-15 for illustration of valid bits in the UART BRM Modulator Register and Table 46-18 for description of the bit fields.

This register can be written by both software and hardware. When enabling the automatic baud rate detection feature hardware can write a proper value into the UBMR based on detected baud rate. Hardware has higher priority when both software and hardware try to write it at the same cycle[1].

Note that software reset resets the register to its reset value.

0x00a8 (UBMR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MOD															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-15. UART BRM Modulator Register

Table 46-18. UART BRM Modulator Register Field Descriptions

Name	Description
31–16	Reserved
15–0 MOD	Modulator Denominator. Holds the value of the denominator minus one of the BRM ratio (see Section 46.4.6, “Binary Rate Multiplier (BRM)”). The UBIR register MUST be updated before the UBMR register for the baud rate to be updated correctly. If only one register is written to by software, the BRM will ignore this data until the other register is written to by software. Updating this register using byte accesses is not recommended and undefined.

1. The write priority in the new design is not same as the original UART. In the original design, software has higher priority than hardware when writing this register at the same time.

46.3.3.14 UART Baud Rate Count Register (UBRC)

See Figure 46-16 for illustration of valid bits in the UART Baud Rate Count Register and Table 46-19 for description of the bit fields.

0x00ac (UBRC) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BCNT															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 46-16. UART Baud Rate Count Register

Table 46-19. UART Baud Rate Count Register Field Descriptions

Name	Description
31–16	Reserved
15–0 BCNT	Baud Rate Count Register. This read only register is used to count the start bit of the incoming baud rate (if ADNIMP=1), or start bit + bit0 (if ADNIMP=0). When the measurement is done, the Baud Rate Count Register contains the number of UART internal clock cycles (clock after divider) present in an incoming bit. BCNT retains its value until the next Automatic Baud Rate Detection sequence has been initiated. The 16 bit Baud Rate Count register is reset to 4 and stays at hex FFFF in the case of an overflow.

46.3.3.15 UART One Millisecond Register (ONEMS)

See Figure 46-17 for illustration of valid bits in the UART One Millisecond Register and Table 46-20 for description of the bit fields.

0x00b0 (ONEMS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0								
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ONEMS															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 46-17. UART One Millisecond Register

Table 46-20. UART One Millisecond Register Field Descriptions

Name	Description
31–24	Reserved
23–0 ONEMS	One Millisecond Register. This 24-bit register must contain the value of the UART internal frequency (*ref_clk* in Figure 46-1) divided by 1000. The internal frequency is obtained after the UART BRM internal divider (F(*ref_clk*) = F(*module_clock*) / RFDIV). In fact this register contains the value corresponding to the number of UART BRM internal clock cycles present in one millisecond. The ONEMS (and UTIM) registers value are used in the escape character detection feature (Section 46.4.5.7, “Escape Sequence Detection”) to count the number of clock cycles left between two escape characters. The ONEMS register is also used in infrared special case mode (IRSC = UCR4[5] = 1’b1), see Section 46.4.7.3, “InfraRed Special Case (IRSC) Bit.”

NOTE

This register has been expanded from 16 bits to 24 bits. In previous versions, the 16-bit ONEMS can only support the maximum 65.535MHz (0xFFFFx1000) *ref_clk*. To support 4Mbps Bluetooth application with 66.5MHz *module_clock*, the value 0x103C4 (66.5M/1000) should be written into this register. In this case, the 16 bits are not enough to contain the 0x103C4. So this register was expanded to 24 bits to support high frequency of the *ref_clk*.

46.3.3.16 UART Test Register (UTS)

See Figure 46-18 for illustration of valid bits in the UART Test Register and Table 46-21 for description of the bit fields.

0x00b4 (UTS) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FRCP ERR	LOOP	DBG EN	LOOP IR	RXDB G	0	0	TXEM PTY	RXE MPTY	TXFU LL	RXFU LL	0	0	SOF TRS T
W																
RESET	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 46-18. UART Test Register

Table 46-21. UART Test Register Field Descriptions

Name	Description
31–14	Reserved
13 FRCPERR	Force Parity Error. Forces the transmitter to generate a parity error if parity is enabled. FRCPERR is provided for system debugging. 0 Generate normal parity 1 Generate inverted parity (error)
12 LOOP	Loop TX and RX for Test. Controls loopback for test purposes. When LOOP is high, the receiver input is internally connected to the transmitter and ignores the RXD pin. The transmitter is unaffected by LOOP. If RXDMUXSEL (UCR3[2]) is set to 1, the loopback is applied on serial and IrDA signals. If RXDMUXSEL is set to 0, the loopback is only applied on serial signals. 0 Normal receiver operation 1 Internally connect the transmitter output to the receiver input
11 DBGEN	$\overline{\text{debug\_enable}}$. This bit controls whether to respond to the *debug_req* input signal. 0 UART will go into debug mode when *debug_req* is HIGH 1 UART will not go into debug mode even if *debug_req* is HIGH
10 LOOPIR	Loop TX and RX for IR Test (LOOPIR). This bit controls loopback from transmitter to receiver in the InfraRed interface. 0 No IR loop 1 Connect IR transmitter to IR receiver
9 RXDBG	RX_fifo_debug_mode. This bit controls the operation of the RX fifo read counter when in debug mode. 0 rx fifo read pointer does not increment 1 rx_fifo read pointer increments as normal
8–7	Reserved
6 TXEMPTY	TxFIFO Empty. Indicates that the TxFIFO is empty. 0 The TxFIFO is not empty 1 The TxFIFO is empty

Table 46-21. UART Test Register Field Descriptions (continued)

Name	Description
5 RXEMPTY	RxFIFO Empty. Indicates the RxFIFO is empty. 0 The RxFIFO is not empty 1 The RxFIFO is empty
4 TXFULL	TxFIFO FULL. Indicates the TxFIFO is full. 0 The TxFIFO is not full 1 The TxFIFO is full
3 RXFULL	RxFIFO FULL. Indicates the RxFIFO is full. 0 The RxFIFO is not full 1 The RxFIFO is full
2–1	Reserved
0 SOFTRST	Software Reset. Indicates the status of the software reset ($\overline{\text{SRST}}$ bit of UCR2). 0 Software reset inactive 1 Software reset active

46.4 Functional Description

This section provides a complete functional description of the module.

46.4.1 Interrupts and DMA Requests

See Table 46-22 for the lists of all interrupt and DMA signals and associated interrupt and DMA sources of the UART. See register description section for explanation of interrupt/DMA enable and status.

Table 46-22. Interrupts and DMA

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
$\overline{\textit{interrupt\_uart}}$	RRDYEN IDEN DREN RXDSEN ATEN	UCR1 (bit 9) UCR1 (bit 12) UCR4 (bit 0) UCR3 (bit 6) UCR2 (bit 3)	RRDY IDLE RDR RXDS AGTIM	USR1 (bit 9) USR2 (bit 12) USR2 (bit 0) USR1 (bit 6) USR1 (bit 8)
$\overline{\textit{interrupt\_uart}}$	TXMPTYEN TRDYEN TCEN	UCR1 (bit 6) UCR1 (bit 13) UCR4 (bit 3)	TXFE TRDY TXDC	USR2 (bit 14) USR1 (bit 13) USR2 (bit 3)

Table 46-22. Interrupts and DMA (continued)

Interrupt/DMA Output	Interrupt/DMA Enable	Enable Register Location	Interrupt/DMA Flag	Flag Register Location
interrupt_uart	OREN BKEN WKEN ADEN ACIEN ESCI ENIRI AIRINTEN AWAKEN FRAERREN PARERREN RTSDEN RTSEN DTREN (DCE) RI (DTE) DCD (DTE) DTRDEN	UCR4 (bit 1) UCR4 (bit 2) UCR4 (bit 7) UCR1 (bit 15) UCR3 (bit 0) UCR2 (bit 15) UCR4 (bit 8) UCR3 (bit 5) UCR3 (bit 4) UCR3 (bit 11) UCR3 (bit 12) UCR1 (bit 5) UCR2 (bit 4) UCR3 (bit 13) UCR3 (bit 8) UCR3 (bit 9) UCR3 (bit 3)	ORE BRCD WAKE ADET ACST ESCF IRINT AIRINT AWAKE FRAERR PARITYERR RTSD RTSF DTRF RIDELT DCDDELT DTRD	USR2 (bit 1) USR2 (bit 2) USR2 (bit 7) USR2 (bit 15) USR2 (bit 11) USR1 (bit 11) USR2 (bit 8) USR1 (bit 5) USR1 (bit 4) USR1 (bit 10) USR1 (bit 15) USR1 (bit 12) USR2 (bit 4) USR2 (bit 13) USR2 (bit 10) USR2 (bit 6) USR1 (bit 7)
dma_req_rx	RXDMAEN ATDMAEN IDDMAEN	UCR1 (bit 8) UCR1 (bit 2) UCR4 (bit 6)	RRDY AGTIM IDLE	USR1 (bit 9) USR1 (bit 8) USR2 (bit 12)
dma_req_tx	TXDMAEN	UCR1 (bit 3)	TRDY	USR1 (bit 13)

46.4.2 Clocks

This section describes clocks and special clocking requirements of the UART.

46.4.2.1 Clock requirements

UART module receives 2 clocks, *peripheral_clock* and *module_clock*. The *peripheral_clock* is used as write clock of the TxFIFO, read clock of the RxFIFO and synchronization of the modem control input pins. It must always be running when UART is enabled. There is an exception in stop mode (see Section 46.4.2.3, “Clocking in Low-Power Modes”).

The *module_clock* is for all the state machines, writing RxFIFO, reading TxFIFO, etc. It must always be running when UART is sending or receiving characters.This clock is used in order to allow frequency scaling on *peripheral_clock* without changing configuration of baud rate (*module_clock* staying at a fixed frequency).

The constraints on *peripheral_clock* and *module_clock* are as follows:

- *peripheral_clock* and *module_clock* can totally be asynchronous. Off course, they can also be synchronous.
- Due to the 16x oversampling of the incoming characters, *module_clock* frequency must always be greater or equal to 16x the maximum baud rate. For example, if max baud rate is 4 Mbit/s, *module_clock* must be greater or equal to 4 M x 16 = 64MHz.

NOTE

The restriction that *peripheral_clock* frequency must be higher or equal to 16x baud rate has been removed. There is no limitation on *peripheral_clock* frequency to baud rate.

46.4.2.2 Maximum Baud Rate

The max baud rate the UART can support is determined by the max frequency of the *module_clock* and logic synthesis results. For example, if SoC can provide the fastest *module_clock* 66.5MHz and the UART synthesis timing is OK under this constraint, the UART can transmit and receive serial data with the max baud rate 66.5M/16 = 4.15 Mbit/s.

The UART supports serial IR interface low speed. In the low speed IrDA mode, the max baud rate is 115.2Kbit/s. To support the 115.2Kbit/s, *module_clock* frequency must be higher or equal to 1.8432MHz.

46.4.2.3 Clocking in Low-Power Modes

The UART supports 2 low-power modes: DOZE and STOP.

In STOP mode (input pin *stop_req* is at '1'), the UART doesn't need any clock. In this mode the UART can wake-up the MCU with the asynchronous interrupts (see Section 46.4.8, "Low Power Modes"). An application of this feature is when the system must be waken-up by the arrival of a frame of characters.

- If before entering in Stop mode the software has enabled RTSDEN interrupt, then when RTS will change state (put at '0' by external device started to send), the asynchronous interrupt will wake-up the system, *peripheral_clock* and *module_clock* will be provided to the UART before first start bit, so that no data will be lost.
- If RTS doesn't change state (already at '0' before entering in Stop mode), then wake-up interrupt (AWAKE) will be sent at the arrival of first Start bit (on falling edge). In this case, the UART must receive the *peripheral_clock* and *module_clock* during the first half of start bit to correctly receive this character (for example, at 115.2 Kbit/s, UART must receive *peripheral_clock* and *module_clock* at maximum 4.3 microseconds after falling edge of Start bit). If the UART receives *peripheral_clock* and *module_clock* too late, first character will be lost, and so should be dropped. Also, if autobaud detection is enabled, the first character won't be correctly received and another autobaud detection will need to be initiated.

In Doze mode, UART behavior is programmable through DOZE bit (UCR1[1]). If DOZE bit is set to '1', then UART is disabled in Doze mode, and in consequence, UART clocks can be switched-off (after being sure UART is not transmitting nor receiving). On the contrary, if DOZE bit is set to '0', UART is enabled and it must receive *peripheral_clock* and *module_clock*.

46.4.3 General UART Definitions

Definitions of terms that occur the following discussions are given in this section.

- Bit Time—The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency).

- Start bit—The bit time of a logic 0 that indicates the beginning of a data frame. A start bit begins with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1.
- Stop bit—1 bit time of logic 1 that indicates the end of a data frame.
- BREAK—A frame in which all of the data bits, including the stop bit, are logic 0. This type of frame is usually sent to signal the end of a message or the beginning of a new message.
- Mark - When no data is being sent, the serial port's transmit pin's voltage is 1 and is said to be in a MARK state.
- Space - The serial port can also be forced to keep the transmit pin at a 0 and is said to be the SPACE or BREAK state.
- Frame—A start bit followed by a specified number of data or information bits and terminated by a stop bit. The number of data or information bits depends on the format specified and must be the same for the transmitting device and the receiving device. The most common frame format is 1 start bit followed by 8 data bits (least significant bit first) and terminated by 1 stop bit. An additional stop bit and a parity bit also can be included.
- Framing Error—An error condition that occurs when the stop bit of a received frame is missing, usually when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. Framing errors can go undetected if a data bit in the expected stop bit time happens to be a logic 1. A framing error is always present on the receiver side when the transmitter is sending BREAKs. However, when the UART is programmed to expect 2 stop bits and only the first stop bit is received, this is not a framing error by definition.
- Parity Error—An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the RXD input. Parity error is calculated only after an entire frame is received.
- Idle—One in NRZ encoding format and selectable polarity in IrDA mode.
- Overrun Error—An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART receive buffer (RxFIFO). An overrun error indicates that the software reading the buffer (RxFIFO) is not keeping up with the actual reception of characters on the RXD input.

46.4.3.1 $\overline{\text{RTS}}$—UART Request To Send

The UART Request To Send input controls the transmitter. The modem or other terminal equipment signals the UART when it is ready to receive by setting '0' on the $\overline{\text{RTS}}$ pin. Normally, the transmitter waits until this signal is active (low) before transmitting a character, however when the Ignore RTS (IRTS) bit is set, the transmitter sends a character as soon as it is ready to transmit. An interrupt (RTSD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion. When $\overline{\text{RTS}}$ is set to '1' during a transmission, the UART transmitter finishes transmitting the current character and shuts off. The contents of the TxFIFO (characters to be transmitted) remain undisturbed. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode.

46.4.3.2 $\overline{\text{RTS}}$ Edge Triggered Interrupt

The input to the $\overline{\text{RTS}}$ pin can be programmed to generate an interrupt on a selectable edge. See Table 46-23 for summary of the operation of the $\overline{\text{RTS}}$ edge triggered interrupt (RTSF).

To enable the $\overline{\text{RTS}}$ pin to generate an interrupt, set the request to send interrupt enable (RTSEN) bit (UCR2[4]) to 1. Writing 1 to the $\overline{\text{RTS}}$ edge triggered interrupt flag (RTSF) bit (USR2[4]) clears the interrupt flag. The interrupt can occur on the rising edge, falling edge, or either edge of the $\overline{\text{RTS}}$ input. The request to send edge control (RTEC) field (UCR2[10:9]) programs the edge that generates the interrupt. When RTEC is set to 0x00 and RTSEN = 1, the interrupt occurs on the rising edge (default). When RTEC is set to 0x01 and RTSEN = 1, the interrupt occurs on the falling edge. When RTEC is set to 0x1X and RTSEN = 1, the interrupt occurs on either edge. This is a synchronous interrupt. The RTSF bit is cleared by writing 1 to it. Writing 0 to RTSF has no effect.

Table 46-23. $\overline{\text{RTS}}$ Edge Triggered Interrupt Truth Table

$\overline{\text{RTS}}$	RTSEN	RTEC [1]	RTEC [0]	RTSF	Interrupt Occurs On...	$\overline{\textit{interrupt\_uart}}$
X	0	X	X	0	Interrupt disabled	1
1–>0	1	0	0	0	Rising edge	1
0–>1	1	0	0	1	Rising edge	0
1–>0	1	0	1	1	Falling edge	0
0–>1	1	0	1	0	Falling edge	1
1–>0	1	1	X	1	Either edge	0
0–>1	1	1	X	1	Either edge	0

There is another $\overline{\text{RTS}}$ interrupt that is not programmable. The status bit RTSD asserts the $\overline{\textit{interrupt\_uart}}$ interrupt when the $\overline{\text{RTS}}$ delta interrupt enable = 1. This is an asynchronous interrupt. The RTSD bit is cleared by writing 1 to it. Writing 0 to the RTSD bit has no effect.

46.4.3.3 $\overline{\text{DTR}}$ - Data Terminal Ready

This signal indicates the general readiness of the Data Terminal Equipment (DTE). This signal is an input in DCE mode and an output in DTE mode. If the connection between the DCE and the DTE is established once, the $\overline{\text{DTR}}$ signal must remain active throughout the whole connection time. In general the $\overline{\text{DTR}}$ and $\overline{\text{DSR}}$ signals are responsible for establishing the connection. $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ are responsible for the data transfer and the transfer direction in the case of a half-duplex configuration. The $\overline{\text{DTR}}$ signal is like a “main switch”. If the $\overline{\text{DTR}}$ signal is inactive the $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals have no effect. In DCE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion.

46.4.3.4 $\overline{\text{DSR}}$ - Data Set Ready

This signal indicates the general readiness of the DCE. This signal is an output in DCE mode and an input in DTE mode. The DCE uses this signal to inform the DTE that it is switched on, has completed all preparations and can communicate with the DTE. In DTE mode, an interrupt (DTRD) can be posted on any transition of this pin and can wake the MCU from STOP mode on its assertion.

46.4.3.5 $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt

The $\overline{\text{DTR}}$ input pin (DCE mode) or $\overline{\text{DSR}}$ input pin (DTE mode) can be configured to cause an interrupt on a selectable edge. See Table 46-24 for summary of the operation of the $\overline{\text{DTR/DSR}}$ edge triggered interrupt. To enable the interrupt, set the DTREN bit (UCR3[13]) to '1'. Write a "one" to the DTRF bit (USR2[13]) to clear the interrupt flag.

The interrupt can be configured to occur on either the rising, falling, or either edge of the $\overline{\text{DTR/DSR}}$ input. Write to the DPEC[1:0] bits (UCR3[15:14]) to program which edge will cause an interrupt. If the bits are set to 00b and DTREN = 1, the interrupt will occur on the rising edge (default). If the bits are set to 01b and DTREN = 1, the interrupt will occur on the falling edge. If the bits are set to 1Xb and DTREN = 1, the interrupt will occur on either edge.

Table 46-24. $\overline{\text{DTR/DSR}}$ Edge Triggered Interrupt Truth Table

$\overline{\text{DTR}}$ / $\overline{\text{DSR}}$	DTREN	DPEC[1]	DPEC[0]	DTRF	Interrupt occurs on:	*$\overline{\text{interrupt\_uart}}$*
X	0	X	X	0	turned off	1
1->0	1	0	0	0	rising edge	1
0->1	1	0	0	1	rising edge	0
1->0	1	0	1	1	falling edge	0
0->1	1	0	1	0	falling edge	1
1->0	1	1	X	1	either edge	0
0->1	1	1	X	1	either edge	0

46.4.3.6 $\overline{\text{DCD}}$ - Data Carrier Detect

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE it has detected the carrier signal and the connection will be set up. This signal remains active while the connection remains established. In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (DCD, UCR3[9]). The change state is reflected in DCDDELT (USR2[6]). Also, the state of the Data Carrier Detect input is mirrored in the status register DCDIN (USR2[5]).

46.4.3.7 $\overline{\text{RI}}$ - Ring Indicator

This signal is an output in DCE mode and an input in DTE mode. If used, the DCE device uses this signal to inform the DTE that a ring just occurred. In DTE mode this input can trigger an interrupt on changing state. This is achieved by setting to '1' the interrupt enable bit (RI, UCR3[8]). The change state is reflected in RIDELT (USR2[10]). Also, the state of the Ring Indicator input is mirrored in the status register RIIN (USR2[9]).

46.4.3.8 $\overline{\text{CTS}}$—Clear To Send

This output pin serves two purposes. Normally, the receiver indicates that it is ready to receive data by asserting this pin (low). When the $\overline{\text{CTS}}$ trigger level is programmed to trigger at 32 characters received and

the receiver detects the valid start bit of the 33 character, it de-asserts this pin. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode.

46.4.3.9 Programmable $\overline{\text{CTS}}$ Deassertion

The $\overline{\text{CTS}}$ output can also be programmed to deassert when the RxFIFO reaches a certain level. Setting the CTS trigger level (UCR4[15:10]) at any value less than 32 negates the $\overline{\text{CTS}}$ pin on detection of the valid start bit of the N + 1 character (where N is the trigger level setting). However, the receiver continues to receive characters until the RxFIFO is full.

46.4.3.10 TXD—UART Transmit

This is the transmitter serial output. When operating in normal mode, NRZ encoded data is output. When operating in infrared mode, a 3/16 bit-period pulse is output for each 0 bit transmitted, and no pulse is output for each 1 bit transmitted. For RS-232 applications, this pin must be connected to an RS-232 transmitter. The operation of this output is the same regardless of whether the UART is in DTE or DCE mode. See Figure 46-19.

46.4.3.11 RXD—UART Receive

This is the receiver serial input. When operating in normal mode, NRZ encoded data is expected. When operating in infrared mode, a narrow pulse is expected for each 0 bit received and no pulse is expected for each 1 bit received. External circuitry must convert the IR signal to an electrical signal. RS-232 applications require an external RS-232 receiver to convert voltage levels. The operation of this input is the same regardless of whether the UART is in DTE or DCE mode. See Figure 46-19.

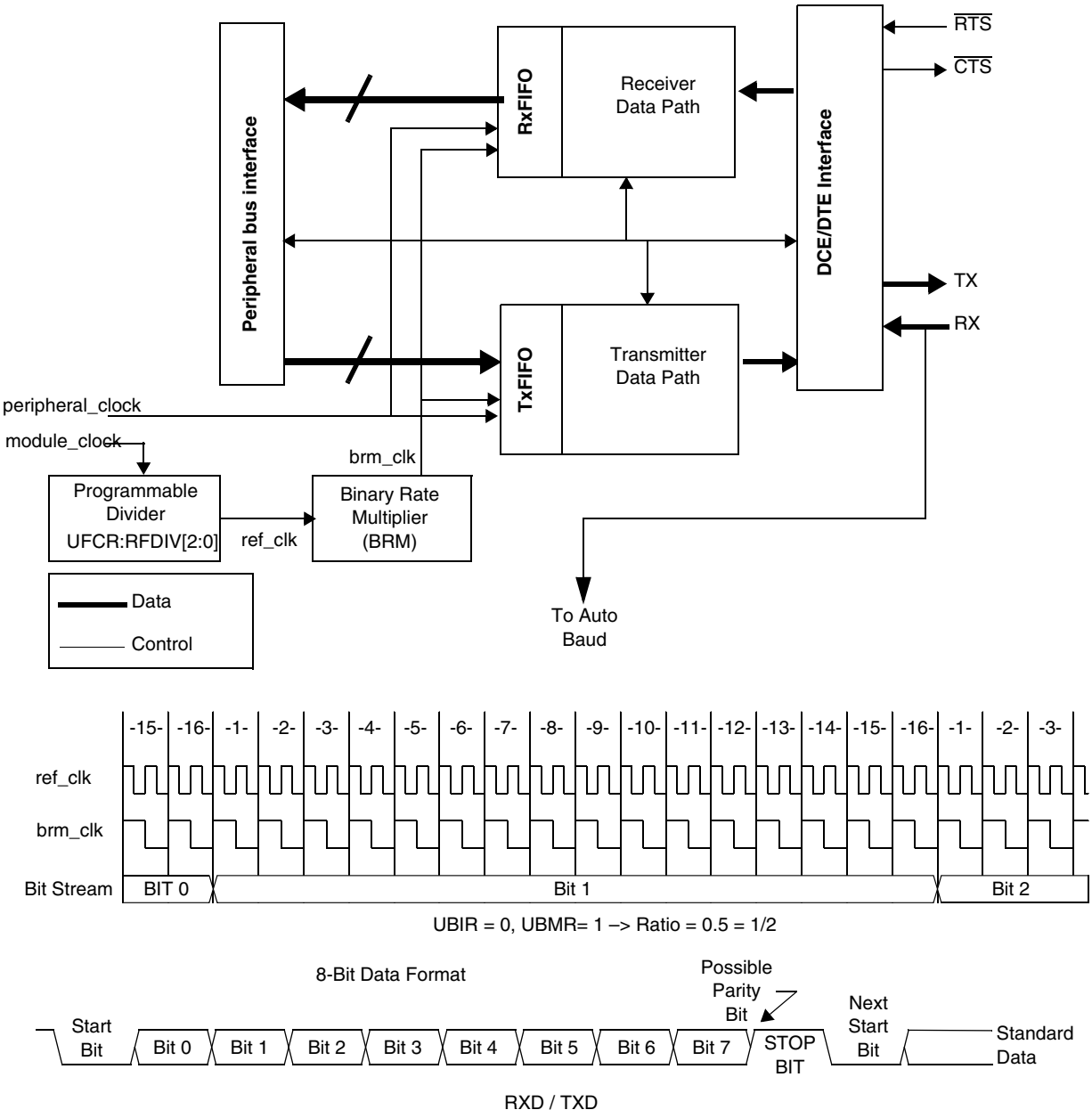


Figure 46-19. UART Simplified Block and Clock Generation Diagrams

46.4.4 Transmitter

The transmitter accepts a parallel character from the MCU and transmits it serially. The start, stop, and parity (when enabled) bits are added to the character. When the ignore RTS bit (IRTS) is set, the transmitter sends a character as soon as it is ready to transmit. $\overline{\text{RTS}}$ can be used to provide flow-control of the serial data. When $\overline{\text{RTS}}$ is set to ‘1’, the transmitter finishes sending the character in progress (if any), stops, and waits for $\overline{\text{RTS}}$ to be set to ‘0’ again. Generation of BREAK characters and parity errors (for debugging

purposes) is supported. The transmitter operates from the clock provided by the BRM. Normal NRZ encoded data is transmitted when the IR interface is disabled.

The transmitter FIFO (TxFIFO) contains 32 bytes. The data is written to TxFIFO by writing to the UTXD register with the byte data to the [7:0] bits. The data is written consecutively if the TxFIFO is not full. It is read (internally) consecutively if the TxFIFO is not empty. TXFULL bit (UTS[4]) can be used to control whether TXFIFO is full or not. The TxFIFO can be written regardless of the transmitter is disabled or enabled. If the UART is disabled, user can write data into the TxFIFO correctly. But in this case the write access will yield to a transfer error if the signal resp_sel is tied to 0.

46.4.4.1 Transmitter FIFO Empty Interrupt Suppression

The transmitter FIFO empty interrupt suppression logic suppresses the TXFE interrupt between writes to the TxFIFO. When TxFIFO is empty, the software can either send one or several characters. If the software sends one character, it would write the character into the UTXD register, then that character is immediately transferred to the transmitter shift register, assuming the transmitter is already enabled. Without interrupt suppression logic, the TXFE interrupt flag would be set immediately. But, with this logic, the interrupt flag is set when the last bit of the character has been transmitted, for example, before the transmission of the parity bit (if exists) and the stop bit(s).

So, the suppression logic doesn't immediately send the TXFE interrupt flag. It allows the software to write another character to the TxFIFO before the interrupt flag is asserted.

When the transmitter shift register empties before another character is written to the TxFIFO, the interrupt flag is asserted. Writing data to the TxFIFO would release the interrupt flag. The interrupt flag is asserted on the following conditions:

- System Reset
- UART software reset
- When a single character has been written to Transmitter FIFO and then the Transmitter FIFO and the Transmitter Shift Register become empty until another character is written to the Transmitter FIFO
- The last character in the TxFIFO is transferred to the shift register, when TxFIFO contains two or more characters. See Figure 46-20.

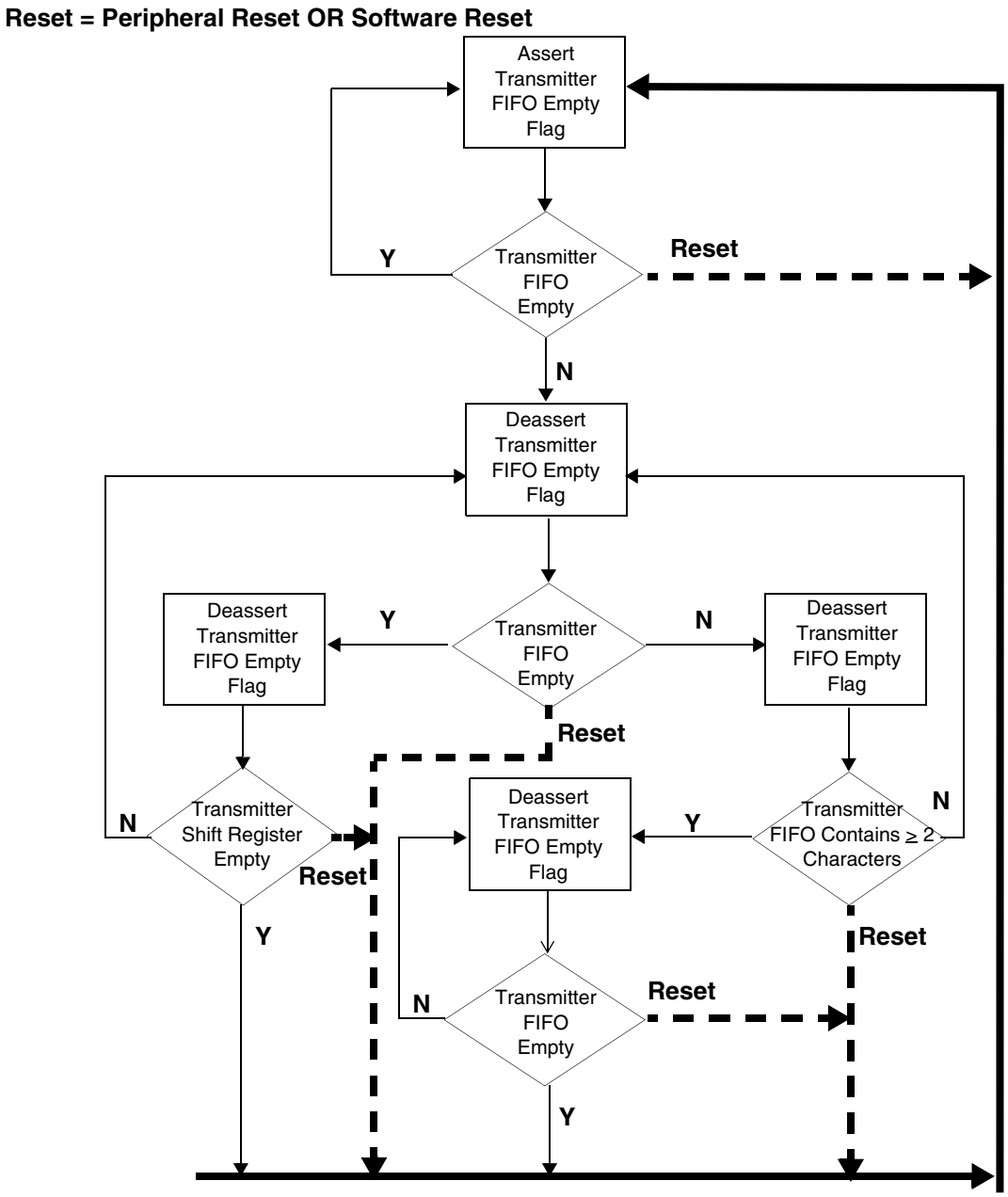


Figure 46-20. Transmitter FIFO Empty Interrupt Suppression Flow Chart

46.4.4.2 Transmitting a Break Condition

Asserting SNDBRK bit of the UCR1 Register forces the transmitter to send a break character (continuous zeros). The transmitter will finish sending the character in progress (if any) before sending break until this bit is reset. The user is responsible to ensure that this bit is high for long enough to generate a valid BREAK. The transmitter samples SNDBRK after every bit is transmitted. Following completion of the BREAK transmission, the UART will transmit two mark bits. The user can continue to fill the FIFO and any character remaining will be transmitted when the break is terminated.

46.4.5 Receiver

See Figure 46-21 for the receiver flow chart. The receiver accepts a serial data stream and converts it into parallel characters. When enabled, it searches for a start bit, qualifies it, and samples the following data bits at the bit-center. Jitter tolerance and noise immunity are provided by sampling at a 16x rate and using voting techniques to clean up the samples. Once the start bit is found, the data bits, parity bit (if enabled), and stop bits (either 1 or 2 depending on user selection) are shifted in. Parity is checked and its status reported in the URXD register when parity is enabled. Frame errors and BREAKs are also checked and reported. When a new character is ready to be read by the MCU from the RxFIFO, the receive data ready (RDR = USR2[0]) bit is asserted and an interrupt is posted (if DREN = UCR4[0] = 1). If the receiver trigger level is set to 2 (RXTL[5:0] = UFCR[5:0] = 2), and 2 chars have been received into RxFIFO, the receiver ready interrupt flag (RRDY = USR1[9]) is asserted and an interrupt is posted if the receiver ready interrupt enable bit is set (RRDYEN = UCR1[9] = 1). If the UART Receiver Register (URXD) is read once, and in consequence there is only 1 character in the RxFIFO, the interrupt generated by the RDR bit is automatically cleared. The RRDY bit is cleared when the data in the RxFIFO falls below the programmed trigger level.

Normal NRZ encoded data is expected when the IR interface is disabled.The RxFIFO contains 32 half-word entries. Characters received are written consecutively into this FIFO. If the FIFO is full and a 33rd characters is received, this character will be ignored and the USR2[ORE] bit will be set.

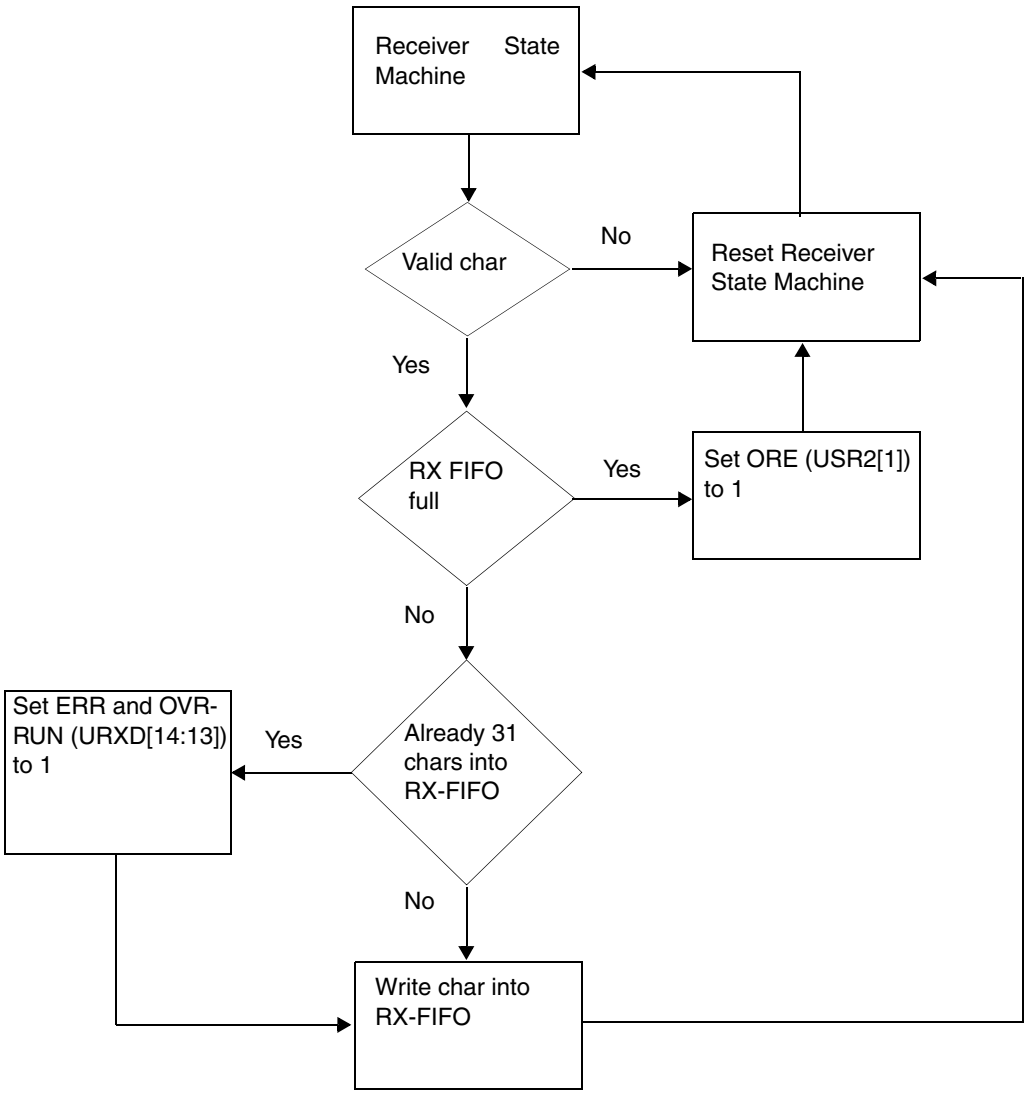


Figure 46-21. Receiver Flow Chart

46.4.5.1 Idle Line Detect

The receiver logic block includes the ability to detect an idle line. Idle lines indicate the end or the beginning of a message.

For an idle condition to occur:

- RxFIFO must be empty and
- RXD pin must be idle for more than a configured number of frames (ICD[1:0] = UCR1[11:10]).

When the idle condition detected interrupt enable (IDEN = UCR1[12]) is set and the line is idle for 4 (default), 8, 16, or 32 (maximum) frames, the detection of an idle condition flags an interrupt (see Table 46-25). When an idle condition is detected, the IDLE (USR2[12]) bit is set. Clear the IDLE bit by writing 1 to it. Writing 0 to the IDLE bit has no effect.

Table 46-25. Detection Truth Table

IDEN	ICD [1]	ICD [0]	IDLE	*interrupt_uart*
0	X	X	0	1
1	0	0	asserted after 4 idle frames	asserted after 4 idle frames
1	0	1	asserted after 8 idle frames	asserted after 8 idle frames
1	1	0	asserted after 16 idle frames	asserted after 16 idle frames
1	1	1	asserted after 32 idle frames	asserted after 32 idle frames

Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal. This table shows how this interrupt affects the *interrupt_uart* signal.

During a normal message there is no idle time between frames. When all of the information bits in a frame are logic 1s, the start bit ensures that at least one logic 0 bit time occurs for each frame so that the IDLE bit is not asserted.

46.4.5.2 Ageing Character Detect

The receiver block also includes the possibility to detect when at least one character has been sitting into the RxFIFO for a time corresponding to 8 characters. This ageing character capability allows the UART to inform the MCU that there is less character into the RxFIFO than the Rx trigger and, no new character has been detected on the RXD line. The ageing capability is a timer which starts to count as soon as the RxFIFO is not empty and its trigger level is not reached (RRDY=0). This counter is reset when either a RxFIFO read is performed or another character starts to present on the RXD line. If none of those two events occurs, the bit AGTIM (USR1[8]) is set when the counter has measured a time corresponding to 8 characters. AGTIM is cleared by writing a 1 to it. AGTIM can flag an interrupt to MCU on *interrupt_uart* if ATEN (UCR2[3]) has been set.

To summarize, AGTIM is set when:

- There is at least one character into RxFIFO.
- No read has occurred on RxFIFO and RXD line has stayed high, for a time corresponding to 8 characters.
- The RxFIFO trigger is not reached (RRDY=0)

46.4.5.3 Receiver Wake

The WAKE bit (USR2[7]) is set when the receiver detects a qualified Start bit. For this, two conditions must be fulfilled, firstly a falling edge on RXD line must be detected and secondly the RXD line must stays at low level for more than a half-bit duration. When the wake interrupt enable WKEN (UCR4[7]) bit is enabled, the receiver flags an interrupt (*interrupt_uart*) if the WAKE status bit is set. The WAKE bit is cleared by writing 1 to it. Writing 0 to the WAKE bit has no effect. The WAKE status bit can be asserted in either serial RS-232 mode or IR mode. The generation of the WAKE interrupt needs the clock *module_clock*.

When the asynchronous wake interrupt (AWAKE) is enabled (AWAKEN = UCR3[4] = 1), and the MCU is in STOP mode, and UART clocks have been shut-off, then a falling edge detected on the receive pin

(RXD) asserts the AWAKE bit (USR1[4]) and the *interrupt_uart* interrupt to wake the MCU from STOP mode. Re-enable UART clocks and clear the AWAKE bit by writing 1 to it. Writing 0 to the AWAKE bit has no effect. When IR interface is enabled (UCR1[7]=1), the AWAKE bit is always not asserted. The generation of the asynchronous AWAKE interrupt does not need any clocks.

In IR mode, if the asynchronous IR WAKE interrupt is enabled (AIRINTEN = UCR3[5] = 1), and if MCU is in STOP mode (UART clocks are off when MCU in STOP mode), then the detection of a falling edge on the receive pin (RXD_IR), asserts the AIRINT bit (USR1[5]), and the *interrupt_uart* interrupt. This interrupt wakes the MCU from STOP mode. Software re-enables UART clocks and clear the AIRINT bit by writing 1 to it. Writing 0 to the AIRINT bit has no effect. When IR interface is disabled (UCR1[7]=0), the AIRINT bit is always not asserted. The generation of the asynchronous AIRINT interrupt does not need any clocks.

Recommended procedure for programming the asynchronous interrupts is to first clear them by writing 1 to the appropriate bit in the UART Status Register 1 (USR1). Poll or enable the interrupt for the Receiver IDLE Interrupt Flag (RXDS) in the USR1. When asserted, the RXDS bit indicates to the software that the receiver state machine is in the idle state, the next state is idle, and the RXD pin is idle (high). After following this procedure, enable the asynchronous interrupt and enter STOP mode.

46.4.5.4 Receiving a BREAK Condition

A BREAK condition is received when the receiver detects all 0s (including a 0 during the bit time of the stop bit) in a frame. The BREAK condition asserts the BRCD bit (USR2[2]) and writes only the first BREAK character to the RxFIFO. Clear the BRCD bit by writing 1 to it. Writing 0 to the BRCD bit has no effect.

Asserting BRCD would generate an interrupt on *interrupt_uart*. The interrupt generation can be masked using the control bit BKEN (UCR4[2]). Receiving a break condition will also effect the following bits in the receiver register URXD:

URXD(11) = BRK. While high this bit indicates that the current char was detected as a break.

URXD(12) = FRMERR. The frame error bit will always be set when BRK is set.

URXD(10) = PRERR. If odd parity was selected the parity error bit will also be set when BRK is set.

URXD(14) = ERR. The error detect bit indicates that the character present in the rx data field has an error status. This can be asserted by a break.

46.4.5.5 Vote Logic

The vote logic block provides jitter tolerance and noise immunity by sampling with respect to a 16x clock (*brm_clk*) and using voting techniques to clean up the samples. The voting is implemented by sampling the incoming signal constantly on the rising edge of the *brm_clk*. See Figure 46-22. The receiver is provided with the majority vote value, which is 2 out of the 3 samples. For examples of the majority vote results of the vote logic, see Table 46-26.

Table 46-26. Majority Vote Results

Samples	Vote
000	0
101	1
001	0
111	1

The vote logic captures a sample on every rising edge of *brm_clk*, however the receiver uses 16x oversampling to take its value in the middle of the sample character.

The receiver starts to count when the Start bit is set however it does not capture the contents of the RxFIFO at the time the Start bit is set. The start bit is validated when 0s are received for 7 consecutive 1/16 of bit times following the 1-to-0 transition. Once the counter reaches 0xF, it starts counting on the next bit and captures it in the middle of the sampling frame (see Table 46-26). All data bits are captured in the same manner. Once the stop bit is detected, the receiver shift register (SIPO_OUT) data is parallel shifted to the RxFIFO.

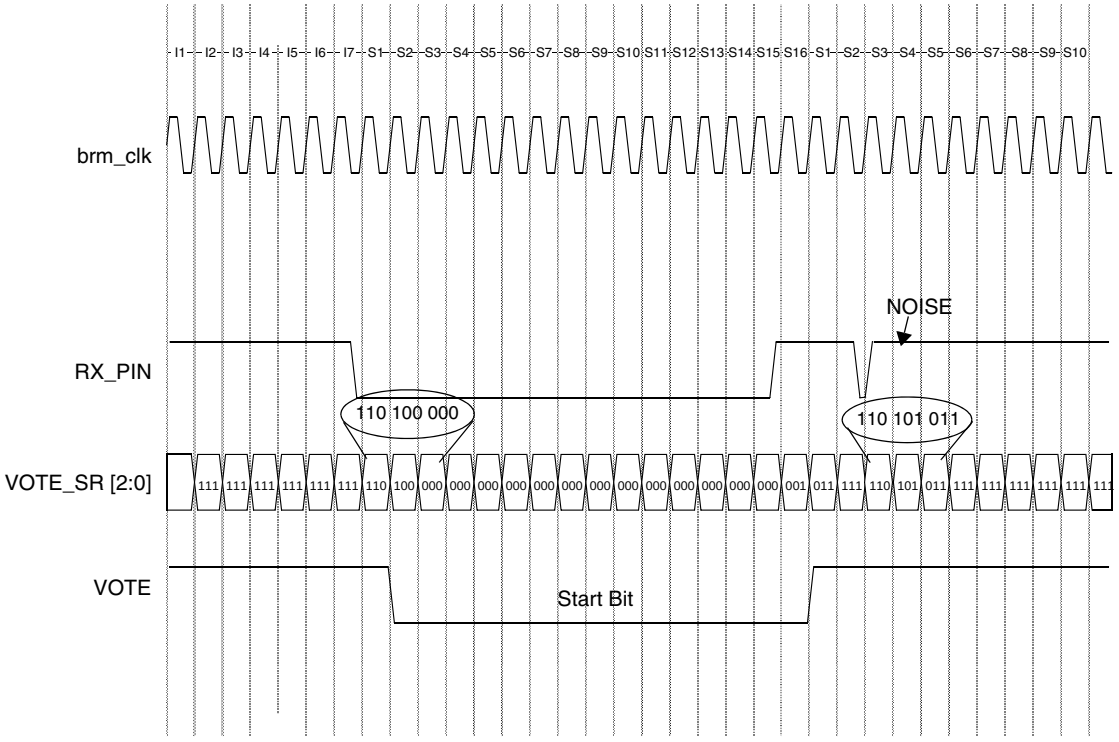


Figure 46-22. Majority Vote Results

A new feature has been recently implemented, it allows to re-synchronize the counter on each edge of RXD line. This is automatic and allows to improve the immunity of UART against signal distortion.

There is a special case when the *brm_clk* frequency is too low and is unable to capture a 0 pulse in IrDA. In this case, the software must set the IRSC (UCR4[5]) bit so that the reference clock (after internal divider) is used for the voting logic. The pulse is validated by counting the length of the pulse.

See Section 46.4.7, “Infrared Interface” for more details.

46.4.5.6 Baud Rate Automatic Detection Logic

When the baud rate automatic detection logic is enabled, the UART locks onto the incoming baud rate. To enable this feature, set the automatic detection of baud rate bit (ADBR = UCR1[14] = 1) and write 1 to the ADET bit (USR2[15]) to clear it. When ADET=0 and ADBR =1, the detection starts. Then, once the beginning of start bit (transition from 1-to-0 of RXD) has been detected, UART start a counter (UBRC) working at reference frequency. Once the end of start bit is detected (transition from 0-to-1 of RXD), the value of UBRC - 1 is directly copied into UBMR register. UBIR register is filled with 0x000F.

So, at the end of start bit, registers gets following values:

```
UBRC = number of reference clock periods (after divider) during Start bit.

UBIR = 0x000F

UBMR = UBRC - 1
```

The updated values of the 3 registers can be read.

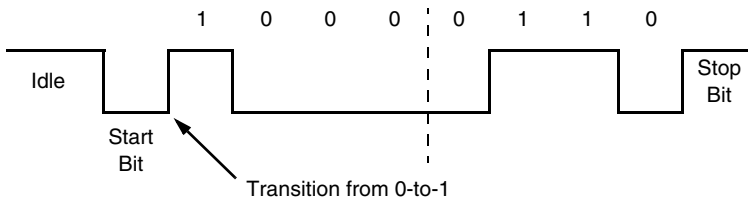
See Table 46-27 for list of parameters for baud rate detection and Figure 46-23 for baud rate detection protocol diagram.

If any of the UART BRM registers are simultaneously written by the baud rate automatic detection logic and by the peripheral data bus, the peripheral data bus would have lower priority.

Table 46-27. Baud Rate Automatic Detection

ADBR	ADET	Baud Rate Detection	*interrupt_uart*
0	X	Manual Configuration	1
1	0	Auto Detection Started	1
1	1	Auto Detection Complete	0

Note: This table assumes that no other interrupt is set at the same time this interrupt is set for the *interrupt_uart* signal.



Note: LSB transmitted first.

Figure 46-23. Baud Rate Detection Protocol Diagram

46.4.5.6.1 Baud Rate Automatic Detection Protocol

The receiver must receive an ASCII character “A” or “a” to verify proper detection of the incoming baud rate. When an ASCII character “A” (0x41) or “a” (0x61) is received and no error occurs, the Automatic Detect baud rate bit is set (ADET=1) and if the interrupt is enabled (ADEN=UCR1[15]=1), an interrupt $\overline{interrupt\_uart}$ is generated.

When an ASCII character “A” or “a” is not received (because of a bit error or the reception of another character), the auto detection sequence restarts and waits for another 1-to-0 transition.

As long as ADET = 0 and ADBR = 1, the UART continues to try to lock onto the incoming baud rate. Once the ASCII character “A” or “a” is detected and the ADET bit is set, the receiver ignores the ADBR bit and continues normal operation with the calculated baud rate.

The UART interrupt is active ($\overline{interrupt\_uart}$ = 0) as long as ADET = 1 and ADBR = 1. This can be disabled by clearing the automatic baud rate detection interrupt enable bit (ADEN = 0). Before starting an automatic baud rate detection sequence, set ADET = 0 and ADBR = 1.

The RxFIFO must contain the ASCII character “A” or “a” following the automatic baud rate detection interrupt.

The 16-bit UART Baud Rate Count Register (UBRC) is reset to 4 and stays at 0xFFFF when an overflow occurs. The UBRC register counts (measures) the duration of start bit. When the start bit is detected and counted, the UART Baud Rate Count Register retains its value until the next automatic baud rate detection sequence is initiated.

The Baud Rate Count Register counts only when auto detection is enabled.

46.4.5.6.2 Baud Rate Automatic Detection Protocol Improved[1]

New Baud Rate Determination

In order to fight against the problems caused by the distortion and the noise on the RXD line, the duration of the baud rate measurement has been extended. Previously, as described above, this determination was based on the measurement of the START bit duration. Now, this measurement is based on the duration of START bit + bit0. Bit0 is the first bit following the START bit. In fact, the counter which is started at the falling edge of START bit is no longer stopped at next rising edge (end of START bit), but it is stopped at the next falling edge (end of bit0). As the character sent is always a “A” (41h) or a “a” (61h), this second falling edge will always be present and it will indicate the end of bit0. Once this counter is stopped, the result is divided by 2 and used by the BRM to determine the incoming baud rate.

NOTE

UBRC register contains the result of this division by two, in consequence it reflects the measurement of the duration of one bit.

1. Several issues have been reported for ICs using the existing autobaud protocol, especially for 57.6 Kbit/s and 115.2 Kbit/s. As a consequence, this protocol has been improved. The old one is still available in the current UART IP, but several modifications can also be used in order to make this autobaud detection more reliable. If the user wants to keep with the old method, he has to set the bit ADNIMP (UCR3[7]) to 1. If this bit isn’t set (default), the autobaud improvements will be used. Those improvements are mainly grouped in two categories: the new baud rate measurement and the new ACST bit (and associated interrupt).

New Autobaud Counter Stopped bit and Interrupt

A new bit has been added in USR2 register: ACST (USR2[11]). This bit is set immediately after the determination of the baud rate, So,

- if ADNIMP is not set (default), ACST is set to 1 after the end of bit0,
- If ADNIMP is set to 1, ACST is set to 1 at the end of START bit.

If ACIEN (UCR3[0]) is set to 1, ACST will flag an interrupt on $\overline{interrupt\_uart}$ signal. This interrupt informs the MCU the BRM has just been set with the result of the bit length measurement. If needed, the MCU can perform a read of UBMR (or UBRC) register and determine by itself the baud rate measured. Then the MCU has the possibility to correct the BRM registers with the nearest standardized baud rate.

NOTE

- ACST is set only if ADBR is set to 1, for example, the UART is autobauding.
- Clear the ACST bit by writing 1 to it. Writing 0 to the ACST bit has no effect.

46.4.5.7 Escape Sequence Detection

An escape sequence typically consists of 3 characters entered in rapid succession (such as +++). Because these are valid characters by themselves, the time between characters determines if it is a valid escape sequence. Too much time between two of the “+” characters is interpreted as two “+” characters, and not part of an escape sequence.

The software chooses the escape character and writes its value to the UART Escape Character Register (UESC). The software must also enable escape detection feature by setting ESCEN (UCR2[11]) to 1. The hardware compares this value to incoming characters in the RxFIFO. When an escape character is detected, the internal escape timer starts to count. The software specifies a time-out value for the maximum allowable time between 2 successive escape characters (see Table 46-28). The escape timer is programmable in intervals of 2 ms to a maximum interval of 8.192 seconds.

Table 46-28. Escape Timer Scaling

UTIM Register	Maximum Time Between Specified Escape Characters
0x000	2 ms
0x001	4 ms
0x002	6 ms
0x003	8 ms
0x004	10 ms
…	…
0F8	498 ms
0F9	500 ms
…	…

Table 46-28. Escape Timer Scaling (continued)

UTIM Register	Maximum Time Between Specified Escape Characters
9C3	5 s
…	…
FFD	8.188 s
FFE	8.190 s
FFF	8.192 s

Note: To calculate the time interval:

(UTIM_Value + 1) × 0.002 = Time_Interval

Example:

(09C3 + 1) × 0.002 = 5 s.

The escape sequence detection feature is available for all the reference frequencies. Before using Escape Sequence Detection, the user must fill the ONEMS register. This 24-bit register must contain the value of the UART internal frequency divided by 1000. The internal frequency is obtained after the UART internal divider which is applied on *module_clock* clock.

Example I:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 2 with the internal divider: UFCR[9:7] = 3’b100

Calculation of Frequency for ONEMS Register ***Eqn. 46-1***

$$ONEMS = \frac{66.5\times10^{6}}{2\times1000} = 33250 = 81E2h$$

Example II:

- If the input clock *module_clock* frequency is 66.5 MHz.
- And if the input clock *module_clock* is divided by 1with the internal divider: UFCR[9:7] = 3’b101

Calculation of Frequency for ONEMS Register ***Eqn. 46-2***

$$ONEMS = \frac{66.5\times10^{6}}{1000} = 66500 = 103C4h$$

The escape sequence detection interrupt is asserted when the escape sequence interrupt enable (ESCI) bit is set and an escape sequence is detected (ESCF set). Clear the ESCF bit by writing 1 to it. Writing 0 to the ESCF bit has no effect.

46.4.6 Binary Rate Multiplier (BRM)

The BRM sub-module receives *ref_clk* (*module_clock* clock after divider). From this clock, and with integer and non-integer division, BRM generates a 16x baud rate clock whose frequency is 16 times of baud rate. The uart transmitter will shift data out based on this 16x baud rate clock. The uart receiver will sample the serial data line based on this 16x baud rate clock. The input and output frequency ratio is programmed in the UART BRM Incremental Register (UBIR) and UART BRM MOD Register (UBMR). The output frequency is divided by the input frequency to produce this ratio. For integer division, set the UBIR = 0x000F and write the divisor to the UBMR register. All values written to these registers must be one less than the actual value to eliminate division by 0 (undefined), and to increase the maximum range of the registers.

Updating the BRM registers requires writing to both registers. The UBIR register must be written before writing to the UBMR register. If only one register is written to by the software, the BRM continues to use the previous values.

The following examples show how to determine what values are to be programmed into UBIR and UBMR for a given reference frequency and desired baud rate. The following equation can be used to help determine these values:

Frequency and Baud Rate for UBIR and UBMR ***Eqn. 46-3***

$$\text{BaudRate} = \frac{\text{RefFreq}}{\left(16 \times \frac{\text{UBMR} + 1}{\text{UBIR} + 1}\right)}$$

With

Reference Frequency (Hz): UART Reference Frequency (*module_clock* after RFDIV divider)

Baud Rate (bit/s): Desired baud rate.

Example 46-1. Integer Division ÷ 21

```
Reference Frequency = 19.44 MHz
UBIR = 0x000F
UBMR = 0x0014
Baud Rate = 925.7 kbit/s
```

NOTE

Observe that each value written to the registers is one less than the actual value.

Example 46-2. Non-Integer Division

```
Reference Frequency = 16 MHz
Desired Baud Rate = 920 Kbits/s
```

$$\frac{UBMR+1}{UBIR+1} = \frac{RefFreq}{16 \times BaudRate} = \frac{16 \times 10^6}{16 \times 920 \times 10^3} = 1.087$$

```
Ratio = 1.087 = 1087 / 1000
UBIR = 999 (decimal)= 0x3E7
UBMR = 1086 (decimal)= 0x43E
Non-Integer Division
Reference Frequency = 25 MHz
Desired Baud Rate = 920 kbit/s
Ratio = 1.69837 = 625 / 368
UBIR = 367 (decimal)= 0x16F
UBMR = 624 (decimal)= 0x270
```

Example 46-3. Non-Integer Division

```
Reference Frequency: 30 MHz
Desired Baud Rate = 115.2 kbit/s
Ratio = 16.276043 = 65153 / 4003
UBIR = 4002 (decimal) = 0x0FA2
UBMR = 65152 (decimal) = 0xFE80
```

46.4.7 Infrared Interface

46.4.7.1 Generalities

The Infrared interface is selected when IREN (UCR1[7]) is set to 1.

The Infrared Interface is compatible with IrDA Serial Infrared Physical Layer Specification. In this specification, a “zero” is represented by a positive pulse, and a “one” is represented by no pulse (line remains low).

In the UART:

In TX: For each “zero” to be transmitted, a narrow positive pulse which is 3/16 of a bit time is generated. For each “one” to be transmitted no pulse is generated (output is low). External circuitry has to be provided to drive an Infrared LED.

In RX: When receiving, a narrow negative pulse is expected for each “zero” transmitted while no pulse is expected for each “one” transmitted (input is high).

NOTE

Rx part of IR block expects to receive an inverted signal compared to IrDA specification. Circuitry external to the IC transforms the Infrared signal to an electrical signal.

The IR interface has an edge triggered interrupt (IRINT). This interrupt validates a zero bit being received. This interrupt is enabled by writing a “one” to ENIRI bit.

The behavior of Infrared Interface is determined by 3 bits INVT (UCR3[1]), INVR (UCR4[9]) and IRSC (UCR4[5]).

46.4.7.2 Inverted Transmission and Reception bits (INVT & INVR)

The values of INVT and INVR depend of the IrDA transceiver connected on the TXD_IR and RXD_IR pins of the UART. If this transceiver is not inverting on both paths Tx and Rx, a Zero is represented by a positive pulse and a One is represented by no pulse (line remains low). In this case, the bit INVT must be set to 0 and the bit INVR must be set to 1 (because Rx IR block expects an inverted signal).

On the contrary user must set INVT=1 and INVR=0 if both paths of the transceiver are inverting, that is, a Zero is represented as a negative pulse and a One is represented by no pulse (line remains high). The transceiver can also be inverting on only one path (Tx or Rx), in this case INVT and INVR must be together equal to 1 or to 0, depending on which path is inverted.

46.4.7.3 InfraRed Special Case (IRSC) Bit

The value to apply to IRSC bit is based on 2 parameters: the baud rate and the Minimum Pulse Duration (MPD) of the transceiver. According to IrDA Standard Specification, for SIR (Serial IR) baud rates from 2.4 Kbit/s to 115.2 Kbit/s this nominal pulse duration is equal to 3/16 of a bit duration (at the selected baud rate). But, for all the baud rates a Minimum Pulse Duration is also specified. According to IrDA Standard, a Zero is represented by a light pulse, so the IrDA transceiver can’t emit a light pulse shorter than the MPD. For SIR, the MPD is constant and equal to 1.41 us.

But user must take into account the electrical MPD associated to the transceiver on the receiver path. Typically this value is 2.0 us, but for some manufacturers MPD can go down to 1.0 us.

In order to understand the meaning of IRSC bit, one must understand how the RX path work in IrDA mode.

When UART is in IrDA mode, a Zero is not only detected by the state of the RXD_IR line, but also with the duration of the pulse. This pulse duration can be measured with 2 different clocks. In this case, clock is selected with the IRSC bit.

- If IRSC = 0, the clock used is the BRM clock.
- If IRSC = 1, the clock used is the UART internal clock (UART clock after the divider (RFDIV)).

In normal operation, IRSC=0. This means at any time, the user must insure the frequency of BRM_clock is high enough to measure the pulse. In the UART and for IRSC=0, the pulse must last at least 2 BRM clock cycles.

If this condition is not fulfilled, IRSC must be set to 1.

Let us take 2 examples, with the Minimum Pulse Duration equals to the MPD of the IrDA specification (in SIR).

Example 46-4. Calculation of BRM Clock Period (Clock Period < 1.41 µs)

The user wants to receive IrDA data at 115.2 Kbit/s. The UBIR and UBMR registers are set in order to create the BRM_clock with a frequency of 16*baud rate = 16 * 115.2 = 1.843 MHz. But at the same time, in order to correctly detect the pulse, the user must be sure that 2* BRM_clock period is lower than 1.41 us. Lets check:

BRM_clock period = 1/1843000 = 542 ns

So 2*BRM_clock period = 1.09 us < 1.41 us. It is fine.

Example 46-5. Calculation of BRM Clock Period (Clock Period > 1.41 µs)

This time the user wants to receive at 19.2 Kbit/s. So, the BRM_clock is set to 16*19200 = 307.2 kHz. Let's check if 2* BRM_clock period < 1.41 us:

BRM_clock period =1/307200 = 3.25 us

So 2*BRM_clock period = 6.50 us >> 1.41 us. It doesn't work.

So, in this case, the BRM clock can't be used to measure the pulse duration and the user must select the UART internal clock by setting IRSC =1.

NOTE

Like for Escape character detection, when IR Special Case is enabled (IRSC=1), the UART must measure a duration. In order to do that, the user must fill the ONEMS register. See Section 46.4.5.7, "Escape Sequence Detection."

46.4.7.4 IrDA interrupt

Serial infrared mode (SIR) uses an edge triggered interrupt flag IRINT (USR2[8]). When INVR =0, detection of a falling edge on the RXD pin asserts the IRINT bit. When INVR=1, detection of a rising edge on the RXD pin asserts the IRINT bit. When IRINT and ENIRI bits are both asserted, the *interrupt_uart* interrupt is asserted. Clear the IRINT bit by writing 1 to it. Writing 0 to the IRINT bit has no effect.

46.4.7.5 Conclusion about IrDA

Before using the UART in IrDA, the baud rate limit must be calculated. This baud rate limit will inform the user if IRSC bit has to be set or not.

Let's determine this limit:

As already described, if IRSC = 0, the following condition must always be fulfilled

Calculation of Baud Rate: ***Eqn. 46-4***

$$2 \times \text{BRMClockPeriod} < \text{MinPulseDuration}$$

So,

$$\text{BRMClockFrequency} > \frac{2}{\text{MPD}}$$

So, knowing BRM_clock frequency = 16 * Baud Rate, we get:

$$\text{BaudRate} > \frac{1}{8 \times \text{MinPulseDuration}}$$

So, the user needs to set IRSC = 0 when:

- If Minimum Pulse Duration = 2.5 us and Baud Rate > 50 Kbit/s.
- If Minimum Pulse Duration = 2.0 us and Baud Rate > 62.5 Kbit/s.
- If Minimum Pulse Duration = 1.41 us and Baud Rate > 88.6 Kbit/s.

NOTE

For baud rates lower than the limit, IRSC must be set to1.

46.4.7.6 Programming IrDA Interface

46.4.7.6.1 High Speed

As an example, the following sequence can be used to program the IrDA interface in order to send and receive characters at 115.2 Kbit/s.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 115.2 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to MCU when 1 char is received into the Rx FIFO (RDR)

Registers values and Programming orders:

UCR1 = 0x0085

```
UCR1[7] = IREN = 1: Enable IR interface
UCR1[0] = UARTEN = 1: Enable UART
```

UTS = 0x0000

UFCR = 0x0981

```
TXTL[5:0] = 0x02: Default value
```

RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30 MHz)

RXTL[5:0] = 0x01: Default value

UBIR = 0x0202

UBMR = 0x20BE Baud rate = 115.2 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027

UCR2[14] = IRTS = 1: Ignore level of RTS input signal

UCR2[5] = WS = 1: Characters are 8-bit length

UCR2[2] = TXEN = 1: Enable Rx path

UCR2 [1] = RXEN = 1: Enable Tx path

UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0000

UCR4 = 0x8201

CTSTL[5:0] = 0x20: Default value

UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)

UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)

The UART is ready to send a character as soon as there is a write into UTXD register. And an interrupt will be sent to MCU when a character is received.

46.4.7.6.2 Low Speed

This time, we keep the same assumptions but the speed is now 9.6 Kbit/s. So, this baud rate is below the limit (even with a Min. Pulse Duration of 2.5 us) and thus IRSC must be set to 1.

Assumptions:

- Input UART clock = 90 MHz
- Internal clock divider = 3 (divide Input UART clock by 3)
- Baud rate = 9.6 Kbit/s
- IrDA transceiver is not inverting on both channels: for Tx and Rx, a Zero is represented by a positive pulse, and a One is represented by no pulse (line stays low).
- Interrupt: Sent to MCU when 1 char is received into the Rx FIFO (RDR).

Registers values and Programming orders:

UCR1 = 0x0085

UCR1[7] = IREN = 1: Enable IR interface

UCR1[0] = UARTEN = 1: Enable UART

UFCR = 0x0981

UFCR[15:10] = TXTL[5:0] = 0x02: Default value

RFDIV[2:0] = 0x3: Divide Input UART clock by 3 (resulting internal clock is 30 MHz)

UFCR[5:0] = RXTL[5:0] = 0x01: Default value

UBIR = 0x00FF

UBMR = 0xC354 Baud rate = 9.6 kbit/s with internal clock = 30 MHz

UCR2 = 0x4027

UCR2[14] = IRTS = 1: Ignore level of RTS input signal

UCR2[5] = WS = 1: Characters are 8-bit length

UCR2[2] = TXEN = 1: Enable Rx path

UCR2 [1] = RXEN = 1: Enable Tx path

UCR2[0] = SRST_B = 1: No software reset

UCR3 = 0x0000

UCR3[1] = INVT = 0: Positive pulse represents 0.

UCR4 = 0x8221

UCR4[15:10] = CTSTL[5:0] = 0x20: Default value

UCR4[9] = INVR = 1: Inverted Infrared Reception (because IrDA transceiver is not inverting)

UCR4[5] = IRSC = 1: Because data rate is below the limit and thus the UART internal clock is used to measure the pulse duration.

UCR4[1] = DREN = 1: To enable RDR interrupt (sent when one char is received)

The UART is now ready to send a character as soon as there is a write into UTXD register. An interrupt will be sent to MCU when a character is received.

46.4.8 Low Power Modes

Table 46-29 shows the UART functionality while in hardware controlled low-power modes. These modes are controlled by the signals *doze_req* and *stop_req*. The control/status/data registers won't change when getting into/out of low power modes.

Table 46-29. UART Low Power State Operation

	Normal State (*doze_req* = 1'b0 & *stop_req* = 1'b0)	Doze State (*doze_req* = 1'b1)		Stop State (*stop_req* = 1'b1)
		DOZE bit = 0	DOZE bit = 1	
UART-Clock	ON	ON	ON	OFF
UART Serial / IrDA	ON	ON	OFF	OFF

46.4.8.1 UART Operation in System Doze Mode

While in Doze State (when *doze_req* input pin is set to 1'b1), the UART behavior depends on the DOZE (UCR1[1]) control bit. While the DOZE bit is negated, the UART serial interface is enabled. While the system is in the Doze State, and the DOZE bit is asserted, the UART is disabled. If the Doze State is entered with the DOZE bit asserted while the UART serial interface was receiving or transmitting data, it will complete the receive/transmit of the current character and signal to the far-end transmitter/receiver to stop sending/receiving.

46.4.8.2 UART Operation in System Stop Mode

The internal baud rate clocks of the transmitter and receiver are gated off if the *stop_req* signal to UART is asserted. Even though the clocks at the input of UART module continue to run during system Stop mode the UART will not do any transmission or reception.

The following UART interrupts wake the MCU processor from STOP mode:

- RTS (RTSD)
- IrDA Asynchronous WAKE (AIRINT)
- Asynchronous WAKE (AWAKE)
- RI (RIDELT in DTE mode only)
- DCD (DCDDELT in DTE mode only)
- DTR (DTRD in DCE mode only)
- DSR (DTRD in DTE mode only)

When an asynchronous WAKE (awake) interrupt exits the MCU from STOP mode, make sure that a dummy character is sent first because the first character may not be received correctly.

46.4.8.3 Power Saving Method in UART

The RXEN (UCR2[1]), TXEN (UCR2[2]) and UARTEN (UCR1[0]) bits are set by the user and provide software control of low-power modes.

Setting the UARTEN (UCR1[0]) bit to 0 shuts off the receiver and transmitter logic and the associated clocks.

If the UART is used only in transmit mode, UARTEN and TXEN must be set to 1. If the UART is used only in receive mode, UARTEN and RXEN must be set to 1. Setting TXEN or RXEN to 0 allows to save a lot of power.

46.4.9 UART Operation in System Debug State

The bit UTS [11] controls whether the UART will respond to the input signal *debug_req*, or whether it will continue to run as normal.

If the UART is programmed to respond to *debug_req*:

1. The UART will halt all operations upon detecting the *debug_req* input.
2. A transfer in progress, either to/from a core (using the IP Bus interface) or to/from an external device, will be completed before halting. This means a single byte/word transfer, not an entire FIFO. Reception of any further data from an external device will be disabled.
3. Internal registers will continue to be writable and readable using the IP Bus interface. A read will leave the contents unaffected.
4. The RX FIFO is affected in debug mode in the following way:
   a) All writes into the RX FIFO are prevented.
   b) The bit RXDBG (UTS[9]) is used to select the readability of the RX FIFO during debug mode:

      RXDBG = 0: hold the read pointer at the location it had upon entering debug mode, and URXD register returns only the data value at that location, no matter how many reads attempted.

      RXDBG = 1, selectable at any time: Allow to read the characters received in Rx FIFO. It will not be possible to re-read previously read locations, nor will it be possible to readjust the read pointer to the value it had prior to entering debug mode.

46.4.10 Reset

This section describes how to reset the module and explains special requirements related to reset.

46.4.10.1 Hardware reset

All of registers, FIFOs, state machines and sequential elements can be reset to their initial values by hardware reset or power on reset.

46.4.10.2 Software reset

The status registers USR1 and USR2, BRM registers UBIR and UBMR, TxFIFO and RxFIFO, and transmitter and receiver state machines can be reset by software reset. Internal logic will remain the software reset signal at active for about 4 *module_clock* cycles. Programmer can follow the following software reset sequence:

1. Clear the $\overline{\text{SRST}}$ bit (UCR2[0])

2. Wait for software reset complete: poll SOFTRST bit (UTS[0]) until it is 0.
3. Re-program baud rate registers: Re-write UBIR and UBMR.

46.4.11 Transfer Error

The UART can generate a transfer error on the peripheral bus in the following cases:

- Core is writing into a read-only register.
- Core is accessing (read or write) an unused location within the assigned address space reserved to UART.
- Core is writing into UTXD register with transmit interface disabled (TXEN=0 or UARTEN=0)
- Core is reading URXD register with receive interface disabled (RXEN=0 or UARTEN=0)

46.4.12 Functional Timing

This section includes timing diagrams for functional signaling.

46.4.12.1 RS-232 Mode

When transmitting a byte, the UART first sends a Start Bit which is logic 0, followed by the data (general 8 bits, but could be 7 bits) followed by parity bit (optional) and one or two Stop Bits, which is logic 1. The sequence is repeated for each byte sent. Figure 46-24 shows a diagram of a what a byte transmission would look like.The receiver also sample data according to this format.

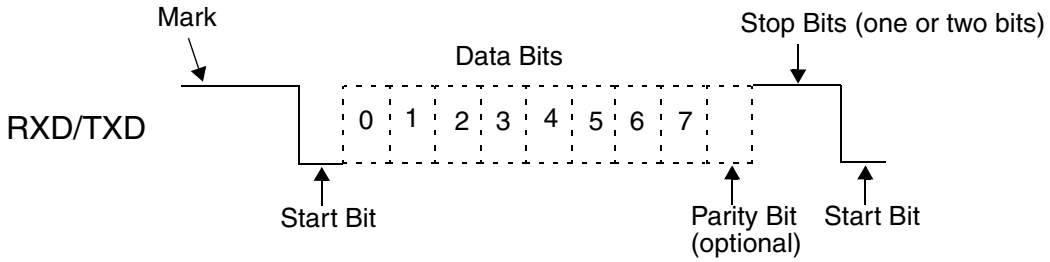


Figure 46-24. Timing Diagram of RS-232 Serial Data Line

46.4.12.2 IrDA Mode

According to IrDA specification, the low speed (115.2Kbit/s and below) IR frame format is compatible with UART frame. Figure 46-25 shows the timing of IR data line and corresponding UART frame. In this figure, an example data 0x65 is used.

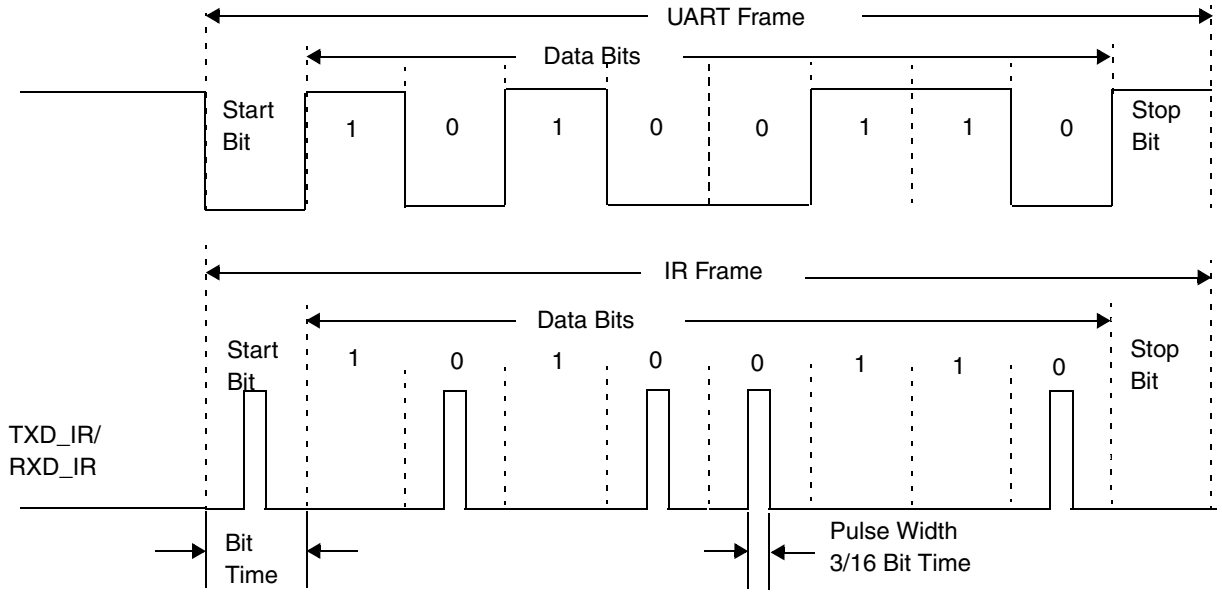


Figure 46-25. Timing diagram of Low Speed IR (<=115.2 Kbit/s) Data Line

46.5 Initialization

As an example, the following sequence can be used to program the UART in order to send and receive characters in RS-232 mode.

Assumptions:

- Input uart clock = 100 MHz
- Baud rate = 921.6Kbps
- Data bits = 8 bits
- Parity = Even
- Stop bits = 1 bit
- Flow control = Hardware

Main program:

1. UCR1 = 0x0001

Enable the UART.

2. UCR2 = 0x2127

Set hardware flow control, data format and enable transmitter and receiver.

3. UCR3 = 0x0704

Set UCR3[RXDMUXSEL] = 1.

4. UCR4 = 0x7C

Set CTS trigger level to 31,

5. UFCR = 0x089E

Set internal clock divider = 5 (divide input uart clock by 5). So the reference clock is 100MHz/5 = 20MHz.

Set TXTL = 2 and RXTL = 30.

6. UBIR = 0x08FF
7. UBMR = 0x0C34

In the above two steps, set baud rate to 921.6Kbps based on the 20MHz reference clock.

8. UCR1 = 0x2201

Enable the TRDY and RRDY interrupts.

Interrupt service routine for the transmitter:

- Write characters into UTXD

The TRDY interrupt will be automatically de-asserted when the data level of the TxFIFO exceeds the TXTL=2. Note: For the first time the interrupt may be de-asserted after 4 characters are written into the TxFIFO because of the shift register.

Interrupt service routine for the receiver:

- Read characters from URXD

The RRDY interrupt will be automatically de-asserted when the data level of the RxFIFO is below the RXTL=30.

Chapter 47
Universal Serial Bus OTG and Host (USBOH)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

The USBOH module contains all of the functionality required to support 2 independent USB ports, compatible with the USB 2.0 specification. In addition to the normal USB functionality, the module also provides support for direct connections to on-board USB peripherals with serial, UTMI or ULPI protocol, and supports multiple interface types for ULPI and serial transceivers.

47.1 Overview

The USBOH module provides high performance USB on-the-go (OTG) functionality, compatible with the USB 2.0 specification, the OTG supplement and the ULPI specification. The module consists of two independent USB cores (host core and OTG core), each with serial and ULPI USB ports.

In addition to the USB cores, the module provides for full-speed transceiverless link (TLL) operation on the OTG and host ports, The OTG core also supplies the UTMI interface for the internal UTMI PHY.

Figure 47-1 shows a block diagram of USBOH.

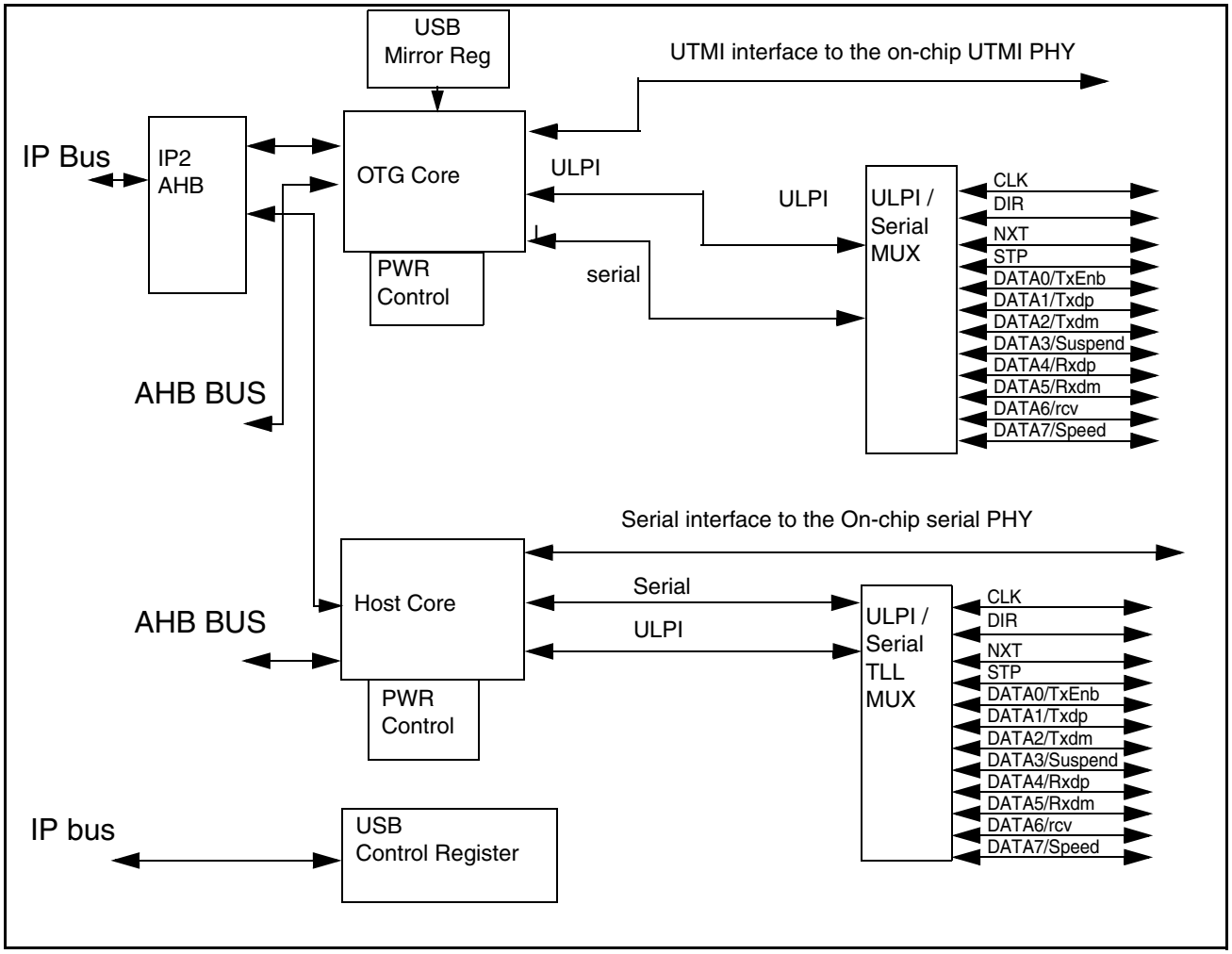


Figure 47-1. USBOH Block Diagram

47.1.1 Features

The USBOH module includes the following features:

- High-speed / full-speed / low-speed host-only core:
  - HS/FS ULPI-compatible interface
  - Software-configurable for full-speed / low-speed interface for Serial transceiver
  - Full-speed transceiverless link logic (FS-TLL) for on-board connection to an FS/LS USB peripheral
  - Software-configurable interface for internal serial PHY, external serial PHY and ULPI PHY selection
- High-speed / full-speed / low-speed OTG core
  - HS/FS ULPI-compatible interface

  - Software-configurable for ULPI or serial transceiver interface
  - High-speed (with ULPI transceiver), full-speed and low-speed operation in host mode
  - High-speed (with ULPI transceiver), and full-speed operation in peripheral mode
  - Hardware support for OTG signaling, session request protocol and host negotiation protocol
  - Up to 8 bidirectional endpoints
  - Software-configurable interface for internal UTMI PHY, external serial PHY and external ULPI PHY selection
- Low power mode with local and remote wake-up capability
- Serial PHY interfaces configurable for bidirectional/unidirectional and differential/single ended
- Embedded DMA controller

47.1.2 Modes of Operation

USBOH modes of operation include normal and low-power modes.

47.1.2.1 Normal Mode

In normal mode, each USB core controls its corresponding port. Both host port and OTG port can work in a number of modes, which are described in Section 47.1.2.1.1, “Host Port Modes,” and Section 47.1.2.1.2, “OTG Port Modes,” respectively.

In addition to these modes, each USB core interface can be configured for high-speed (480 Mbps) and/or full/low-speed operation (12/1.5 Mbps).

47.1.2.1.1 Host Port Modes

The host port supports ULPI and serial transceivers, as well as the internal USB transceiver. The following modes are supported by the host port:

- Internal serial interface mode, used for connecting an on-chip Serial USB PHY
- External serial interface mode, used for connecting an on-board Serial USB PHY
- ULPI interface mode, which is the low pin-count standard for connecting off-chip high-speed USB transceivers to a USB device. When the port is configured for ULPI mode, only a ULPI-compatible transceiver can be used.
- FS TLL mode, which is typically used for on-board USB connections to USB-capable peripherals. This mode emulates the functionality of two back-to-back connected transceivers, so host and peripheral can be connected without PHY. The FS-TLL is used in serial interface mode for full/low-speed transfers.

47.1.2.1.2 OTG Port Modes

The OTG port requires a transceiver and is intended for off-board USB connections. The following modes are supported by the OTG port:

- Serial interface mode. The port does not support dedicated signals for OTG signaling: instead, a transceiver with built-in OTG registers must be used. Typically, the transceiver registers are accessible over an I^2C or SPI interface
- ULPI mode. In this mode, a ULPI transceiver is connected to the port contacts to support high-speed off-board USB connections. ULPI mode is activated by writing the relevant register.
- UTMI mode. In this mode, the on-chip UTMI transceiver is connected to the USB module.

47.1.2.2 Low-Power Mode

Each USB core has an associated power control module that is controlled by the USB core and clocked on a 32 kHz clock. When a USB bus is idle, the transceiver can be placed in low-power (suspend) mode, after which the clocks to the USB core can be stopped. The 32 kHz low power clock must remain active, as it is needed for wake-up detection.

Either the local CPU or the remote USB host/peripheral can initiate a wake-up sequence to resume USB communication.

47.2 External Signal Description

47.2.1 Overview

See Table 47-1 for the list of signals entering and exiting this module to peripherals within the chip.

Table 47-1. Signal Properties

Name	Direction	Reset state	Pull up	Width	Function
OTG External Signals					
ipp_ind_otg_clk	Input	—	—	1	PHY interface clock input
ipp_ind_otg_dir	Input	—	—	1	Controls the direction of the data bus
ipp_ind_otg_nxt	Input	—	—	1	The PHY asserts to throttle the data
ipp_do_otg_clk	Output	0	—	1	Link interface clock output
ipp_obe_otg_clk	Output	0	—	1	PAD Clock I/O direction control
ipp_do_otg_stp	Output	0	—	1	The Link asserts to stop the data stream
ipp_do_otg_data_0	Output	1	—	0	Output Transceiver data[0]
ipp_do_otg_data_1	Output	1	—	0	Output Transceiver data[1]
ipp_do_otg_data_2	Output	1	—	0	Output Transceiver data[2]
ipp_do_otg_data_3	Output	1	—	0	Output Transceiver data[3]
ipp_do_otg_data_4	Output	1	—	0	Output Transceiver data[4]

Table 47-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
ipp_do_otg_data_5	Output	1	—	0	Output Transceiver data[5]
ipp_do_otg_data_6	Output	1	—	0	Output Transceiver data[6]
ipp_do_otg_data_7	Output	1	—	0	Output Transceiver data[7]
ipp_ind_otg_data_0	Input	—	—	0	Input Transceiver data[0]
ipp_ind_otg_data_1	Input	—	—	0	Input Transceiver data[1]
ipp_ind_otg_data_2	Input	—	—	0	Input Transceiver data[2]
ipp_ind_otg_data_3	Input	—	—	0	Input Transceiver data[3]
ipp_ind_otg_data_4	Input	—	—	0	Input Transceiver data[4]
ipp_ind_otg_data_5	Input	—	—	0	Input Transceiver data[5]
ipp_ind_otg_data_6	Input	—	—	0	Input Transceiver data[6]
ipp_ind_otg_data_7	Input	—	—	0	Input Transceiver data[7]
ipp_obe_otg_data_0	Output	1	—	0	Control direction of Transceiver data[0]
ipp_obe_otg_data_1	Output	1	—	0	Control direction of Transceiver data[1]
ipp_obe_otg_data_2	Output	1	—	0	Control direction of Transceiver data[2]
ipp_obe_otg_data_3	Output	1	—	0	Control direction of Transceiver data[3]
ipp_obe_otg_data_4	Output	1	—	0	Control direction of Transceiver data[4]
ipp_obe_otg_data_5	Output	1	—	0	Control direction of Transceiver data[5]
ipp_obe_otg_data_6	Output	1	—	0	Control direction of Transceiver data[6]
ipp_obe_otg_data_7	Output	1	—	0	Control direction of Transceiver data[7]
Host Signals					
ipp_ind_uh1_clk	Input	—	—	1	PHY Transceiver interface clock input
ipp_ind_uh1_dir	Input	—	—	1	Controls the direction of the data bus
ipp_ind_uh1_nxt	Input	—	—	1	The PHY asserts to throttle the data
ipp_ind_uh1_stp	Input	—	—	1	stp input from ULPI PHY, In HS-TLL mode
ipp_do_uh1_clk	Output	0	—	1	Link interface clock Output
ipp_do_uh1_dir	Output	0	—	1	dir output to ULPI PHY, In HS-TLL mode
ipp_do_uh1_nxt	Output	0	—	1	nxt output to ULPI PHY, In HS-TLL mode
ipp_do_uh1_stp	Output	0	—	1	The Link asserts to stop the data stream
ipp_obe_uh1_dir	Output	0	—	1	Direction control of dir, HS-TLL mode = 1, Normal mode = 0
ipp_obe_uh1_nxt	Output	0	—	1	Direction control of nxt, HS-TLL mode = 1, Normal mode = 0
ipp_obe_uh1_stp	Output	0	—	1	Direction control of stp, HS-TLL mode = 0, Normal mode = 1
ipp_obe_uh1_clk	Output	0	—	1	PAD clock I/O direction control

Table 47-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
ipp_do_uh1_data_0	Output	1	—	0	Output Transceiver data[0]
ipp_do_uh1_data_1	Output	1	—	0	Output Transceiver data[1]
ipp_do_uh1_data_2	Output	1	—	0	Output Transceiver data[2]
ipp_do_uh1_data_3	Output	1	—	0	Output Transceiver data[3]
ipp_do_uh1_data_4	Output	1	—	0	Output Transceiver data[4]
ipp_do_uh1_data_5	Output	1	—	0	Output Transceiver data[5]
ipp_do_uh1_data_6	Output	1	—	0	Output Transceiver data[6]
ipp_do_uh1_data_7	Output	1	—	0	Output Transceiver data[7]
ipp_ind_uh1_data_0	Input	—	—	0	Input Transceiver data[0]
ipp_ind_uh1_data_1	Input	—	—	0	Input Transceiver data[1]
ipp_ind_uh1_data_2	Input	—	—	0	Input Transceiver data[2]
ipp_ind_uh1_data_3	Input	—	—	0	Input Transceiver data[3]
ipp_ind_uh1_data_4	Input	—	—	0	Input Transceiver data[4]
ipp_ind_uh1_data_5	Input	—	—	0	Input Transceiver data[5]
ipp_ind_uh1_data_6	Input	—	—	0	Input Transceiver data[6]
ipp_ind_uh1_data_7	Input	—	—	0	Input Transceiver data[7]
ipp_obe_uh1_data_0	Output	1	—	0	Control direction of Transceiver data[0]
ipp_obe_uh1_data_1	Output	1	—	0	Control direction of Transceiver data[1]
ipp_obe_uh1_data_2	Output	1	—	0	Control direction of Transceiver data[2]
ipp_obe_uh1_data_3	Output	1	—	0	Control direction of Transceiver data[3]
ipp_obe_uh1_data_4	Output	1	—	0	Control direction of Transceiver data[4]
ipp_obe_uh1_data_5	Output	1	—	0	Control direction of Transceiver data[5]
ipp_obe_uh1_data_6	Output	1	—	0	Control direction of Transceiver data[6]
ipp_obe_uh1_data_7	Output	1	—	0	Control direction of Transceiver data[7]
Host IC_USB Signals					
ipp_ind_ic_dp	Input	—	—	1	IC-USB DP input
ipp_ind_ic_dm	Input	—	—	1	IC-USB DM input
ipp_do_ic_dp	Output	0	—	1	IC-USB DP output
ipp_do_ic_dm	Output	0	—	1	IC-USB DM output
ipp_obe_ic_dp	Output	1	—	1	DP I/O control
ipp_obe_ic_dm	Output	1	—	1	DM I/O control
ipp_pue	Output	0	—	1	Pad PD/PU enable control
dp_resister_control	Output	0	—	2	DP Pad resister control
dm_resister_control	Output	0	—	2	DM Pad resister control

Table 47-1. Signal Properties (continued)

Name	Direction	Reset state	Pull up	Width	Function
ipp_do_usb_pwr	Output	0	—	1	USB port power control.
ipp_ind_usb_oc	Input	—	—	1	USB overcurrent.
ipp_otg_pwr_select	Output	0	—	1	OTG USB PWR indictor.
ipp_uh1_pwr_select	Output	0	—	1	HOST USB PWR indictor.

47.3 Memory Map/Register Definition

Table 47-2 shows the USBOH module memory map. For the base address of a particular module instantiation, see the system memory map.

The buffers are not accessible using the IP bus, so they do not appear in the memory map.

Table 47-2. USBOH Module Memory Map

Base Address Offset	Controller	Register Name (Name Abbreviation)	AccessSo
0x0000	OTG	ID (UOG_ID)	RO
0x0004	OTG	Hardware General (UOG_HWGENERAL)	RO
0x0008	OTG	Host Hardware Parameters (UOG_HWHOST)	RO
0x0010	OTG	TX Buffer Hardware Parameters (UOG_HWTXBUF)	RO
0x0014	OTG	RX Buffer Hardware Parameters (UOG_HWRXBUF)	RO
0x0080	OTG	General Purpose Timer #0 Load(UOG_GPTIMER0LD)	RW
0x0084	OTG	General Purpose Timer #0 Controller(UOG_GPTIMER0CTRL)	RW
0x0088	OTG	General Purpose Timer #1 Load(UOG_GPTIMER0LD)	RW
0x008c	OTG	General Purpose Timer #1 Controller(UOG_GPTIMER0CTRL)	RW
0x0090	OTG	System Bus interface control (UOG_SBUSCFG)	RW
0x0100	OTG	Capability Register Length (UOG_CAPLENGTH)	RO
0x0102	OTG	Host Interface Version (UOG_HCIVERSION)	RO
0x0104	OTG	Host Control Structural Parameters (UOG_HCSPARAMS)	RO
0x0108	OTG	Control Capability Parameters (UOG_HCCPARAMS)	RO
0x0120	OTG	Device Interface Version (UOG_DCIVERSION)	RO
0x0124	OTG	Device Controller Capability Parameters (UOG_DCCPARAMS)	RO
0x0140	OTG	USB Command Register (UOG_USBCMD)	RW
0x0144	OTG	USB Status Register (UOG_USBSTS)	RW
0x0148	OTG	Interrupt Enable Register (UOG_USBINTR)	RW
0x014C	OTG	USB Frame Index (UOG_FRINDEX)	RW

Table 47-2. USBOH Module Memory Map (continued)

Base Address Offset	Controller	Register Name (Name Abbreviation)	AccessSo
0x0154	OTG	Host Controller Frame List Base Address (UOG_PERIODICLISTBASE)	RW
0x0158	OTG	Host Controller Next Asynch. Address (UOG_ASYNCLISTADDR)	RW
0x0160	OTG	Host Controller Embedded TT Asynch. Buffer Status (UOG_BURSTSIZE)	RW
0x0164	OTG	TX FIFO Fill Tuning (UOG_TXFILLTUNING)	RW
0x0170	OTG	ULPI Viewport (UOG_ULPIVIEW)	RW
0x0180	OTG	Config Flag (UOG_CFGFLAG)	RO
0x0184	OTG	Port Status & Control (UOG_PORTSC1)	RW
0x01A4	OTG	On-The-Go Status & control (UOG_OTGSC	RW
0x01A8	OTG	USB Device Mode (UOG_USBMODE)	RW
0x01AC	OTG	Endpoint Setup Status (UOG_ENDPTSETUPSTAT)	RW
0x01B0	OTG	Endpoint Initialization (UOG_ENDPTPRIME)	RW
0x01B4	OTG	Endpoint De-Initialize (UOG_ENDPTFLUSH)	RW
0x01B8	OTG	Endpoint Status (UOG_ENDPTSTAT)	RO
0x01BC	OTG	Endpoint Complete (UOG_ENDPTCOMPLETE)(	RW
0x01C0	OTG	Endpoint Control0 (UOG_ENDPTCTRL0)	RW
0x01C4	OTG	Endpoint Control1 (UOG_ENDPTCTRL1)	RW
0x01C8	OTG	Endpoint Control2 (UOG_ENDPTCTRL2)	RW
0x01CC	OTG	Endpoint Control3 (UOG_ENDPTCTRL3)	RW
0x01D0	OTG	Endpoint Control4 (UOG_ENDPTCTRL4)	RW
0x01D4	OTG	Endpoint Control5 (UOG_ENDPTCTRL5)	RW
0x01D8	OTG	Endpoint Control6 (UOG_ENDPTCTRL6)	RW
0x01DC	OTG	Endpoint Control07(UOG_ENDPTCTRL7)	RW
0x0200	Host	Host ID (UH1_ID)	RO
0x0204	Host	Hardware General (UH1_HWGENERAL)	RO
0x0208	Host	Host Hardware Parameters (UH1_HWHOST)	RO
0x0210	Host	TX Buffer Hardware Parameters (UH1_HWTXBUF)	RO
0x0214	Host	RX Buffer Hardware Parameters (UH1_HWRXBUF)	RO
0x0280	Host	General Purpose Timer #0 Load(UH1_GPTIMER0LD)	RW
0x0284	Host	General Purpose Timer #0 Controller(UH1_GPTIMER0CTRL)	RW
0x0288	Host	General Purpose Timer #1 Load(UH1_GPTIMER0LD)	RW
0x028c	Host	General Purpose Timer #1 Controller(UH1_GPTIMER0CTRL)	RW
0x0300	Host	Capability Register Length (UH1_CAPLENGTH)	RO

Table 47-2. USBOH Module Memory Map (continued)

Base Address Offset	Controller	Register Name (Name Abbreviation)	AccessSo
0x0302	Host	Host Interface Version (UH1_HCIVERSION)	RO
0x0304	Host	Host Control Structural Parameters (UH1_HCSPARAMS)	RO
0x0308	Host	Control Capability Parameters (UH1_HCCPARAMS)	RO
0x0340	Host	USB Command Register (UH1_USBCMD)	RW
0x0344	Host	USB Status Register (UH1_USBSTS)	RW
0x0348	Host	Interrupt Enable Register (UH1_USBINTR)	RW
0x034C	Host	USB Frame Index (UH1_FRINDEX)	RW
0x0354	Host	Host Controller Frame List Base Address (UH1_PERIODICLISTBASE)	RW
0x0358	Host	Host Controller Next Asynch. Address (UH1_ASYNCLISTADDR)	RW
0x0360	Host	Host Controller Embedded TT Asynch. Buffer Status (UH1_BURSTSIZE)	RW
0x0364	Host	TX FIFO Fill Tuning (UH1_TXFILLTUNING)	RW
0x0370	Host	ULPI Viewport (UH1_ULPIVIEW)	RW
0x0380	Host	Reserved	RO
0x0384	Host	Port Status & Control (UH1_PORTSC1)	RW
0x03A8	Host	USB Device Mode (UH1_USBMODE)	RW
0x0600	USBOH	USB Control Register (USB_CTRL)	RW
0x0604	USBOH	USB OTG Mirror Register (USB_OTG_MIRROR)	RW
0x0608	USBOH	USB OTG PHY Function Control Register (USB_PHY_CTRL_FUNC)	RW
0x060c	USBOH	USB OTG UTMI PHY Test Control Register (USB_FHY_CTRL_TEST)	RW

47.3.1 Register Descriptions

This section describes only the registers that are additional to the HS-USB Core register set. For a detailed description of the registers inside the HS-USB controllers, see Section 47.5.2, “Register Interface”.

47.3.1.1 USB Control Register (USB_CTRL)

The USB control register controls the integration-specific features of the USB module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Offset 0x0600 (USB_CTRL) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWIR	OSIC		OUIE	OWIE	HEXT EN	OEXT EN	OPM	H2WIR	HSIC		HUIE	HWIE	PP_ HST	0	HPM
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VBUS _WK EN	ID_W KEN	OLK EN	HLK EN	PP_ OTG	XCSO	XCSH	IPPUI DP	IPPUE-UP	IPPUE -DWN	HSTD	USBTE	OCP OL_ OTG	OCP OL_ HST	HOCS	OOC S
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Figure 47-2. USB_CTRL Register

Table 47-3. USB_CTRL Register Field Description

Field	Description
31 OWIR	OTG wake-up interrupt request. This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt. 0 No wake-up detected (default) 1 Wake-up interrupt request received
30–29 OSIC	OTG serial interface configuration. Controls the interface type of the OTG port when used with a serial transceiver. This bit field allows for configuring the serial interface for single ended or differential operation combined with bidirectional or unidirectional operation. 00 Differential/unidirectional (6-wire) 01 Differential/bidirectional (4-wire) 10 Single-ended/unidirectional (6-wire) (default) 11 Single-ended/bidirectional (3-wire)
28 OUIE	OTG ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver trigger the wake-up logic. This bit is only meaningful when a ULPI transceiver is selected. 0 ULPI transceiver interrupts are ignored by the wake-up logic (default) 1 ULPI transceiver interrupts activate the wake-up logic
27 OWIE	OTG wake-up interrupt enable. This bit enables or disables the OTG wake-up interrupt. Disabling the interrupt also clears the interrupt request bit. W789y6*666655*ake-up interrupt enable must be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode. 0 Interrupt disabled (default) 1 Interrupt enabled
26 HEX_TEN	Host external ULPI clock enable. Selects whether the host clock comes from external PHY or internal PLL. This bit must be set to 1 before setting the host core into ULPI mode. It must be cleared when setting the host core into FS serial mode 0 Select host clock from internal PLL (serial mode) (default) 1 Select host clock from external PHY (ULPI mode)
25 OEX_TEN	OTG external ULPI clock enable. Select whether the OTG clock comes from external PHY or internal PLL. This bit must be set to 1 before setting the OTG core into ULPI mode. It must be cleared when setting OTG core into FS serial mode 0 Select OTG clock from internal PLL (serial mode) (default) 1 Select OTG clock from external PHY (ULPI mode)

Table 47-3. USB_CTRL Register Field Description (continued)

Field	Description
24 OPM	OTG power mask. Controls whether or not the external Vbus power and overcurrent detection are active for the OTG port. 0 The USBPWR signal asserts with the OTG core's Vbus power enable, and the assertion of the OC input is reported to the OTG core. (default) 1 The USBPWR and OC signals are not used by the OTG core.
23 HWIR	Host wake-up interrupt request. Indicates a pending wake-up request on host port 2. This bit is cleared by disabling the interrupt. The interrupt must be disabled for at least 2 clock cycles of the standby clock. 0 No wake-up interrupt received (default) 1 Wake-up interrupt received
22–21 HSIC	Host serial interface configuration. Controls the interface type of the Host port when used with a serial transceiver. This bit field allows for configuring the serial interface for single ended or differential operation combined with bidirectional or unidirectional operation. 00 Differential/unidirectional (6-wire) 01 Differential/bidirectional (4-wire) 10 Single-ended/unidirectional (6-wire) (default) 11 Single-ended/bidirectional (3-wire)
20 HUIE	Host ULPI interrupt enable. Controls whether or not interrupts from the ULPI transceiver triggers the wake-up logic. This bit is only effective when a ULPI transceiver is selected. 0 ULPI transceiver interrupts are ignored by the wake-up logic. (default) 1 ULPI transceiver interrupts activate the wake-up logic
19 HWIE	Host wake-up interrupt enable. This bit enables or disables the host wake-up interrupt. Disabling the interrupt also clears the interrupt request bit. Wake-up interrupt enable must be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 0 Interrupt disabled (default) 1 Interrupt enabled
18 PP_HST	Power polarity for Host. Controls the polarity of the PWR output signal of Host 0 Low active (power is supplied when PWR is negated) (default) 1 High active (power is supplied when PWR is asserted)
17	Reserved.
16 HPM	Host power mask. The power mask bit controls whether or not the external Vbus power and overcurrent detection are active for the host port. 0 The USBPWR signal asserts the host core's Vbus power enable and the assertion of the OC input is reported to the host core.(default) 1 The USBPWR and OC signals are not used by the host core.
15 VBUS_WKEN	OTG VBUS wake-up enable The VBUS wake-up enable/disable of OTG. When using internal UTMI PHY and OTG works in device mode, if nothing attached to the OTG port and user want to enter into suspend mode to save power, set this bit will enable a wake-up event after VBUS pin changed (external USB Host is attached to the OTG port), this wake-up event will generate interrupt if OWIE is enabled. 0 OTG VBUS wake-up disable.(default) 1 OTG VBUS wake-up enable.

Table 47-3. USB_CTRL Register Field Description (continued)

Field	Description
14 ID_WKEN	OTG ID wake-up enable The ID wake-up enable/disable of OTG. When using internal UTMI PHY and OTG works in device mode, if nothing attached to the OTG port and user want to enter into suspend mode to save power, set this bit will enable a wake-up event after ID pin changed (external USB Device is attached to the OTG port), this wake-up event will generate interrupt if OWIE is enabled. 0 OTG ID wake-up disable.(default) 1 OTG ID wake-up enable.
13 OLKEN	OTG AHB lock enable The AHB lock enable/disable of OTG. 0 OTG lock disable. The OTG core will never use locked transfer to access memory (default) 1 OTG lock enable. The OTG core will always use locked transfer to access memory
12 HLKEN	Host AHB lock enable The AHB lock enable/disable of Host. 0 Host lock disable. The Host core will never use locked transfer to access memory (default) 1 Host lock enable. The Host core will always use locked transfer to access memory
11 PP_OTG	Power polarity. Controls the polarity of the PWR output signal of OTG. 1 High active (power is supplied when PWR is asserted) 0 Low active (power is supplied when PWR is negated)
10 XCSO	XCVR clock select for OTG port. This bit enables software to force the OTG port to use the internal 60 MHz clock. This is used when the PHY's 60 MHz clocks have not been supplied to the OTG port but the OTG port is in the XCVR supply clock mode. 0 OTG port is not forced to use the internal 60 MHz (default) 1 OTG port is forced to use the internal 60 MHz clock
9 XCSH	XCVR clock select for host port. This bit enables software to force the host port to use the internal 60 MHz clock. This is used when the PHY's 60 MHz clocks have not been supplied to the host port but the host port is in the XCVR supply clock mode. 0 Host port is not forced to use the internal 60 MHz (default) 1 Host port is forced to use the internal 60 MHz clock
8 IP_PUIDP	This bit enables software to force the signal ipp_puimpel_pullup_dp to the on-chip full-speed PHY. 0 Software does not control ipp_puimpel_pullup_dp (default) 1 Software has control over ipp_puimpel_pullup_dp.
7 IP_PUE_UP	This bit enables software to force the signal ipp_pue_pullup_dpdm to the on-chip full-speed PHY. 0 Software does not control ipp_pue_pullup_dpdm (default) 1 Software has control over ipp_pue_pullup_dpdm.
6 IP_PUE_DWN	This bit enables software to force the signal ipp_pue_pulldwn_dpdm to the on-chip full-speed PHY. 0 Software does not control ipp_pue_pulldwn_dpdm (default) 1 Software has control over ipp_pue_pulldwn_dpdm.
5 HSTD	Host serial TLL disable. This bit controls whether or not the serial transceiverless link logic (FS-TLL) is enabled for the serial interface of host port. 0 Serial TLL is enabled (default) 1 Serial TLL is disabled
4 USBTE	USB transceiver enable. This bit controls to select the on-chip serial for host 0 USBT is disabled (default) 1 USBT is enabled

Table 47-3. USB_CTRL Register Field Description (continued)

Field	Description
3 OCPOL_OTG	Overcurrent polarity of OTG. This bit selects the polarity of OTG overcurrent. 0 Low active 1 High active (default)
2 OCPOL_HST	Overcurrent polarity of host. This bit selects the polarity of host overcurrent. 0 Low active 1 High active (default)
1 HOCS	Host overcurrent state. This bit shows the overcurrent state of the host. 1 The host has an overcurrent event 0 The host has no overcurrent event
0 OOCS	OTG overcurrent state. This bit is to show the overcurrent state of the OTG. 0 The OTG has no overcurrent event 1 The OTG has an overcurrent event

47.3.1.2 OTG Port Mirror Register (USB_OTG_MIRROR)

The OTG port is designed for operation with an external OTG transceiver. When a ULPI transceiver is in use, all OTG signaling is communicated over the ULPI data bus as described in the ULPI specification. However, when a serial transceiver is used, the interface for OTG signaling is not standardized. Most OTG transceivers use a serial interface like I2C or SPI to transfer the OTG signaling back to the CPU and/or USB core. In this case, the USB CORE has no direct connection the OTG signals in the transceiver.

The USB_OTG_MIRROR register provides a soft interface between the OTG signals in the transceiver and the OTG signal inputs to the USB core. The USB driver software is responsible for reading the OTG status registers in the transceiver over the serial interface and set the bits accordingly in the USB_OTG_MIRROR register, such that the USB controller knows the status of the transceiver.

The USB driver must be designed to meet the latency requirements as defined in the USB 2.0 OTG supplement specification.

Offset 0x0604 (USB_OTG_MIRROR)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OUTMI CLK	OULP ICLK	HULP ICLK	0	SESEND	VBUS VAL	BSES VLD	ASES VLD	IDDIG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-3. USB_OTG_MIRROR Register

Table 47-4. USB_OTG_MIRROR Register Field Descriptions

Field	Description
31–9	Reserved.
8 OUTMICLK	OTG UTMI PHY clock on detection. This read-only status bit indicates the external USB UTMI PHY clock is on and supplied to the module. 0 The OTG UTMI input clock is off 1 The OTG UTMI input clock is on
7 OULPICLK	OTG ULPI PHY clock on detection. This read-only status bit indicates the external USB ULPI PHY clock is on and supplied to the module. 0 The OTG ULPI input clock is off 1 The OTG ULPI input clock is on
6 HULPICLK	Host ULPI PHY clock on detection. This read-only status bit indicates the external USB ULPI PHY clock is on and is supplied to module. 0 The host ULPI input clock is off 1 The host ULPI input clock is on
5	Reserved.
4 SESEND	B device session end. This bit is set by the USB driver when the PHY reports a session end condition. 0 Session active 1 Session end (0.2 V < Vbus < 0.8 V)
3 VBUSVAL	Vbus valid. The USB driver sets this bit when the transceiver reports Vbus valid. 0 Vbus invalid (Vbus < 4.4 V) 1 Vbus is valid (Vbus > 4.4 V)
2 BSESVLD	B session valid. This bit is set by the USB driver when a valid ‘B session’ level is detected on Vbus. 0 B Session is not valid (Vbus < 0.8 V) 1 B session is valid (0.8V < Vbus < 4.0 V)
1 ASESVLD	A session valid. This bit is set by the USB driver when a valid ‘A session’ level is detected on Vbus. 0 Session is not valid for A device. 1 A session is Valid (0.8 V < Vbus < 2.0 V)
0 IDDIG	OTG ID-contact status. This bit indicates to the USB core whether it operates as A-device or as B-device 0 ID contact is low -- operate as A-device 1 ID contact is high -- operate as B-device

47.3.1.3 OTG UTMI PHY Function Control Register (USB_PHY_CTRL_FUNC)

This register controls the on-chip UTMI PHY function signals.

Offset 0x0608 (USB_PHY_CTRL_FUNC) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	CLKVLD	SUSP	RESET	USBEN	EVDO	LSFE	0	VCTL_LD	VCTL_ADDR			
W																
Reset	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VCTL_DATA								VSTS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-4. USB_PHY_CTRL_FUNC Register Diagram

Table 47-5. USB_PHY_CTRL_FUNC Register Field Description

Field	Description
31–28	Reserved.
27 CLKVLD	UTMI clock valid. This read-only bit indicates the clock from the on-chip UTMI PHY is valid and stable 0 The clock is not stable 1 The clock is stable
26 SUSP	UTMI suspend. This is to enabled the software give a suspend to the PHY, if the suspend from the UTMI core is not stable. 0 Suspend 1 No suspend (default)
25 RESET	UTMI reset. This is to enable the software give a reset to the PHY, if the reset from the UTMI core is not stable. Set this bit will reset all in the UTMI PHY except the clock generation logic 0 No PHY reset (default) 1 PHY reset
24 USBEN	UTMI USB enable. Enables/disables the USB UTMI PHY. Clear this bit will reset all in the UTMI PHY include the clock generation logic. 0 Disable UTMI PHY 1 Enable UTMI PHY (default)
23 EVDO	UTMI external Vbus divider option. Enables off-chip resistor divider for Vbus 0 Disable (default) 1 Enable
22 LSFE	UTMI line state filter enable. Enables filtering of line state to account for skew between D+/D- signals 0 Disable 1 Enable (default)
21	Reserved.

Table 47-5. USB_PHY_CTRL_FUNC Register Field Description (continued)

Field	Description
20 VCTL_LD	UTMI vendor control load This bit enables the signal to load the vendor control register of the on-chip UTMI PHY from the VCTL_DATA and VCTL_ADDR fields 0 Vendor control register load enabled 1 Vendor control register load disabled (default)
19–16 VCTL_ADDR	UTMI vendor control address. This field indicates the vendor-defined control used to address the vendor control registers. In normal operating mode, the VCTL_ADDR field is cleared (all zeros).
15–8 VCTL_DATA	UTMI vendor control data. These bits indicate data to be written into the vendor control registers.The UTMI contains a number of diagnostic control registers. The UTMI implements a protocol whereby the Vendor Control register is loaded to specify a write to one of these registers. The data to be written is provided on these registers. In normal operating mode, the VCTL_DATA field is cleared (all zeros).
7–0 VSTS	UTMI vendor control status. These read-only bits indicate the contents of the vendor control register. The UTMI contains a number of diagnostic status and monitoring registers—their status is reflected in the vendor control register (status source register). The UTMI implements a protocol whereby the contents of the vendor control register are loaded, and can be read from this register field.

47.3.1.4 OTG UTMI PHY Test Control Register (USB_PHY_CTRL_TEST)

This register controls UTMI PHY test signals.

Offset 0x060C (USB_PHY_CTRL_TEST) Access: User read-write

	7	6	5	4	3	2	1	0
R	0	0	0	TM		LB	NFC	FT
W								
Reset	0	0	0	0	0	0	0	0

Figure 47-5. USB_PHY_CTRL_TEST Register

Table 47-6. USB_PHY_CTRL_TEST Register Field Descriptions

Field	Description
7–5	Reserved.
4–3 TM	UTMI test mode selection.The TM[1:0] are for test mode selection. Along with the other test signals mentioned in this section, this is sampled asynchronously and permits the UTMI to be placed into test and functional debug modes. See the UTMI PHY specification for more details.
2 LB	UTMI loopback. Enables the PHY to receive its own packets 0 Disable loopback (default) 1 Enable loopback

Table 47-6. USB_PHY_CTRL_TEST Register Field Descriptions (continued)

Field	Description
1 NFC	UTMI no Frequency check. This causes PHY to disregard the UTMI specification for clock accuracy(± 0.5%). 0 PHY checks frequency for compliance with UTMI specification 1 PHY does not check frequency
0 FT	UTMI function test. This bit enables PHY to use the ipt_scan_clk instead of the internally-generated PLL clocks. 0 Use internally-generated PLL clocks (default) 1 Enable ipt_scan_clk

47.4 Functional Description

This section describes the functionality and the topology of the different building blocks of the USB module.

47.4.1 USB Host Controller

The host controller core can be configured for high-speed, full-speed or low-speed operation.

47.4.2 USB OTG Controller

The OTG controller offers high-speed (HS), full-speed (FS) and low-speed (LS) capabilities in host mode, and HS/FS in device mode.

47.4.2.1 OTG Controller Host Mode

The controller supports direct connection of a FS/LS device (without external hub) with external serial transceiver and HS/FS device with external ULPI transceiver. There is no separate transaction translator block in the system: this function has been implemented within the DMA and protocol engine blocks to support connection to FS/LS devices.

47.4.2.2 OTG Controller Peripheral (Device) Mode

Peripheral mode has the following features:

- Up to 8 bidirectional endpoints
- HS/FS operation
- Supports HNP and SRP OTG protocols
- Remote wake-up capable

47.4.3 USB Power Control Module

The HS-USB module supports suspend and wake-up functionality.

47.4.3.1 Entering Suspend Mode

Suspend mode is always entered under control of driver software by setting the appropriate bit in the PORTSC register. After the controller is suspended, the clocks to the USB block can be stopped.

47.4.3.2 Wake-up Events

The power control module monitors the USB bus when the USB core is in the suspend state. Depending on whether the core is on host or device mode, a number of wake-up conditions are detected. Upon detection of a wake-up condition, an interrupt (asynchronous) is generated on the CPU complex. This interrupt also re-activates the clocks if these were stopped during the suspend.

47.4.3.2.1 Host Mode Events

The host controller wakes up on the following events:

- Remote wake-up request

  A peripheral can request the host to reactivate the bus by driving wake-up signaling on the Dm/Dp lines. The power control module detects a J-K transition on the Dm/Dp lines and signal the wake-up request to the core.
- Wake on over-current

  If wake on over-current is enabled in the PORTSC registers, the power control module signals a wake-up condition to the USB core.
- Wake on disconnect

  The power control module detects disconnect events by monitoring the Dp/Dm lines. When a disconnect event is detected (Dm = Dp = 0) and the wake on disconnect is enabled in the PORTSC register, the core is notified.
- Wake on connect

  Similar to the wake on disconnect, the power control module detects a connect event (Dm or Dp high) and signals this to the USB core by setting the pwrctl_wake-up signal if enabled in the PORTSC register.

47.4.3.2.2 Device Mode Events

When the OTG controller is configured for peripheral operation, the power control module detects the following events:

- Bus activity detection

  Any non-idle condition on the USB bus activates the wake-up output of the power control module to notify the USB core of the wake-up event.
- Device connection detection

  When using on-chip UTMI PHY, the USB module can enter into suspend mode if nothing is attached to the OTG port. After a device is connected to the port, the ID pin changes from 1 to 0, activating the wake-up output of the power control module which notifies the USB core of the wake-up event. This event is generated only if the proper bit is set in USB_CTL register
- Host connection detection

When using on-chip UTMI PHY, the USB module can enter into suspend mode if nothing is attached to the OTG port. After a device is connected to the port, the VBUS pin changes from 0 to 1, activating the wake-up output of the power control module which notifies the USB core of the wake-up event.This event is generated only when proper bit is set in USB_CTL register

47.4.4 Full-Speed Transceiverless Link Logic (FS-TLL) Module

The transceiverless link logic circuit allows two microcontrollers to use USB as an interprocessor communication link (ICL) without using conventional USB transceivers. The TLL multiplexers support serial-type (FS-TLL) interfacing, which is available on the host port. FSL-TLL mode has the following properties:

- FS-TLL can be selected based on the type of USB peripheral. The FS/LS serial interface protocol is supported.
- The TLL can be disabled by setting the HSTD bit in the USB CONTROL register. This disables the FS-TLL module, and the host USB core works in conventional USB mode.
- The TLL meets the timing requirements of both host and device.

47.4.4.1 Serial FS-TLL Functional Description

The Serial FS-TLL is a logical representation of two serial transceivers connected by a USB cable. The USB bus DM/DP states are modeled internally in the FS-TLL function, such that the USB I/O port acts as if it were a transceiver.

In a regular USB implementation with serial PHYs, the speed selection on the USB bus is done by means of a pull-up resistor on the peripheral side either on the DM (low-speed) or the DP (full-speed) line. This pull-up resistor pulls one of the USB lines high when the bus is idle. This option cannot be modeled in logic, as the serial USB interface does not provide for such a signal. The FS-TLL multiplexer is therefore configured for full-speed only operation.

The IDLE condition of the bus is determined by the TxEnb signals and suspend signals on both sides of the multiplexer. When both TxEnb signals are high, or when one or both suspend signals are high, the idle condition is assumed. The FS-TLL module then drives TxDp high and TxDm low on both sides of the block.

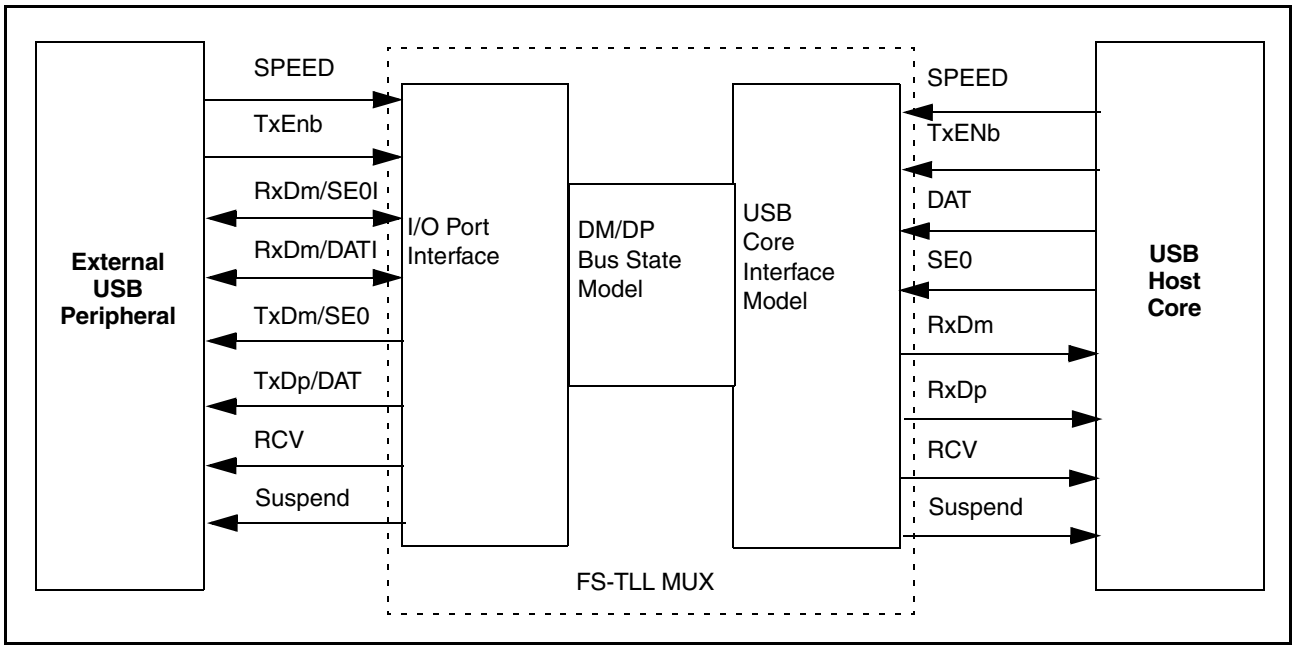


Figure 47-6. FS-TLL Multiplexer Functional Diagram

47.4.5 ULPI/Serial Multiplexer

Host and OTG cores can be configured by software for ULPI or serial PHY operation. The ULPI/serial multiplexer selects between ULPI interface signals and serial PHY interface signals. The multiplexer is controlled by the PHY select signals from the USB core and is switched when the software selects the interface mode.

The default configuration for the multiplexer is serial mode. Switching to ULPI mode is done by writing 0b10 to the parallel transceiver select (PTS) bits in the PORTSC register.

47.4.6 Interrupts

47.4.6.1 USB Core Interrupts

Each USB core uses one dedicated vector in the interrupt table. The vector numbers associated with each of the cores can be found in the “Interrupts and DMA Events” chapter.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB cores. See the USB core documentation for more details.

47.4.6.2 USB Wake-up Interrupts

Each USB core has an associated wake-up interrupt. The wake-up interrupts are generated outside the USB cores, but use the same vector as the corresponding core’s interrupt. These interrupt are generated by the power control modules that run on the 32 kHz standby clock.The wake-up interrupt works even when the

USB and CPU clocks are disabled, so that a wake-up condition on the USB bus can reactivate the CPU clocks.

Because the wake-up interrupt is generated and cleared on a 32 kHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously, because the request is clocked by the CPU clock. The software must then wait for at least three 32 kHz clock cycles before re-enabling this interrupt to allow sufficient time for the request flag to clear. As this interrupt is only used for USB low-power modes, it is sufficient to enable the wake-up interrupt just prior to entering USB suspend mode.

47.5 Initialization/Application Information

This section describes detailed application information for host and OTG ports. Some of the following content is related to host, and some relates to device. Device-related content is specific to OTG, while Host-related content applies to both ports.

47.5.1 Software Model

The device application program interface (API) provides a framework of routines to control the USB-HS OTG High-Speed USB On-The-Go peripheral in USB device applications. It includes an application to respond to the Chapter 9 device framework commands issued by a USB host.

The USB-HS OTG High-Speed USB On-The-Go device API is designed to significantly simplify the software tasks required to develop a USB device application. The API presents a high-level data transfer interface to the user's application code. All the register, interrupt and DMA interactions with the USB-HS OTG High-Speed USB On-The-Go core are managed by the API. The API also includes routines that handle all the USB device framework (Chapter 9) commands which are required for all USB devices.

The Host Stack provides a layered software architecture to control all aspects of a USB bus system. The Host Controller Device (HCD) interface controls the functions of an embedded EHCI host controller. The USB driver layer provides all the USB driver functions to enumerate, manage and schedule a USB bus system, while the upper layers of the stack support standard USB device class interfaces to the device drives running on the embedded system.

For details on the HS-USB OTG High-Speed USB On-The-Go software stack, refer the documentation provided with the software products.

- The ANSI-C OTG software stack provides host and device application support. USB software included with the HS-USB OTG High-Speed USB On-The-Go core is tested with the hardware.
- The OTG API handles OTG protocols. Connect and disconnect events are handled as well as the OTG Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) state machines. The OTG code calls the Host or Device API functions based on the connection state of the OTG state machines.
- The host API speeds up host software development. Simple API calls allow direct interaction with USB pipes. Additional layers support bus enumeration, bus management and a growing set of supported USB classes.

- The device API speeds up peripheral development. USB peripheral characteristics such as endpoints, configurations, interfaces, and alternate settings are controlled by supplied ANSI-C firmware. Chapter 9 Device framework command set reduces software development time. Simple API interface allows quick coding of USB device applications.

47.5.1.1 Device Data Structure

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers.

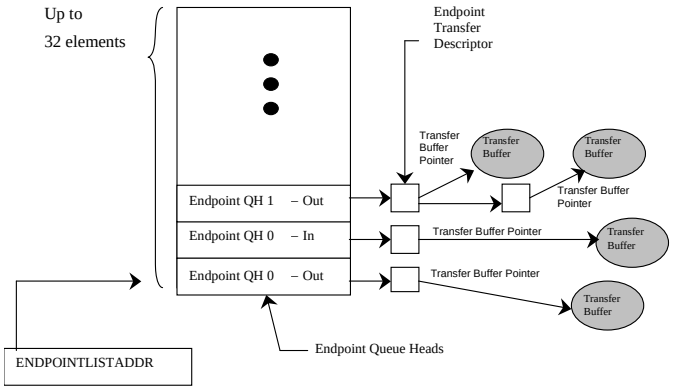


Figure 47-7. Endpoint Queue Head Organization

The HS-USB OTG High-Speed USB On-The-Go device API incorporates and abstracts for the application developer all of the information contained in the device operational model.

47.5.1.2 Host Data Structure

The host data structures are used to communicate control, status, and data between software and the Host Controller. The Periodic Frame List is an array of pointers for the periodic schedule. A sliding window on the Periodic Frame List is used. The Isochronous Transfer List is where all the control and bulk transfers are managed. The HS-USB OTG High-Speed USB On-The-Go Host API incorporates and abstracts for the application developer all of the information contained in the host operational model.

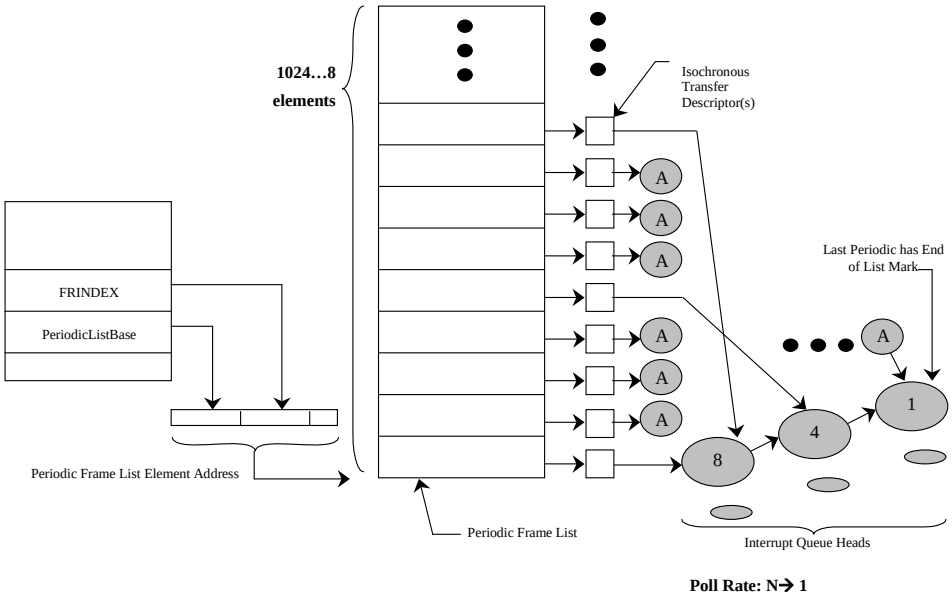


Figure 47-8. Periodic Schedule Organization

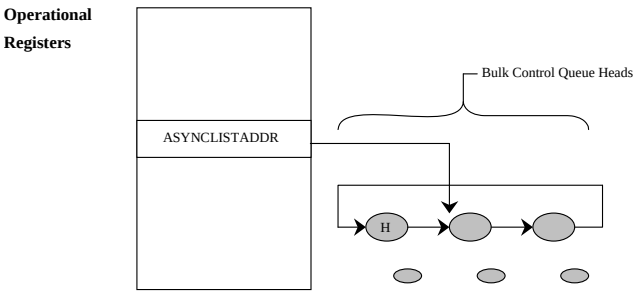


Figure 47-9. Asynchronous Schedule Organization

47.5.2 Register Interface

Slave accesses from the controlling processor enables access to the configuration, control, and status registers. One function of the system address map is the registers base address, which must begin on a dword (32-bit) boundary. Register offset definitions are listed in the table below.

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are comprised of dynamic control or status registers that may be read-only, read/write, or read/write to clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

NOTE

Host mode EHCI compatibility begins at offset 0x100. If it is necessary to begin the EHCI register set at offset 0x000, the identification registers can be disabled from the address map by connecting the uppermost address bit of the slave interface to a logic level 1 and adjusting the offsets below accordingly.

Table 47-7. Interface Register Sets

Offset	Register Set	Explanation
0x000–0x0FC	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
0x100–0x124	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
0x140–0x1FC	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

47.5.2.1 Configuration, Control and Status Registers

Table 47-8 shows configuration, control and status registers, and their usage by different core types.

Table 47-8. Configuration, Control, and Status Registers by Core Type

Base Address Offset	Size (Bytes)	Name Abbreviation	Register Name	Core Type			
				DEV	OTG	SPH	MPH
0x0000	4	ID	Identification Register	Yes	Yes	Yes	Yes
0x0004	4	HWGENERAL	General Hardware Parameters	Yes	Yes	Yes	Yes
0x0008	4	HWHOST	Host Hardware Parameters	—	Yes	Yes	Yes
0x000C	4	HWDEVICE	Device Hardware Parameters	Yes	Yes	—	—
0x0010	4	HWTXBUF	TX Buffer Hardware Parameters	Yes	Yes	Yes	Yes
0x0014	4	HWRXBUF	RX Buffer Hardware Parameters	Yes	Yes	Yes	Yes
0x0018	4	HWTTTXBUF	TT-TX Buffer Hardware Parameters	—	—	—	Yes
0x001C	4	HWTTRXBUF	TT-RX Buffer Hardware Parameters	—	—	—	Yes
0x0020 – 0x00FC	232	Reserved	N/A	—	—	—	—
0x0080	4	GPTIMER0LD	General Purpose Timer #0 Load Register	—	—	—	—

Table 47-8. Configuration, Control, and Status Registers by Core Type (continued)

Base Address Offset	Size (Bytes)	Name Abbreviation	Register Name	Core Type			
				DEV	OTG	SPH	MPH
0x0084	4	GPTIMER0CTRL	General Purpose Timer #0 Control Register	—	—	—	—
0x0088	4	GPTIMER1LD	General Purpose Timer #1 Load Register	—	—	—	—
0x008C	4	GPTIMER1CTRL	General Purpose Timer #1 Control Register	—	—	—	—
0x0090	4	SBUSCFG	Control for the system bus interface	Yes	Yes	Yes	Yes
0x0100	1	CAPLENGTH	Capability Register Length	Yes	Yes	Yes	Yes
0x0101	1	Reserved	N/A	—	—	—	—
0x0102	2	HCIVERSION	Host Interface Version Number	—	Yes	Yes	Yes
0x0104	4	HCSPARAMS	Host Control Structural Parameters	—	Yes	Yes	Yes
0x0108	4	HCCPARAMS	Host Control Capability Parameters	—	Yes	Yes	Yes
0x010C – 0x011F	20	Reserved	N/A	—	—	—	—
0x0120	2	DCIVERSION	Dev. Interface Version Number	Yes	Yes	—	—
0x0122	2	Reserved	N/A	÷			
0x0124	4	DCCPARAMS	Device Control Capability Parameters	Yes	Yes	—	—
0x0128 – 0x013C	24	Reserved	N/A	—	—	—	—
0x0140	4	USBCMD	USB Command	Yes	Yes	Yes	Yes
0x0144	4	USBSTS	USB Status	Yes	Yes	Yes	Yes
0x0148	4	USBINTR	USB Interrupt Enable	Yes	Yes	Yes	Yes
0x014C	4	FRINDEX	USB Frame Index	Yes	Yes	Yes	Yes
0x0150	4	Reserved	4G Segment Selector	—	—	—	—
0x0154	4	PERIODICLISTBASE	Frame List Base Address	—	Yes	Yes	Yes
		Device Addr	USB Device Address	Yes	Yes	—	—
0x0158	4	ASYNCLISTADDR	Next Asynchronous List Address	—	Yes	Yes	Yes
		Endpointlist Addr	Address at Endpoint list in memory	Yes	Yes	—	—
0x015C	4	ASYNCTTSTS	Asynchronous Buffer Status For Embedded TT.	—	—	—	Yes
0x0160	4	BURSTSIZE	Programmable Burst Size	Yes	Yes	Yes	Yes
0x0164	4	TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	—	Yes	Yes	Yes
0x0168	4	TXTTFILLTUNING	Host TT Transmit Pre-Buffer Packet Tuning	—	—	—	Yes
0x016C	4	N/A	Reserved	—	—	—	—

Table 47-8. Configuration, Control, and Status Registers by Core Type (continued)

Base Address Offset	Size (Bytes)	Name Abbreviation	Register Name	Core Type			
				DEV	OTG	SPH	MPH
0x0170 – 0x017C	16	N/A	Reserved	—	—	—	—
0x0180	4	CONFIGFLAG	Configured Flag Register	—	Yes	Yes	Yes
0x0184	4	PORTSC1	Port Status/Control 1	Yes	Yes	Yes	Yes
0x0188 – 0x01A0	28	PORTSC2–8	Port Status/Control 2–8	—	—	—	Yes
0x01A4	4	OTGSC	On-The-Go (OTG) Status and Control	—	Yes	—	—
0x01A8	4	USBMODE	USB Device Mode	Yes	Yes	Yes	Yes
0x01AC	4	ENPDTSETUPSTAT	Endpoint Setup Status	Yes	Yes	—	—
0x01B0	4	ENDPTPRIME	Endpoint Initialization	Yes	Yes	—	—
0x01B4	4	ENDPTFLUSH	Endpoint De-Initialize	Yes	Yes	—	—
0x01B8	4	ENDPTSTATUS	Endpoint Status	Yes	Yes	—	—
0x01BC	4	ENDPTCOMPLETE	Endpoint Complete	Yes	Yes	—	—
0x01C0 – 0x01FC	64	ENDPTCTRL0–15	Endpoint Control 0–15	Yes	Yes	—	—

47.5.2.2 Register Summary

Table 47-9 is the HS-USB register summary. Italicized text in Table 47-9 indicates a deviation from EHCI for the device.

Table 47-9. HS-USB Register Summary

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 ID	reserved								REVISION								NID								ID							
0x004 HWGENERAL	reserved																						SM	PHYM			PHYW		BWT	CLKC		RT
0x008 HWHOST	TTPER								TTASY								reserved												NPORT			HC
0x00C HWDEVICE	reserved																										DEVP					DC
0x010 HWTXBUF	TXLC		reserved						TXCHANADD								TXADD								TXBURST							
0x014 HWRXBUF	reserved																RXADD								RXBURST							
0x020 Reserved	reserved																															

Table 47-9. HS-USB Register Summary (continued)

… Reserved	reserved																															
0x080 GPTIMER0LD	reserved																GPTLD															
0x084 GPTIMER0CTRL	GPTRUN	GPTRSR	reserved												GPTMODE	GPTCNT																
0x088 GPTIMER1LD	reserved																GPTLD															
0x08C GPTIMER1CTRL	GPTRUN	GPTRSR	reserved												GPTMODE	GPTCNT																
0x090 SBUSCFG	reserved																													AHBBRST		
… Reserved	reserved																															
0x0FC Reserved	reserved																															
0x100 CAPLENGTH																														CAPLENGTH		
0x101 Reserved																														reserved		
0x102 HCIVERSION																	HCIVERSION															
0x104 HCSPARAMS	reserved		N_TT		N_PTT		reserved		PI		N_CC		N_PCC		reserved		PPC		N_PORTS													
0x108 HCCPARAMS	reserved										EEC{[7:0]				IST[7:4]		R	R	PFL	ADC												
0x10C Reserved	reserved																															
… Reserved	reserved																															
0x11F Reserved	reserved																															
0x120 DCIVERSION																	*DCIVERSION*															
0x122 Reserved																	*reserved*															

Table 47-9. HS-USB Register Summary (continued)

0x124 DCCPARAMS	*reserved*																												*HC*	*DC*	*R*	*R*	*DEN*
0x128 Reserved	reserved																																
… Reserved	reserved																																
0x13C Reserved	reserved																																
0x140 USBCMD	reserved								ITC												FS2	R	SUTW	ATDTW	ASPE	R	ASP1	ASP0	LR	IAA	ASE	PSE	FS1	FS0	RST	RS
0x144 USBSTS	reserved																				AS	PS	RCL	HCH	reserved				SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
0x148 USBINTR	reserved																												SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
0x14C FRINDEX	reserved																												FRINDEX[13:0]								
0x150 Reserved	reserved																																
0x154 PERIODICLIST BASE	PERBASE[31:12]																												reserved								
Device Addr	*USBADR[31:25]*								*reserved*																												
0x158 ASYNCLISTADDR	ASYBASE[31:5]																																reserved				
Endpointlist Addr	*EPBASE[31:11]*																												*reserved*								
0x15C ASYNCTTSTS	reserved																																		TTAC	TTAS
0x160 BURSTSIZE	reserved																				TXPBURST												RXPBURST			
0x164 TXFILLTUNING	reserved																TXFIFOTHRES					R			TXSCHHEALTH								TXSCHOH			
0x168 TXTTFILLTUNING	reserved																												TXTTSCHHEALTH				R		TXTTSCHOH		
0x16C Reserved	reserved																																				
0x170 ULPI Viewport	ULPI Viewport [optional]																																				
0x174 Reserved	reserved																																				

Table 47-9. HS-USB Register Summary (continued)

… Reserved	reserved																															
0x17C Reserved	reserved																															
0x180 CONFIGFLAG	set to zero																															1
0x184 PORTSC1	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC		PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
0x188 PORTSC2	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC		PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
… PORTSCx	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC		PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
0x1A0 PORTSC8	PTS		STS	PTW	PSPD		R	PFSC	PHCD	WKOC	WKDS	WKCN	PTC		PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
0x1A4 OTGSC	R	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	R	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS	R	DPS	1msT	BSE	BSV	ASV	AVV	ID	HABA HADP			DP	OT	HAAR	VC	VD
0x1A8 USBMODE	*reserved*																										*SDIS*	*SLOM*	*ES*	*CM*		
0x1AC ENPDTSETUPSTAT	*reserved*																*ENDPTSETUPSTAT*															
0x1B0 ENDPTPRIME	*PETB[15:0]*																*PERB[15:0]*															
0x1B4 ENDPTFLUSH	*FETB[15:0]*																*FERB[15:0]*															
0x1B8 ENDPTSTATUS	*ETBR[15:0]*																*ERBR[15:0]*															
0x1BC ENDPTCOMPLETE	*ETCE[15:0]*																*ERCE[15:0]*															
0x1C0 ENDPTCTRL0	*reserved*							*TXE*	*reserved*				*TXT*		*R*	*TXS*	*reserved*							*RXE*	*reserved*			*RXT*		*R*	*RXS*
0x1C4 ENDPTCTRL1	*reserved*							*TXE*	*TXR*	*TXI*	*R*		*TXT*		*TXD*	*TXS*	*reserved*							*RXE*	*RXR*	*RXI*	*R*	*RXT*		*RXD*	*RXS*

Table 47-9. HS-USB Register Summary (continued)

... ENDPTCTRLn	reserved	TXE	TXR	TXI	R	TXT	TXD	TXS	reserved	RXE	RXR	RXI	R	RXT	RXD	RXS
0x1FC ENDPTCTRL15	reserved	TXE	TXR	TXI	R	TXT	TXD	TXS	reserved	RXE	RXR	RXI	R	RXT	RXD	RXS

... Reserved	reserved

47.5.2.3 Identification Registers

Identification registers provide information about the slave interface, and include a table of the hardware configuration parameters.

47.5.2.3.1 Identification Register (ID)

The Identification register (ID) provides a simple way to determine if the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core is provided in the system. The ID register identifies the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core and its revision. Figure 47-10 shows the register, and Table 47-10 provides field descriptions.

Offset 0x0000 (ID) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	REVISION							
W																
Reset	0	0	0	0	0	0	0	0	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1	1	NID						0	0	ID					
W																
Reset	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1

Figure 47-10. Identification Register (ID) Fields

Table 47-10. Identification Register Field Descriptions

Field	Description
31—24	Reserved.
23—16 REVISION[7:0]	Revision number of the core.
15—14	Reserved
13—8 NID[5:0]	Ones complement version of ID[5:0].
7—6	Reserved
5—0 ID	Configuration number. This field is set to 0x05, and indicates that the peripheral is the USB-HS OTG High-Speed USB On-The-Go USB 2.0 core.

47.5.2.3.2 General Hardware Parameters Register (HWGENERAL)

This register contains general hardware parameters as defined in System Level Issues and Core Configuration]. Figure 47-11 shows the register, and Table 47-11 provides field descriptions.

Offset 0x0004 (HWGENERAL) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	SM	PHYM			PHYW		BWT	CLKC		RT
W																
Reset	0	0	0	0	0	0	—	—	—	—	—	—	—	—	—	—

Figure 47-11. General Hardware Parameters Register (HWGENERAL)

Table 47-11. General Hardware Parameters Register Field Descriptions

Field	Description
31—10	Reserved.
9 SM	VUSB_HS_PHY_SERIAL
8—6 PHYM	VUSB_HS_PHY_TYPE
5—4 PHYW	VUSB_HS_PHY16_8
3 BWT	Reserved for internal testing.
2—1 CLKC	VUSB_HS_CLOCK_CONFIGURATION
0 RT	VUSB_HS_RESET_TYPE

47.5.2.3.3 Host Hardware Parameters Register (HWHOST)

This register contains host hardware parameters, as defined in System Level Issues and Core Configuration. Figure 47-12 shows the register, and Table 47-12 provides field descriptions.

Offset 0x0008 (HWHOST) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TTPER								TTASY							
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	NPORT			HC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—

Figure 47-12. Host Hardware Parameters Register (HWHOST)

Table 47-12. Host Hardware Parameters Register Field Descriptions

Field	Description
31—24 TTPER	VUSB_HS_TT_PERIODIC_CONTEXTS
23—16 TTASY	VUSB_HS_TT_ASYNC_CONTEXTS
15—4	Reserved
3—1 NPORT	VUSB_HS_NUM_PORT – 1
0 HC	VUSB_HS_HOST

47.5.2.3.4 Device Hardware Parameters Register (HWDEVICE)

This register specifies device hardware parameters as defined in System Level Issues and Core Configuration. Figure 47-13 shows the register, and Table 47-13 provides field descriptions.

Offset 0x000C (HWDEVICE) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	DEVEP					DC
W																
Reset	0	0	0	0	0	0	0	0	0	0	*n*	*n*	*n*	*n*	*n*	*n*

Figure 47-13. Device Hardware Parameters Register (HWDEVICE)

Table 47-13. Device Hardware Parameters Register Field Descriptions

Field	Description
31—6	Reserved.
5—1 DEVEP	VUSB_HS_DEV_EP
0 DC	Device capable; [VUSB_HS_DEV ≠ 0]

47.5.2.3.5 TX Buffer Hardware Parameters Register (HWTXBUF)

This register defines TX buffer hardware parameters as defined in System Level Issues and Core Configuration. Figure 47-14 shows the register, and Table 47-14 provides field descriptions.

Offset 0x0010 (HWTXBUF) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXLCR	0	0	0	0	0	0	0	TXCHANADD							
W																
Reset	*n*	0	0	0	0	0	0	0	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXADD								TCBURST							
W																
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

Figure 47-14. TX Buffer Hardware Parameters Register (HWTXBUF)

Table 47-14. TX Buffer Hardware Parameters Register Field Descriptions

Field	Description
31 TXLCR	VUSB_HS_TX_LOCAL_CONTEXT_REGISTERS
30–24	Reserved.
23–16 TXCHANADD	VUSB_HS_TX_CHAN_ADD
15–8 TXADD	VUSB_HS_TX_ADD
7–0 TCBURST	VUSB_HS_TX_BURST

47.5.2.3.6 RX Buffer Hardware Register (HWRXBUF)

RX buffer hardware parameters as defined in System Level Issues and Core Configuration. Figure 47-15 shows the register, and Table 47-15 provides field descriptions.

Offset 0x0014 (HWRXBUF) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXADD								RXBURST							
W																
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

Figure 47-15. RX Buffer Hardware Parameters Register (HWRXBUF) Fields

Table 47-15. RX Buffer Hardware Parameters Register Field Descriptions

Field	Description
31–16	Reserved.
15–8 RXADD	VUSB_HS_RX_ADD
7–0 RXBURST	VUSB_HS_RX_BURST

47.5.2.3.7 System Bus Interface Register (SBUSCFG)

This non-EHCI register controls the AMBA AHB burst length using the field AHBBRST.Figure 47-16 shows the register, and Table 47-16 provides field descriptions.

In all cases where unspecified burst lengths are allowed, single accesses can occur (usually when the transaction is not 32-bit word aligned).

When an INCR*x* burst size is selected and the transfer is not a multiple of the INCR*x* burst, the burst is decomposed in the different ways. With AHBBRST[2] = 1, the smaller bursts will be unspecified length. with AHBBRST[2] = 0, the smaller bursts will be smaller INCR*x* or singles.

For example, in a 22-word transfer the master sequences for different AHBBRST settings are:

- 101: INCR4+INCR4+INCR4+INCR4+INCR4+INCR(unspecified length)
- 110: INCR8+INCR8+INCR4+INCR(unspecified length)
- 111: INCR16+INCR4+INCR(unspecified length)
- 001: INCR4+INCR4+INCR4+INCR4+INCR4+SINGLE+SINGLE
- 010: INCR8+INCR8+INCR4+SINGLE+SINGLE
- 011: INCR16+INCR4+SINGLE+SINGLE

Offset 0x0090 (SBUSCFG) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	AHBBRST		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*n*	*n*	*n*

Figure 47-16. System Bus Interface Register (SBUSCFG)

Table 47-16. System Bus Interface Register Field Descriptions

Field	Description
31–3	Reserved.
2–0 AHBBRST	AMBA AHB BURST. This field configures the m_hburst signal of the AMBA master interface. The AHBBRST field is only used if the AMBA-AHB system interface has been selected. It has no effect for cores featuring BVCI interface, in which case reads return zeros. When this field is different from zero, the burst size is set internally to the value of the INCR*x* AMBA burst, and the value of the fields TXBURST and RXBURST in register BURSTSIZE are ignored (but the BURSTSIZE register can still be written/read while the AHBBRST field is nonzero). The reset value of AHBBURST can be configured in the file vusb_hs_cfg. 000 INCR burst of unspecified length 001 INCR4, non-multiple transfers of INCR4 are decomposed into singles 010 INCR8, non-multiple transfers of INCR8, are decomposed into INCR4 or singles 011 INCR16, non-multiple transfers of INCR16, are decomposed into INCR8, INCR4 or singles 100 Reserved, not used 101 INCR4, non-multiple transfers of INCR4 are decomposed into smaller unspecified length bursts 110 INCR8, non-multiple transfers of INCR8 are decomposed into smaller unspecified length bursts 111 INCR16, non-multiple transfers of INCR16 are decomposed into smaller unspecified length bursts

47.5.2.4 Device/Host Capability Registers

Device/Host Capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation.

47.5.2.4.1 Capability Registers Length Register (CAPLENGTH)

This EHCI-compatible register is used to indicate which offset to add to the register base address at the beginning of the operational registers. Figure 47-17 shows the register.

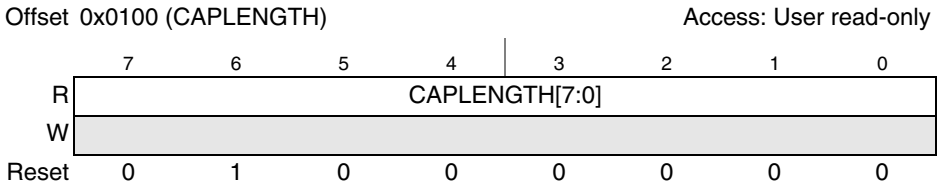


Figure 47-17. Capability Registers Length Register (CAPLENGTH)

47.5.2.4.2 Host Interface Version Number Register (HCIVERSION)

This register is EHCI-compatible. Figure 47-18 shows the register.

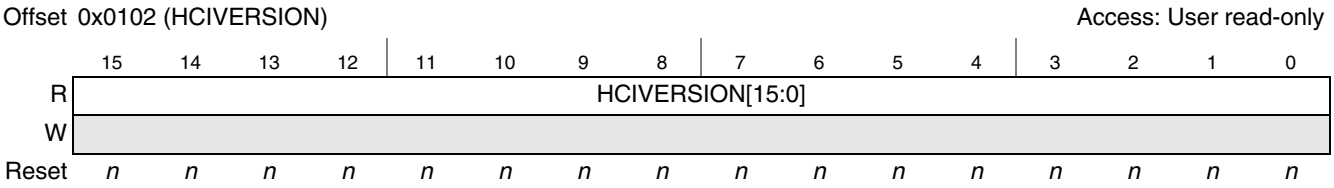


Figure 47-18. Host Interface Version Number Register Fields (HCIVERSION)

47.5.2.4.3 Host Control Structural Parameters Register (HCSPARAMS)

This register is EHCI compatible, with some extensions. Port steering logic capabilities are described in this register. Figure 47-19 shows the register, and Table 47-17 provides field descriptions.

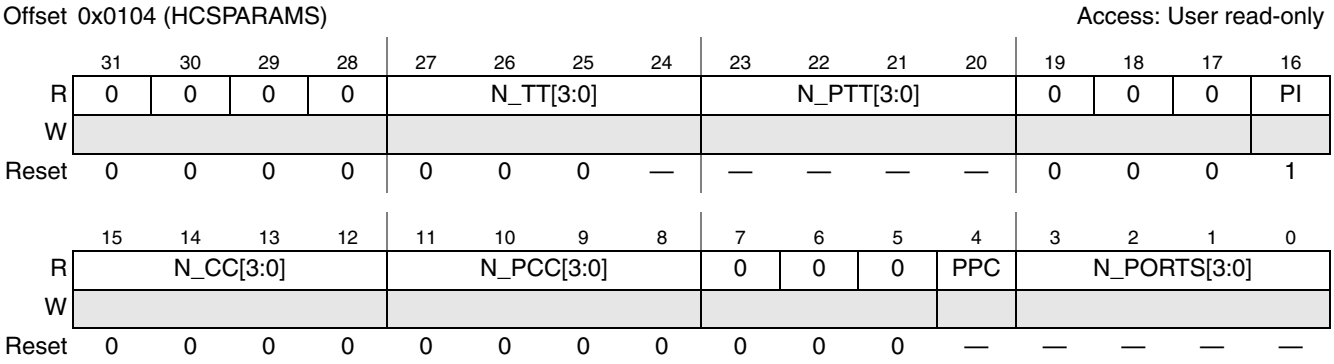


Figure 47-19. Host Control Structural Parameters Register (HCSPARAMS)

Table 47-17. Host Control Structural Parameters Register Field Descriptions

Field	Description
31–28	Reserved.
27–24 N_TT[3:0]	Number of Transaction Translators (N_TT). This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. For multi-port host this field is equal 0001. For all other implementations, N_TT = 0000. This in a non-EHCI field to support embedded TT.

Table 47-17. Host Control Structural Parameters Register Field Descriptions

Field	Description
23–20 N_PTT[3:0]	Number of Ports per Transaction Translator (N_PTT). This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. For multi-port hosts this field always equal N_PORTS. For all other implementations, N_PTT = “0000”. This in a non-EHCI field to support embedded TT.
19–17	Reserved.
16 PI	Port Indicator (P INDICATOR). This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writable field for controlling the state of the port indicator. This field is always 1.
15–12 N_CC[3:0]	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Port-ownership handoffs are supported. High, Full- and Low-speed devices are supported on the host controller root ports. In this implementation this field is always 0.
11–8 N_PCC[3:0]	Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. In this implementation this field is always 0.
7–5	Reserved.
4 PPC	Port Power Control. This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each PORTSC*x* register. This field is always 0 for a device-only implementation.
3–0 N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 0x1 to 0xF. A zero in this field is undefined. The number of ports for a host implementation is parametrizable from 1 to 8. This field is always 1 for device-only implementation.

47.5.2.4.4 Host Control Parameters Register (HCCPARAMS)

This EHCI-compatible register identifies multiple mode control (time-base bit functionality) addressing capability. Figure 47-20 shows the register, and Table 47-18 provides field descriptions.

Offset 0x0108 (HCCPARAMS) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EECP[7:0]								IST[7:4]				0	ASP	PFL	ADC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 47-20. Host Control Capability Parameters Register (HCCPARAMS)

Table 47-18. Host Control Capability Parameters Register Field Descriptions

Field	Description
31–16	Reserved.
15–8 EECP[7:0]	EHCI extended capabilities pointer. For this implementation this field is always 0x00.
7–4 IST[7:4]	Isochronous scheduling threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is 0, the value of the least significant 3 bits indicates the number of microframes a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is 1, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field is always 0x0 for this implementation.
3	Reserved.
2 ASP	Asynchronous schedule park capability. This bit is always set to 1 in this implementation, indicating that the host controller supports the park feature for high-speed queue heads in the asynchronous schedule. The park feature can be disabled or enabled and set to a specific level by using the asynchronous schedule park mode enable (ASPE) and asynchronous schedule park mode count (ASP) fields in the USBCMD register.
1 PFL	Programmable frame list flag. This bit is always set to 1 in this implementation, indicating that the system software can program the frame list size using the frame list size field (USBCMD register). The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous.
0 ADC	64-bit addressing capability. This bit is always 0 in this implementation (no 64-bit addressing capability is supported).

47.5.2.4.5 Device Interface Version Number Register (DCIVERSION)

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register. This register is not EHCI-compatible. Figure 47-21 shows the register.

Offset 0x0120 (DCIVERSION) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DCIVERSION[15:0]															
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Figure 47-21. Device Interface Version Number Register (DCIVERSION)

47.5.2.4.6 Device Control Capability Parameters (DCCPARAMS)

The (non–EHCI-compatible) DCCPARAMS register fields describe the overall host/device capability of the controller. Figure 47-22 shows the register, and Table 47-19 provides field descriptions.

Offset 0x0124 (DCCPARAMS) Access: User read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HC	DC	0	0	DEN[4:0]				
W																
Reset	0	0	0	0	0	0	0	—	—	0	0	—	—	—	—	—

Figure 47-22. Device Control Capability Parameters Register (DCCPARAMS)

Table 47-19. Device Control Capability Parameters Register Field Descriptions

Field	Description
31–9	Reserved.
8 HC	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5	Reserved
4–0 DEN[4:0]	Device Endpoint Number. This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field is 0. Valid values are 0–16.

47.5.2.5 Device/Host Timer Registers

The host/device controller drivers can measure time related activities using these timer registers. These registers are not part of the standard EHCI controller.

47.5.2.5.1 General Purpose Timer *n* Load Register (GPTIMER*n*LD, *n* = 0,1)

These registers contains timer duration or load values. It is not EHCI-compatible. See Section 47.5.2.5.2, "General Purpose Timer n Control Registers (GPTIMERnCTRL, n = 0, 1)" for a description of the timer functions.

Figure 47-23 shows the register, and Table 47-20 provides field descriptions.

Offset 0x0080 (GPTIMER0LD)
0x0088 (GPTIMER1LD)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	GPTLD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPTLD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-23. General Purpose Timer *n* Load Registers (GPTIMER*n*LD)

Table 47-20. General Purpose Timer *n* Load Registers Field Descriptions

Field	Description
31–24	Reserved.
23–0 GPTLD	General Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus 1 for the timer duration.

47.5.2.5.2 General Purpose Timer *n* Control Registers (GPTIMER*n*CTRL, *n* = 0, 1)

These non–EHCI-compatible registers contains the controls for the timers, and data fields that can be queried to determine the running count value. This timer has granularity of 1 µs, and can be programmed to a little over 16 seconds. There are two modes supported by the timers: one-shot and looped count. When a timer counter value transitions to zero, an interrupt can be generated though the use of the timer interrupts in the USBSTS and USBINTR registers.

Figure 47-24 shows the register, and Table 47-21 provcompatibleides field descriptions.

Offset 0x0084 (GPTIMER0CTRL)
0x008C (GPTIMER1CTRL)

Access: Mixed

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GPTRUN	GPTRST	0	0	0	0	0	GPTMOD	GPTCNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPTCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-24. General Purpose Timer *n* Controller Registers (GPTIMER*n*CTRL)

Table 47-21. General Purpose Timer *n* Controller Registers Field Descriptions

Field	Description
31 GPTRUN	General purpose timer run. This bit enables the GPT to run. 0 Timer Stop 1 Timer Run.
30 GPTRST	General purpose timer reset. Setting this bit to 1 reloads the GPTCNT field with the GPTLD value in the corresponding GPTIMER*n*LD register. 0 No action 1 Load counter value from GPTLD field (GPTIMER*n*LD register)
29–25	Reserved.
24 GPMOD	General purpose timer mode. In one-shot mode the timer counts to zero, generates an interrupt and stop until the timer is reset by software. In repeat mode the timer counts to zero, generates an interrupt and automatically reloads the counter to begin again. 0 One-shot mode 1 Repeat mode
23–0 GPTCNT	General purpose timer counter. This field is the value of running timer.

47.5.2.6 Device/Host Operational Registers

Operational registers are comprised of dynamic control or status registers that may be read-only, read/write, or read/write-to-clear. The following sections define the use of these registers.

47.5.2.6.1 USB Command Register (USBCMD)

The serial bus host/device controller executes the command indicated in this register. Figure 47-25 shows the register, and Table 47-22 provides field descriptions.

Offset 0x0140 (USBCMD) Access: Field-dependent

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	ITC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS[2]	ATDTW	SUTW	0	ASPE	0	ASP		LR	IAA	ASE	PSE	FS		RST	RS
W																
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0

Figure 47-25. USB Command Register (USBCMD)

Table 47-22. USB Command Register Field Descriptions

Field	Description
31–24	Reserved.
23–16 ITC	Interrupt Threshold Control. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in microframes. 0x00 Immediate (no threshold) 0x01 1 microframe 0x02 2 microframes 0x04 4 microframes 0x08 8 microframes (default) 0x10 16 microframes 0x20 32 microframes 0x40 64 microframes
15, 3–2 FS	Frame list size. This field is Read/Write only if the programmable frame list (PFL) flag in the HCCPARAMS registers is set to 1: otherwise it is read-only. This field specifies the size of the frame list that controls which bits in the Frame Index Register are used for the Frame List Current index. 000 1024 elements (4096 bytes—Default value) 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes) Only the host controller uses this field.
12	Reserved.

Table 47-22. USB Command Register Field Descriptions

Field	Description
13 SUTW	Set up trip wire (device mode only). This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a queue head (QH) by the DCD without being corrupted. If the setup lockout mode is off (See Section 47.5.2.6.1, "USB Command Register (USBCMD)") then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software, and will be cleared by hardware when a hazard exists. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.
14 ATDTW	Add dTD Trip Wire (device mode only). This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software, and is also cleared by hardware when the state machine is in a state for which adding a dTD to a primed endpoint may go unrecognized. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.
11 ASPE	Asynchronous Schedule Park Mode Enable (OPTIONAL) — Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is set to 1, then this bit defaults to a 1 and is R/W. Otherwise the bit is cleared and read-only. Software uses this bit to enable or disable Park mode. When this bit is 1, park mode is enabled. When this bit is cleared, park mode is disabled. This bit defaults to 1 in this implementation.
10	Reserved.
9–8 ASP	Asynchronous Schedule Park Mode Count (OPTIONAL) If the Asynchronous Park Capability bit in the HCCPARAMS register is set to 1, then this field defaults to 11 and is R/W: otherwise it defaults to 0 and is read-only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the asynchronous schedule. Valid values are 01, 10, and 11. Software must not write a 0 to these bits when the ASPE bit in this register is set to 1, as this will result in undefined behavior. This field defaults to 11 in this implementation.
7 LR	Light Host/Device Controller Reset (OPTIONAL). This read-only.bit is always 0.
6 IAA	Interrupt on Async Advance Doorbell. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a 1 to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 Do not process the Asynchronous Schedule (default) 1 Use the ASYNCLISTADDR register to access theAsynchronous Schedule. Only the host controller uses this bit.

Table 47-22. USB Command Register Field Descriptions

Field	Description
4 PSE	Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 Do not process the Periodic Schedule (default) 1 Use the PERIODICLISTBASE register to access the PeriodicSchedule. Only the host controller uses this bit.
1 RST	Controller Reset. Software uses this bit to reset the controller. This bit is cleared by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a 0 to this register. • Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HC Halted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. • Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) • Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to 1. When this bit is cleared, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a 1 to this field unless the host controller is in the Halted state (the HC Halted bit in the USBSTS register is set to 1). • Device Controller: Writing a 1 to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this bit will cause a detach event.

47.5.2.6.2 USB Status Register (USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits

in this register by writing a 1 to them, as described in Table 47-23. Figure 47-26 shows the register, and Table 47-23 provides field descriptions.

Offset 0x0144 (USBSTS) Access: Field-dependent

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	GPTINT		0	0	0	0	0	0	0	0
W							w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	0	ULPII	0	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W						w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-26. USB Status Register Fields (USBSTS)

Table 47-23. USB Status Register Field Descriptions

Field	Description
31–26	Reserved.
25–24 GPTINT[1:0}	General Purpose Timer Interrupt *n*. TheGPTINT[*n*] bit is set to 1 when the counter in the GPTIMER*n*CTRL register transitions to zero. Writing a 1 to GPTINT[*n*] clears it.
23—16	Reserved.
15 AS	Asynchronous Schedule Status. This read-only bit reports the current real status of the Asynchronous Schedule. When cleared the asynchronous schedule status is disabled and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). This bit is only used by the host controller.
14 PS	Periodic Schedule Status. This read-only bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). This bit is only used by the host controller.
13 RCL	Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. This bit is only used by the host controller.
12 HCH	Host Controller Halted. This read-only bit is 0 whenever the Run/Stop bit is set to 1. The Host Controller sets this bit to 1 after it has stopped executing because of the Run/Stop bit being cleared either by software or by the Host Controller hardware (due to an internal error). This bit is only used by the host controller.
11	Reserved.

Table 47-23. USB Status Register Field Descriptions (continued)

Field	Description
10 ULPII	ULPI Interrupt. When the ULPI Viewport is present in the design, an event completion will set this interrupt. Used by both host & device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1
9	Reserved.
8 SLI	DC Suspend. When a device controller enters a suspend state from an active state, this bit is set to 1. The device controller clears the bit upon exiting from a suspend state. This bit is write 1 to clear, and is only used by the device controller.
7 SRI	SOF Received. When the device controller detects a Start Of (micro) Frame, this bit is set to 1. When a SOF is extremely late, the device controller automatically sets this bit to 1 to indicate that an SOF was expected. Consequently this bit is set roughly every 1 ms in device FS mode and every 125 μs in HS mode, and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit is set at intervals of 1 ms during the prelude to connect and chirp. In host mode, this bit is set every 125 μs, and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.
6 URI	USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to 1. Software can write a 1 to this bit to clear it. This bit is only used by the device controller.
5 AAI	Interrupt on Async Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Software can clear this bit by writing 1 to it. This bit is only used by the host controller.
4 SEI	System Error. This bit is set to 1 when an error response is seen to a read on the system interface. Software can clear this bit by writing 1 to it.
3 FRI	Frame List Rollover. The host controller sets this bit to1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the frame list size (FS) field of the USBCMD register) is 1024, the frame index register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Software can clear this bit by writing 1 to it. This bit is only used by the host controller.
2 PCI	Port Change Detect. • The Host Controller sets this bit to 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. • The Device Controller sets this bit to 1 when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DC Suspend bits, respectively. This bit is not EHCI-compatible.

Table 47-23. USB Status Register Field Descriptions (continued)

Field	Description
1 UEI	USB Error Interrupt (USBERRINT) — R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1, “Reference Host Operation Model: Transfer/Transaction Based Interrupt” in the EHCI Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com) for a complete list of host error interrupt conditions.See section Device Error Matrix in the USB-HS OTG High-Speed USB On-The-Go DEV reference manual. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) — R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

47.5.2.6.3 USB Interrupt Enable Register (USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software. Figure 47-27 shows the register, and Table 47-24 provides field descriptions.

Offset 0x0148 (USBINTR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	TIE[1:0]		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	ULPIE	0	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-27. USB Interrupt Enable Register Fields (USBINTR)

Table 47-24. USB Interrupt Enable Register Field Descriptions

Field	Description
31–26	Reserved.
25–24 TIE[1:0]	When the TIE[*n*] bit is set to 1and the GPTINT[*n*] bit in the USBSTS register is set to 1, the controller issues an interrupt. The interrupt is acknowledged by software clearing the GPTINT[*n*] bit.
23–11	Reserved
10 ULPIE	When this bit is set to 1, and the ULPI Interrupt bit in the USBSTS register transitions, the controller will issue and interrupt. The interrupt is acknowledged by software writing a one to the ULPI Interrupt bit. Used by both host & device controller. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1.
9	Reserved.

Table 47-24. USB Interrupt Enable Register Field Descriptions

Field	Description
8 SLE	When this bit is set to 1, and the DC Suspend bit in the USBSTS register transitions, the device controller issues an interrupt. The interrupt is acknowledged by software writing a one to the DC Suspend bit. Only used by the device controller.
7 SRE	When this bit is set to 1, and the SOF Received bit in the USBSTS register is set to 1, the device controller issues an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.
6 URE	When this bit is set to 1, and the USB Reset Received bit in the USBSTS register is set to 1, the device controller issues an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller.
5 AAE	When this bit is set to 1, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller.
4 SEE	When this bit is a one, and the System Error bit in the USBSTS register is a one, the host/device controller issues an interrupt. The interrupt is acknowledged by software clearing the *System Error* bit.
3 FRE	When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller issues an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller.
2 PCE	When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host/device controller issues an interrupt. The interrupt is acknowledged by software clearing the *Port Change Detect* bit.
1 UEE	When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.
0 UE	When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host/device controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the *USBINT* bit.

47.5.2.6.4 USB Frame Index Register (FRINDEX)

In host mode, this register is used by the host controller to index the periodic frame list. The register updates every 125 µsec (once each microframe). Bits [*N*: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution, where *N* depends on the size of the frame list as set by system software in the frame list size field in the USBCMD register. This register must be written as a dword. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the Halted state as indicated by the HC halted (HCH) bit (USBSTS register). A write to this register while the Run/Stop (RS) bit (USBCMD register) is set to 1 produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read-only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] is checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] is set to the SOF value and FRINDEX [2:0] is set to zero (that is, SOF for 1 ms

frames). If FRINDEX [13:3] is equal to the SOF value, then FRINDEX [2:0] is incremented (that is, SOF for 125 µs microframes.)

Figure 47-28 shows the register, and Table 47-25 provides field descriptions.

Offset 0x014C (FRINDEX)

Access: User read/write (host mode)
User read-only (device mode)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FRINDEX													
W																
Reset	0	0	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*	*u*

Figure 47-28. USB Frame Index Register Fields (FRINDEX)

Table 47-25. USB Frame Index Register Field Descriptions

Field	Description
31–14	Reserved.
13–0 FRINDEX	Frame Index. • In host mode, the value in this register increments at the end of each time frame (microframe). Bits [*N*: 3] are used for the frame list current index, so that each location of the frame list is accessed 8 times (frames or microframes) before moving to the next index. The following table illustrates values of *N* based on the value of the Frame List Size (FS) field in the USBCMD register, when used in host mode.

Frame List Size Field (USBCMD Register)	Number of Elements	*N*
0b000	1024	12
0b001	512	11
0b010	256	10
0b011	128	9
0b100	64	8
0b101	32	7
0b110	16	6
0b111	8	5

• In device mode the value is the current frame number of the last frame transmitted. It is not used as an index.
In either mode, bits 2:0 indicate the current microframe.
This field is a free-running counter: the reset value is undefined.

47.5.2.6.5 CTRLDSSEGMENT Register

This register is located at base address offset 0x0150, and is not used in this implementation.

47.5.2.6.6 Host Controller Frame List Base Address Register (PERIODICLISTBASE) and USB Device Address Register (DEVICEADDR)

Base address offset 0x0154 is occupied by the PERIODICLISTBASE register for host mode, and the DEVICEADDR register for device mode.

Host Controller Frame List Base Address Register (PERIODICLISTBASE)

This 32-bit register contains the beginning address of the periodic frame list in the system memory. HCD loads this register prior to starting the schedule execution by the host controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the frame index register (FRINDEX) to enable the host controller to step through the periodic frame list in sequence. Figure 47-29 shows the register, and Table 47-26 provides field descriptions.

Offset 0x0154 (PERIODICLISTBASE) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BASEADR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASEADR				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-29. Host Controller Frame List Base Address Register (PERIODICLISTBASE)

Table 47-26. Host Controller Frame List Base Address Register Field Descriptions

Field	Description
31–12 BASEADR]	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. This field is only used by the host controller.
11–0	Reserved.

Device Controller USB Device Address Register (DEVICEADDR)

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor. Figure 47-30 shows the register, and Table 47-27 provides field descriptions.

Offset 0x0154 (DEVICEADDR) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBADR[31:25]							0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-30. Device Controller USB Device Address (DEVICEADDR)

Table 47-27. Device Controller USB Device Address Field Descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24–0	Reserved.

47.5.2.6.7 Host Controller Next Asynchronous Address Register (ASYNCLISTADDR) and Device Controller Endpoint List Address Register (ENDPOINTLISTADDR)

Base address offset 0x0158 is occupied by the ASYNCLISTADDR register for host mode, and the ENDPOINTLISTADDR register for device mode.

Host Controller Next Asynchronous Address Register (ASYNCLISTADDR)

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read. Figure 47-31 shows the register, and Table 47-28 provides field descriptions.

Offset 0x0158 (ASYNCLISTADDR) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ASYBASE[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ASYBASE[15:5]											0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-31. Host Controller Next Asynchronous Address (ASYNCLISTADDR)

Table 47-28. Host Controller Next Asynchronous Address Field Descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). This field is only used by the host controller.
4–0	Reserved.

Device Controller Endpoint List Address Register (ENDPOINTLISTADDR)

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software, and always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed to be 64-byte.

shows the register, and Table 47-29 provides field descriptions.

Offset 0x0158 (ENDPOINTLISTADDR) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EPBASE{31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EPBASE[15:11]					0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-32. Device Controller Endpoint List Address (ENDPOINTLISTADDR)

Table 47-29. Device Controller Endpoint List Address Field Descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH) (that is, one queue head per endpoint and direction.)
10–0	Reserved.

47.5.2.6.8 Host Controller Embedded TT Asynchronous Buffer Status Register (BURSTSIZE)

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface. Figure 47-33 shows the register, and Table 47-30 provides field descriptions.

Offset 0x0160 (BURSTSIZE) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TXPBURST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPBURST								RXPBURST							
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 47-33. Host Controller Embedded TT Asynchronous Buffer Status (BURSTSIZE)

Table 47-30. Host Controller Embedded TT Asynchronous Buffer Status Field Descriptions

Field	Description
31–17	Reserved.
16–8 TXPBURST	Programmable TX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus. Default is the constant VUSB_HS_TX_BURST
7–0 RXPBURST	Programmable RX Burst Length. (Read/Write) This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory. Default is the constant VUSB_HS_RX_BURST.

47.5.2.6.9 TXFILLTUNING Register

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Offset 0x0164 (TXFILLTUNING) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	TXFIFOTHRES					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	TXSCHHEALTH					TXSCHOH							
W				wtc	wtc	wtc	wtc	wtc								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-34. TXFILLTUNING Register

Timing parameter definitions are as follows:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total packet flight time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total packet time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “backoff” event. When a backoff event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many backoff events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Backoffs can be minimized with use of the TSCHHEALTH (T_{ff}) described below.

Table 47-31. TXFILLTUNING Field Descriptions

Field	Description
31–22	Reserved.
21–16 TXFIFOTHRES	FIFO Burst Threshold. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the stream disable bit in USBMODE register is set.
15–13	Reserved.

Table 47-31. TXFILLTUNING Field Descriptions

Field	Description
12–8 TXSCHHEALTH	Scheduler Health Counter. (Write To Clear) This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7–0 TXSCHOH	Scheduler overhead. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.26 μs when a device is connected in High-Speed Mode for OTG and SPH. The time unit represented in this register is 6.33 μs when a device is connected in Low/Full Speed Mode for OTG & SPH. The time unit represented in this register is always 1.267 the MPH product.

47.5.2.6.10 ULPI VIEWPORT (Optional)

The register provides indirect access to the ULPI PHY register set. Although the core performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access. Figure 47-35 shows the register, and Table 47-32 provides field descriptions.

Offset 0x0170 (ULPIVIEWPORT) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ULPIWU	ULPIRUN	ULPIRW	0	ULPISS	ULPIPORT			ULPIADDR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ULPIDATRD								ULPIDATWR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-35. ULPI VIEWPORT

NOTE

- Writes to the ULPI through the VIEWPORT register can substantially disrupt standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI.
- Executing read operations though the ULPI Viewport should have no harmful side effects to standard USB operations.
- The ULPI Viewport is only synthesized in the design if the ULPI option has been purchased & installed and the constant `VUSB_HS_PHY_ULPI` is set to `1`. If the ULPI interface is not enabled, this register will always read as 0.

The ULPI Viewport, can perform two types of operations: wake-up and read /write. The wake-up operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wake-up operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync. state bit (ULPISS). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS indicates a ‘0’ then read/write operations will not be able execute. Undefined behavior will result if ULPISS = 0 and a read or write operation is performed. To execute a wake-up operation, write all 32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately, the ULPIWU bit is 1 and the ULPIRUN bit is a 0. Poll the ULPI Viewport until ULPIWU is zero for the operation to complete.

To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPIRUN bit is a 1. Poll the ULPI Viewport until ULPIRUN is 0 for the operation to complete. After ULPIRUN is 0, the ULPIDATRD is valid if the operation was a read.

An alternative to using the polling method above is to use the ULPI interrupt defined in the USB Status Register (USBSTS) and USB Interrupt Enable Register (USBINTR). When a wake-up or read/write operation is completed, the ULPI interrupt is set.

Table 47-32. ULPI VIEWPORT Field Descriptions

Field	Description
31 ULPIWU	ULPI Wake-up – Read/Write. Writing the ‘1’ to this bit will begin the wake-up operation. The bit will automatically transition to 0 after the wake-up is complete. Once this bit is set, the driver can not set it back to ‘0’. Note: The driver must never execute a wake-up and a read/write operation at the same time.
30 ULPIRUN	ULPI Read/Write Run – Read/Write. Writing the ‘1’ to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver can not set it back to ‘0’. Note: The driver must never execute a wake-up and a read/write operation at the same time.
29 ULPIRW	ULPI Read/Write Control – Read/Write. (0) – Read; (1) – Write. This bit selects between running a read or write operation.
28	Reserved, read as 0.
27 ULPISS	ULPI Sync State – Read Only. (1) – Normal Sync. State. (0) In another state (i.e. carkit, serial, low power) This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0.
26–24 ULPIPORT	ULPI Port Number – Read/Write. For the wake-up or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi-port host. The range is 0 to 7. This field should always be written as a 0 for the non-multi port products.
23–16 ULPIADDR	ULPI Data Address – Read/Write. When a read or write operation is commanded, the address of the operation is written to this field.
15–8 ULPIDATRD	ULPI Data Read – Read Only. After a read operation completes, the result is placed in this field.
7–0 ULPIDATWR	ULPI Data Write – Read/Write. When a write operation is commanded, the data to be sent is written to this field.

47.5.2.6.11 CONFIGFLAG Register

This register (located at base address offset 0x0180), is not used in this implementation. A read from this register returns a constant of 0x0000_0001, to indicate that all port routines default to this host controller.

47.5.2.6.12 Port Status Control *x* Registers (PORTSC*x*, *x* = 1…8)

These registers are used differently for host and device controllers, as follows:

- Host Controller

  A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register. Software uses this information as an input parameter to determine how many ports require servicing.

  This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

  — No device connected

  — Port disabled

  If the port has port power control, this state remains until software applies power to the port by setting port power to one.

- Device Controller

  A device controller must implement only port register 1, and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Figure 47-36 shows the register. Table 47-33 provides field descriptions.

Offset 0x0184 (PORTSC1) Access: Field-dependent

…

0x01A0 (PORTSC8)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS		STS	PTW	PSPD		0	PFSC	PHCD	WKOC	WKDS	WKCN	PTC			
W																
Reset	*n*	*n*	*n*	*n*	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W																
Reset	0	0	0	1/0[1]	*u*	*u*	0	0	0	0	0	0	0	1/0[1]	0	0

[1] Resets to 1 in host mode, 0 in device mode.

Figure 47-36. Port Status Control *x* Registers (PORTSC*x*)

Table 47-33. Port Status Control *x* Registers Field Descriptions

Field	Description
31–30 PTS	Parallel Transceiver Select. This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected. If VUSB_HS_PHY_TYPE is set for 0,1,2, or 3 then this bit is read only. If VUSB_HS_PHY_TYPE is 3,4,5, or 6 then this bit is read/write. This field is reset to: 00 if VUSB_HS_PHY_TYPE = 0,4 – UTMI/UTMI+ 01 if VUSB_HS_PHY_TYPE = 1,5 – Reserved 10 if VUSB_HS_PHY_TYPE = 2,6 – ULPI 11 if VUSB_HS_PHY_TYPE = 3,7 – Serial/1.1 PHY (FS Only) This bit is not defined in the EHCI specification.
29 STS	Serial Transceiver Select. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ or ULPI physical interface to provide FS/LS signaling instead of the parallel interface. If VUSB_HS_PHY_SERIAL is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY_SERIAL is 3 or 4 then this bit is read/write. This bit has no effect unless the parallel transceiver select (PTS) bit is set to UTMI+ or ULPI. The Serial/1.1 physical interface uses the serial interface engine for FS/LS signaling regardless of this bit value. This bit is not defined in the EHCI specification. **Note:** This bit is reserved for future operation and is a placeholder adding dynamic use of the serial engine in accord with UMTI+ and ULPI characterization logic.
28 PTW	Parallel Transceiver Width. This register bit is used in conjunction with the configuration constant VUSB_HS_PHY8_16 to control whether the data bus width of the UTMI transceiver interface. If VUSB_HS_PHY8_16 is set for 0 or 1 then this bit is read only. If VUSB_HS_PHY8_16 is 2 or 3 then this bit is read/write. This bit is reset to 1 if VUSB_HS_PHY8_16 selects a default UTMI interface width of 16-bits else it is reset to 0. 0 8-bit [60MHz] UTMI interface is selected. 1 16-bit [30MHz] UTMI interface is selected. This bit has no effect if the serial interface is selected. This bit is not defined in the EHCI specification. **Note:** If this bit changes from 0 to 1, software must reset the clock generation logic in the UTMI PHY before enabling the controller core as follows: • Clear the UTMI USB enable bit (USBEN in the USB_PHY_CTRL_FUNC register) • Set USBEN.
27–26 PSPD	Port speed. This read-only field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator. 00 Full Speed 01 Low Speed 10 High Speed 11 Reserved, not used. This bit is not defined in the EHCI specification.
25	Reserved.
24 PFSC	Port Force Full Speed Connect – Read/Write. Default = 0. Setting this bit to 1 forces the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification, and is intended for debugging purposes.

Table 47-33. Port Status Control *x* Registers Field Descriptions (continued)

Field	Description
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) – Read/Write. Default = 0. 0 PHY clock enabled. 1 PHY clock disabled. **Note:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend – Clock Disable when the device is not running (USBCMD Run/Stop=0) or the host has signaled suspend (PORTSC SUSPEND=1). Low power suspend will be cleared automatically when the host has signaled resume if using a circuit similar to that in. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend – Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. See for more discussion on clock disable and power down issues. This bit is not defined in the EHCI specification.
22 WKOC	Wake on Over-current Enable. Setting this bit to 1 enables the port to be sensitive to over-current conditions as wake-up events. This field is 0 if the Port Power *(*PP*)* is 0. This bit is output from the controller as signal pwrctl_wake_ovrcurr_en (OTG/host core only) for use by an external power control circuit.
21 WKDS	Wake on Disconnect Enable. Setting this bit to 1 enables the port to be sensitive to device disconnects as wake-up events. This field is 0 if Port Power *(*PP*)* is 0 or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.
20 WKCN	Wake on Connect Enable (WKCNNT_E) — Read/Write. Default = 0. Setting this bit to 1 enables the port to be sensitive to device connects as wake-up events. This bit is zero if Port Power (PP) is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.
19–16 PTC	Port Test Control. Any value other than 0x0 indicates that the port is operating in test mode. 0x0 TEST_MODE_DISABLE 0x1 J_ STATE 0x2 K_STATE 0x3 SE0 (host) / NAK (device) 0x4 Packet 0x5 FORCE_ENABLE_HS 0x6 FORCE_ENABLE_FS 0x7 FORCE_ENABLE_LS 0x8–0xFReserved See Chapter 7 of the USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **Note:** Low speed operations are not supported as a peripheral device.[

Table 47-33. Port Status Control *x* Registers Field Descriptions (continued)

Field	Description
15–14 PIC	Port Indicator Control — Read/Write. Default = 00. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bit is: Bit ValueMeaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined See the USB Specification Revision 2.0 Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org for a description on how these bits are to be used. This field is output from the controller as signals port_ind_ctl_1 & port_ind_ctl_0 for use by an external led driving circuit.
13 PO	Port Owner—Read/Write. Default = 0. In this implementation, this bit is always 0.
12 PP	Port Power (PP)—Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. • If PPC = 0 (device-only implementation with no OTG capability), then PP = 0 (read-only). • If PPC = 1 (host/OTG controller), then this bit represents the current setting of the port power control switch (0=off, 1=on). A non-powered port (PP = 0), is non-functional and does not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port (PP = 1), the PP bit in each affected port can be changed by the host controller driver from a one to a zero (removing power from the port).
11–10 LS[1:0]	Line Status—Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits are: Bits [11:10]Meaning 00b SE0 10b J-state 01b K-state 11b Undefined In host mode, the use of line state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line state by the device controller driver is not necessary.
9 HSP	High-Speed Port — Read Only. Default = 0. 0 Host/device connected to the port is not in high-speed mode. 1 Host/device connected to the port is in high-speed mode. **Note:** HSP is redundant with PSPD(27:26) but remains in the design for compatibility. This bit is not defined in the EHCI specification.
8 PR	Port Reset. This bit is 0 if the Port Power (PP) bit is 0. • In Host Mode: This bit is read/write accessible. 0 Port is not in reset (Default) 1 Port is in reset. When software writes a 1 to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit automatically changes to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.* • In Device Mode: This bit is a read-only status bit. Device reset from the USB bus is also indicated in the USBSTS register.

Table 47-33. Port Status Control *x* Registers Field Descriptions (continued)

Field	Description
7 SUSP	Suspend. Function in host and device modes is described as follows. In Host Mode: The bit is read/write, with the following settings: 0 Port not in suspend state (default). 1 Port in suspend state. The {Port Enabled, Suspend} bits of this register define the port states as follows: 0*n* Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The bit is a read-only status bit, with the following settings: 0 Port is not in suspend state (default). 1 Port is in suspend state.
6 FPR	Force Port Resume • Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* When the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not EHCI-compatible. • Device mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, the Port Change Detect bit in the USBSTS register is also set to one. 0 No resume (K-state) detected/driven on port (default). 1 Resume detected/driven on port.
5 OCC	Overcurrent Change—R/WC. Software clears this bit by writing 1. For host/OTG implementations the user can provide overcurrent detection to the vbus_pwr_fault input for this condition. 0 No change has occurred to overcurrent active status. 1 Change has occurred to overcurrent active status. For device-only implementations this bit is always 0.

Table 47-33. Port Status Control *x* Registers Field Descriptions (continued)

Field	Description
4 OCA	Over-current Active—Read Only. This bit will automatically transition from one to zero when the overcurrent condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit is always 0. 0 This port does not have an over-current condition. (default) 1 This port currently has an over-current condition.
3 PEC	Port Enable/Disable Change—R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. • In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. • In Device mode: The device port is always enabled. (This bit is always 0)
2 PE	Port Enabled/Disabled—Read/Write. • In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. 0 Port disabled (default) 1 Port enabled • In Device Mode: The device port is always enabled. (This bit is always 1)

Table 47-33. Port Status Control *x* Registers Field Descriptions (continued)

Field	Description
1 CSC	Connect Status Change—R/WC. • In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. 0 No change in current connect status (default) 1 Change in current connect status. • In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status—Read Only. • In Host Mode: 0 No device is present. (default) 1 Device is present on port. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. • In Device Mode: A setting of 1 indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 Device is not attached. (default) 1 Device is attached.

47.5.2.6.13 On-the-Go Status and Control Register (OTGSC)

A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller.

The OTGSC register has four sections:

- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls(Read/Write)

The status inputs are debounced using a 1 msec time constant. Values on the status inputs that do not persist for more than 1 msec will not cause a status input register update or OTG interrupt. See also the USBMODE register.

Figure 47-37 shows the register, and Table 47-34 provides field descriptions.

Offset 0x01A4 (OTGSC) Access: field dependent

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	DPIE	1msE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	0	DPIS	1msS	BSEIS	BSVIS	ASVIS	AVIVS	IDIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DPS	1msT	BSE	BSV	ASV	AVV	ID	HADP	HABA	IDPU	DP	OT	HAAR	VC	VD
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 47-37. OTG Status Control Register (OTGSC)

Table 47-34. OTG Status Control Register Field Descriptions

Field	Description
31	Reserved.
30 DPIE	Data Pulse Interrupt Enable
29 1msE	1 millisecond timer Interrupt Enable – Read/Write
28 BSEIE	B Session End Interrupt Enable – Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable – Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable – Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable – Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable – Read/Write. Setting this bit enables the USB ID interrupt.
23	Reserved.
22 DPIS	Data Pulse Interrupt Status – Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when the CM field (USBMODE register) = 11(denoting host) and the port power bit (PORTSC0 register) = 0 (power off). Software must write 1 to clear this bit.
21 1msS	1 millisecond timer Interrupt Status – Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status – Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit
19 BSVIS	B Session Valid Interrupt Status – Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status – Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a one to clear this bit.

Table 47-34. OTG Status Control Register Field Descriptions (continued)

Field	Description
17 AVVIS	A VBus Valid Interrupt Status – Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status – Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.
15	Reserved.
14 DPS	Data Bus Pulsing Status – Read Only. A ‘1’ indicates data bus pulsing is being detected on the port.
13 1msT	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End – Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid – Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid – Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid – Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID – Read Only. 0 = A device, 1 = B device
7 HABA	Hardware Assist B-Disconnect to A-connect ñ Read/Write. 0 = Disabled. 1 = Enable automatic B-disconnect to A-connect sequence.
6 HADP	Hardware Assist Data-Pulse ñ Write to Set. 1 = Start Data Pulse Sequence.
5 IDPU	ID Pullup – Read/Write. This bit provide control over the ID pull-up resister; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing – Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination – Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 HAAR	Hardware Assist Auto-Reset ñ Read/Write. 0 = Disabled. 1 = Enable automatic reset after connect on host port.
1 VC	VBUS Charge – Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge – Read/Write. Setting this bit causes VBus to discharge through a resistor.

47.5.2.6.14 USB Device Mode Register (USBMODE)

Figure 47-38 shows the register, and Table 47-35 provides field descriptions.

Offset 0x01A8 (USBMODE) Access: field dependent

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SDIS	SLOM	ES	CM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—	—

Figure 47-38. USB Device Mode Register (USBMODE)

Table 47-35. USB Device Mode Register Field Descriptions

Field	Description
31–5	Reserved.
4 SDIS	Stream Disable Mode. (0 – Inactive [default]; 1 – Active) • Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active. • Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **Note:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature. **Note:** The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See the Control Endpoint Operation Model. 0 Setup Lockouts On (default) 1 Setup Lockouts Off DCD requires use of the setup data buffer tripwire in the USB Command Register (USBCMD)

Table 47-35. USB Device Mode Register Field Descriptions (continued)

Field	Description
2 ES	Endian Select – Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. 0 Little endian [Default] 1 Big endian
1–0 CM	Controller Mode – R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. 00Idle [Default for combination host/device] 01Reserved 10Device Controller [Default for device only controller] 11Host Controller [Default for host only controller]

47.5.2.6.15 Endpoint Setup Status Register (ENDPTSETUPSTAT)

This register is used in device mode only. Figure 47-39 shows the register, and Table 47-36 provides field descriptions.

Offset 0x01AC (ENDPTSETUPSTAT) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENDPTSETUPSTAT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-39. Endpoint Setup Status Register (ENDPTSETUPSTAT)

Table 47-36. Endpoint Setup Status Register Field Descriptions

Field	Description
31–16	Reserved.
15–0 ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from the Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See “Managing Endpoints” in the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual. This register is only used in device mode.

47.5.2.6.16 Endpoint Initialization Register (ENDPTPRIME)

This register is only used in device mode. Figure 47-40 shows the register, and Table 47-37 provides field descriptions.

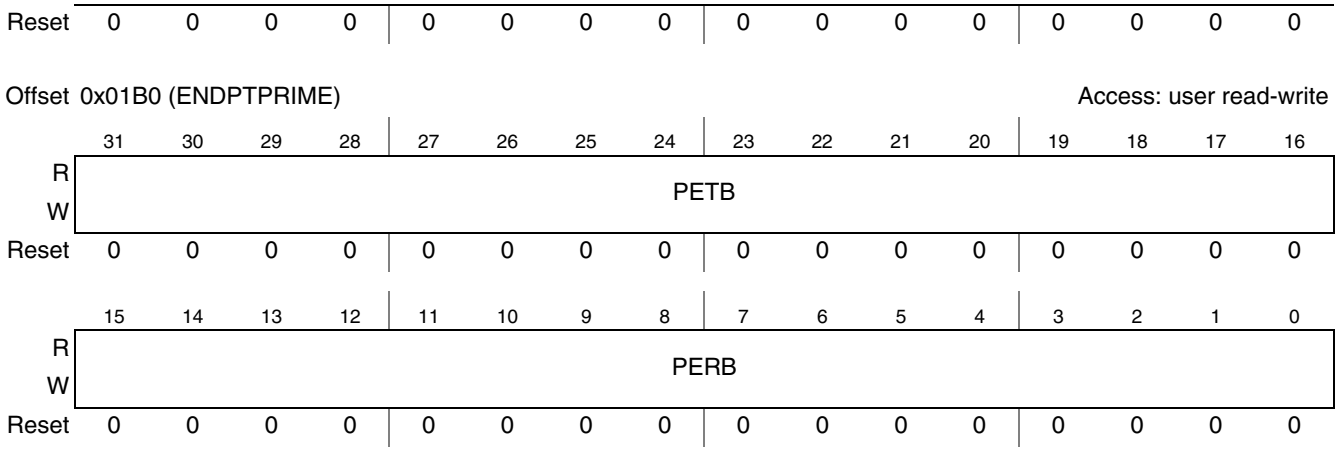


Figure 47-40. Endpoint Initialization Register (ENDPTPRIME)

Table 47-37. Endpoint Initialization Register Field Descriptions

Field	Description
31–16 PETB	Prime Endpoint Transmit Buffer – R/WS. For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[15] – Endpoint #15 PETB[1] – Endpoint #1 PETB[0] – Endpoint #0
15–0 PERB	Prime Endpoint Receive Buffer – R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. Bit 15 – Endpoint #15 … Bit 0 – Endpoint #0

47.5.2.6.17 Endpoint De-Initialize Register (ENDPTFLUSH)

This register is only used in device mode. Figure 47-41 shows the register, and Table 47-38 provides field descriptions.

Offset 0x01B4 (ENDPTFLUSH) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FETB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FERB															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-41. Endpoint De-Initialize Register (ENDPTFLUSH)

Table 47-38. Endpoint De-Initialize Register Field Descriptions

Field	Description
31–16 FETB	Flush Endpoint Transmit Buffer – R/WS. Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[15] – Endpoint #15 … FETB[1] – Endpoint #1 FETB[0] – Endpoint #0
15–0 FERB	Flush Endpoint Receive Buffer – R/WS. Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. Bit 15 – Endpoint #15 … Bit 1 – Endpoint #1 Bit 0 – Endpoint #0

47.5.2.6.18 Endpoint Status Register (ENDPTSTAT)

This register is only used in device mode. Figure 47-42 shows the register, and Table 47-39 provides field descriptions.

Offset 0x01B8 (ENDPTSTAT) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ETBR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERBR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-42. Endpoint Status Register (ENDPTSTAT)

Table 47-39. Endpoint Status Field Descriptions

Field	Description
31–16 ETBR	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[15]– Endpoint #15 … ETBR[0]– Endpoint #0
15–0 ERBR	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[15]– Endpoint #15 … ERBR[0]– Endpoint #0

47.5.2.6.19 Endpoint Complete Register (ENDPTCOMPLETE)

This register is only used in device mode. Figure 47-43 shows the register, and Table 47-40 provides field descriptions.

Offset 0x01BC (ENDPTCOMPLETE) Access: user read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ETCE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERCE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 47-43. Endpoint Complete Register (ENDPTCOMPLETE)

Table 47-40. Endpoint Complete Register Field Descriptions

Field	Description
31–16 ETCE	Endpoint Transmit Complete Event—R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ETCE[15]—Endpoint #15 … ETCE[0]—Endpoint #0
15–0 ERCE	Endpoint Receive Complete Event—RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the *USBINT*. Writing a one will clear the corresponding bit in this register. ERCE[15]—Endpoint #15 … ERCE[0]—Endpoint #0

47.5.2.6.20 Endpoint Control 0 Register (ENDPTCTRL0)

Every device implements Endpoint0 as a control endpoint.

Figure 47-44 shows the register, and Table 47-41 provides field descriptions.

Offset 0x01C0 (ENDPTCTRL0) Access: field dependent

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TXE	0	0	0	TXT		0	TXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RXE	0	0	0	RXT		0	RXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 47-44. Endpoint Control 0 Register (ENDPTCTRL0)

Table 47-41. Endpoint Control 0 Register Field Descriptions

Field	Description
31–24	Reserved.
23 TXE	TX Endpoint Enable 0 Reserved, does not occur 1 Enabled Endpoint 0 is always enabled.
22–20	Reserved.
19–18 TXT	TX Endpoint Type – Read/Write 00 Control 01–11 Reserved, do not occur Endpoint0 is fixed as a control endpoint.
17	Reserved
16 TXS	TX Endpoint Stall – Read/Write 0 Endpoint OK (Default) 1 Endpoint Stalled Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.
15—8	Reserved.
7 RXE	RX endpoint enable 0 Reserved, does not occur 1 Enabled Endpoint0 is always enabled.
6—4	Reserved.

Table 47-41. Endpoint Control 0 Register Field Descriptions (continued)

Field	Description
3—2 RXT	RX endpoint type – read/write 00 Control 01–11 Reserved, does not occur Endpoint0 is fixed as a control endpoint.
1	Reserved.
0 RXS	RX endpoint stall – read/write 0 Endpoint OK. [default] 1 Endpoint stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.

47.5.2.6.21 Endpoint Control *x* Registers (ENDPTCTRL*x*, *x* = 1…15)

There is an ENDPTCTRL*x* register implemented for each endpoint in a device. Figure 47-45 shows the register, and Table 47-42 provides field descriptions.

Offset 0x01C0 (ENDPTCTRL1) Access: user read-write

…

0x01F8 (ENDPTCTRL15)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	TXE	TXR	TXI	0	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	RXE	RXR	RXI	0	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Figure 47-45. Endpoint Control *x* Registers (ENDPTCTRL*x*)

NOTE

If one endpoint direction is enabled and the paired endpoint of the opposite direction is disabled, then the unused direction type must be changed from the default control-type to any other type (IE. Bulk-type). leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Table 47-42. Endpoint Control *x* Registers Field Descriptions

Field	Description
31–24	Reserved.
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 – Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20	Reserved.
19–18 TXT	TX Endpoint Type – Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source – Read/Write 0 Dual port memory buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall – Read/Write 0 Endpoint OK 1 Endpoint stalled This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.
15–8	Reserved.
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 – Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.

Table 47-42. Endpoint Control *x* Registers Field Descriptions (continued)

Field	Description
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4	Reserved.
3–2 RXT	RX endpoint type – Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt Endpoint
1 RXD	RX endpoint data sink – Read/Write 0 Dual Port Memory Buffer/DMA Engine (Default) Should always be written as zero.
0 RXS	RX Endpoint Stall – Read/Write 0 Endpoint OK. [Default] 1 Endpoint stalled This bit is set to 1 automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It will be cleared automatically upon receipt a SETUP request if this endpoint is configured as a control endpoint, Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above,

47.5.2.7 OTG Operations

47.5.2.7.1 Register Bits Used for OTG Operations

The Register interface described in Section 47.5.2.6, “Device/Host Operational Registers,” has separate behaviors for device mode and for host mode. However, during OTG operations it is necessary to perform tasks independent of the controller mode.

As described in Section 47.5.2.6.21, “Endpoint Control x Registers (ENDPTCTRLx, x = 1…15),” the only way to transition the controller mode out of host or device mode is by programming the controller reset bit in the USBCMD register. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

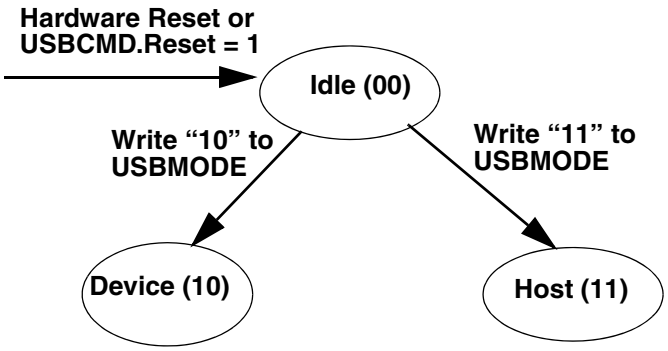


Figure 47-46. Controller Mode

The following register bits can be used for OTG operations. These bits are independent of the controller mode, and are not affected by a write to the reset bit in the USBCMD register:

- All identification registers
- All device/host capability registers
- OTG states control register (OTGSC): all bits
- PORTSC register:
  - Physical interface select
  - Physical interface serial select
  - Physical interface data width
  - Physical interface low power
  - Physical interface wake signals
  - Port indicators
  - Port power

47.5.2.7.2 Hardware Assist

The hardware assist provides automated response and sequencing that avoid significant interrupt latency response times introduced by software. The use of this additional circuitry is optional, and can be used to assist the auto-reset, data-pulse, and B-connect to A-connect sequences described below.

Auto-Reset Sequence

When the HAAR bit (OTGSC register) is set to 1, the host automatically starts a reset after a connect event. This shortcuts the normal process, where software is notified of the connect event and starts the reset. When the HAAR bit is set, software still receives notification of the connect event, but is not responsible to write the reset bit. Software will be notified again after the reset is complete using the enable change bit in the PORTSC register, which causes a port change interrupt.

This hardware assist ensures the OTG parameter TB_ACON_BSE0_MAX = 1 ms is met.

Data-Pulse Sequence

Writing a 1 to HADP (OTGSC register) will start a data pulse of approximately 7 ms in duration and then automatically cease the data pulsing. During the data pulse, the DP will be set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion or simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist ensures that data pulsing meets the OTG requirement of between 5–10 ms.

B-Disconnect to A-Connect Sequence

During HNP, the B-disconnect occurs from the OTG A_suspend state and within 3 ms, the A device must enable the pullup on the DP leg in the A-peripheral state. If the HABA bit (OTGSC register) is set to 1, the host controller port is in suspend mode, and the device disconnects, then hardware assist performs the following actions:

1. Reset the OTG core.
2. Write the OTG core into device mode.
3. Write the device run bit to a 1 and enable necessary interrupts, including:
    - USB reset enable (URE); enables interrupt on USB bus reset to device
    - Sleep enable (SLE); enables interrupt on device suspend
    - Port change detect enable (PCE); enables interrupt on device connect

When software has enabled this hardware assist, it must not interfere during the transition or write any register in the core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred, or at least first verify that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (SOF, for example) that may or may not cascade and my be cleared by the soft reset depending on the software response time.

After the core has entered device mode by the hardware assist, the DCD must ensure that the ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, will require at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pullup, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist ensures the parameter TA_BDIS_ACON_MAX = 3 ms is met.

47.5.3 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the enhanced host controller (hardware). The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a periodic schedule, periodic frame

list, asynchronous schedule, isochronous transaction descriptors, split-transaction isochronous transfer descriptors, queue heads, and queue element transfer descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. isochronous data streams are managed using isochronous transaction descriptors. isochronous split-transaction data streams are managed with split-transaction isochronous transfer descriptors. all interrupt, control, and bulk data streams are managed using queue heads and queue element transfer descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writable fields. The host controller must preserve the read-only fields on all data structure writes.

47.5.3.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the PERIODICLISTBASE address register and the FRINDEX register. The periodic schedule is based on an array of pointers called the periodic frame list. The PERIODICLISTBASE address register is combined with the FRINDEX register to produce a memory pointer into the frame list. The periodic frame list implements a sliding window of work over time.

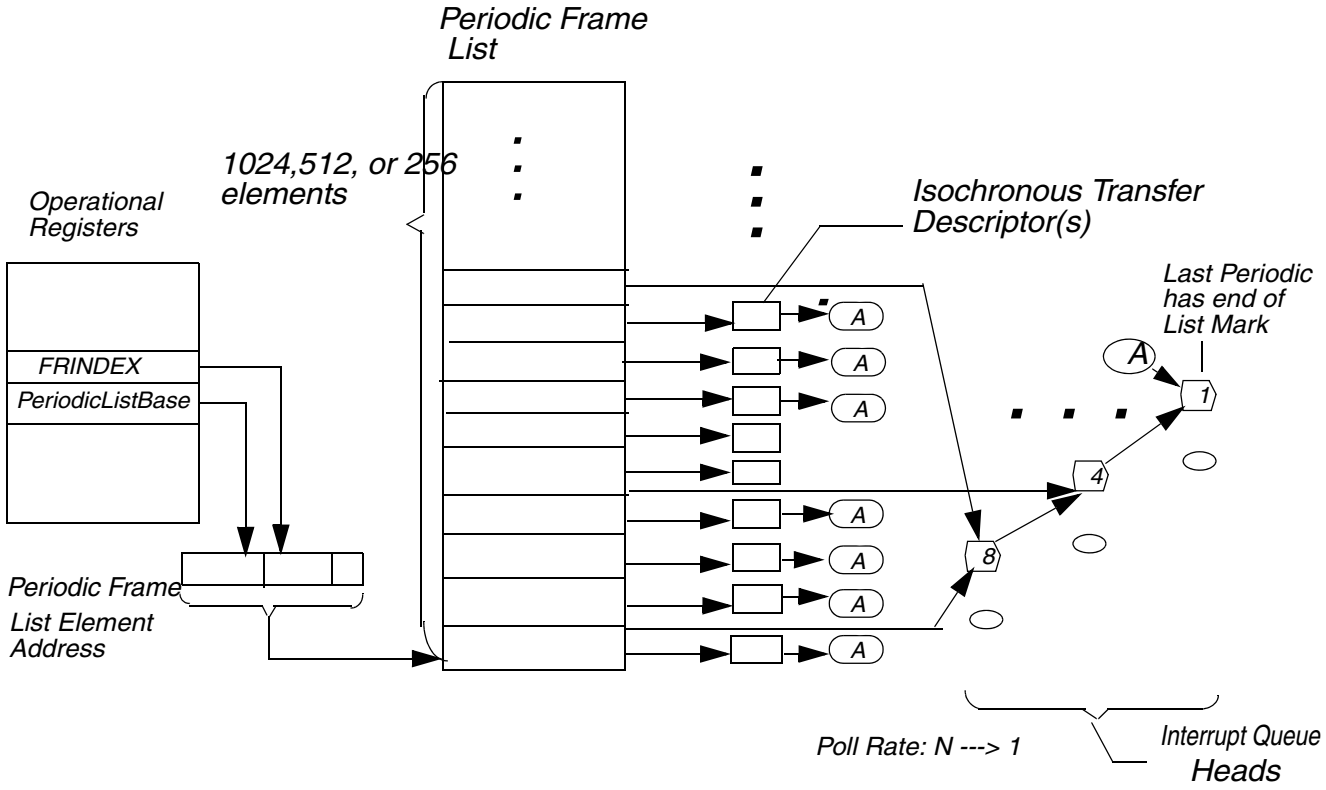


Figure 47-47. Periodic Schedule Organization

[1] Split transaction interrupt, bulk and control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4K-page aligned array of frame list link pointers. The length of the frame list is programmable using the frame list size (FS) bit in the USBCMD register.Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current microframe. The link pointers are aligned on dword boundaries within the frame list.

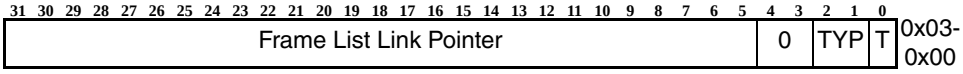


Figure 47-48. Format of Frame List Element Pointer

Frame list link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupts). System software should not place non-periodic schedule items into the periodic schedule.

The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing:

- The TYP field indicates the exact type of data structure being referenced by this pointer. TYP field encodings are given in Table 47-43.

Table 47-43. TYP Field Value Definitions

TYP Field Value	Meaning
00	Isochronous Transfer Descriptor
01	Queue Head
10	Split Transaction Isochronous Transfer Descriptor.
11	Frame Span Traversal Node.

- When the T bit (bit 0) is set to 1, the host controller never uses the value of the frame list pointer as a physical memory pointer.

47.5.3.2 Asynchronous List Queue Head Pointer

The asynchronous transfer list (at the location specified by the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. The host controller uses this list only if at least one of the following conditions holds:

- The host controller reaches the end of the periodic list.
- The periodic list is disabled,
- The periodic list is empty.

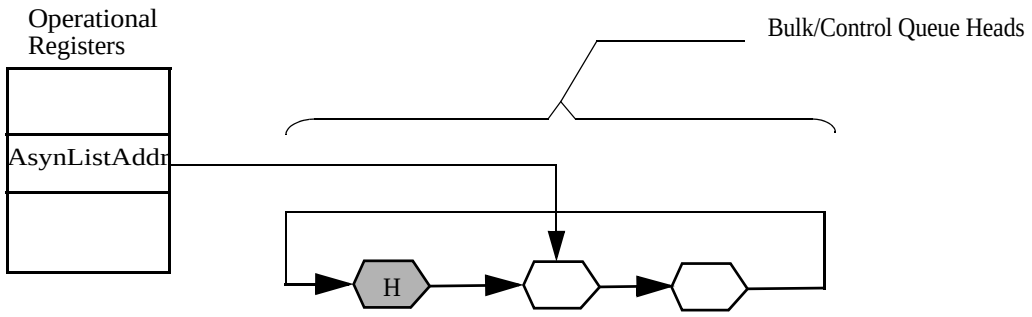
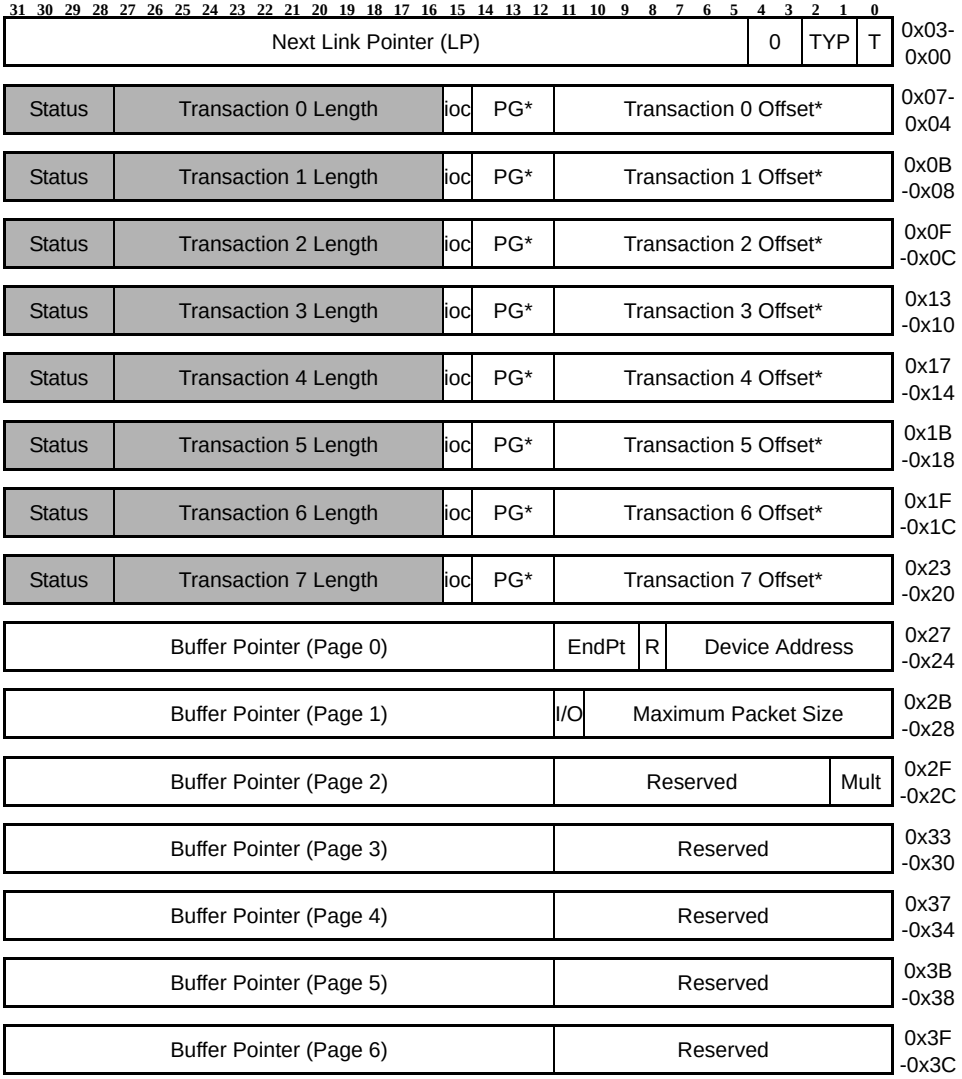


Figure 47-49. Asynchronous Schedule Organization

The asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply pointer to the *next* queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

47.5.3.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

Figure 47-50 shows the format of an isochronous transfer descriptor. This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. iTDs must be aligned on a 32-byte boundary.



Note: *Note: these fields may be modified by the host controller if the I/O field indicates an OUT.

Figure 47-50. Isochronous Transaction Descriptor (iTD)

47.5.3.3.1 Next Link Pointer

The first dword of an iTD is a pointer to the next schedule data structure.

Table 47-44. Next Schedule Element Pointer

Bits	Description
31–5 LP	Link Pointer. These bits correspond to memory address signals [31:5], respectively. This field points to another isochronous transaction descriptor (iTD/siTD) or Queue Head (QH).
4–3	Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2–1 TYP	QH/(s)iTD Select. This field indicates to the host controller whether the item referenced is an iTD, siTD or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor 11 FSTN (frame span traversal node)
0 T	Terminate. 0 Link Pointer field is valid. 1 Link Pointer field is not valid

47.5.3.3.2 iTD Transaction Status and Control List

Dwords 1 through 8 are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The *PG* and *Transaction X Offset* fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three dwords of the buffer page pointer list, to execute a transaction on the USB.

Table 47-45. iTD Transaction Status and Control

Bits	Description	
31–28	**Status.** This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	**Active.** Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	**Data Buffer Error.** Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	**Babble Detected.** Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
	28	**Transaction Error (XactErr).** Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27–16	**Transaction X Length**. For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (e.g. 0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15	**Interrupt On Complete (IOC).** If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14–12	**Page Select (PG).** These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11–0	**Transaction X Offset**. This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

47.5.3.3.3 iTD Buffer Page Pointer List (Plus)

Dwords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Since each pointer is a 4K-aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

Table 47-46. iTD Buffer Pointer Page 0 (Plus)

Bits	Description
31–12	**Buffer Pointer (Page 0).** This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–8	**Endpoint Number (Endpt).** This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	**Reserved.** Bit reserved for future use and should be initialized by software to zero.
6–0	**Device Address.** This field selects the specific device serving as the data source or sink.

Table 47-47. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31–12	**Buffer Pointer (Page 1).** This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11	**Direction (I/O). 0 = OUT; 1 = IN.** This field encodes whether the high-speed transaction should use an IN or OUT PID.
10–0	**Maximum Packet Size.** This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (.e.g. per microframe). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 47-48. iTD Buffer Pointer Page 2 (Plus)

Bits	Description
31–12	**Buffer Pointer.** This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–2	**Reserved.** This bit reserved for future use and should be set to zero.
1–0	**Multi.** This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (e.g. per microframe). The valid values are: 00 Reserved. A zero in this field yields undefined results. 01 One transaction to be issued for this endpoint per microframe 10 Two transactions to be issued for this endpoint per microframe 11 Three transactions to be issued for this endpoint per microframe

Table 47-49. iTD Buffer Pointer Page 3-6

Bit	Description
31–12	**Buffer Pointer.** This is a 4K-aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11–0	**Reserved.** These bits reserved for future use and should be set to zero.

47.5.3.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

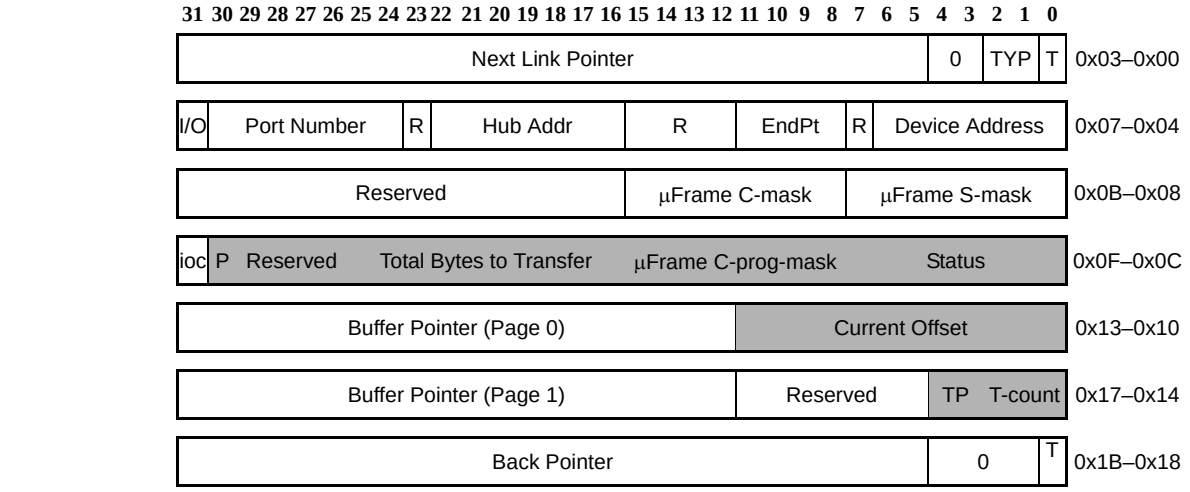


Figure 47-51. Split-transaction Isochronous Transaction Descriptor (siTD)

47.5.3.4.1 Next Link Pointer

Dword0 of a siTD is a pointer to the next schedule data structure.

Table 47-50. Next Link Pointer

Bits	Description
31–5	**Next Link Pointer (LP).** This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4–3	**Reserved.** These bits must be written as zeros.
2–1	**QH/(s)iTD Select (TYP).** This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor 11 FSTN (frame span traversal node)
0	**Terminate (T).** 1=Link Pointer field is not valid. 0=Link Pointer is valid.

47.5.3.4.2 siTD Endpoint Capabilities/Characteristics

Dwords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and microframe scheduling control.

Table 47-51. Endpoint and Transaction Translator Characteristics

Bits	Description
31	**Direction (I/O).0 = OUT; 1 = IN.** This field encodes whether the full-speed transaction should be an IN or OUT.
30–24	**Port Number.** This field is the port number of the recipient Transaction Translator.
23	**Reserved.** Bit reserved and should be set to zero.
22–16	**Hub Address.** This field holds the device address of the Companion Controllers' hub.
15–12	**Reserved.** Field reserved and should be set to zero.
11–8	**Endpoint Number (Endpt).** This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	**Reserved.** Bit is reserved for future use. It should be set to zero.
6–0	**Device Address.** This field selects the specific device serving as the data source or sink.

Table 47-52. Microframe Schedule Control

Bits	Description
31–16	**Reserved.** This field reserved for future use. It should be set to zero.
15–8	**Split Completion Mask (μFrame C-Mask).** This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which microframes the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7–0	**Split Start Mask (μFrame S-mask).** This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which microframes the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

47.5.3.4.3 siTD Transfer State

Dwords 3-6 are used to manage the state of the transfer.

Table 47-53. siTD Transfer Status and Control

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>31</td><td>Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.</td></tr>
<tr><td>30</td><td>Page Select (P). Used to indicate which data page pointer should be concatenated with the CurrentOffset field to construct a data buffer pointer (0 selects Page 0 pointer and 1 selects Page 1). The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a one to a zero).</td></tr>
<tr><td>29–26</td><td>Reserved. This field reserved for future use and should be set to zero.</td></tr>
<tr><td>25–16</td><td>Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)</td></tr>
<tr><td>15–8</td><td>μFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.</td></tr>
<tr><td>7–0</td><td>Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>7</td><td>Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.</td></tr>
<tr><td>6</td><td>ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.</td></tr>
<tr><td>5</td><td>Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.</td></tr>
<tr><td>4</td><td>Babble Detected. Set to a one by the Host Controller during status update when” babble” is detected during the transaction generated by this descriptor.</td></tr>
<tr><td>3</td><td>Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, etc.). This bit will only be set for IN transactions.</td></tr>
<tr><td>2</td><td>Missed microframe. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.</td></tr>
<tr><td>1</td><td>Split Transaction State (SplitXstate). The bit encodings are as follows:
00 Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask.
01 Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.</td></tr>
<tr><td>0</td><td>Reserved. Bit reserved for future use and should be set to zero.</td></tr>
</table></td></tr>
</table>

47.5.3.4.4 siTD Buffer Pointer List (plus)

Dwords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each dword in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each dword are used as additional transfer state.

Table 47-54. Buffer Page Pointer List (Plus)

Bits	Description
31–12	**Buffer Pointer List.** Bits [31:12] of dwords 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* specifies the *current* active pointer
11–0	• Page 0: The 12 least significant bits of the Page 0 pointer give the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field)). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero). • Page 1: The least significant bits of Page 1 pointer is split into three sub-fields, as shown in the following table.

Bits	Description
11–5	Reserved.
4–3	**Transaction position (TP).** This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: 00 All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes). 01 Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction 10 Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes. 11 End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2–0	**Transaction count (T-Count).** Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

47.5.3.4.5 siTD Back Link Pointer

Dword 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

Table 47-55. siTD Back Link Pointer

Bits	Description
31–5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4–1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

47.5.3.5 Queue Element Transfer Descriptor (qTD)

This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20480 (5*4096) bytes. The structure contains two structure pointers used for queue advancement, a dword of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

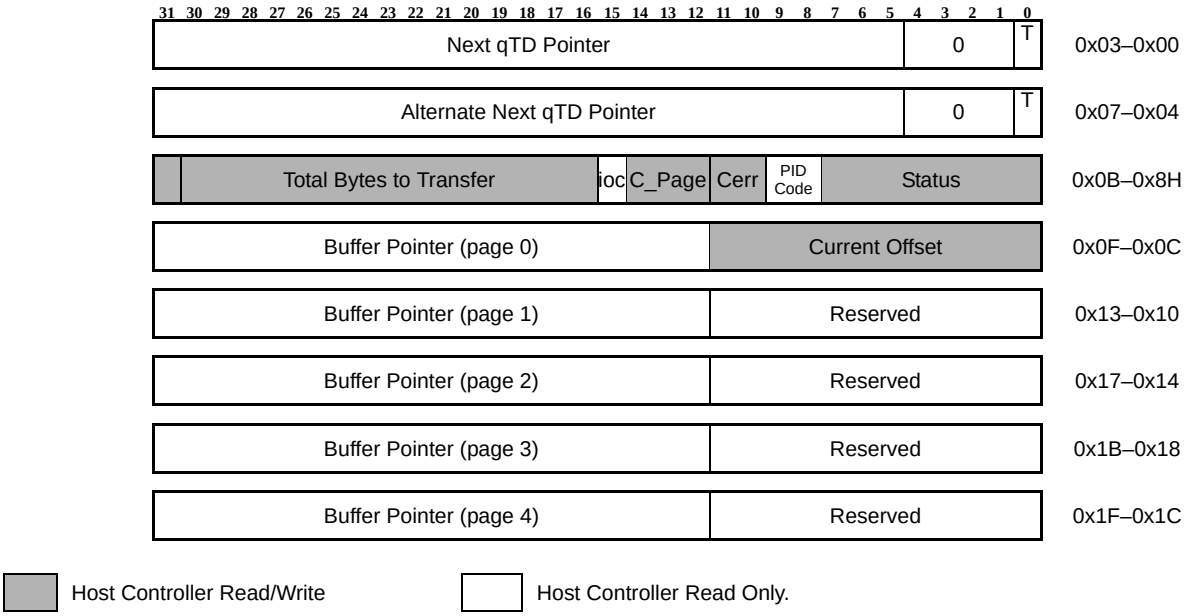


Figure 47-52. Queue Element Transfer Descriptor Block Diagram

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

47.5.3.5.1 Next qTD Pointer

The first dword of an element transfer descriptor is a pointer to another transfer element descriptor.

Table 47-56. qTD Next Element Transfer Pointer (Dword 0)

Bits	Description
31–5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.

Table 47-56. qTD Next Element Transfer Pointer (Dword 0)

4–1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). This bit indicates to the Host Controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid Transfer Element Descriptor) 1 Pointer is invalid.

47.5.3.5.2 Alternate Next qTD Pointer

The second dword of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet. To be more explicit the host controller will always use this pointer when the current qTD is retired due to short packet.

Table 47-57. TD Alternate Next Element Transfer Pointer (Dword 1)

Bits	Description
31–5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4–1	Reserved. These bits are reserved and their value has no effect on operation.
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

47.5.3.5.3 qTD Token

The third dword of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

Table 47-58. qTD Token (Dword 2)

Bits	Description
31	Data Toggle. This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.
30–16	Total Bytes to Transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4 Kbytes (0x5000). This is the maximum number of bytes that 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that *Total Bytes To Transfer* be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QHD.Maximum Packet Length. Although it is possible to create a transfer up to 20 Kbytes, this assumes the first offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16 Kbytes**. Therefore, the maximum recommended transfer is 16 Kbytes (0x4000).

Table 47-58. qTD Token (Dword 2) (continued)

15	Interrupt On Complete (IOC). If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.
14–12	Current Page (C_Page). This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0x0 to 0x4. The host controller is not required to write this field back when the qTD is retired.
11–10	Error Counter (CERR). This field is a 2-bit down-counter that keeps track of the number of consecutive errors detected while executing this qTD. • If this field is programmed with a nonzero value during setup, the host controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the Halted bit to a 1, and sets an error status bit for the error that caused CERR to decrement to zero. An interrupt is generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one. • If HCD programs this field to zero during set-up, the Host Controller will not count errors for this qTD and there will be no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD. The behavior of the CERR field for different error types is as follows: • Babble or stall detection automatically halts the queue head—CERR is not decremented. • Data buffer errors are host problems— CERR is not decremented. • Transaction errors cause CERR to be decremented. If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller is responsible to reset CERR to extend the total number of errors for this transaction. For example, CERR should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 0b00. See Section 47.5.4.12.2, "Split Transaction Interrupt," for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See Section , "Asynchronous—Do Complete Split," for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the *PIDCode* indicates a SETUP. **Note:** Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.
9–8	PID Code. This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. 00 OUT Token generates token (E1H) 01 IN Token generates token (69H) 10 SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt the queue head is non-zero.) transfer type, e.g. μ*Frame S-mask* field in 11 Reserved
7–0	**Status.** This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. Table 47-59 shows the bit encodings.

Table 47-59. qTD Token (Dword 2) Status Field (bits 7–0) Bits Description

Status Field Bit	Description
7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
6	Halted. Set to a one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.

Table 47-59. qTD Token (Dword 2) Status Field (bits 7–0) Bits Description (continued)

5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller will force a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Since "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
2	Missed microframe. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. Bit encodings are: 0 Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1 Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: 0 Do OUT. This value directs the host controller to issue anOUT PID to the endpoint. 1 Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

47.5.3.5.4 qTD Buffer Page Pointer List

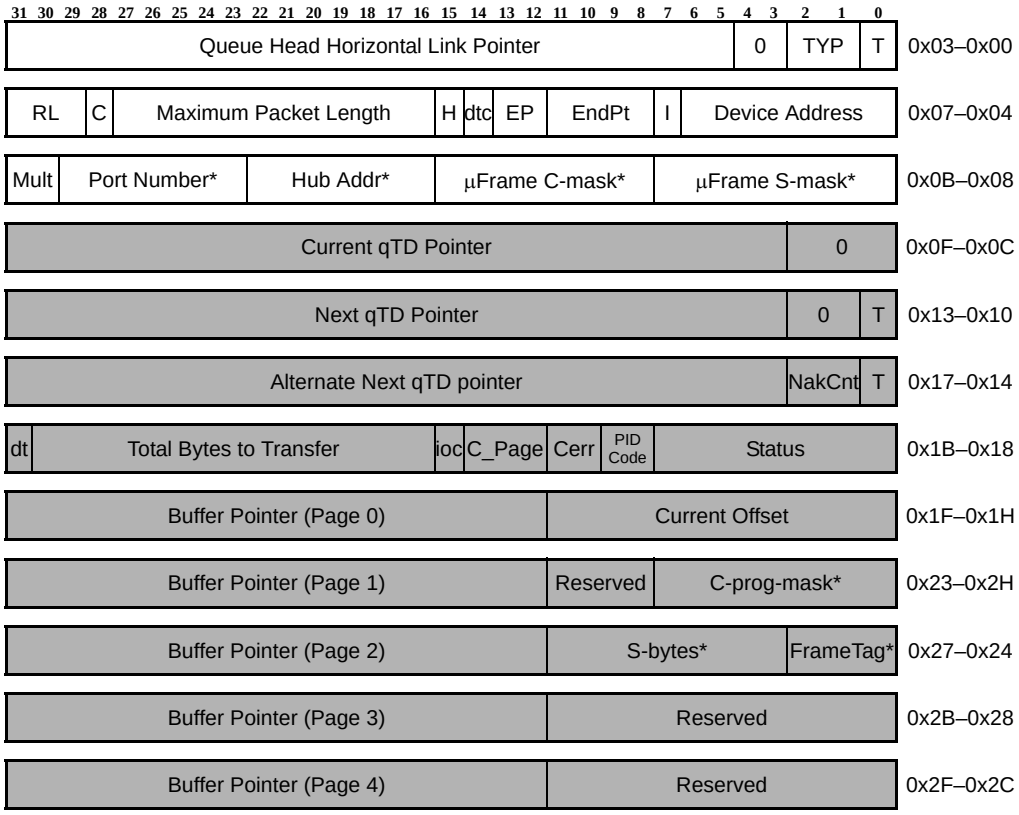
The last five dwords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes *Current Offset* field to the starting offset into the current page, where current page is selected using the value in the *C_Page* field.

Table 47-60. qTD Buffer Pointer(s) (Dwords 3-7)

Bits	Description
31–12	**Buffer Pointer List.** Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction using the new buffer pointer.
11–0	**Current Offset (Reserved).** This field is reserved in all pointers except the first one (e.g. Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zeros.

47.5.3.6 Queue Head



Transfer Overlay Transfer Results

Static Endpoint State

*These fields are used exclusively to support split transactions to USB 2.0 Hubs

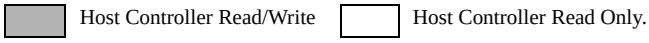


Figure 47-53. Queue Head Structure Layout

Queue Head Horizontal Link Pointer

The first dword of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

Table 47-61. Queue Head Dword 0

Bits	Description
31–5	**Queue Head Horizontal Link Pointer (QHLP).** This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4–3	**Reserved.** These bits must be written as zeros.
2–1	**QH/(s)iTD Select (TYP).** This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: ValueMeaning 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	**Terminate (T).** 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

47.5.3.6.1 Endpoint Capabilities/Characteristics

The second and third dwords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt using split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

Table 47-62. Endpoint Characteristics: Queue Head Dword 1

Bit	Description
31:28	**Nak Count Reload (RL).** This field contains a value, which is used by the host controller to reload the Nak Counter field.
27	**Control Endpoint Flag (C).** If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to a zero.
26:16	**Maximum Packet Length.** This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).
15	**Head of Reclamation List Flag (H).** This bit is set by System Software to mark a queue head as being the head of the reclamation list.
14	**Data Toggle Control (DTC).** This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0 Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1 Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.
13–12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are: 00 Full-speed (12 Mbps) 01 Low-speed (1.5 Mbps) 10 High-speed (480 Mbps) 11 Reserved. This field must not be modified by the host controller.
11:8	**Endpoint Number (Endpt).** This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	**Inactivate on Next Transaction (I).** This bit is used by system software to request that the host controller set the Active bit to zero. See Section , “Rebalancing the Periodic Schedule,” for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the EPS field indicates a Full or Low-speed endpoint. Setting this bit to a one when the queue head is in the Asynchronous Schedule or the EPS field indicates a high-speed device yields undefined results.
6:0	**Device Address.** This field selects the specific device serving as the data source or sink.

Table 47-63. Endpoint Capabilities: Queue Head Dword 2

Bit	Description
31:30	**High-Bandwidth Pipe Multiplier (Mult).** This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: ValueMeaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per microframe 10b Two transactions to be issued for this endpoint per microframe 11b Three transactions to be issued for this endpoint per microframe
29:23	**Port Number.** This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.

Table 47-63. Endpoint Capabilities: Queue Head Dword 2 (continued)

Bit	Description
22:16	**Hub Addr.** This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15:8	**Split Completion Mask (μFrame C-Mask).** This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which microframes the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the μ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7:0	**Interrupt Schedule Mask (μFrame S-mask).** This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the μ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

47.5.3.6.2 Transfer Overlay

The nine dwords in this area represent a *transaction working space* for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the *Queue Head Horizontal Link Pointer* to the next queue head. The host controller will never follow the *Next Transfer Queue Element or Alternate Queue Element* pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The dword3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 47-64. Current qTD Link Pointer

Bit	Description
31:5	**Current Element Transaction Descriptor Link Pointer.** This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4:0	**Reserved (R).** These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The dwords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an *overlay* because when the queue is advanced to the next queue element, the source queue element is *merged* onto this area. This area serves an execution cache for the transfer.

Table 47-65. Host-Controller Rules for Bits in Overlay (Dwords 5, 6, 8 and 9)

Dword	Bit	Description
5	4:1	**Nak Counter (NakCnt)μRW**. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	**Data Toggle.** The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	**Interrupt On Complete (IOC).** The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11:10	**Error Counter (C_ERR).** This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	**Ping State (P)/ERR.** If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7:0	**Split-transaction Complete-split Progress (C-prog-mask).** This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4:0	**Split-transaction Frame Tag (Frame Tag).** This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11:5	**S-bytes.** Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

47.5.3.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary. See section Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0x0096. FSTNs are not defined for implementations before 0x0096 and their use will yield undefined results.

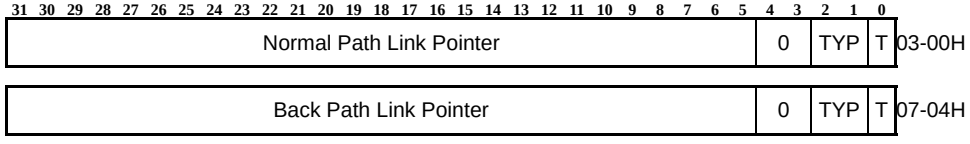


Figure 47-54. Frame Span Traversal Node Structure Layout

47.5.3.7.1 FSTN Normal Path Pointer

The first dword of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

Table 47-66. FSTN First Dword Fields Description

Bits	Description
31–5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4–3	Reserved. These bits must be written as 0s.
2–1	QH/(s)iTD/FSTN Select (TYP). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (Frame Span Traversal Node)
0	Terminate (T). 0 Link Pointer is valid. 1 Link Pointer field is not valid.

47.5.3.7.2 FSTN Back Path Link Pointer

The second dword of an FTSN node contains a link pointer to a queue head. If the T bit in this pointer is a zero, then this FSTN is a *Save-Place* indicator. Its TYP field must be set by software to indicate the target data structure is a queue head. If the T bit in this pointer is set to 1, then this FSTN is the *Restore* indicator. When the T bit is 1, the host controller ignores the TYP field.

Table 47-67. FSTN Second Dword Field Descriptions

Bits	Description
31:5	**Back Path Link Pointer (BPLP).** This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4:3	**Reserved.** These bits must be written as 0s.
2:1	**TYP.** Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	**Terminate (T).** 1=Link Pointer field is not valid (i.e. the host controller must not use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (i.e. the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

47.5.4 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software). Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

47.5.4.1 Host Controller Initialization

When the system boots, the host controller is enumerated, assigned a base address for the register space and BIOS sets the FLADJ register to a system-specific value. After initial power-on or software reset (by setting the RST bit in the USBCMD register, all of the operational registers are set at their default values, as illustrated in Table 47-68. After a hardware reset, only the operational registers not contained in the auxiliary power well will be at their default values.

Table 47-68. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USBCMD	0x0008_0B00[
USBSTS	0x0000_1000
USBINTR	0x0000_0000
FRINDEX	0x0000_0000
CTRLDSSEGMENT	0x0000_0000
PERIODICLISTBASE	Undefined
ASYNCLISTADDR	Undefined
CONFIGFLAG	0x0000_0001
PORTSC	0x0000_2000 (if PPC = 1, HCSPARAMS register) 0x0000_3000 (if PPC = 0, HCSPARAMS register)

In order to initialize the host controller, software should perform the following steps

- Program the CTRLDSSEGMENT register with 4-Gigabyte segment where all of the interface data structures are allocated.
- Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the Periodic Frame List should have their *T-Bits* set to a one.
- Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller *ON* using setting the *Run/Stop* bit.
- Write a 1 to CONFIGFLAG register to route all ports to the EHCI controller.

At this point, the host controller is up and running and the port registers will begin reporting device connects, etc. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled

High-speed ports, but the schedules have not yet been enabled. The EHCI Host controller will not transmit SOFs to enabled Full- or Low-speed ports. In order to communicate with devices using the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the *Asynchronous Schedule Enable* bit in the USBCMD register. In order to communicate with devices using the periodic schedule, system software must enable the periodic schedule by writing a one to the *Periodic Schedule Enable* bit in the USBCMD register. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

47.5.4.2 Port Routing and Control

A USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers. Companion host controllers (cHCs) may be implementations of either universal or open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; e.g. Low-, Full-, and High-speed capability for every port. Figure 47-55 illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

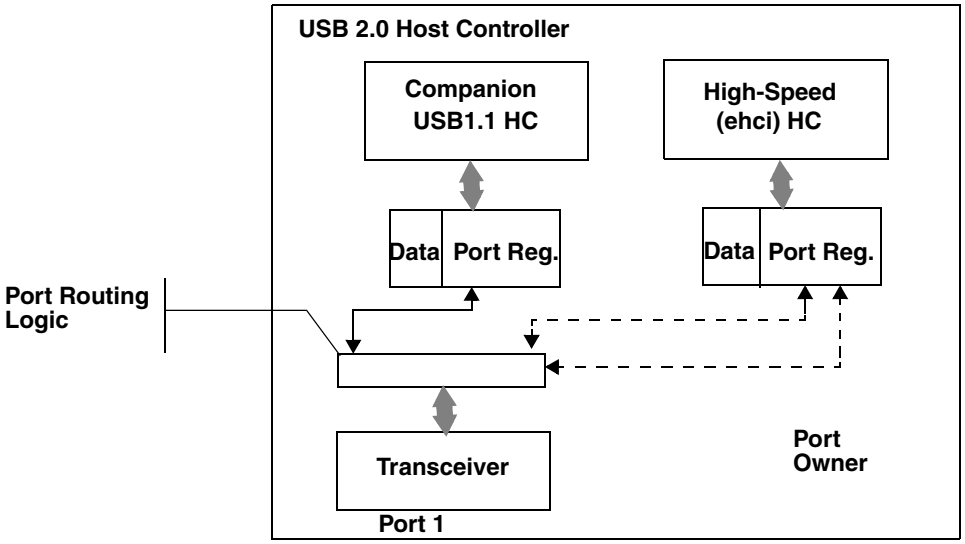


Figure 47-55. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller module has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Each transceiver can be controlled by either the EHCI host controller or one companion host controller. Routing logic lies between the transceiver and the port status and control registers.

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a *global* routing policy control field and per-port *ownership* control fields. The *Configured*

Flag (*CF*) bit (defined in Section 47.5.2.6.8, “Host Controller Embedded TT Asynchronous Buffer Status Register (BURSTSIZE)”) is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes companion controllers, then the ports will still work in full- and low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers’ function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (i.e. no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The *N_CC* field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When *N_CC* has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports will always fail the high-speed chirp during reset and the ports will not be enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Section 47.5.2.4.3, “Host Control Structural Parameters Register (HCSPARAMS)”.

47.5.4.2.1 Port Routing Control using EHCI *Configured* (*CF*) Bit

Each port in the USB 2.0 host controller can be routed either to a single companion host controller or to the EHCI host controller. The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The *Configured Flag* (*CF*) bit (defined in Section 47.5.2.6.8, “Host Controller Embedded TT Asynchronous Buffer Status Register (BURSTSIZE)”), is used to globally set the policy of the routing logic. Each port register has a *Port Owner* control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the *CF bit* transitions from a zero to a one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all *Port Owner* bits go to zero). While the *CF-bit* is a one, the EHCI Driver can control individual ports' routing using the *Port Owner* control bit. Likewise, whenever the *CF bit* transitions from a one to a zero (as a result of AUX power application, setting the RST bit in the USB command register, or software writing a zero to CF-bit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC’s *CF bit* (after AUX power application or setting RST to 1) is zero. Table 47-69 summarizes the default routing for all the ports, based on the value of the EHCI HC’s CF bit.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

Table 47-69. Default Port Routing Depending on EHCI HC CF Bit

HC CF Bit Setting	Default Port Ownership	Explanation
0	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Section 47.5.2.6.12, "Port Status Control x Registers (PORTSCx, x = 1…8)"). The EHCI HC can temporarily release control of the port to a companion HC by setting the PortOwner bit in the PORTSC register to a one.

47.5.4.2.2 Port Routing Control using *PortOwner* and Disconnect Event

Manipulating the port routing using the CF-bit is an extreme process and not intended to be used during normal operation. The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's PORTSC register (for handoffs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI CF-bit is set to a one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.
- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the *LineStatus* bits in the PORTSC register. If they indicate the attached device is a full-speed device (e.g. D+ is asserted), then the EHCI Driver sets the *PortReset* control bit to a one (and sets the *PortEnable* bit to a zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing a zero to the port reset bit. The reset process is actually complete when software reads a zero in the *PortReset* bit. The EHCI Driver checks the PortOwner bit in the PORTSC register. If set to a one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the *LineStatus* bits might indicate a low-speed device. Additionally, when the port reset process is complete, the *PortEnable* field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the PORTSC register to a one to release port ownership to a companion host controller.

- When the EHCI Driver sets *PortOwner* bit to a one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI PORTSC register observes and reports a disconnect event using the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to a one, a connect change set to a one and a connect status set to a zero. This information is derived directly from the EHCI port register. This will allow the hub driver to assume the device was disconnected during reset. It will acknowledge the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's *CF-bit* transitions from a 1b to a 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects will be detected by the EHCI port register and the process will repeat.

47.5.4.2.3 Example Port Routing State Machine

Figure 47-56 illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

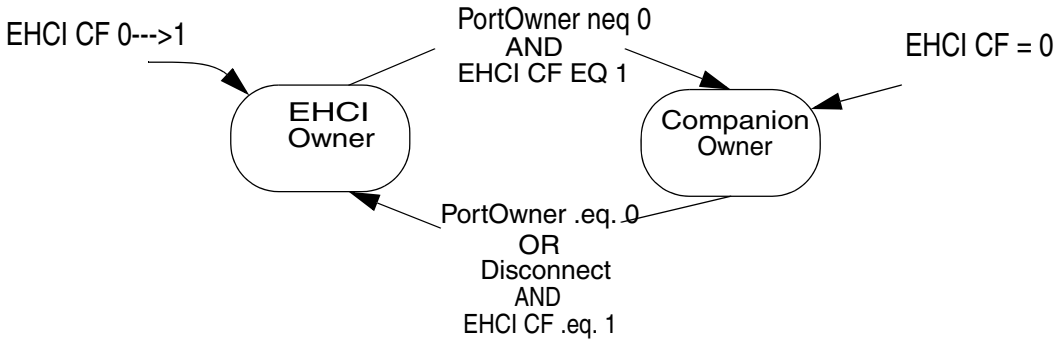


Figure 47-56. Port Owner Handoff State Machine

EHCI HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the EHCI HC's *Configure Flag* (*CF*) bit in the CONFIGFLAG register transitions from a zero to a one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.

- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver will acknowledge the disconnect by setting the connect status change bit to a zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes a zero to the *PortOwner* bit in the PORTSC register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the *Port Owner* field transitions from a zero to a one.
- When the HS-mode HC's *Configure Flag* (*CF*) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

47.5.4.2.4 Port Power

The Port Power Control (PPC) bit in the HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (See Section 47.5.2.4.3, "Host Control Structural Parameters Register (HCSPARAMS)"). When this bit is a zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the PPC bit is set to one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is denoted as Port Power Enable (PPE). PPE is controlled based on the states of the PPC bit, EHCI Configured (CF) bit and individual Port Power (PP) bits. Table 47-70 summarizes the PPE behavior.

Table 47-70. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	*n*	CHC	0	When the EHCI controller has not been configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	*n*	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.

Table 47-70. Port Power Enable Control Rules (continued)

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1 CHC (Companion Host Controller)

2 EHC (EHCI Host Controller)

3 PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists)

47.5.4.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- Over-current Active bits are set to one. When the over-current condition goes away, the Over-current Active bit transitions from 1 to 0.
- Over-current Change bits are set to1. On every transition of the Over-current Active bit the host controller will set the Over-current Change bit to 1. Software clears the Over-current Change bit by writing 1 to this bit.
- Port Enabled/Disabled bit is cleared. When this change bit gets set to 1, then the Port Change Detect bit in the USBSTS register is set to1.
- Port Power (PP) bits may optionally be cleared. There is no requirement in USB that a power provider shut off power in an over-current condition. It is sufficient to limit the current and leave power applied. When the Over-current Change bit transitions from 0 to 1, the host controller also sets the Port Change Detect bit in the USBSTS register to 1. In addition, if the Port Change Interrupt Enable bit in the USBINTR register is set to 1, then the host controller issues an interrupt to the system. See Table 47-71 for summary behavior for over-current detection when the host controller is halted (suspended from a device component point of view).

47.5.4.3 Suspend/Resume

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely using software initiation. Other control mechanisms are provided to parameterize the host controller's response (or

sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wake-up events are:

- Remote-wake-up enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the PORTSC registers.

Selective suspend is a feature supported by every PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the *Run/Stop* bit in the USBCMD register to a zero. The EHCI module can then be placed into a lower device state using the PCI power management interface (See Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs the system will resume operation and system software will eventually set the *Run/Stop* bit to a one and resume the suspended ports. Software must not set the *Run/Stop* bit to a one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the CPU is restarted. So, by definition, if software is running, clocks in the system are stable and the *Run/Stop* bit in the USBCMD register can be set to a one. There are also minimum system software delays defined in the PCI Power Management Specification. See this specification for more information.

47.5.4.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing a one into the appropriate PORTSC *Suspend* bit. Software must only set the *Suspend* bit when the port is in the enabled state (*Port Enabled* bit is a one) and the EHCI is the port owner (*Port Owner* bit is a zero).

The host controller may evaluate the *Suspend* bit immediately or wait until a microframe or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several microframes of activity on the port until the host controller evaluates the *Suspend* bit. The host controller must evaluate the *Suspend* bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing a one to the *Force Port Resume* bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Section 47.5.2.6.12, “Port Status Control x Registers (PORTSCx, x = 1…8)”). If system software sets *Force Port Resume* bit to a one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (*Suspend* bit is a one) before initiating a port resume using the *Force Port Resume* bit. When *Force Port Resume* bit is a one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then sets the *Force Port Resume* bit to a zero. When the host controller receives the write to transition *Force Port*

Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the *Force Port Resume* and *Suspend* bits to zero. Software-initiated port resumes do not affect the *Port Change Detect* bit in the USBSTS register nor do they cause an interrupt if the *Port Change Interrupt Enable* bit in the USBINTR register is a one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's *Force Port Resume* bit is set to a one and the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one the host controller will issue a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 msec), then terminates the resume sequence by writing zero to the *Force Port Resume* bit in the port. The host controller receives the write of zero to *Force Port Resume*, terminates the resume sequence and sets *Force Port Resume* and *Suspend* port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the *Suspend* and *Force Port Resume* bits are zero. Software must ensure that the host controller is running (i.e. *HCHalted* bit in the USBSTS register is a zero), before terminating a resume by writing a zero to a port's *Force Port Resume* bit. If *HCHalted* is a one when *Force Port Resume* is set to a zero, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

Table 47-71 summarizes the wake-up events. Whenever a resume event is detected, the *Port Change Detect* bit in the USBSTS register is set to a one. If the *Port Change Interrupt Enable* bit is a one in the USBINTR register, the host controller will also generate an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the *Port Change Detect* status bit in the USBSTS register.

Table 47-71. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in PORTSC register is set to a one. Port Change Detect bit in USBSTS register set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a one. A disconnect is detected.	Depending in the initial port state, the PORTSC Connected Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a zero. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected.	PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]

Table 47-71. Behavior During Wake-up Events (continued)

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port is connected and the port's WKOC_E bit is a one. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is a zero. An over-current condition occurs.	PORTSC Over-current Active, Over-current Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USBINTR register is a one.
[2] PME# asserted if enabled (Note: PME Status must always be set to a one).
[3] PME# not asserted.

47.5.4.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the PERIODICLISTBASE register (see Section 47.5.2.6.6, “Host Controller Frame List Base Address Register (PERIODICLISTBASE) and USB Device Address Register (DEVICEADDR)”). The PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structure. In each microframe, if the periodic schedule is enabled (see Section , “Periodic Scheduling Threshold”) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the PERIODICLISTBASE and the FRINDEX registers (see Figure 47-57). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a *next* link pointer of a schedule data structure having its *T-bit* set to a one. When the host controller encounters a *T-Bit* set to a one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the microframe.

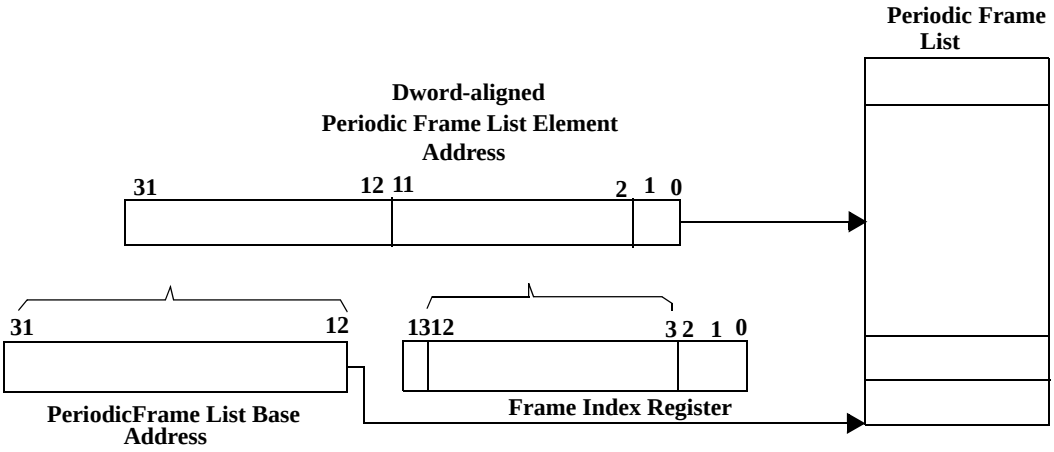


Figure 47-57. Derivation of Pointer into Frame List Array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register *ASYNCLISTADDR* to access the asynchronous schedule, see Figure 47-58.

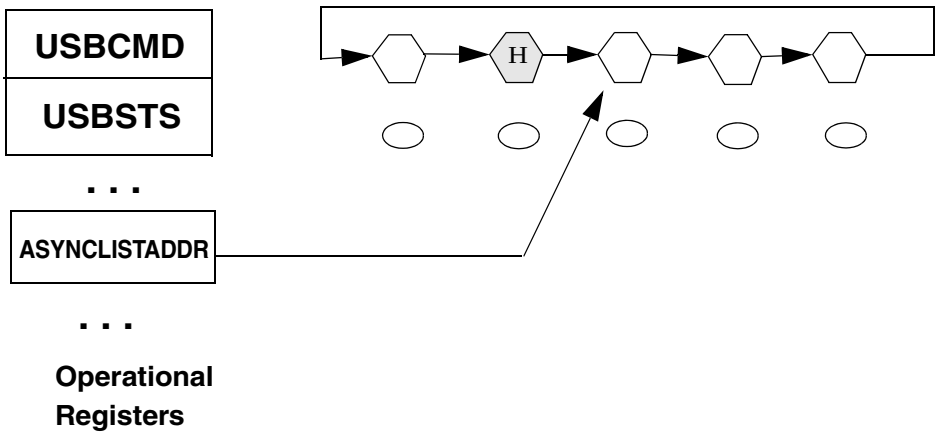


Figure 47-58. General Format of Asynchronous Schedule List

The *ASYNCLISTADDR* register contains a physical memory pointer to the *next* queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the *ASYNCLISTADDR* register. Software must set queue head *horizontal* pointer *T-bits* to a zero for queue heads in the asynchronous schedule. See Section 47.5.4.8, “Asynchronous Schedule” for complete operational details.

47.5.4.4.1 Example—Preserving Microframe Integrity

One of the requirements of a USB host controller is to maintain *Frame Integrity*. This means that the HC must preserve the microframe boundaries. For example: SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of microframe timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One

implication of this responsibility is that the HC must ensure that it does not start transactions that will not be completed before the end of the microframe. More precisely, no transactions should be started by the host controller, which cannot be completed in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it will complete before the end of the microframe.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction will take. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch *all* of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers could allow the host controller to know exactly whether there was enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the microframe. It is a reasonable assumption that software will never over-commit the microframe to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction will not be executed that *could* have been executed. However, under all circumstances, a transaction will never be started unless there is enough time in the frame to complete the transaction.

Transaction Fit—A Best-Fit Approximation Algorithm

A curve is calculated which represents the *latest* start time for every packet size, at which software will schedule the start of a periodic transaction. This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the microframe. A plot of these two curves is illustrated in Figure 47-59. The plot Y-axis represents the number of byte-times left in a frame.

The space between the *80%* and the *Last Start* plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and *Last Start* curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land *above* the function curve. The host controller will not start transactions whose results land below the function curve.

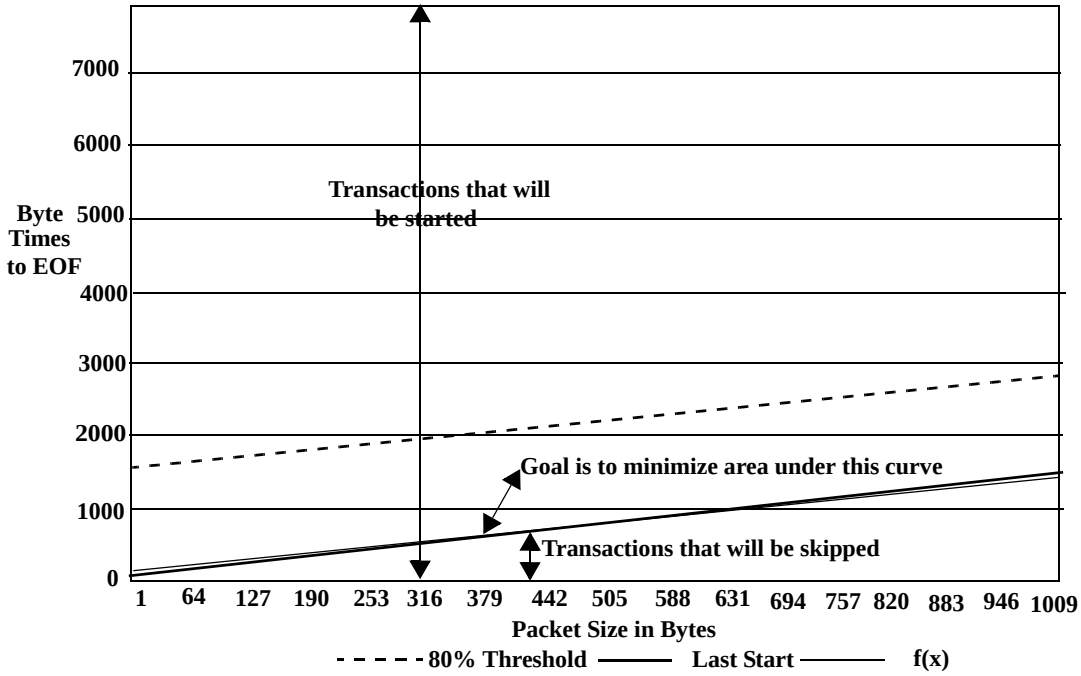


Figure 47-59. Best Fit Approximation

The *LastStart* line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used was a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in Table 47-72. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 47-72. Example Worse-Case Transaction Timing Components

Component	Bit time	Byte Time	Explanation
Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop….
Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop…
Host IPG	88	11	Same as above
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop…
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop…
		144	**Total**

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the *Last Start* curve, without dipping below the *LastStart* line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 47-59 was constructed using the

following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Algorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
        Local Temp = MaximumPacketSize + 192
        Local rvalue = TRUE

        If MaximumPacketSize >= 128 then
                Temp += 128
        End If

        If Temp > HC_BytesLeftInFrame then
                Rvalue = FALSE
        End If
        Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and a hardware counter of the number of bytes left in the current microframe. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the *f*(*x*) plot was getting *close* to the *LastStart* line.

47.5.4.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned. Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions using a microframe pipeline (see start- (SS) and complete- (CS) splits illustrated in Figure 47-60). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

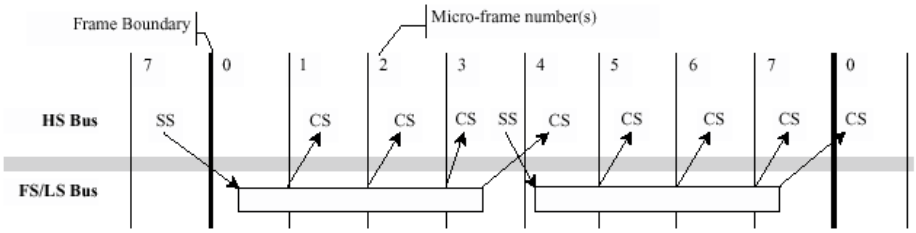


Figure 47-60. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as Figure 47-61 illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement a one microframe phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed using the Frame List Index Register (FRINDEX) documented in Section 47.5.2.6.4, “USB Frame Index Register (FRINDEX),” and initially illustrated in Section 47.5.4.4, “Schedule Traversal Rules.” Bits FRINDEX[2:0], represent the microframe number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one microframe. The one microframe delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 47-61. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

Figure 47-61 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called *H-Frames*. The high-speed bus's view of the 1-millisecond boundaries is called *B-Frames*.

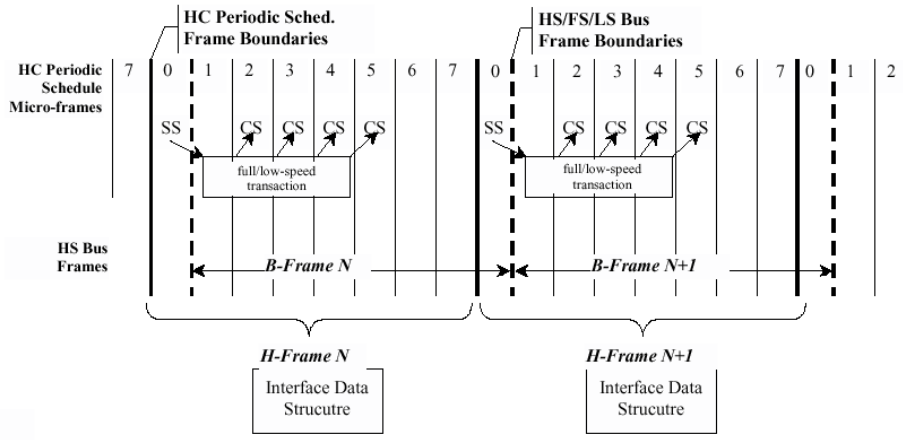


Figure 47-61. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Microframe numbers for the *H-Frame* are tracked by FRINDEX[2:0]. *B-Frame* boundaries are visible on the high-speed bus using changes in the SOF token's frame number. Microframe numbers on the high-speed bus are only derived from the SOF token's frame number (i.e. the high-speed bus will see eight SOFs with the same frame number value). *H-Frames* and *B-Frames* have the fixed relationship (i.e. *B-Frames* lag *H-Frames* by one microframe time) illustrated in Figure 47-61 . The host controller's periodic schedule is naturally aligned to *H-Frames*. Software schedules transactions for full- and low-speed periodic endpoints relative the *H-Frames*. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in Section 47.5.2.6.4, "USB Frame Index Register (FRINDEX)," the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one microframe count. Table 47-73 illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. Section 47.5.2.6.4, "USB Frame Index Register (FRINDEX)," provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 47-73. Operation of FRINDEX and SOFV (SOF Value Register)

Current			Next		
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

Where [F] = [13:3]; [μF] = [2:0]

47.5.4.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled using the *Periodic Schedule Enable* bit in the USBCMD register. If the *Periodic Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the periodic frame list using the *PERIODICLISTBASE* register. Likewise, when the *Periodic Schedule Enable* bit is a one, then the host controller does use the *PERIODICLISTBASE* register to traverse the periodic schedule. The host controller will not react to modifications to the *Periodic Schedule Enable* immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the *Periodic Schedule Enable* bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b microframe. These work items must be removed from the schedule before the *Periodic Schedule*

Enable bit is written to a zero. The *Periodic Schedule Status* bit in the USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing a one (or zero) to the *Periodic Schedule Enable* bit in the USBCMD register. Software then can poll the *Periodic Schedule Status* bit to determine when the periodic schedule has made the desired transition. Software must not modify the *Periodic Schedule Enable* bit unless the value of the *Periodic Schedule Enable* bit equals that of the *Periodic Schedule Status* bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. Figure 47-62 illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (e.g. closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

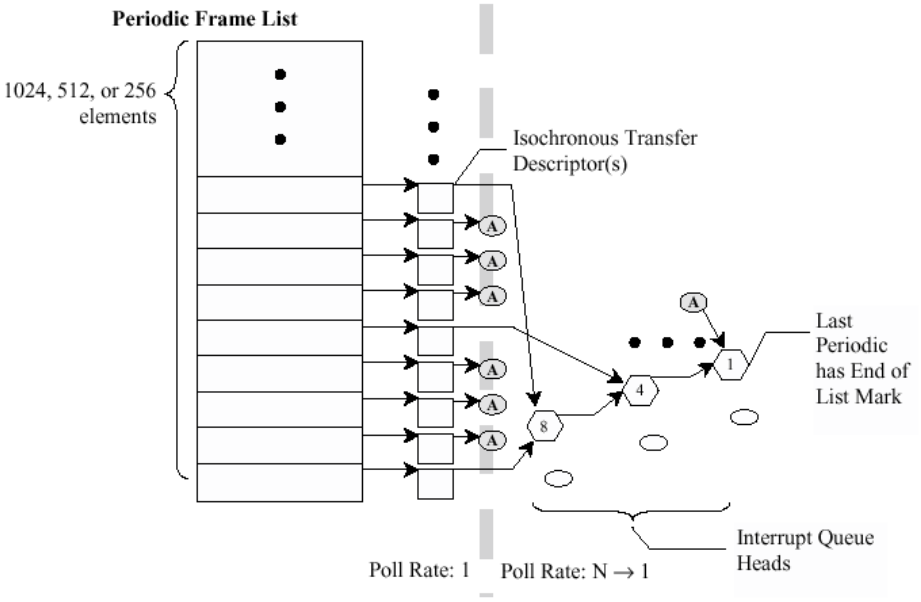


Figure 47-62. Example Periodic Schedule

47.5.4.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). There are four distinct sections to an iTD:

- The first field is the *Next Link Pointer.* This field is for schedule linkage purposes only;
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one microframe's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.

- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

47.5.4.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Each iTD can span 8 microframes worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array. If the *active* bit in the *Status* field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the *Next* pointer to the next schedule data structure.

When the indexed *active* bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, etc.). It also uses the *Page Select (PG)* field to index the *buffer pointer* array, storing the selected *buffer pointer* and the next sequential *buffer pointer.* For example, if *PG* field is a 0, then the host controller will store Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's *PG* field) and the transaction description's *Transaction Offset* field. The host controller uses the endpoint addressing information and *I/O-bit* to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the *Status* field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet’s offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions’ *PG* (example value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When this occurs the host controller simply replaces the current buffer pointer’s page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the *Maximum Packet Size* field. An iTD supports high-bandwidth pipes using the *Mult* (multiplier) field. When the *Mult* field is 1, 2, or 3, the host controller executes the specified number of *Maximum Packet* sized bus transactions for the endpoint in the current microframe. In other words, the *Mult* field represents a transaction count for the endpoint in the current microframe. If the *Mult* field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the *Mult* field.

For OUT transfers, the value of the *Transaction X Length* field represents the total bytes to be sent during the microframe. The *Mult* field must be set by software to be consistent with *Transaction X Length* and *Maximum Packet Sixe.* The host controller will send the bytes in *Maximum Packet Size*'d portions. After each transaction, the host controller decrements its local copy of *Transaction X Length* by *Maximum Packet Size*. The number of bytes the host controller sends is always *Maximum Packet Size* or *Transaction X Length*, whichever is less. The host controller advances the transfer state in the transfer description,

updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues *Mult* transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use *Maximum Packet Size* to detect packet babble errors. The host controller keeps the sum of bytes received in the *Transaction X Length* field. After all transactions for the endpoint have completed for the microframe, *Transaction X Length* contains the total bytes received. If the final value of *Transaction X Length* is less than the value of *Maximum Packet Size*, then less data than was allowed for was received from the associated endpoint. This *short packet* condition does not set the *USBINT* bit in the USBSTS register to a one. The host controller will not detect this condition. If the device sends more than *Transaction X Length* or *Maximum Packet Size* bytes (whichever is less), then the host controller will set the *Babble Detected* bit to a one and set the *Active* bit to a zero. Note, that the host controller is not required to update the iTD field *Transaction X Length* in this error scenario. If the *Mult* field is greater than one, then the host controller will automatically execute the value of *Mult* transactions. The host controller will not execute all *Mult* transactions if:

- The endpoint is an OUT and *Transaction X Length* goes to zero before all the *Mult* transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before *Mult* transactions have been executed. The end of microframe may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next microframe.

47.5.4.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N microframes. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure 47-63 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (i.e. the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one microframe's worth of transactions. The EHCI controller does not provide per-transaction results within a microframe. It treats the per-microframe transactions as a single logical transfer. On the left is the host controller’s frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

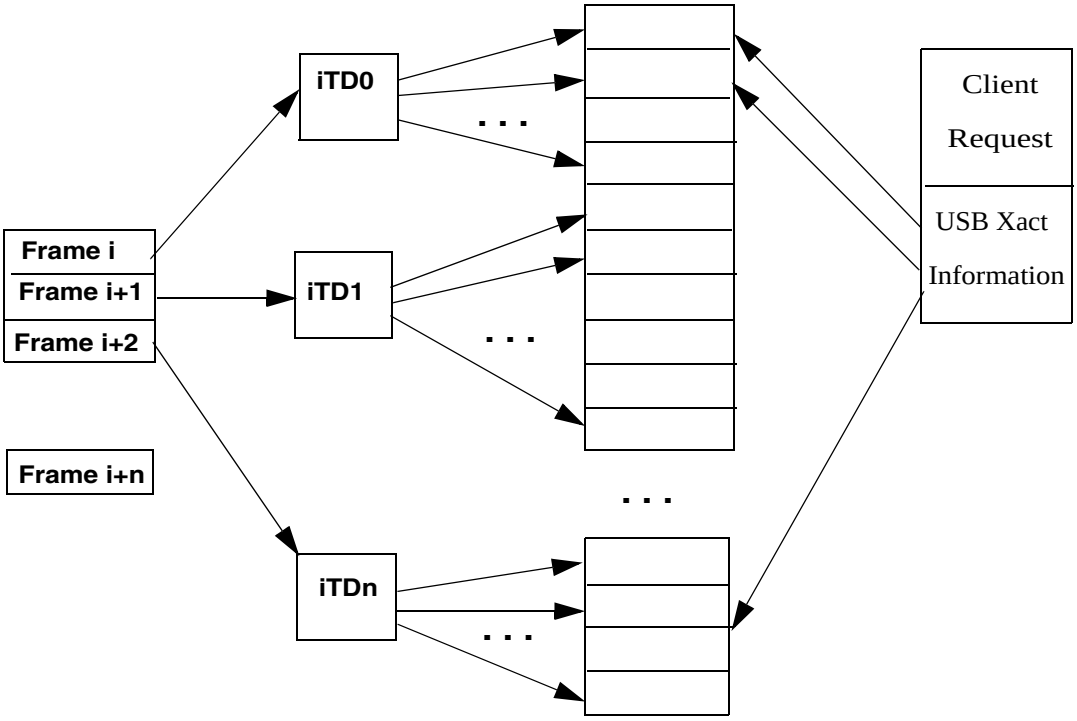


Figure 47-63. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the *PG* field is set to index the correct physical buffer page pointer and the *Transaction Offset* field is set relative to the correct buffer pointer page (e.g. the same one referenced by the *PG* field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current *Page Buffer Pointer* (as selected by the *PG* field) and concatenates to the *transaction offset* field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available *Page Buffer Pointer*. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so will yield undefined behavior. The host controller hardware is not required to 'alias' the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

Periodic Scheduling Threshold

The *Isochronous Scheduling Threshold* field in the HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures. It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the

current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 microframes worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 microframes. The three caching models are: no caching, microframe caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and microframe the host controller is currently executing. Of course, there is no information about where in the microframe the host controller is, so a constant uncertainty-factor of one microframe has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the *Isochronous Scheduling Threshold* field. The host controller may pre-fetch data structures during a periodic schedule traversal (per microframe) but will always dump any accumulated schedule state at the end of the microframe. At the appropriate time relative to the beginning of every microframe, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 microframes in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the *Isochronous Scheduling Threshold* field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 microframes). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the *current* microframe/frame (assume modulo 8 arithmetic in adding the constant 1 to the microframe number). For any current frame N, if the current microframe is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current microframe is 7, then software can add isochronous transactions to Frame N + 2.

Microframe caching is indicated with a non-zero value in the least-significant 3 bits of the *Isochronous Scheduling Threshold* field. System software assumes the host controller caches one or more periodic data structures for the number of microframes indicated in the *Isochronous Scheduling Threshold* field. For example, if the count value were 2, then the host controller keeps a *window* of 2 microframes worth of state (current microframe, plus the next) on-chip. On each microframe boundary, the host controller releases the current microframe state and begins accumulating the next microframe state.

47.5.4.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled using the *Asynchronous Schedule Enable* bit in the USBCMD register. If the *Asynchronous Schedule Enable* bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule using the *ASYNCLISTADDR* register. Likewise, when the *Asynchronous Schedule Enable* bit is a one, then the host controller does use the *ASYNCLISTADDR* register to traverse the asynchronous schedule. Modifications to the *Asynchronous Schedule Enable* bit are not necessarily immediate. Rather the new value of the bit will only be taken into

consideration the next time the host controller needs to use the value of the *ASYNCLISTADDR* register to get the next queue head.

The *Asynchronous Schedule Status* bit in the USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the *Asynchronous Schedule Enable* bit in the USBCMD register. Software then can poll the *Asynchronous Schedule Status* bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the *Asynchronous Schedule Enable* bit unless the value of the *Asynchronous Schedule Enable* bit equals that of the *Asynchronous Schedule Status* bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the *ASYNCLISTADDR* register. The default value of the *ASYNCLISTADDR* register after reset is undefined and the schedule is disabled when the *Asynchronous Schedule Enable* bit is a zero.

Software may only write this register with defined results when the schedule is disabled. e.g. *Asynchronous Schedule Enable* bit in the USBCMD and the *Asynchronous Schedule Status* bit in the USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit is set to one. The asynchronous schedule is actually enabled when the *Asynchronous Schedule Status* bit is a one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the *ASYNCLISTADDR* register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller "completes" processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the *ASYNCLISTADDR* register. Next time the asynchronous schedule is accessed, this is the first data structure that will be serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller "completes" processing the asynchronous schedule when one of the following events occur:

- The end of a microframe occurs.
- The host controller detects an empty list condition (see Section 47.5.4.8.3, “Empty Asynchronous Schedule Detection”)
- The schedule has been disabled using the *Asynchronous Schedule Enable* bit in the USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 47-58. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

47.5.4.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the *ASYNCLISTADDR* register, then enables the list by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example qTD pointers have *T-Bit*s set to a one or reference valid qTDs and the *Horizontal Pointer* references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)

        --
        -- Requirement: all inputs must be properly initialized.
        --
        -- pQHeadCurrent is a pointer to a queue head that is
        -- already in the active list
        -- pQHeadNew is a pointer to the queue head to be added
        --
        -- This algorithm links a new queue head into a existing
        -- list
        --
        pQueueHeadNew.HorizontalPointer = pQueueHeadCurrent.HorizontalPointer


        pQueueHeadCurrent.HorizontalPointer = physicalAddressOf(pQueueHeadNew)


End InsertQueueHead
```

47.5.4.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the *Asynchronous Schedule Enable* bit in the USBCMD register to a zero. Software can determine when the list is idle when the *Asynchronous Schedule Status* bit in the USBSTS register is a zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first

deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list using the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)

        --

        -- Requirement: all inputs must be properly initialized.

        --

        -- pQHeadPrevious is a pointer to a queue head that

        -- references the queue head to remove

        -- pQHeadToUnlink is a pointer to the queue head to be

        -- removed

        -- pQheadNext is a pointer to a queue head still in the

        -- schedule. Software provides this pointer with the

        -- following strict rules:

        --      if the host software is one queue head, then

        --      pQHeadNext must be the same as

        --      QueueheadToUnlink.HorizontalPointer. If the host

        --      software is unlinking a consecutive series of

        --      queue heads, QHeadNext must be set by software to

        --      the  queue head remaining in the schedule.

        --

        -- This algorithm unlinks a queue head from a circular list

        --

        pQueueHeadPrevious.HorizontalPointer = pQueueHeadToUnlink.HorizontalPointer
pQueueHeadToUnlink.HorizontalPointer = pQHeadNext

End UnlinkQueueHead
```

If software removes the queue head with the *H-bit* set to a one, it must select another queue head still linked into the schedule and set its *H-bit* to a one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its *H-bit* set to a one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, etc.). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (*Interrupt on Async Advance Doorbell* bit in the USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (*Interrupt on Async Advance* bit in the USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to a one, it also sets the command bit to a zero. The third bit is an interrupt enable (*Interrupt on Async Advance* bit in the USBINTR register) that is matched with the status bit. If the status bit is a one and the interrupt enable bit is a one, then the host controller will assert a hardware interrupt.

Figure 47-64 illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set to a one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (i.e. traversed beyond queue head (B) in this example).

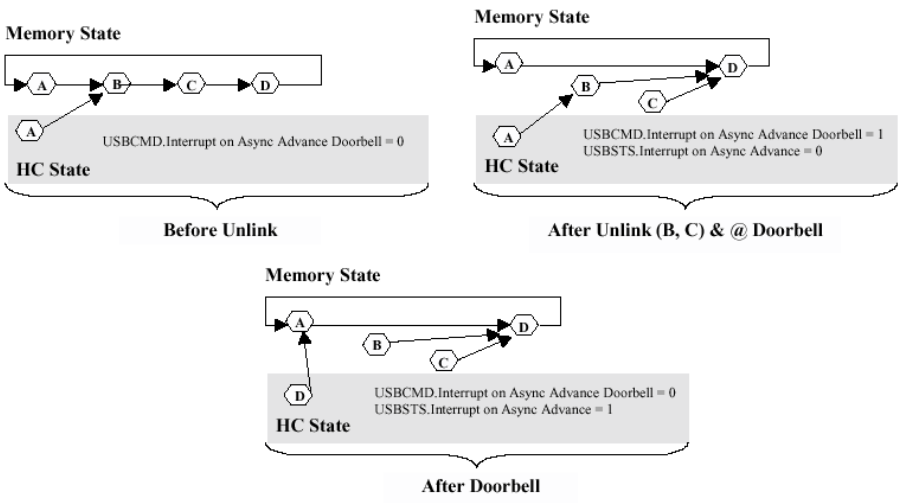


Figure 47-64. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (e.g. observed the head of the queue (twice)) before setting the *Advance on Async* status bit to a one.

Software may re-use the memory associated with the removed queue heads after it observes the *Interrupt on Async Advance* status bit is set to a one, following assertion of the doorbell. Software should acknowledge the *Interrupt on Async Advance* status as indicated in the USBSTS register, before using the doorbell handshake again.

47.5.4.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see Figure 47-53) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USBSTS register (*Reclamation*) that is set to a zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to a one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts: see Section 47.5.4.8.5, “Asynchronous Schedule Traversal—Start Event”).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is illustrated in Figure 47-65.

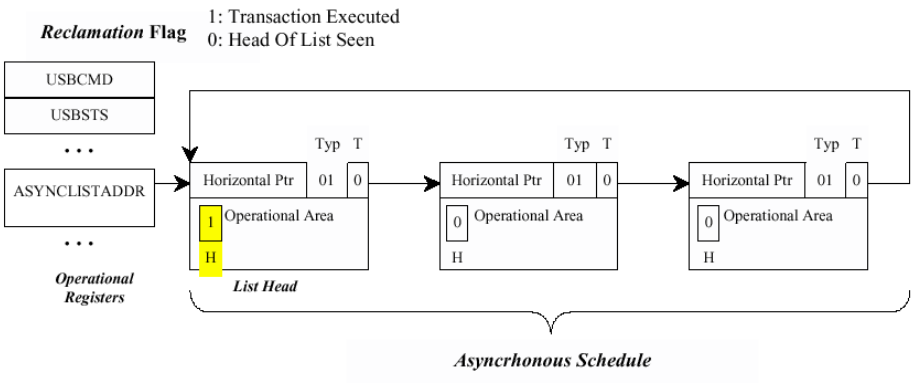


Figure 47-65. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to a one, and that it is always coherent with respect to the schedule.

47.5.4.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the microframe. It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the microframe. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller will cease traversal of the Asynchronous schedule very early in the microframe. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current microframe, instead of waiting until the next microframe. When the reason for host controller idling asynchronous

schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10µsec. The value is derived based on the analysis in Example Derivation for AsyncSchedSleepTime. The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-Microframe event occurs, or an empty list is detected. If the event is an End-of-Microframe, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, etc. so the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each microframe. Figure 47-66 illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

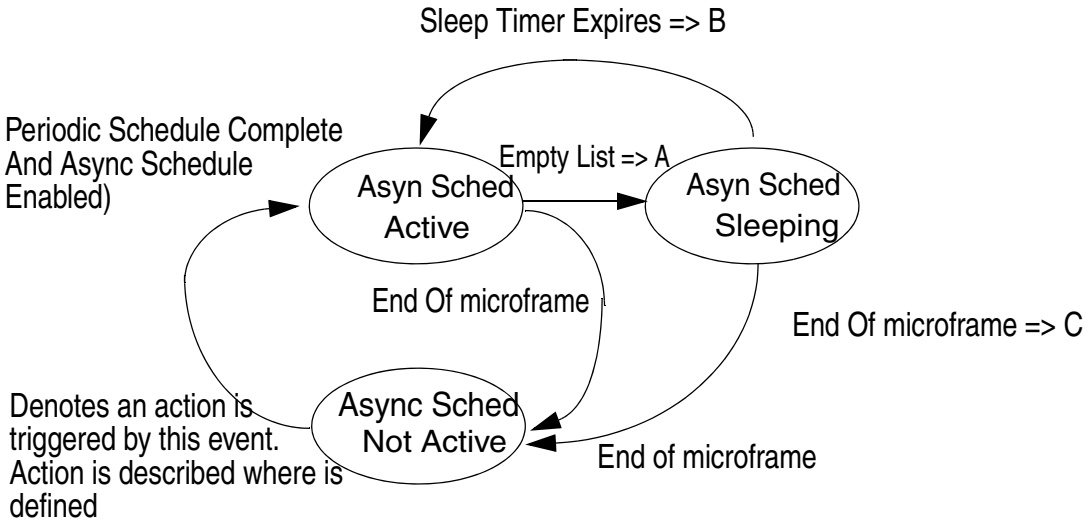


Figure 47-66. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in Figure 47-66 are defined in Table 47-74.

Table 47-74. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to a one and moves the Nak Counter reload state machine to WaitForListHead (see Section 47.5.4.9.1, “Nak Count Reload Control”).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine will not leave this state when the *Asynchronous Schedule Enable* bit in the USBCMD register is a zero.

This state is entered from **Async Sched Active** or **Async Sched Sleeping** states when the end-of-microframe event is detected.

Async Sched Active

This state is entered from the **Async Sched Not Active** state when the periodic schedule is not active. It is also entered from the **Async Sched Sleeping** states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USBSTS register to a one.

While in this state, the host controller will continually traverse the asynchronous schedule until either the end of microframe or an empty list condition is detected.

Async Sched Sleeping

The state is entered from the **Async Sched Active** state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next. It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests. Table 47-75 summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example, *footprint*, or *wall clock*) the transaction will take to complete.

Table 47-75. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk	—	—
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64	—	—
Type	Bulk	—	—
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (i.e. setup)
Size	8	—	—
Type	Cntrl	—	—

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller will get the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10µs sleep period would allow the host controller to get the NAK results on the first complete-split.

47.5.4.8.5 Asynchronous Schedule Traversal—*Start Event*

Once the HC has *idled* itself using the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each microframe. In addition, it may have idled itself early in a microframe. When this occurs (idles early in the microframe) the HC must occasionally *re-activate* during the microframe and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Section 47.5.4.8.4, “Restarting Asynchronous Schedule Before EOF.” Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the microframe is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Section 47.5.4.8.4, “Restarting Asynchronous Schedule Before EOF”).

47.5.4.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (Section 47.5.4.8.3, “Empty Asynchronous Schedule Detection”) depends on the proper management of the *Reclamation* bit in the USBSTS register. The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (See Section 47.5.4.10.1, “Fetch Queue Head”).

It is required that the host controller sets the *Reclamation* bit to a one whenever an asynchronous schedule traversal *Start Event* occurs (as documented in Section 47.5.4.8.5, “Asynchronous Schedule Traversal—Start Event”). The *Reclamation* bit is also set to a one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Section 47.5.4.10.3, “Execute Transaction”). The host controller sets the *Reclamation* bit to a zero whenever it finds a queue head with its *H-bit* set to a one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to a one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to a zero when executing from the periodic schedule.

47.5.4.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head (see Section 47.5.7.4.1, “Queue Head Initialization”). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control using the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced using the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (i.e. like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

There are two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- **Not Used.** This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- **Nak Throttle Mode.** This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller will tolerate on each endpoint. In this mode, the HC will decrement the *NakCnt* field based on the token/handshake criteria listed in Table 47-76 The host controller must reload *NakCnt* when the endpoint successfully moves data (e.g. policy to reward device for moving data).

Table 47-76. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1]Start

Table 47-76. NakCnt Field Adjustment Rules (continued)

Token	Handshake	
	Handshake	NYET
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

[1] Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller will not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Section 47.5.4.9.1, "Nak Count Reload Control."

Note that when all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller will detect an empty Asynchronous Schedule and idle schedule traversal (see Section 47.5.4.8.5, "Asynchronous Schedule Traversal—Start Event").

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Section 47.5.4.8.5, "Asynchronous Schedule Traversal—Start Event." Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

47.5.4.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Section 47.5.4.10.3, "Execute Transaction"). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Figure 47-53) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section 47.5.4.8.5, "Asynchronous Schedule Traversal—Start Event," for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 47-65). Figure 47-67 illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: **Execute Transaction** (Figure 47-67). The host controller does not perform the nak counter reload operation if the RL field (see Figure 47-53) is set to zero.

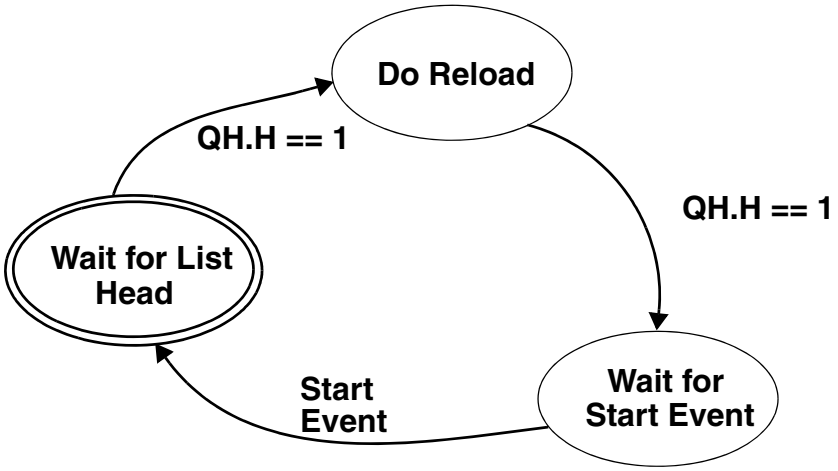


Figure 47-67. Example HC State Machine for Controlling Nak Counter Reloads

Wait for List Head

This is the initial state. The state machine enters this state from **Wait for Start Event** when a start event (as defined in Section 47.5.4.8.5, “Asynchronous Schedule Traversal—Start Event”) occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose *H-bit* is set to a one.

Do Reload

This state is entered from the **Wait for List Head** state when the host controller fetches a queue head with the *H-bit* set to a one. While in this state, the host controller will perform nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to a one is fetched. While in this state, the host controller will not perform nak counter reloads.

47.5.4.10 Managing Control/Bulk/Interrupt Transfers using Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Figure 47-53). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer.*

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,

- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller will set one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (e.g. the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (e.g. buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 47-68. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller will always *find* them if/when they are reachable.

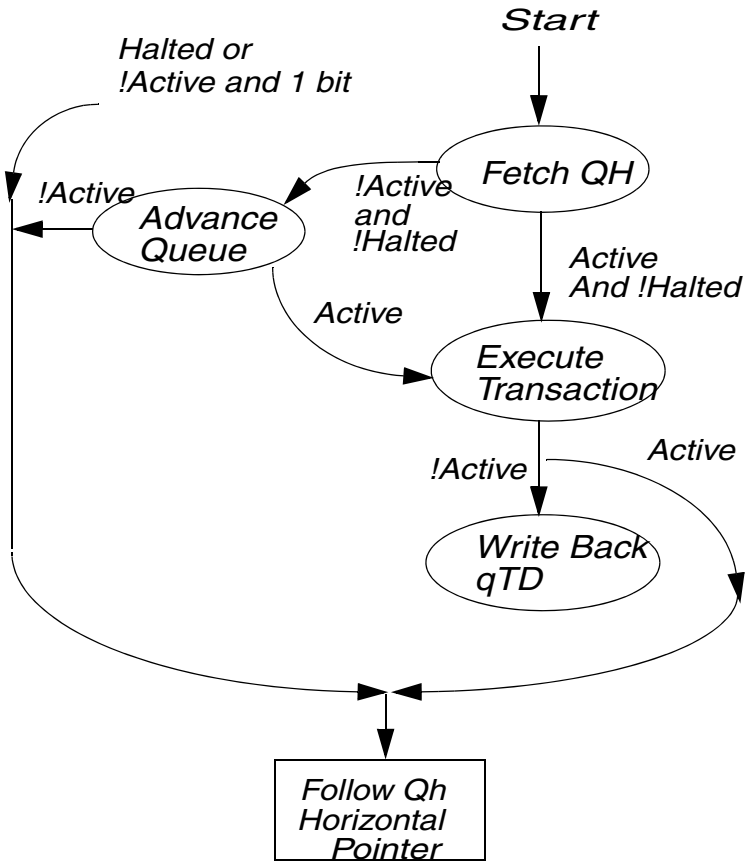


Figure 47-68. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The **Execute Transaction** state (Section 47.5.4.10.3, "Execute Transaction") describes the basic requirements for all endpoints. Section 47.5.4.12.1, "Split Transactions for Asynchronous Transfers," and Section 47.5.4.12.2, "Split Transaction Interrupt," describe details of the required extensions to the **Execute Transaction** state for endpoints requiring split transactions.

Note: Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section 47.5.3.6, "Queue Head," for layout of a queue head):

- Valid static endpoint state
- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

47.5.4.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (Section 47.5.2.6.7, "Host Controller Next Asynchronous Address Register (ASYNCLISTADDR) and Device Controller Endpoint List Address Register (ENDPOINTLISTADDR)." Additionally, it may be referenced from the *Next Link Pointer* field of an iTD, siTD, FSTN or another queue head. If the referencing link pointer has the TYP field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Figure 47-53.

While in this state, the host controller performs operations to implement empty schedule detection (Section 47.5.4.8.3, "Empty Asynchronous Schedule Detection") and Nak Counter reloads (Section 47.5.4.9, "Operational Model for Nak Counter"). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (i.e. *S-mask* is a zero), and
- The *H-bit* is a one, and
- The *Reclamation* bit in the USBSTS register is a zero.

When these criteria are met, the host controller will stop traversing the asynchronous list (as described in Section 47.5.4.8.3, "Empty Asynchronous Schedule Detection"). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is a one and the *Reclamation* bit is a one, then the host controller sets the *Reclamation* bit in the USBSTS register to a zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Section 47.5.4.9, "Operational Model for Nak Counter."

This state is complete when the queue head has been read on-chip.

47.5.4.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the **FetchQHD** state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and

determines whether or not to perform an overlay. Note that if the *I-bit* is a one and the *Active* bit is a zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer.* Otherwise, the host controller uses the *Next qTD Pointer.* If *Next qTD Pointer*'s *T-bit* is set to a one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to a one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 47-62).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

47.5.4.10.3 Execute Transaction

The host controller enters this state from the **Fetch Queue Head** state only if the *Active* bit in *Status* field of the queue head is set to a one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Section 47.5.4.12.1, "Split Transactions for Asynchronous Transfers," and Section 47.5.4.12.2, "Split Transaction Interrupt."

Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has a one in it. For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (e.g. a zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria: The pre-operation is:

Checks the Nak counter reload state (Section 47.5.4.9, "Operational Model for Nak Counter"). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the microframe to complete this transaction (see Transaction Fit—A Best-Fit Approximation Algorithm, for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller will exit this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to a zero, or
- There is not enough time in the microframe left to execute the next transaction (see Transaction Fit—A Best-Fit Approximation Algorithm, for an example method for implementing the frame boundary test).

[4] Note that for a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer

state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (e.g. advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Section 47.5.4.10.6, "Buffer Pointer List Use for Data Streaming with qTDs."

Note that the *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller will execute a zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller will detect a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to a zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (e.g. not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (e.g. decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory.

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *Maximum Packet Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section , "Transaction Error."

The following events will cause the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from a one to a zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to a one and *Active* is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to a one and the *Active* bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes. Note that for a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.

- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the **Advance Queue** state. See Section 47.5.4.12, "Split Transactions," for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (e.g. *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (e.g. a packet babble). This results in the host controller setting the *Halted* bit to a one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed using queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USBSTS register to a one. To halt the queue head, the *Active* bit is set to a zero and the *Halted* bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller will not advance the transfer state on a transaction that results in a *Halt* condition (e.g. no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset*, and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USBSTS register is set to a one. If the *USB Error Interrupt Enable* bit in the USBINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule. This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a

characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section 47.5.4.10.3, "Execute Transaction," apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the HCCPARAMs register to a one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (e.g. *Asynchronous Schedule Park Mode Enable* bit in the USBCMD register is a zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (i.e. only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to a one and also set *Asynchronous Schedule Park Mode Count* field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Section 47.5.4.10.3, "Execute Transaction." It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is a one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake. Table 47-77 summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 47-77. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1,2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.
	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction. [2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

[1] Note, the host controller may continue to execute bus transactions from the current high-speed queue head (if PM-Count is not equal to zero), if a PID mismatch is detected (e.g. expected DATA1 and received DATA0, or visa-versa).,

[2] Note, this specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

47.5.4.10.4 Write Back qTD

This state is entered from the **Execute Transaction** state when the *Active* bit is set to a zero. The source data for the write-back is the transfer results area of the queue head overlay area (see Table 47-77). The host controller uses the *Current qTD Pointer* field as the target address for the qTD. The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer, Cerr,* and *Status*.

The duration of this state depends on when the qTD write-back has been committed.

47.5.4.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is a one on exit from the **Execute Transaction** state, or
- When the host controller exits the **Write Back qTD** state, or
- If the **Advance Queue** state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is a one on exit from the **Fetch QH** state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

47.5.4.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*. This means: if the buffer spans more than one physical page, it must obey the following rules (Figure 47-69 illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction will span a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

Figure 47-69 illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page e.g. 2049 (e.g. 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

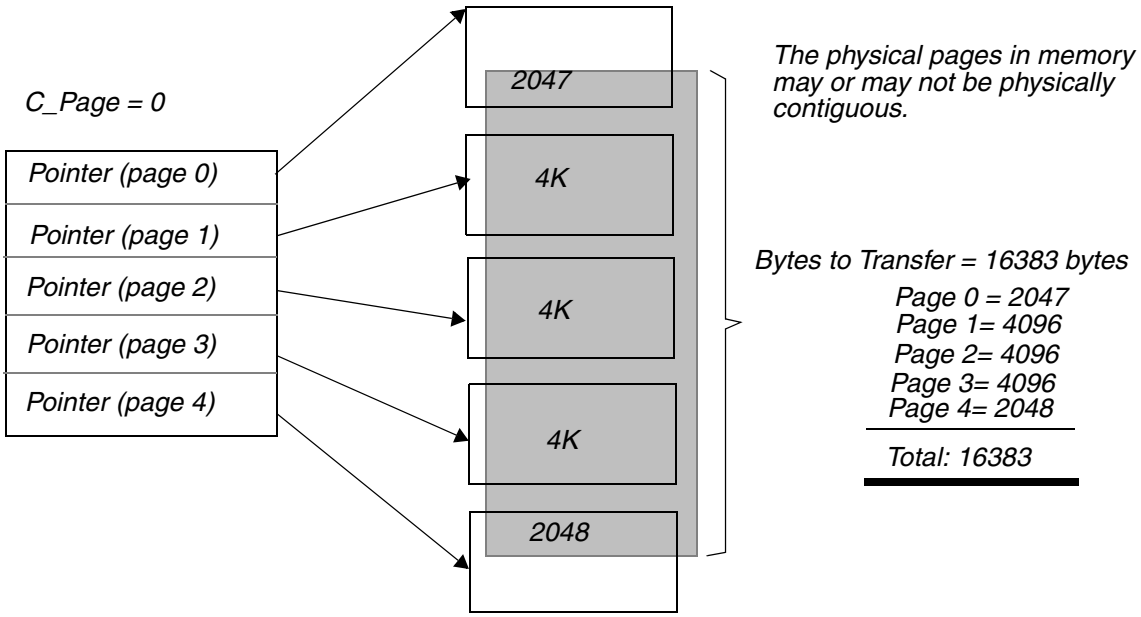


Figure 47-69. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is, *C_Page*) when necessary. There are three conditions for how the host controller handles *C_Page*.

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is, the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to *C_Page* is to increment by one.

47.5.4.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's *S-Mask* to indicate which microframe with-in

a 1 millisecond period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in Table 47-78.

Table 47-78. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, etc. *S-Mask* = 0x01	A queue head for the *bInterval* of 2 milliseconds (16 microframes) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during microframe 0 of the frame.
0, 2, 4, 6, 8, etc. *S-Mask* = 0x02	Another example of a queue head with a *bInterval* of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during microframe 1 of the frame.

47.5.4.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller will set an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to a one, or whenever a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (i.e. like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

47.5.4.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it will use in the next transaction to the endpoint (see Table 47-79). The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

Table 47-79 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. See Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 47-79. Ping Control State Transition Table

Current	Event		Next
	Host	Device	
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr[1]	Do Ping
Do Ping	PING	Stall	N/C[2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C[2]

1 Transaction Error (XactErr) is any time the host misses the handshake.

2 No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (e.g. Active set to zero and Halt set to a one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to **Do Ping**. The Ping State bit has the following encoding:

Value	Meaning
0B	Do OUT The host controller will use an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller will use a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (i.e. start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

47.5.4.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs. This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol. See USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction

protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Section 47.5.3.4, “Split Transaction Isochronous Transfer Descriptor (siTD).” Control, Bulk and Interrupt are managed using the queuing data structures (see Section 47.5.3.6, “Queue Head”). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

47.5.4.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an EPS field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head. All full-speed bulk and full-, low-speed control are managed using queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to **Do-Start-Split**. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to a one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is a one, the split transaction token's ET field is set to indicate a control endpoint. See Chapter 8 of USB Specification Revision 2.0 for details.

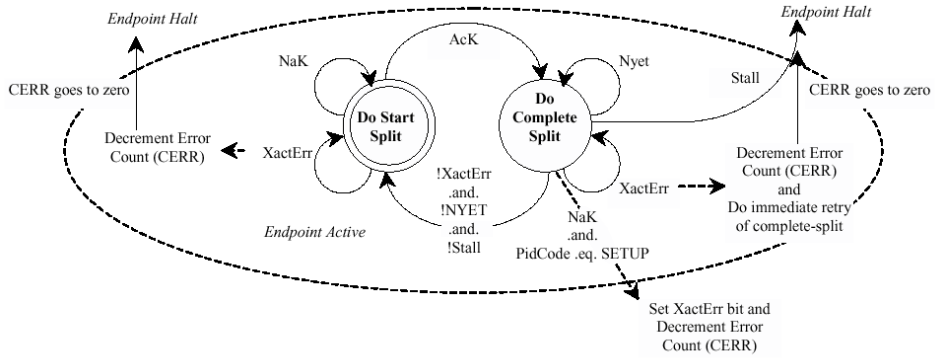


Figure 47-70. Host Controller Asynchronous Schedule Split-Transaction State Machine

Asynchronous—Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the **Do Complete Split** state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller will execute a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or

OUT transaction, then the host controller will reload the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller will not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

Asynchronous—Do Complete Split

This state is entered from the **Do Start Split** state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller will execute a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller will reload the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr.*

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, etc. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the microframe to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next microframe, the first transaction from the schedule will be the retry for this endpoint. If *Cerr* went to zero, the host controller must halt the queue.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to a one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.
- If the *PidCode* indicates an IN, then any of following responses are expected:
- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller will advance the state of the transfer, e.g. move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller will then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

- If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:
- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller will then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section 47.5.4.10, “Managing Control/Bulk/Interrupt Transfers using Queue Heads”).

47.5.4.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed using the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller will visit a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (i.e. takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, etc. occurs as defined in Section 47.5.4.10, “Managing Control/Bulk/Interrupt Transfers using Queue Heads,” but only occurs on the completion of a split transaction.

Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which microframes the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit microframes, or the data or response information in the pipeline is lost. Figure 47-71 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The **S** and ^{C}x labels indicate microframes where software can schedule start-splits and complete splits (respectively).

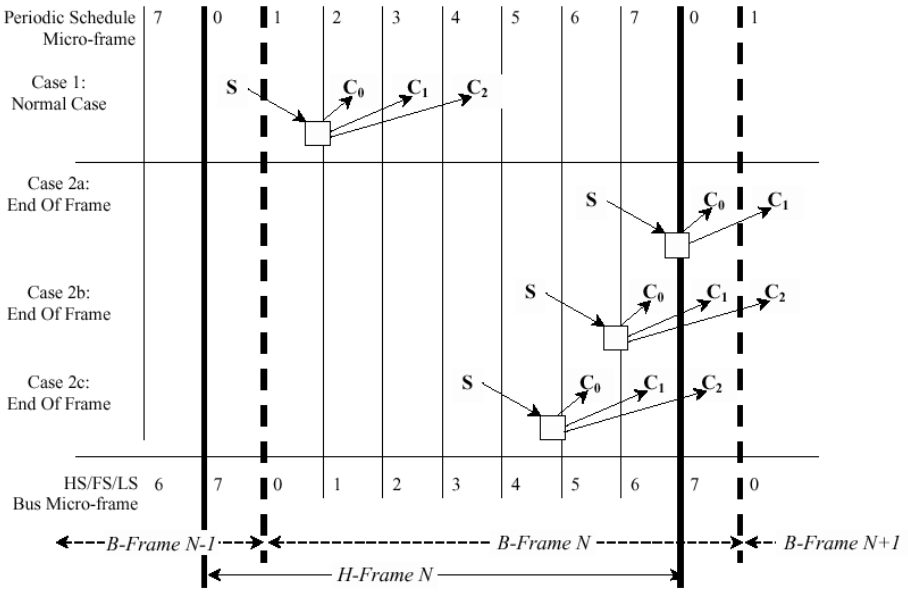


Figure 47-71. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in microframe 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 47-72 illustrates the general layout of the periodic schedule.

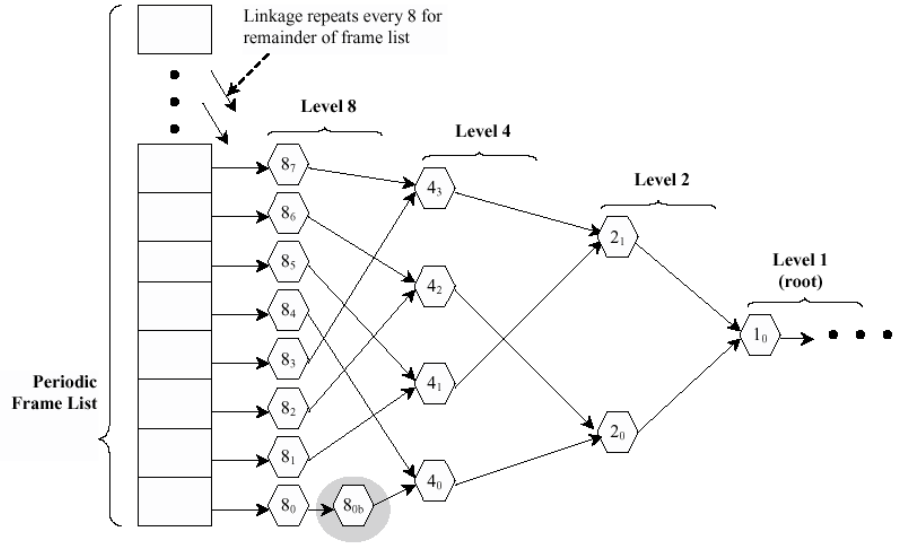


Figure 47-72. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Section , “Host Controller Operational Model for FSTNs,” defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of a queue head (see Table 47-79). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the microframe (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 47-71, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Start**, and the current microframe as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.

- *Frame C-mask*. This is a bit-field where system software sets one or more bits corresponding to the microframes (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 47-71, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates **Do_Complete**, and the current microframe as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (i.e. boundary cases 2a through 2c). An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see Section 47.5.3.4.5, “siTD Back Link Pointer”).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer. T-bit* set to zero.
- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
- Host controller FSTN traversal rules.

Host Controller Operational Model for FSTNs

When the host controller encounters an FSTN during microframes 2 through 7 it simply follows the node’s *Normal Path Link Pointer* to access the next schedule data structure. Note that the FSTN’s *Normal Path Link Pointer.T-bit* may set to a one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in microframes 0 or 1, it will save the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller’s rules for how it traverses the schedule and limits which data structures will be considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the microframe (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN’s *Back Path Link Pointer.*
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer.*

- Do not process a queue head (QH) whose EPS field indicates a high-speed device. Simply follow its horizontal link pointer.
- When a QH's EPS field indicates a Full/Low-speed device, the host controller will only consider it for execution if its *SplitXState* is **DoComplete** (note: this applies whether the *PID Code* indicates an IN or an OUT). See Section 47.5.4.10.3, "Execute Transaction" and Tracking Split Transaction Progress for Interrupt Transfers, for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Isochronous - Do Complete Split, for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer.*
- If the host controller determines that there is not enough time left in the microframe to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive microframe, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in Figure 47-73.

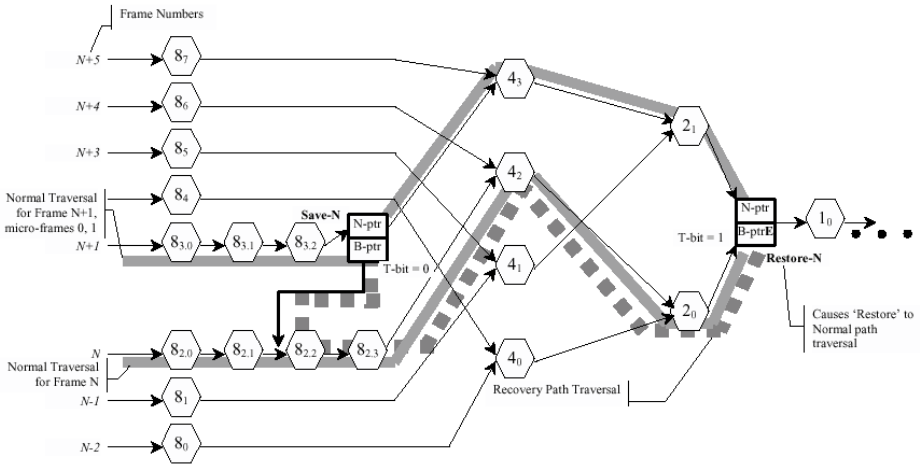


Figure 47-73. Example Host Controller Traversal of Recovery Path using FSTNs

In frame N+1 (microframes 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in Figure 47-73 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to a one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the

saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (e.g. Save-N.Normal Path Link Pointer). The nodes traversed during these microframes include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, **$8_{2.2}$**, **$8_{2.3}$**, **4_2**, **2_0**, **Restore-N**, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are bolded.In frame N (microframes 0-7), for this example, the host controller will traverse all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during microframes 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (i.e. normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (microframes 2-7), when the host controller encounters Save-Path FSTN Save-N, it will unconditionally follow Save-N.Normal Path Link Pointer. The nodes traversed during these microframes include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero. Note that *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to a one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to a one, as this will be use to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there will be times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth rebalance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the microframe number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to microframe are in the context of a microframe within an *H-Frame*.

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit*. This is a single-bit encoding of the current microframe number. It is an eight-bit value calculated by the host controller at the beginning of every microframe. It is calculated from the three least significant bits of the *FRINDEX* register (i.e. *cMicroFrameBit* = (1

shifted-left(*FRINDEX*[2:0]))). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current microframe number. For example, if the current microframe is 0, then *cMicroFrameBit* will equal 0b00000001. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current microframe.

Figure 47-74 illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at **Do_Start** and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to **Do_Complete**. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the **Do_Complete** state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the **Do_Complete** state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (e.g. after the host tries the same transaction three times, and each encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

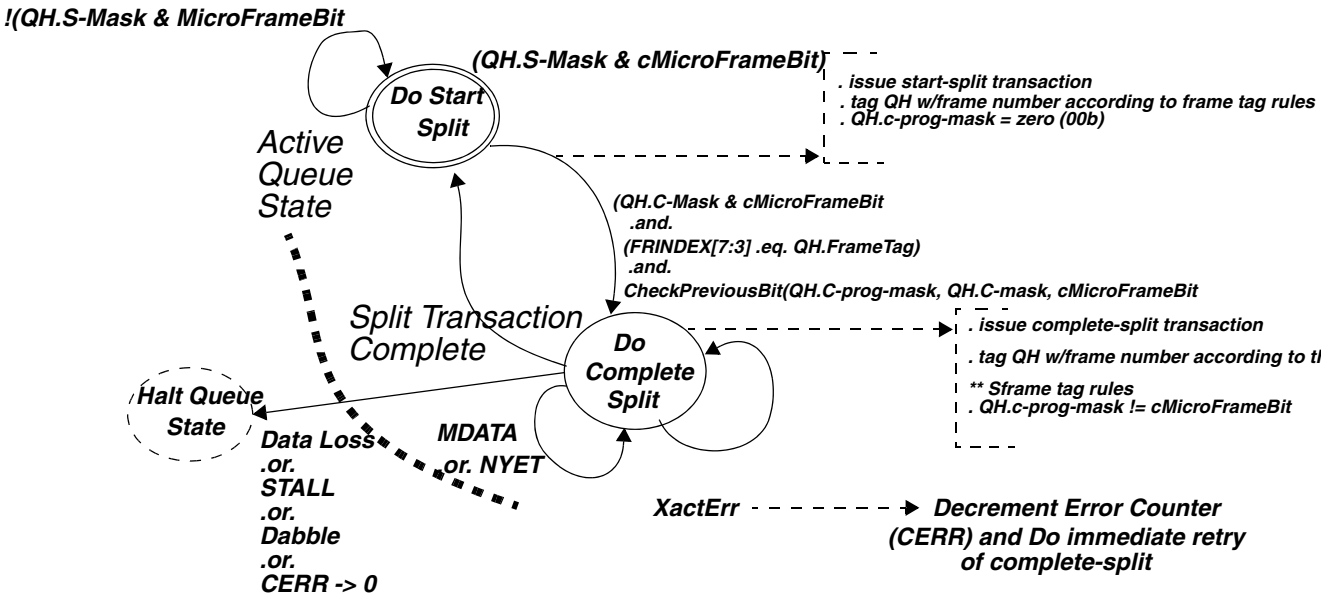


Figure 47-74. Split Transaction State Machine for Interrupt

**See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the **Do_Complete Split** state only after the split transaction is complete. This occurs when

one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (e.g. timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section , “Periodic Isochronous - Do Complete Split,” for the definition of ‘Last’.

Each time the host controller visits a queue head in this state (once within the **Execute Transaction** state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section 47.5.7.4, “Managing Queue Heads”), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the **Do Start Split** state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the **Execute Transaction** state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- **Test A**. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this microframe.
- **Test B**. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- **Test C**. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

Algorithm Boolean CheckPreviousBit`(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)`

```
Begin

-- Return values:

-- TRUE - no error
```

```
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous microframe. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                          If not(previousBit bitAND QH.C-prog-mask) then
                 rvalue = FALSE;
         End if
End If
-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- **Test D.** Check to see if a start-split should be executed in this microframe. Note this is the same test performed in the **Do Start Split** state (see Periodic Isochronous - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this microframe. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by

bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section 47.5.7.4, “Managing Queue Heads”). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the **Last** complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for **Last**. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the microframe to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the microframe to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to **Do Start Split**). This results in a retry of the entire OUT split transaction, at the next poll period. See Chapter 11 “Hubs” (specifically the section describing full- and low-speed interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section 47.5.4.10, “Managing Control/Bulk/Interrupt Transfers using Queue Heads”).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.

- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section 47.5.4.10, “Managing Control/Bulk/Interrupt Transfers using Queue Heads”).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.
- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the **Execute Transaction** state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. Table 47-80 lists the possible combinations and the appropriate action.

Table 47-80. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current microframe.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If PIDCode is an IN, then the Queue head must be halted. If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	QH.FrameTag test failed. This means that exactly one or more H-Frames have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If PIDCode is an IN, then the Queue head must be halted. If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one.

Table 47-80. Interrupt IN/OUT Do Complete Split State Execution Criteria (continued)

Condition	Action	Description
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If PIDCode is an IN, then the Queue head must be halted. If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. start-split is retried). This is a host-induced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one. Note: When executing in the context of a Recovery Path mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a Recovery Path mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- **Rule 1:** If transitioning from **Do Start Split** to **Do Complete Split** and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 47-71).
- **Rule 2:** If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in **Do Complete Split** for cases 2a, 2b, and 2c (Figure 47-71).
- **Rule 3:** If transitioning from **Do_Start Split** to **Do Complete Split** and the current value of *FRINDEX*[2:0] is not 6, or currently in **Do Complete Split** and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 47-71).

Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (i.e. new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is **DoStart** (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is not required to write the transfer

state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current microframe, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed its final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped using the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

47.5.4.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (see Section 47.5.4.7, “Managing Isochronous Transfers Using iTDs,” for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which microframes the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section , “Split Transaction Scheduling Mechanisms for Interrupt,” apply. Figure 47-75 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{s}x and ^{c}x labels indicate microframes where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

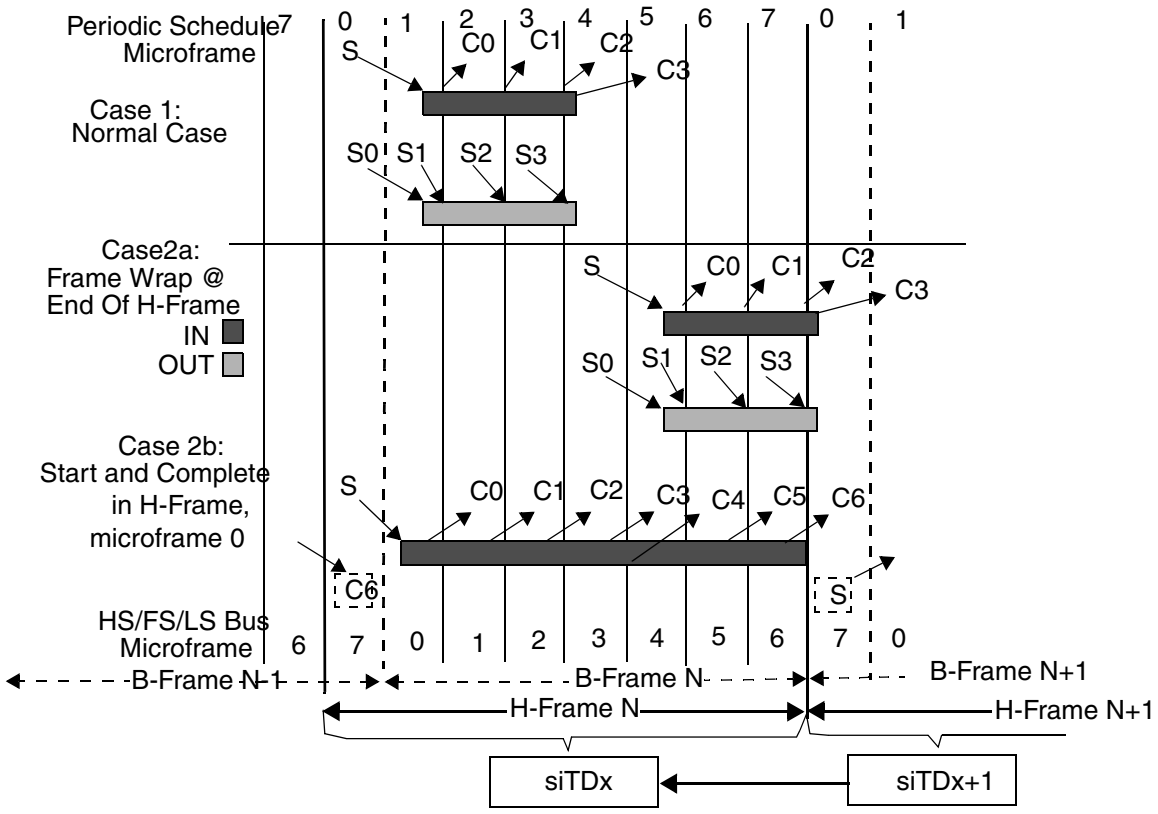


Figure 47-75. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame.* For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.
- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame*, microframes 6 or 7 (*H-Frame* microframe 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(e.g. it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same microframe. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Table 47-53). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section , “Split Transaction Execution State Machine for Interrupt.”
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the microframe (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 47-75, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates **Do Start Split**, and the current microframe as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask*. This is a bit-field where system software sets one or more bits corresponding to the microframes (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 47-75, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates **Do Complete Split**, and the current microframe as indicated by *FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.
- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. Figure 47-76 illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

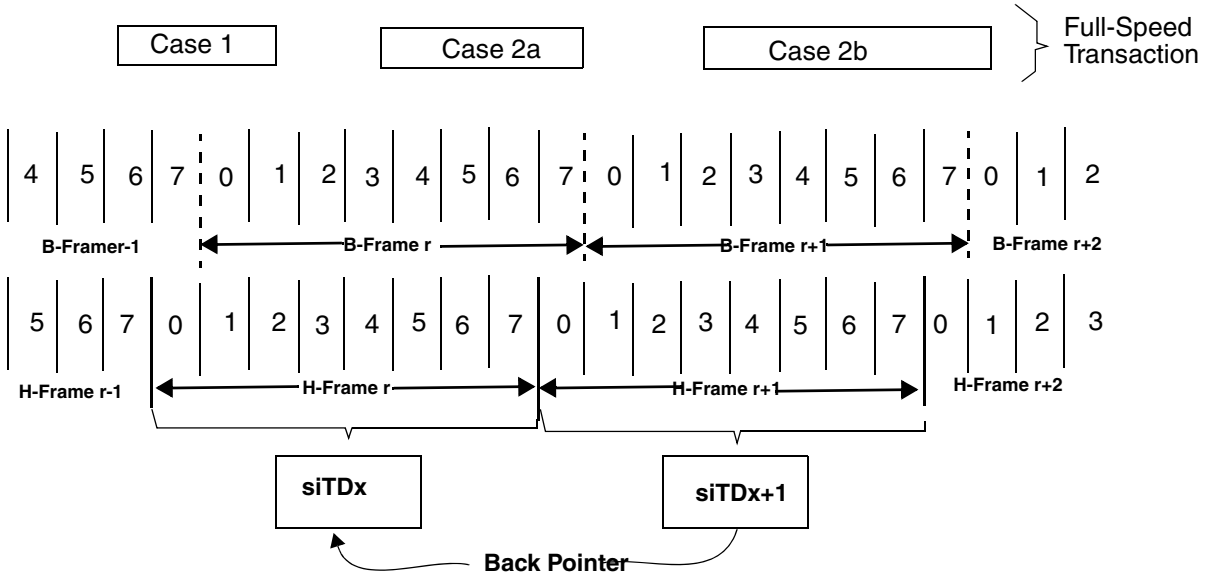


Figure 47-76. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during microframe 7 of $H\text{-}Frame_{Y+1}$, or microframe 0 of $H\text{-}Frame_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from $H\text{-}Frame_{Y+2}$ is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section , “Periodic Isochronous - Do Complete Split.”

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, microframe 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in microframe 1 of *H-Frame* N and the last complete-split would need to occur in microframe 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped microframes), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the microframes they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable using the schedule in the exact microframe in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future microframe).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD using the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section , “Periodic Isochronous - Do Start Split,” for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the microframe (FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. See Asynchronous—Do Complete Split, for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (e.g. high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a microframe boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the remaining bytes to the high-speed pipeline stage. At the microframe boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next microframe, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (e.g. a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (e.g. state not advanced) and report the appropriate error to the client driver.

Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to microframe are in the context of a microframe within an *H-Frame*.

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Tracking Split Transaction Progress for Interrupt Transfers, plus the variable *cMicroFrameBit* defined in Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. Figure 47-77 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the

states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

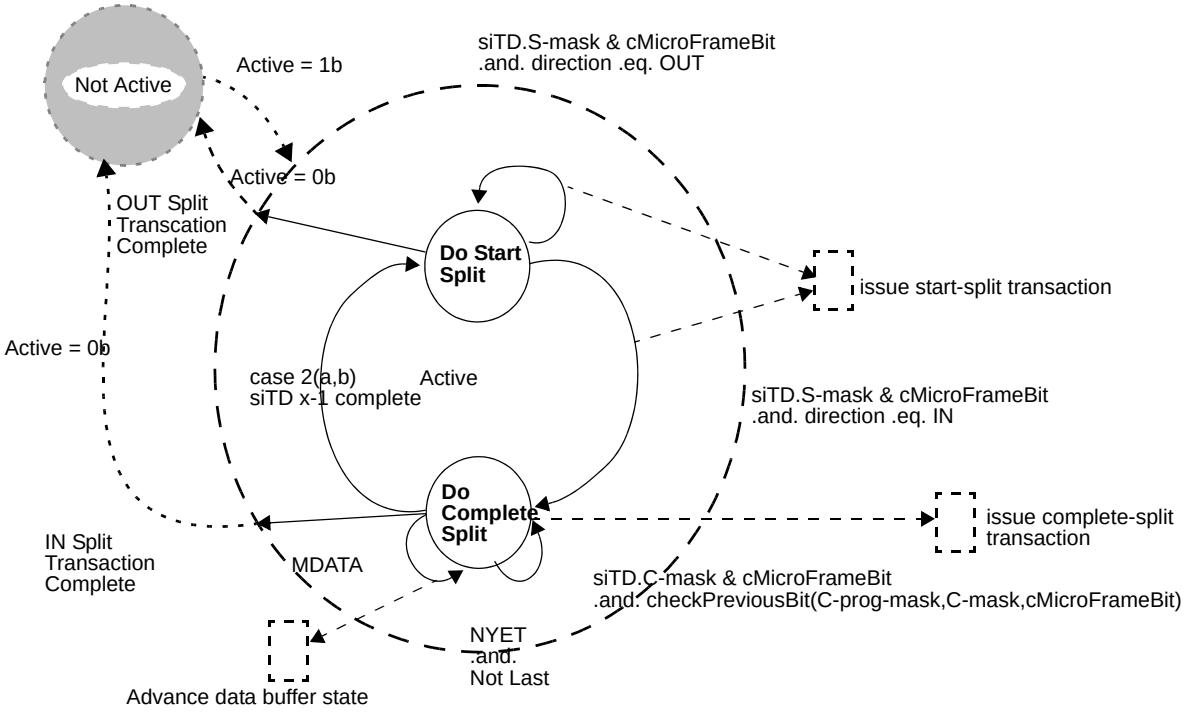


Figure 47-77. Split Transaction State Machine for Isochronous

Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state. An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from **Do Complete Split** when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (i.e. the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory

address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 47-76) is used to determine the initial value of *TP*. The initial cases are summarized in Table 47-81.

Table 47-81. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated. Table 47-82 illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

Table 47-82. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (e.g. greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 47-82.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host

controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (i.e. the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Split Transaction for Isochronous - Processing Examples.

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Periodic Isochronous - Do Complete Split, can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed microframes. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the **Do Start State** after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which microframe the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- **Test A**. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this microframe. This test is always applied to a newly fetched siTD that is in this state.
- **Test B**. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Periodic Isochronous - Do Complete Split). The sequence in which this test is applied depends on the current value of FRINDEX[2:0]. If FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

Algorithm Boolean CheckPreviousBit(`siTD.C-prog-mask, siTD.C-mask, cMicroFrameBi`**t)**

```
Begin

        Boolean rvalue = TRUE;

        previousBit = cMicroFrameBit rotate-right(1)

        -- Bit-wise anding previousBit with C-mask indicates whether there was an intent

        -- to send a complete split in the previous microframe. So, if the

        -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it

        -- happened.
```

```
        if previousBit bitAND siTD.C-mask then
                if not (previousBit bitAND siTD.C-prog-mask) then
                        rvalue = FALSE
                End if
        End if
        Return rvalue
End Algorithm
```

If Test A is true and FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 47-75). See Periodic Isochronous - Do Complete Split, for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,
- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the microframe occurs, the *Active* bit is set to zero.

- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Periodic Isochronous - Do Complete Split. If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.
- MDATA (and Last). See above description for testing for **Last**. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for **Last**. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from microframe X to X+1 and during microframe X, the transaction translator will respond with an MDATA and the data accumulated up to the end of microframe X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Microframe* status bit and sets the *Active* bit to a zero.

Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 47-75) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. Figure 47-77 enumerates the transaction state fields.

Table 47-83. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

Note: *TP* and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is **Do Complete Split**.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$*.Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Complete Split**), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 47-83) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is **Do Start Split**), then the host controller must set *Active* bit to a zero and *Missed microframe* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$'s *Active* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ using $siTD_X$'s back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$'s complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to **Do Start Split**. The host controller then determines whether the case 2b start split boundary condition exists (i.e. if *cMicroframeBit* is a 1b and $siTD_X$*.S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$*.Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$*.Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns to the context of $siTD_X$. Also, note that software should not initialize an

siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines. The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced using the **Execute Transaction** queue head traversal state machine (see Figure 47-68).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, Table 47-84 illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 47-84. Example Case 2a—Software Scheduling siTDs for an IN Endpoint

siTDx		microframes								Initial SplitXState
#	Masks	0	1	2	3	4	5	6	7	
X	S-Mask					1				Do Start Split
	C-Mask	1	1					1	1	
X+1	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask					1				Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Since this is the case-2a frame-wrap case, *S-mask*s of all siTDs for this endpoint have a value of 10h (a one bit in microframe 4) and *C-mask* value of C3h (one-bits in microframes 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back Pointer T-bits* are set to zero).

The initial *SplitXState* of the first siTD is **Do Start Split**. The host controller will visit the first siTD eight times during frame X. The C-mask bits in microframes 0 and 1 are ignored because the state is **Do Start Split**. During microframe 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to **Do Complete Split**. During microframes 6 and 7, the host controller executes

complete-splits. Notice the siTD for frame X+1 has its *SplitXState* initialized to **Do Complete Split**. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During microframes 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, microframe 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to **Do Start Split.** At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches microframe 4. If the split-transaction completes early (transaction-complete is defined in Periodic Isochronous - Do Complete Split), that is, before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to **Do Start Split** early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, microframe 1.

During *H-Frame X+2*, microframe 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses it's next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, microframe 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to **Do Start Split**. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it reaches microframe 4.

47.5.4.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller will access the schedules in main memory each microframe. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the CPU power management software from placing the CPU into its lowest power savings state.

USB Host controllers do not access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule

enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the CPU to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing its shared memory based data structures (schedule lists or otherwise).

47.5.4.14 Port Test Modes

EHCI host controllers must implement the port test modes **Test J_State**, **Test K_State**, **Test_Packet**, **Test Force_Enable**, and **Test SE0_NAK** as described in the USB Specification Revision 2.0. The system is only allowed to test ports that are owned by the EHCI controller (e.g. *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is **Test_Force_Enable**, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting the RST bit (USBCMD register) to a one.

47.5.4.15 Interrupts

The EHCI Host Controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section 47.5.2.6.3, "USB Interrupt Enable Register (USBINTR)"). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter using the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight microframes. This means that the host controller will not generate interrupts any more frequently than once every eight microframes.

Host System Error, details the effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (using an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

The host controller is not required to negate a currently active interrupt condition when software clears the interrupt enables in the USBINR register (see Section 47.5.2.6.3, "USB Interrupt Enable Register (USBINTR)"). The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USB Status Register (USBSTS) from 1 to 0.

47.5.4.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 47-85 lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the

XactErr status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 47-85. Summary of Transaction Errors

Event /Result	Queue Head/qTD/iTD/siTD Side-Effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	–1	XactErr set to a one.	1[1]
Timeout	–1	XactErr set to a one.	1[1]
Bad PID[2]	–1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	—

1 If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Section , "Halting a Queue Head".

2 The host controller received a response from the device, but it could not recognize the PID as a valid PID.

Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*. When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a microframe EOF.

NOTE

When a host controller detects a data PID mismatch, it must either disable the packet babble checking for the duration of the bus transaction, or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (e.g. expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USBSTS register to be set to a one. In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USBSTS register is also set to a one.

Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USBSTS register is set to a one. If the *USB Interrupt Enable* bit is set in the USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

47.5.4.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section , “Interrupt on Async Advance”).

Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one. If the *Port Change Interrupt Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USBCMD register. If it is a one, it sets the *Interrupt on Async Advance* bit in the USBSTS register to a one. If the *Interrupt on Async Advance Enable* bit in the USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section 47.5.4.8.2, “Removing Queue Heads from Asynchronous Schedule.”

Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion. In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USBCMD register is set to a zero.
- The following bits in the USBSTS register are set:
  - *Host System Error* bit is to a one.

— *HCHalted* bit is set to a one.

- If the *Host System Error Enable* bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. Table 47-86 summarizes the required actions taken on the various host errors.

Table 47-86. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

[o] Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a Host System Error, software must reset the host controller using the RST bit in the USBCMD register before re-initializing and restarting the host controller.

47.5.5 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation and On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary. The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator – Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

47.5.5.1 Embedded Transaction Translator Function

The ARC USB-HS OTG High-Speed USB On-The-Go OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator. Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

47.5.5.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to HCSPARAMS – Host Control Structural Parameters
- N_PTT added toHCSPARAMS – Host Control Structural Parameters

47.5.5.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- is a new register.
- Addition of two-bit Port Speed (PSPD) to the PORTSC*x* registers.

47.5.5.1.3 Discovery

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation. The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, chirp completes successfully).

Since this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in the PORTSC*x* register.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in Table 47-87.

Table 47-87. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSCx.

Table 47-87. Summary of EHCI

FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. **[where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (that is, Split target hub)]**	FS and LS device can be either downstream from a HS hub with HubAddr = X **[HubAddr > 0]** or directly attached **[where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (that is, Split target hub is the root hub)**

47.5.5.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator. Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) – Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = 0
   - Transactions to direct attached device/hub.
     - QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub.
     - QH.EPS = Downstream Device Speed

Note: When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

   - Maximum Packet Size must be less than or equal 64 or undefined behavior may result.

2. siTD (for direct attach FS) – Periodic (ISO Endpoint)
   - All FS ISO transactions:
     - Hub Address = 0
     - siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behavior may result.

47.5.5.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Since the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the microframe pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator. Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. Table 47-88 summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 47-88. Summary of the Conditons of Handshakes

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

Note: The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits.

Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0 – 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

USB 2.0 – 11.17.4

- Transaction tracking for 2 data pipes.

USB 2.0 – 11.17.5

- Clear_TT_Buffer capability provided though the use of the register.

Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

USB 2.0 – 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in microframes 6)
  - Idle for more than 4 microframes
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 microframes

USB 2.0 – 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Some applications may not require transaction tracking up to a maximum of 16 periodic data pipes. The option to limit the tracking to only 4 periodic data pipes exists in the by changing the configuration constant VUSB_HS_TT_PERIODIC_CONTEXTS to 4. The result is a significant gate count savings to the core given the limitations implied.

CAUTION	**Note:** Limiting the number of tracking pipes in the EMBedded –TT to four (4) will impose the restriction that no more than 4 periodic transactions (INTERRUPT/ISOCHRONOUS) can be scheduled through the embedded-TT per frame. The number 16 was chosen in the USB specification because it is sufficient to ensure that the high-speed to full-speed periodic pipeline can remain full. Keeping the pipeline full puts no constraint on the number of periodic transactions that can be scheduled in a frame and the only limit becomes the flight time of the packets on the bus.

- Complete-split transaction searching.

Note: There is no data schedule mechanism for these transactions other than the microframe pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the Host Control Structural Parameters Register (HCSPARAMS) register.

47.5.5.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

47.5.5.3 USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode Register (USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

47.5.5.3.1 Non-Zero Fields the Register File.

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode.

47.5.5.3.2 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125 µs interrupt for host mode. EHCI does not specify this interrupt, but it has been added for convenience and as a potential software time base. See Section 47.5.2.6.2, “USB Status Register (USBSTS),” and Section 47.5.2.6.3, “USB Interrupt Enable Register (USBINTR).”

47.5.5.4 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

47.5.5.4.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of microframes are timed precisely to 125 us using the transceiver clock as a reference clock. i.e. 60 Mhz transceiver clock for 8-bit physical interfaces and full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

47.5.5.5 Miscellaneous Variations from EHCI

47.5.5.5.1 Programmable Physical Interface Behavior

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the PORTSC*x* registers, providing a capability that is not defined by EHCI.

47.5.5.5.2 Discovery

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the Port Status Control *x* Registers (*x* = 1…8) register for a duration of 10 ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed – *This information is redundant with the 2-bit Port Speed indicator above.*

47.5.5.5.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI since the release of revision 3.2.1. In earlier product revisions, the test packet mode was not EHCI compatible. An alternate host controller driver procedure is no longer necessary or supported.

47.5.6 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

Note: Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The ARC USB-HS OTG High-Speed USB On-The-Go core includes DCD Software called the USB 2.0 Device API. The Device API provides an easy to use Application Program Interface for developing device (peripheral) applications using the ARC USB-HS OTG High-Speed USB On-The-Go core. The Device API incorporates and abstracts for the application developer all of the elements of the program interface.

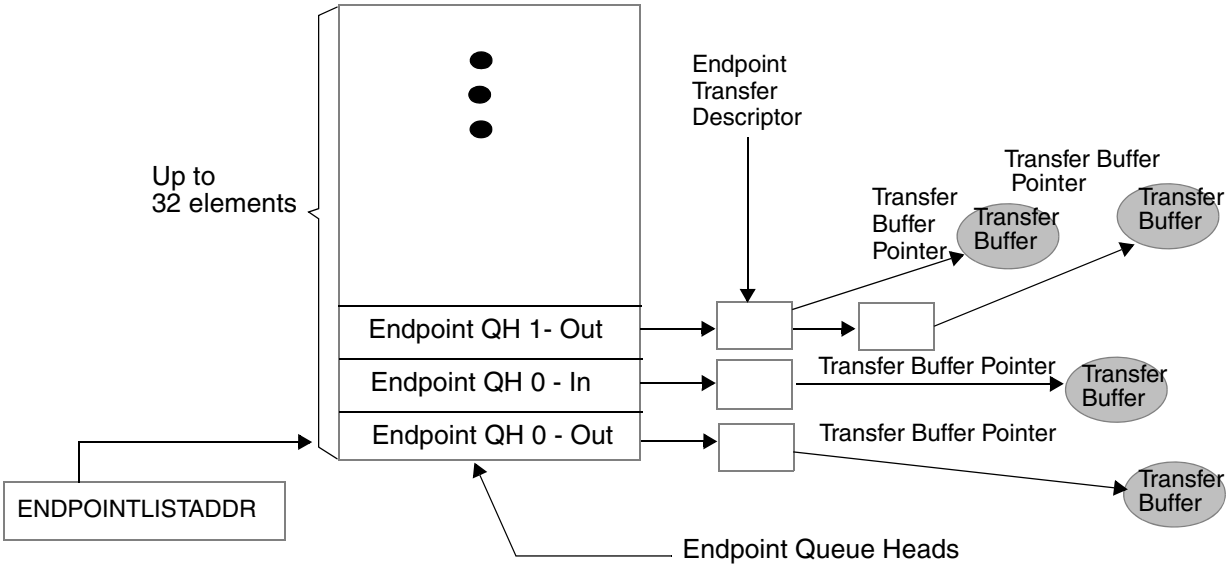


Figure 47-78. Endpoint Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the *ENDPOINTLISTADDR* pointer. The even –numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number

received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE The Endpoint Queue Head List must be aligned to a 2k boundary.

47.5.6.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

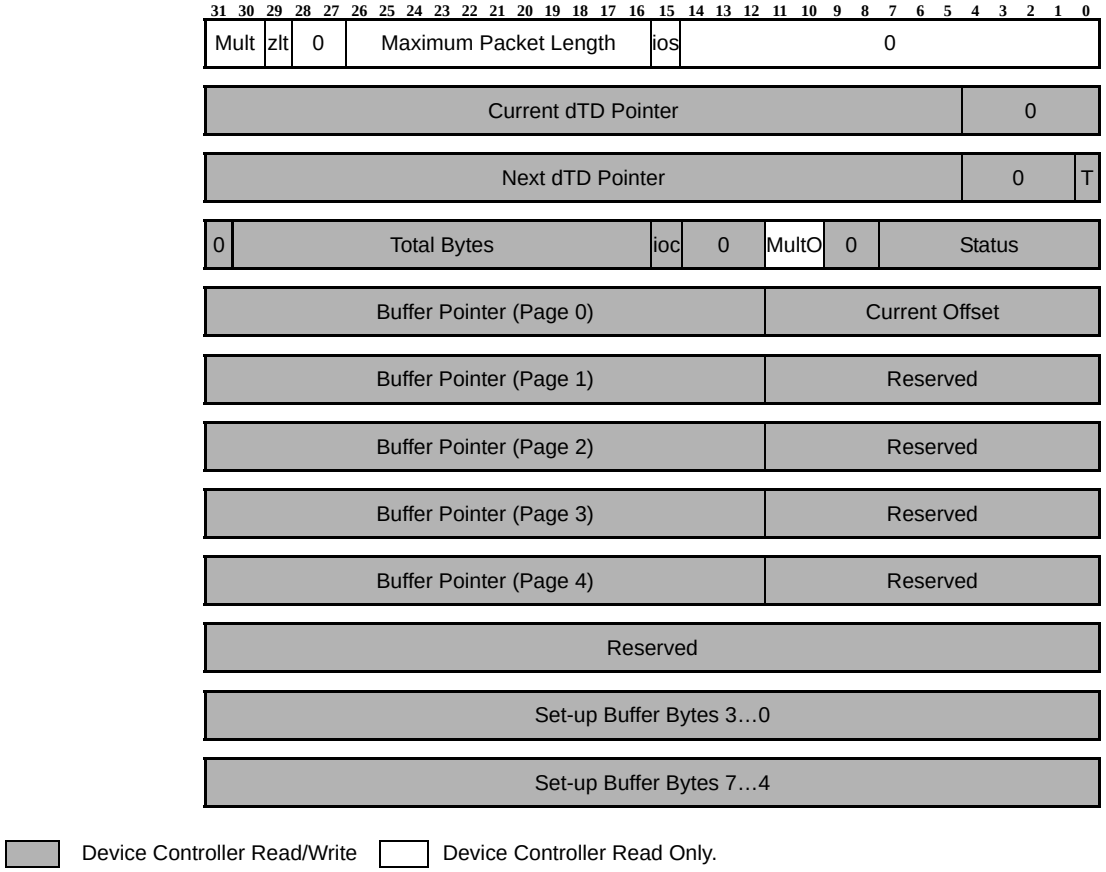


Figure 47-79. Endpoint Queue Head (dQH)

47.5.6.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 47-89. Endpoint Capabilities/Characteristics

Bit	Description
31:30	**Mult.** This field is used to indicate the number of packets executed per transaction description as given by the following: 00 – Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 – Execute 1 Transaction. 10 – Execute 2 Transactions. 11 – Execute 3 Transactions. Note: Non-ISO endpoints must set Mult=”00”. Note: ISO endpoints must set Mult=”01”, “10”, or “11” as needed.
29	**Zero Length Termination Select.** This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple. This bit is not relevant for Isochronous 0 – Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 – Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28:27	**Reserved.** These bit reserved for future use and should be set to zero.
26:16	**Maximum Packet Length.** This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	**Interrupt On Setup (IOS).** This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14:0	**Reserved.** Bits reserved for future use and should be set to zero.

47.5.6.1.2 Transfer Overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

47.5.6.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

Table 47-90. Next dTD Pointer

Bit	Description
31:5	**Current dTD.** This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4:0	**Reserved.** Bit reserved for future use and should be set to zero.

47.5.6.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

Note: Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 47-91. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31:0	**Setup Buffer 0.** This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31:0	**Setup Buffer 1.** This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

47.5.6.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

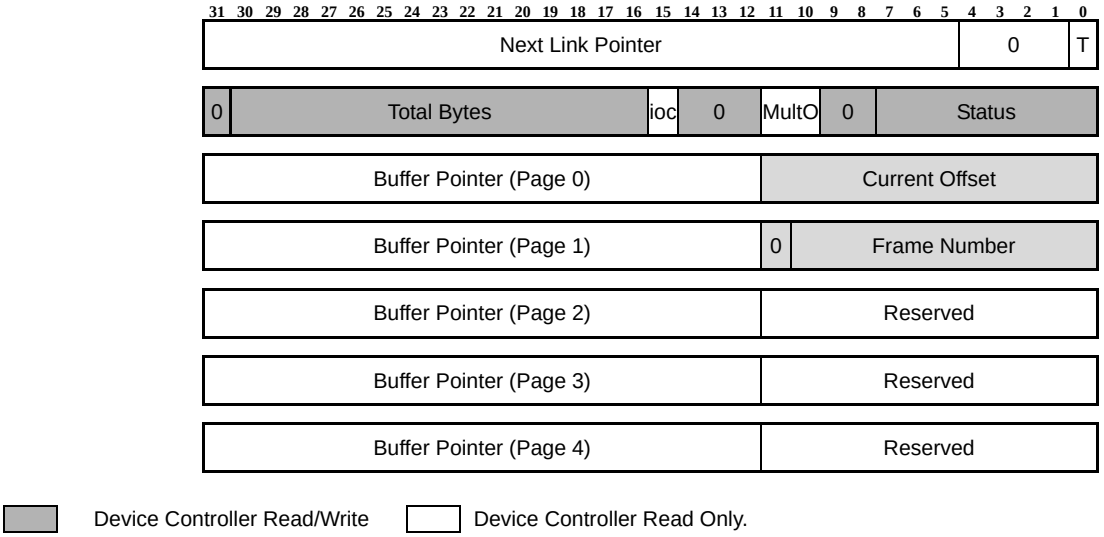


Figure 47-80. Endpoint Transfer Descriptor (dTD)

Table 47-92. Next dTD Pointer

Bit	Description
31:5	**Next Transfer Element Pointer.** This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4:1	**Reserved.** Bits reserved for future use and should be set to zero.
0	**Terminate (T).** 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

Table 47-93. dTD Token

Bit	Description
31	**Reserved.** Bit reserved for future use and should be set to zero.
30:16	**Total Bytes.** This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. **Therefore, the maximum recommended transfer is 16K(4000H).** If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	**Interrupt On Complete (IOC).** This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14:12	**Reserved**. Bits reserved for future use and should be set to zero.
11:10	**Multiplier Override (MultO).** This field can be used for transmit ISO's (that is, ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO="00".
9:8	**Reserved**. Bits reserved for future use and should be set to zero.
7:0	**Status.** This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0Reserved.

Table 47-94. dTD Buffer Page Pointer List

Bit	Description
31:12	**Buffer Pointer.** Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0;11:0	**Current Offset.** Offset into the 4kb buffer where the packet is to begin.
1;10:0	**Frame Number.** Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

47.5.7 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list **transfer descriptors**, pointed to by a **queue head**, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

The ARC USB-HS OTG High-Speed USB On-The-Go is shipped with a DCD called the ARC USB-HS OTG High-Speed USB On-The-Go Device API. The ARC USB-HS OTG High-Speed USB On-The-Go Device API provides an easy to use application interface for developing USB device (peripheral) applications. The ARC USB-HS OTG High-Speed USB On-The-Go Device API incorporates and abstracts for the application developer all of the information contained in the device operational model. For more information on the ARC USB-HS OTG High-Speed USB On-The-Go Device API, see the “Software Design document for the Precise USB 2.0 Device API.”

47.5.7.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a ‘1’. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.
2. Allocate and Initialize device queue heads in system memory.

Note: Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

   — Minimum: Initialize device queue heads 0 Tx & 0 Rx.
   — For information on device queue heads, see the Device Data Structures section.

Note: All device queue heads must be initialized for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

3. Configure ENDPOINTLISTADDR Pointer.
   — For additional information on ENDPOINTLISTADDR, see the register table.
4. Enable the microprocessor interrupt associated with the ARC USB-HS OTG High-Speed USB On-The-Go core.
   — Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
   — For a list of available interrupts see the register tables in Section 47.5.2.6.3, “USB Interrupt Enable Register (USBINTR),” and Section 47.5.2.6.2, “USB Status Register (USBSTS).”
5. Set Run/Stop bit to Run Mode in USB CMD register.

— After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register. It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

47.5.7.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'. After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram depicts the state of a USB 2.0 device.

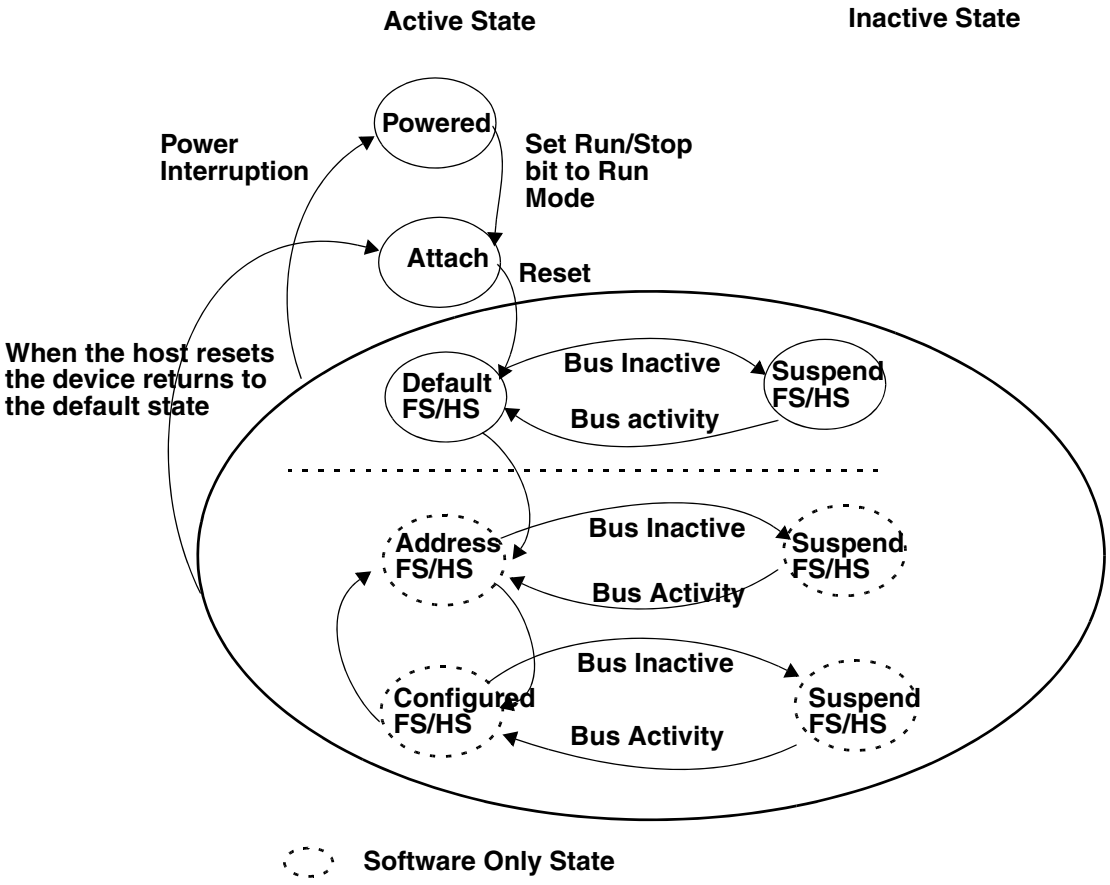


Figure 47-81. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

Table 47-95. Device Controller State Information Bits

Bit	Register
DCSuspend	USB Status Register (USBSTS)
USB Reset Received	USBSTS
Port Change Detect	USBSTS
High-Speed Port	Port Status Control x Registers (PORTSCx, x = 1…8)

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

47.5.7.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.

Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the ENDPTPRIME register are 0 and then writing 0xFFFFFFFF to the ENDPTFLUSH register.

Read the reset bit in the PORTSC*x* register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before the end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare).

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSC*x*, register to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to Chapter 9 (“Device Framework”) of the USB Specification.

Note: The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

47.5.7.2.2 Suspend/Resume

Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wake-up. The ability of a device to signal remote wake-up is optional. If the USB device is capable of remote wake-up signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wake-up signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the PORTSCx register is set to 1, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

Note: Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing 1 to the Resume bit in the in the PORTSC*x* register while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

Note: Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

Port Test Modes

Contact ARC International for port test mode capabilities.

47.5.7.2.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The ARC USB-HS OTG High-Speed USB On-The-Go device controller hardware supports up to the USB 2.0 maximum of 32 endpoint specified numbers. Each additional endpoint beyond the required endpoint position adds additional hardware logic. The maximum number of endpoint numbers available to the DCD is configured at hardware synthesis timer. After synthesis, the DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. If the maximum of 16 endpoint numbers, one for each endpoint direction are being used by the device controller, then 32 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

47.5.7.2.4 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRL*x* register. Each 32-bit ENDPTCTRL*x* is split into an upper and lower half. The lower half of ENDPTCTRL*x* is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRL*x* register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Table 47-96. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	'1'
Data Toggle Inhibit	'0'
Endpoint Type	"00" – Control "01" – Isochronous "10" – Bulk "11" – Interrupt
Endpoint Stall	'0'

47.5.7.2.5 Stalling

There are two occasions where the device controller may need to return to the host a STALL

The first occasion is the **functional stall**, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRL*x* register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A **protocol stall**, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRL*x* register can ensure that both stall bits are set at the same instant.

Note: Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

Table 47-97. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET

Table 47-97. Device Controller Stall Response Matrix (continued)

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	‘1’	None	STALL
IN/OUT/PING packet received by a control endpoint.	‘0’	None.	ACK/ NAK/ NYET

47.5.7.2.6 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, see the USB 2.0 specification.

Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a ‘1’ to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit

Note: This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active (‘1’) causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

47.5.7.3 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will

send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "**priming**" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "**flushing**" is used to describe the action of clearing a packet that was queued for execution.

Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus.

Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

47.5.7.3.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 47-98. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	—
512	256	3	256	256	0
512	512	2	512	0	—

Table 47-99. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	—
512	256	2	256	256	—
512	512	1	512	—	—

Note: The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

Note: All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

Interrupt/Bulk Endpoint Bus Response Matrix

Table 47-100. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

47.5.7.3.2 Control Endpoint Operation Model

Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

In hardware versions 2.3 and later, the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling (Pre-2.3 hardware)

- After receiving an interrupt and inspecting the USBMODE register to determine that a setup packet was received on a particular pipe:
  - 1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
  - 2. Write 1 to clear the corresponding ENDPTSETUPSTAT bit and thereby disabling Setup Lockout. (i.e. the Setup Lockout activates as soon as a setup arrives. By writing to the ENDPTSETUPSTAT register, the device controller will accept new setup packets.)

— 3. Process setup packet using local software byte array copy and execute status/handshake phases.
    - Note: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed and deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Note: To limit the exposure of setup packets to the setup lockout mechanism (if used), the DCD should designate the priority of responding to setup packets above responding to other packet completions.

Setup Packet Handling (2.3 hardware and later)

- Disable Setup Lockout by writing ‘1’ to Setup Lockout Mode (SLOM) in USB Device Mode Register (USBMODE). (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

Note: leaving the Setup Lockout Mode As ‘0’ will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting the ENDPTSETUPSTAT register to determine that a setup packet was received on a particular pipe:
    — 1. Write 1 to clear the corresponding ENDPTSETUPSTAT register bit.
    — 2. Write 1 to the Setup Tripwire (SUTW) bit in the USBCMD register.
    — 3. Duplicate contents of dQH.SetupBuffer into local software byte array.
    — 4. Read Setup TripWire (SUTW) in USB Command Register (USBCMD) register. (if set - continue; if cleared - goto 2)
    — 5. Write '0' to clear Setup Tripwire (SUTW) in USB Command Register (USBCMD) register.
    — 6. Process setup packet using local software byte array copy and execute status/handshake phases.

Note: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed and deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTAT register is a one. If a prime fails, that is the ENDPTPRIME bit goes to 0 and the ENDPTSTAT bit is not set to 1, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTAT) to enforce data coherency with the setup packet.

Note: The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

Note: Error handling of data phase packets is the same as bulk packets described previously.

Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

Note: The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

Note: Error handling of data phase packets is the same as bulk packets described previously.

Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Table 47-101. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	—
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

47.5.7.3.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to IN requests to unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - — MULT counter reaches zero.
  - — Fulfillment Error [*Transaction Error* bit is set]
    - – # Packets Occurred > 0 AND # Packets Occurred < MULT

Note: For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:

— MULT counter reaches zero.
— Non-MDATA Data PID is received**
  – ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
— Overflow Error:
  – Packet received is > maximum packet length. [*Buffer Error* bit is set]
  – Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
— Fulfillment Error [*Transaction Error* bit is set]
  – # Packets Occurred > 0 AND # Packets Occurred < MULT
— CRC Error [*Transaction Error* bit is set

Note: For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N–1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N–1 so that the device controller will execute delivery during (micro)frame N.

CAUTION

Priming an endpoint towards the end of (micro)frame N-1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

Isochronous Endpoint Bus Response Matrix

Table 47-102. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NULL Packet = Zero Length Packet

47.5.7.4 Managing Queue Heads

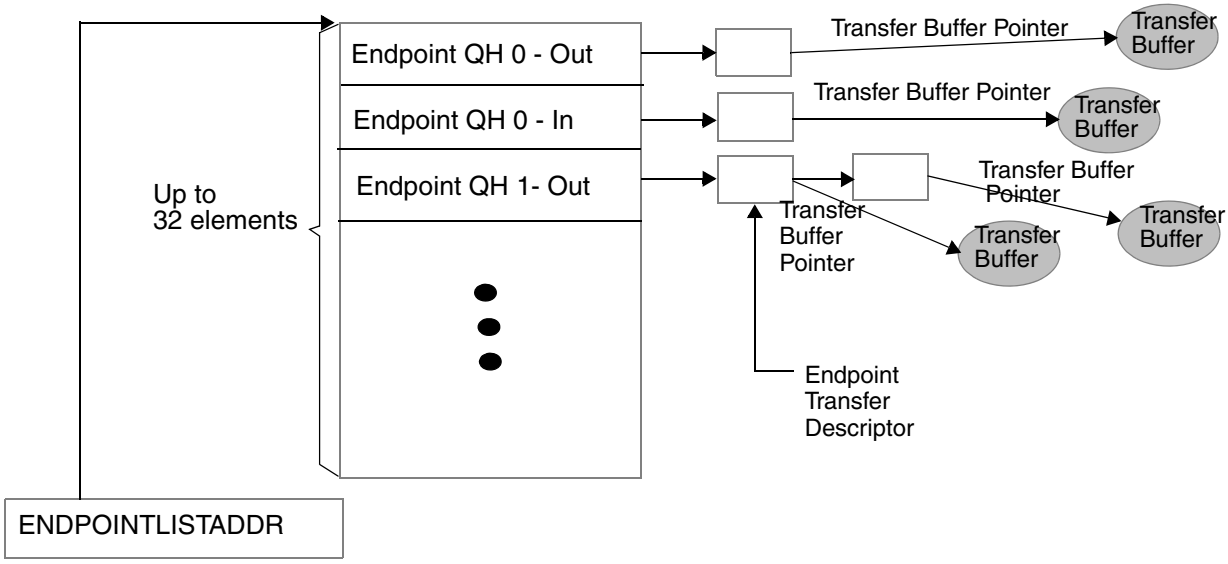


Figure 47-82. Endpoint Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 47-82. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

47.5.7.4.1 Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required for bandwidth, and in conjuction with the USB Chapter 9 protocol. *Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.*
- Write the next dTD Terminate bit field to "1".
- Write the Active bit in the status field to "0".
- Write the Halt bit in the status field to "0".

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

47.5.7.4.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

The acknowledge must occur before continuing to process the setup packet.

After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH – RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in Section 47.5.7.5.5, "Flushing/De-priming an Endpoint."
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

47.5.7.5 Managing Transfers with Transfer Descriptors

47.5.7.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

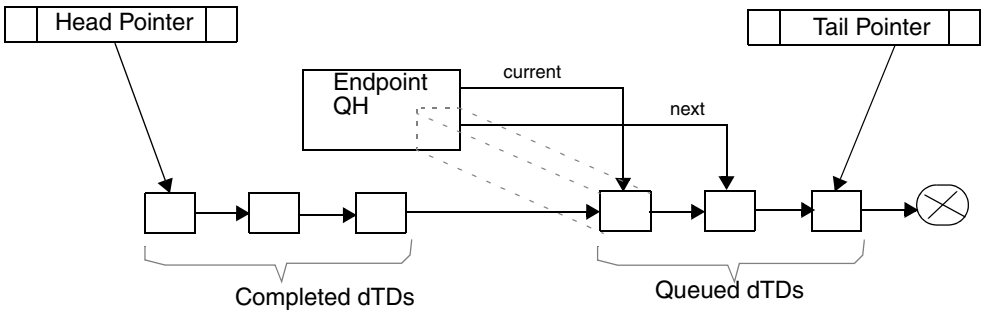


Figure 47-83. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

47.5.7.5.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to 0b00000.

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

47.5.7.5.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

— Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding)

- Case 1: Link list is empty

— 1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.

- 2. Clear active & halt bit in dQH (in case set from a previous error).
- 3. Prime endpoint by writing 1 to the correct bit position in the ENDPTPRIME register.

- Case 2: Link list is not empty
  - 1. Add dTD to end of linked list.
  - 2. Read correct prime bit in the ENDPTPRIME register – if '1' DONE.
  - 3. Set ATDTW bit in the USBCMD register to 1.
  - 4. Read correct status bit in the ENDPTSTAT register (store in tmp. variable for later)
  - 5. Read ATDTW bit in the USBCMD register.
    - If 0 goto 3.
    - If 1 continue to 6.
  - 6. Write ATDTW bit in the USBCMD register to 0.
  - 7. If status bit read in (4) is 1 DONE.
  - 8. If status bit read in (4) is 0 then Goto Case 1: Step 1.

47.5.7.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

CAUTION

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Section 47.5.7.5.6, "Device Error Matrix."

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

47.5.7.5.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a 1 to the corresponding bit(s) in the ENDPTFLUSH register.
2. Wait until all bits in ENDPTFLUSH are 0.
   - Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
3. Read the ENDPTSTAT register to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using the ENDPTFLUSH register. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

47.5.7.5.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 47-103. Device Error Matrix

Error	Description	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow	Number of bytes received exceeded max. packet size or total buffer length. **Note:** This error also sets the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those are not executed.	RX	Any	1	0
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.	RX	ISO	0	1
ISO Fulfillment Error	Hst failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated.

47.5.7.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

47.5.7.6.1 High-Frequency Interrupts

High frequency interrupts in particular should be handed in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 47-104. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt [1] - **ENDPTSETUPSTATUS**	Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ** - **ENDPTCOMPLETE**	Handle completion of dTD as indicated in section Managing Queue Heads.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

[1] It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

47.5.7.6.2 Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don’t occur often in comparison to the high-frequency interrupts.

Table 47-105. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

47.5.7.6.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Table 47-106. Error Interrupt Events

Interrupt	Action
USB Error Interrupt.	This error is redundant because it combines **USB Interrupt** and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of **USB Interrupt (w/ ENDPTCOMPLETE).**
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

Chapter 48
Watchdog Timer (WDOG)

This chapter describes a module integrated into an SoC. The chapter is intended for a module driver software developer. It describes module-level operation and programming. To understand how the module is integrated at the SoC level, a system software developer should see discussions of the module in the appropriate SoC-level chapter(s).

Preface

Terminology: Table 48-1 defines the following types of terms:

- New or invented terms, acronyms, and abbreviated terms
- Terms that are in common use, but are have an unusual meaning in the current context
- Terms that are unfamiliar to readers who are new to the module

Table 48-1. Definitions of Terms

Term	Definition
Active low	A signal is asserted when it changes to logic-level zero.
Active high	A signal is asserted when it changes to logic-level one.
Application	Software focus: A combination of operations and modes (generally one mode at a time) orchestrated by a programmer toward a goal or purpose.
Asserted	A discrete signal that is asserted is in an active logic state.
Mode	Hardware focus: A high-level module state that has broad implications for behavior. For example, a mode could affect available functions, protocol/signal timing, and the programming model (such as change or restrict the definitions and options in register fields).
Negated	A discrete signal that is negated is in an inactive logic state.
Operation	Hardware and low-level software focus: A low-level/atomic function provided by module hardware. Usually involves some minimal, low-level software to initialize parameters and initiate action (a basic operation). The action is then generally carried out by a state machine sequence.

48.1 Overview

This section briefly introduces the module. The full description of the module is in Section 48.4, “Functional Description.”

This section includes a top level diagram that shows the functional organization of the module, including all off-chip signals. Figure 48-1 is the block diagram.

The watchdog (WDOG) timer module protects against system failures by providing a method of escaping from unexpected events or programming errors. After the WDOG module is activated, it must be serviced

by the software on a periodic basis. If servicing does not take place, the timer times out. After a time-out, the WDOG Timer module asserts the internal system reset signal, wdog_rst, which goes to the System Reset Controller. There is also a provision for WDOG signal assertion by time-out counter expiration. There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter time-out is programmable. There is a power-down counter that gets enabled out of any reset (POR, Warm / Cold). This counter has a fixed time out period of 16 seconds after which it asserts the WDOG signal. Flow diagrams for the time-out counter, power-down counter, and interrupt operations are shown in Figure 48-11, Figure 48-12, and Figure 48-13.

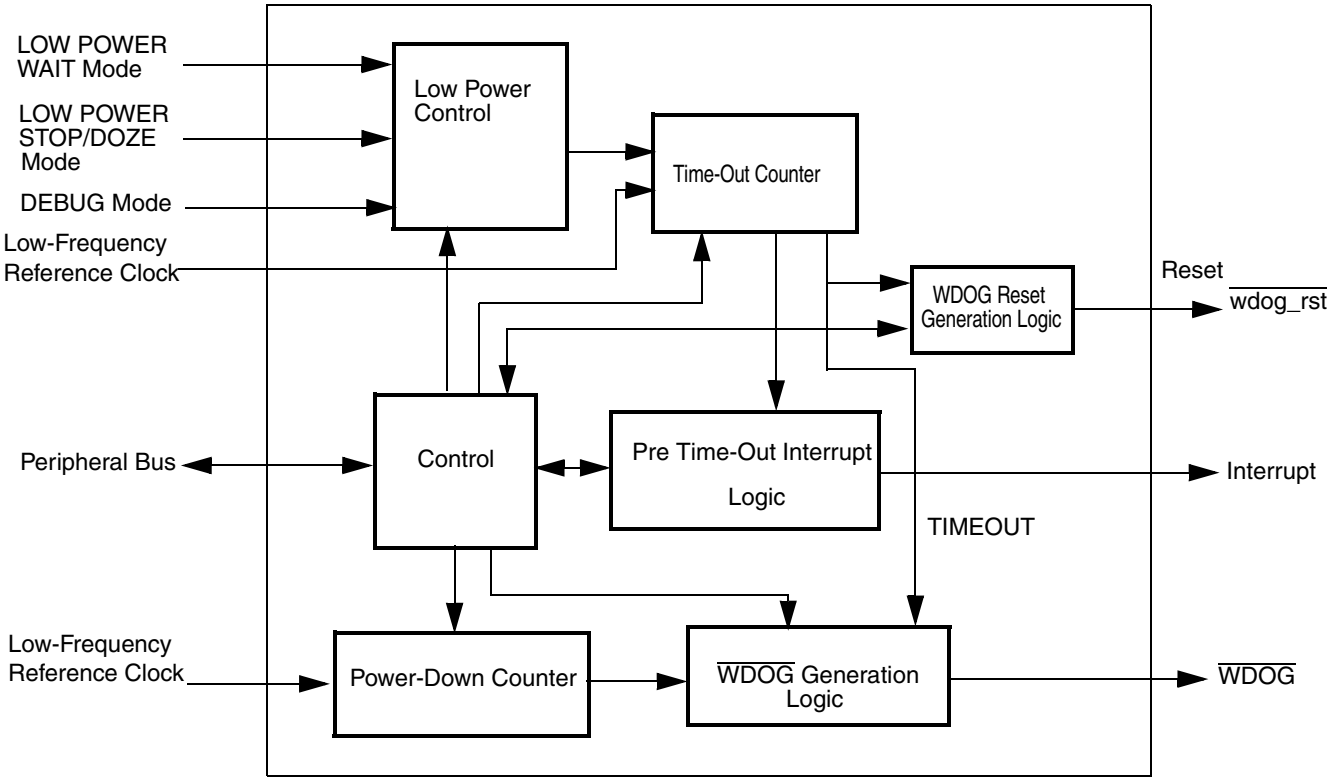


Figure 48-1. WDOG Block Diagram

48.1.1 Features

The WDOG features include the following:

- A configurable time-out counter with Time-out periods from 0.5 seconds up to 128 seconds and after time-out expiration results in assertion of the wdog_rst reset signal.
- Time resolution of 0.5 seconds.
- A configurable time-out counter that can be programmed to run or stop during low-power modes.
- A configurable time-out counter that can be programmed to run or stop during DEBUG mode.
- Programmable interrupt generation prior to time-out.
- The time duration between interrupt and time-out events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.

- A power-down counter with a fixed time-out period of 16 seconds which if not disabled after reset asserts the $\overline{\text{WDOG}}$ signal low.
  - The power-down counter is enabled out of any reset (POR, Warm / Cold reset) by default.

48.1.2 Modes and Operations

The WDOG supports the following modes described in the indicated sections:

- Section 48.4.4, "Low-Power Modes"
- Section 48.4.5, "Debug Mode"

As described in Section 48.4.6, "Operations," the WDOG supports the operations described in the indicated sections:

- Section 48.4.6.1, "Watchdog Reset Generation"
- Section 48.4.6.2, "WDOG_B Generation"

48.2 External Signals

Table 48-2. Off-Chip Module Signals

Signal	I/O	Description	Reset State[1]	Pull-Up/ Down[1]
$\overline{\text{WDOG}}$	O	This signal powers down the Chip. See Section 48.4.6.2, "WDOG_B Generation."	1	—
$\overline{\text{wdog\_rst}}$	O	This signal is a reset source for the chip. See Section 48.4.6.1, "Watchdog Reset Generation."	1	—

[1] The reset state values and pull-up/down requirements provided in this table are from the module-level perspective. To understand how the module is integrated at the SoC level, the system software developer must see discussions of the module in the appropriate SoC-level chapter(s). For example, a module signal that requires a pull-up could be integrated with a pull-up option built into the SoC. In this case, the system software developer must ensure the proper programming at the SoC level.

48.3 Memory Map and Register Definitions

The WDOG module has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Section 48.3.3, "Register Descriptions" provides the detailed descriptions for all of the WDOG registers. Byte operations can be performed on these registers. If a 32-bit access is performed on these registers then the WDOG does not generate any bus error and behaves normally, like a 16-bit access, making read/write possible. A 32-bit access should be avoided as the system might go to an unknown state.

48.3.1 Memory Map

Table 48-3 is the module memory map.

Table 48-3. Module Memory Map

Base Address Offset (Register Abbreviation)	Register	Access	Reset Value	Section/Page
0x0000 (WCR)	Watchdog Control Register	R/W	0x0030	48.3.3.1/48-5
0x0002 (WSR)	Watchdog Service Register	R/W	0x0000	48.3.3.2/48-7
0x0004 (WRSR)	Watchdog Reset Status Register	R	0x000X	48.3.3.3/48-7
0x0006 (WICR)	Watchdog Interrupt Control Register	R/W	0x0004	48.3.3.4/48-8
0x0008 (WMCR)	Watchdog Miscellaneous Control Register	R/W	0x0001	48.3.3.5/48-9

48.3.2 Register Summary

Table 48-4 is the register summary table.

Table 48-4. Module Register Summary

Offset (and Name Abbreviation)		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (WCR)	R	WT								WDW	0	WDA	SRS	WDT	WDE	WDBG	WDZST
	W																
0x0002 (WSR)	R	WSR															
	W																
0x0004 (WRSR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOUT	SFTW
	W																
0x0006 (WICR)	R	WIE	WTIS	0	0	0	0	0	0	WICT[7:0]							
	W		w1c														
0x0008 (WMCR)	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDE
	W																

48.3.3 Register Descriptions

This section provides detailed descriptions of the module's registers.

Register conventions: Figure 48-2 and Table 48-5 explain conventions used in register diagrams and tables.

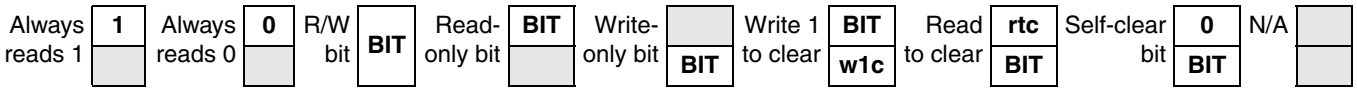


Figure 48-2. Register Field Conventions

48.3.3.1 Watchdog Control Register (WCR)

Table 48-5. General Register Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
BIT	Bit or field name. Its presence in the read or write row indicates that it can correspondingly be read or written.
Register Field Types	
R	Read only. Writing this bit has no effect.
W	Write only.
R/W	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
rtc	Read to clear. A read-only status bit that is automatically cleared when read.
Self-clearing bit	Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr)
Reset Values	
0	Resets to 0 (zero).
1	Resets to 1 (one).
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

The Watchdog Control Register (WCR) controls the WDOG operation. Figure 48-3 shows the register. Table 48-6 provides its field descriptions.

Offset 0x0000 (WCR) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WT								WDW	0	WDA	SRS	WDT	WDE	WDBG	WDZST
Reset	0	0	0	0	0	0	0	0	0	0	1	1	--	0	0	0

Figure 48-3. WDOG Control Register

NOTE

- The WDZST, WDBG, and WDW fields are write once only bits. After the software performs a write access to these bits, all these bits are locked and cannot be reprogrammed until the next system-reset assertion.
- WDE is a write one, once only bit. After software writes a 1 to this bit, the bit cannot be reset/cleared until the next system reset.
- WDT is also a write one, once only bit. After software writes a 1 to this bit, the bit cannot be reset/cleared until the next POR (Power-on Reset). This bit does not gets reset/cleared by any system reset.

Table 48-6. WDOG Control Register Descriptions

Field	Description
15–8 WT	Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the watchdog counter after the service routine has been performed or after the watchdog timer is enabled. After reset, WT[7:0] must have a value written to it before enabling the watchdog timer. Otherwise, the default count value is loaded into the counter. 0x00 0.5 Seconds (default) 0x01 1.0 Seconds. 0x02 1.5 Seconds. 0x03 2.0 Seconds. … 0xFF 128 Seconds. The time-out value can be written at any point in time, but it is loaded to the counter when WDOG is enabled or after the service routine has been performed. For more information, see Section 48.4.1, “Time-Out Event.”
7 WDW	Watchdog Disable for Wait. This bit determines the operation of WDOG during low-power WAIT mode. This is a write once only bit. 0 Continue WDOG timer operation (default). 1 Suspend WDOG timer operation.
6	Reserved.
5 WDA	$\overline{\text{WDOG}}$ assertion. Controls the software assertion of the $\overline{\text{WDOG}}$ signal. 0 Assert $\overline{\text{WDOG}}$ output. 1 No effect on system (default).
4 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal $\overline{\text{wdog\_rst}}$. This bit automatically resets to 1 after it has been asserted to 0. **Note:** This bit does not generate a software reset to the module. 0 Assert system reset signal. 1 No effect on the system (default).
3 WDT	$\overline{\text{WDOG}}$ Time-out assertion. Determines if the $\overline{\text{WDOG}}$ signal gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. 0 No effect on $\overline{\text{WDOG}}$ (default). 1 Assert $\overline{\text{WDOG}}$ upon a Watchdog Time-out event. **Note:** There is no effect on $\overline{\text{wdog\_rst}}$ (WDOG Reset) upon writing on this bit. The $\overline{\text{WDOG}}$ signal gets asserted along with $\overline{\text{wdog\_rst}}$ if this bit is set.

Table 48-6. WDOG Control Register Descriptions (continued)

Field	Description
2 WDE	Watchdog Enable. Enables or disables the WDOG module. This is a write one once only bit. It is not possible to clear this bit by a software write after the bit is set. **Note:** This bit can be set/reset in debug mode (exception). 0 Disable the Watchdog (default). 1 Enable the Watchdog.
1 WDBG	Watchdog DEBUG Enable. Determines the operation of the WDOG module during DEBUG mode. This bit is write once-only. 0 Continue WDOG timer operation (default). 1 Suspend the watchdog timer.
0 WDZST	Watchdog Low-Power. Determines the operation of the WDOG module during low-power modes. This bit is write once-only. **Note:** The WDOG module can continue or suspend the timer operation in the low-power modes (STOP mode). 0 Continue timer operation (default). 1 Suspend the watchdog timer.

48.3.3.2 Watchdog Service Register (WSR)

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the time-out condition. Figure 48-4 shows the register, Table 48-7 provides its field descriptions.

Offset 0x0002 (WSR) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 48-4. Watchdog Service Register (WSR)

Table 48-7. Watchdog Service Register Description

Field	Description
15–0 WSR	Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: Write 0x 5555 to the Watchdog Service Register (WSR). Write 0x AAAA to the Watchdog Service Register (WSR).

NOTE

Executing the service sequence reloads the WDOG time-out counter.

48.3.3.3 Watchdog Reset Status Register (WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR is always asserted high. The register always indicates

the source of the last reset generated due to WDOG. Read access to this register is with one wait state. Any write performed on this register generate a Peripheral Bus Error.

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Watchdog Time-out
- Software Reset

Figure 48-5 shows the register; Table 48-8 provides its field descriptions.

Offset 0x0004 (WRSR) Access: User read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOUT	SFTW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—	—

Figure 48-5. Watchdog Reset Status Register (WRSR)

Table 48-8. Watchdog Reset Status Register Description

Field	Description
15–2	Reserved.
1 TOUT	Time-out. Indicates whether the reset is the result of a WDOG time-out. 0 Reset is not the result of a WDOG time-out. 1 Reset is the result of a WDOG time-out.
0 SFTW	Software Reset. Indicates whether the reset is the result of a WDOG software reset by asserting SRS bit 0 Reset is not the result of a software reset. 1 Reset is the result of a software reset.

48.3.3.4 Watchdog Interrupt Control Register (WICR)

The WICR controls the WDOG interrupt generation.

Figure 48-6 shows the register. Table 48-9 provides its field descriptions.

Offset 0x0006 (WICR) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WIE	WTIS	0	0	0	0	0	0	WICT[7:0]							
W		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 48-6. Watchdog Interrupt Control Register (WICR)

Table 48-9. Watchdog Interrupt Control Register Description

Field	Description
15 WIE	Watchdog Timer Interrupt enable bit. Reset value is 0. 0 Disable Interrupt (default). 1 Enable Interrupt. **Note:** This bit is a write once only bit. After the software makes a write access to this bit, the bit is locked and cannot be reprogrammed until the next system reset assertion.
14 WTIS	Watchdog TImer Interrupt Status bit reflects the timer interrupt status, whether interrupt has occurred or not. After the interrupt has been triggered, software must clear this bit by writing 1 to it. 0 No interrupt has occurred (default). 1 Interrupt has occurred
13–8	Reserved.
7–0 WICT	Watchdog Interrupt Count Time-out (WICT) field determines how long before the counter time-out the interrupt must occur. The reset value is 0x04 and implies that an interrupt occurs 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. WICT[7:0] = 0x00 Time duration between interrupt and time-out is 0 seconds. WICT[7:0] = 0x01 Time duration between interrupt and time-out is 0.5 seconds. … WICT[7:0] = 0x04 Time duration between interrupt and time-out is 2 seconds (default). … WICT[7:0] = 0xFF Time duration between interrupt and time-out is 127.5 seconds. **Note:** This field is write once only. After the software does a write access to this field, the field is locked and cannot be reprogrammed until the next system reset assertion

48.3.3.5 Watchdog Miscellaneous Control Register (WMCR)

WMCR Controls the power-down counter operation.

Figure 48-7 shows the Register. Table 48-10 shows the description.

Offset 0x0008 (WMCR) Access: User read-write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 48-7. Watchdog Miscellaneous Control Register

Table 48-10. Watchdog Miscellaneous Control Register Description

Field	Description
15–1	Reserved.
0 PDE	Power-Down Enable bit. Reset value of this bit is1 which means the power-down counter inside the WDOG is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. After it is disabled, this counter cannot be enabled again. See Section 48.4.3, “Power-Down Counter Event,” for operation of this counter. 0 Power-Down Counter of WDOG is disabled. 1 Power-Down Counter of WDOG is enabled (default). **Note:** This bit is a write one once only bit. After software sets this bit, it cannot be reset until the next system reset.

48.4 Functional Description

48.4.1 Time-Out Event

The WDOG provides time-out periods from 0.5 seconds up to 128 seconds with a time resolution of 0.5 seconds. The user can determine the time-out period by writing to the WDOG Time-out field (WT[7:0]) in the Watchdog Control Register (WCR).” The WDOG has to be enabled by setting the WDE bit of Watchdog Control Register (WCR),” for the time-out counter to start running. After the WDOG is enabled, the counter is activated, loads the time-out value and begins to count down from this programmed value. The timer times out when the counter reaches zero and the WDOG outputs a system reset signal, wdog_rst and asserts WDOG (the WDT bit is set in the Watchdog Control Register (WCR)”).

However the Time-out condition can be prevented by reloading the counter with the new time-out value(WT[7:0] of WCR) if a service routine (See Section 48.4.1.1, “Servicing The WDOG To Reload The Counter”) is performed before the counter reaches zero. If any system errors occur that prevent the software from servicing the Watchdog Service Register (WSR),” then the time-out condition occurs. By performing the service routine, the WDOG reloads its counter to the time-out value indicated by bits WT[7:0] of the Watchdog Control Register (WCR),” and it re-starts the countdown.

A system reset resets the counter and places it in the idle state at any time during the countdown. The counter flow diagram is shown in Figure 48-11.

NOTE

The time-out value is reloaded to the counter at the time when WDOG is enabled or after the service routine has been performed.

48.4.1.1 Servicing The WDOG To Reload The Counter

The proper service sequence to reload a time-out value to the counter begins by writing 0x 5555 followed by 0x AAAA to the Watchdog Service Register (WSR).” Any number of instructions can be executed between the two writes. If the WSR is not loaded with 0x 5555 prior to writing 0x AAAA to the WSR, the counter is not reloaded. If any value other than 0x AAAA is written to the WSR after 0x 5555, the counter is not reloaded. This service sequence reloads the counter with the time-out value WT[7:0] of the Watchdog Control Register (WCR).” The time-out value can be changed at any point of the time and the counter reloads this value whenever the core services the Watchdog.

48.4.2 Interrupt Event

Prior to time-out the WDOG can generate an interrupt which can be considered as a warning signal to indicate that the time-out occurs shortly. The time duration between the interrupt event and the time-out event can be controlled by writing to the WICT field of the Watchdog Interrupt Control Register (WICR).” It can vary between 0 and 127.5 seconds. If the WDOG is serviced (Section 48.4.1.1, “Servicing The WDOG To Reload The Counter”) before interrupt generation, then the counter is reloaded with the time-out value WT[7:0] of the Watchdog Control Register (WCR),” and an interrupt is not triggered.

48.4.3 Power-Down Counter Event

The power-down counter inside WDOG is enabled out of reset. This counter has a fixed time out value of 16 seconds, after which it drives the $\overline{\text{WDOG}}$ signal low to prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control Register (WMCR)," within 16 seconds of reset de-assertion. After it is disabled, this counter cannot be enabled again until the next system reset occurs. This feature is provided to prevent cores being hung up after reset, as WDOG is not enabled out of reset.

48.4.4 Low-Power Modes

48.4.4.1 STOP Mode

If the WDOG Timer Disable bit for Low-power STOP mode (WDZST) bit in the Watchdog Control Register (WCR)," is 0, the WDOG Timer continues to operate using the Low-Frequency Reference clock. If the Low-power Enable (WDZST) bit is set to 1, then the WDOG Timer operation is suspended in Low-power STOP mode. Upon exiting Low-power STOP mode, the WDOG operation returns to what it was prior to entering the STOP mode.

48.4.4.2 WAIT Mode

If the WDOG Timer Disable bit for low-power WAIT mode (WDW) bit in the Watchdog Control Register (WCR)," is 0, the WDOG Timer continues to operate using the low-frequency reference clock. If the low-power WAIT Enable (WDW) bit is set to 1, then the WDOG Timer operation is suspended. Upon exiting low-power WAIT mode, the WDOG operation returns to what it was prior to entering the WAIT mode.

NOTE

WDOG Timer is not able to detect events that occur for periods less than one low-frequency reference clock cycle. For example, in repeated WAIT mode entry/exit, if the RUN mode time is less than one low-frequency reference clock cycle and if the WDW bit is set, then the WDOG timer may never time out even though the system is in RUN mode for a finite duration, because the WDOG timer may not see any low-frequency reference clock edge during its wake time.

48.4.5 Debug Mode

The WDOG Timer can be configured for continual operation, or the operation can be suspended during debug mode. If the WDOG debug enable (WDBG) bit is set to 1 in the Watchdog Control Register (WCR)," the WDOG Timer operation is suspended in debug mode. In the case of the WDBG bit being set to 1 and Debug mode being entered, WDOG Timer operation is suspended after 2 low-frequency reference clocks and similarly WDOG Timer operation is continued after 2 low-frequency reference clocks of Debug mode Exit. Register read and write accesses in Debug mode continue to function normally. Also, while in DEBUG mode, the WDE bit of Watchdog Control Register (WCR)," can be enabled-disabled directly. If

the WDOG debug enable (WDBG) bit is cleared then WDOG Timer operation is not suspended. Power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Section 48.3.3.1, "Watchdog Control Register (WCR)," is set/cleared while in DEBUG mode, it remains set/cleared even after exiting DEBUG mode.

48.4.6 Operations

This section describes the module's operations.

48.4.6.1 Watchdog Reset Generation

The WDOG generated reset signal $\overline{\text{wdog\_rst}}$ is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control Register (WCR)."
- WDOG time-out. See Section 48.4.1, "Time-Out Event."

The $\overline{\text{wdog\_rst}}$ signal is asserted for one clock cycle of the low-frequency reference clock for both the time-out Condition and software write occurrence. It remains asserted for 1 clock cycle of the low-frequency reference clock, even if a system reset is asserted in between. Figure 48-9 shows the timing diagram of this signal due to a time-out condition.

48.4.6.2 WDOG_B Generation

The WDOG module asserts $\overline{\text{WDOG}}$ under the following scenarios:

- Software write to WDA bit of Watchdog Control Register (WCR)." $\overline{\text{WDOG}}$ Signal remains asserted as long as the WDA bit is 0.
- WDOG time-out condition, WDT bit of Watchdog Control Register (WCR), must be set for this scenario. Description of the time-out condition can be found in the Section 48.4.1, "Time-Out Event." $\overline{\text{WDOG}}$ Signal remains asserted until a Power-on Reset (POR) occurs. It gets cleared after the POR occurs and not due to any other system reset. Figure 48-10 shows the timing diagram of $\overline{\text{WDOG}}$ due to time-out condition.
- WDOG Power-down Counter time-out, PDE bit of Watchdog Miscellaneous Control Register (WMCR), is not cleared in this scenario. Description of this counter can be found in the Power-Down Counter Event. The $\overline{\text{WDOG}}$ Signal remains asserted for one clock cycle of the low-frequency reference clock.

The Figure 48-8 shows the scenarios under which $\overline{\text{WDOG}}$ gets asserted.

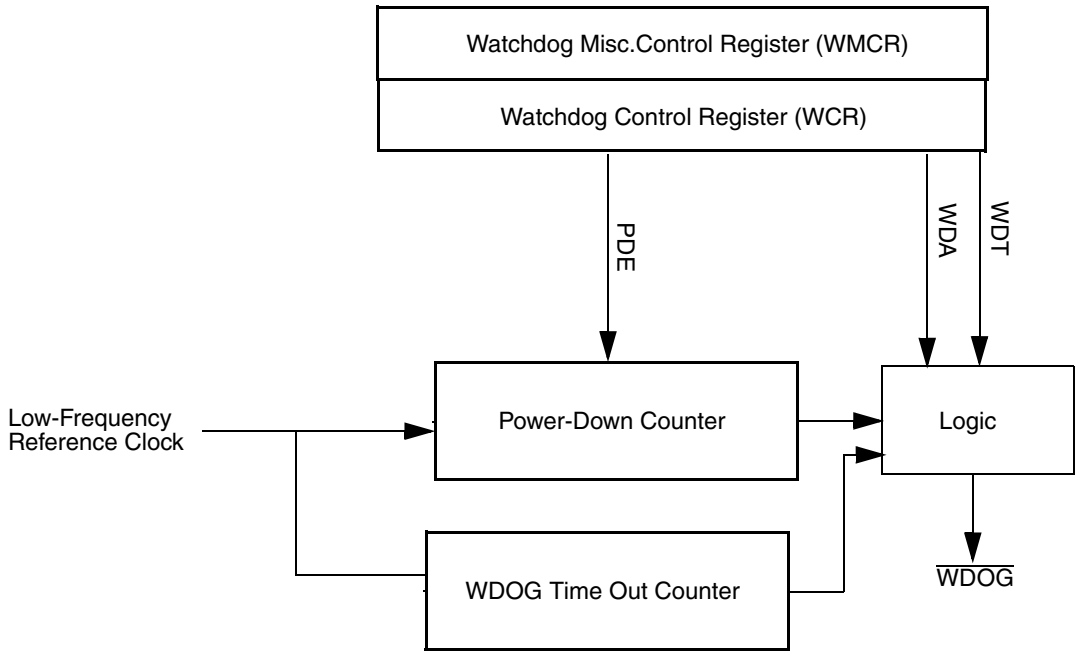


Figure 48-8. $\overline{\text{WDOG}}$ Generation

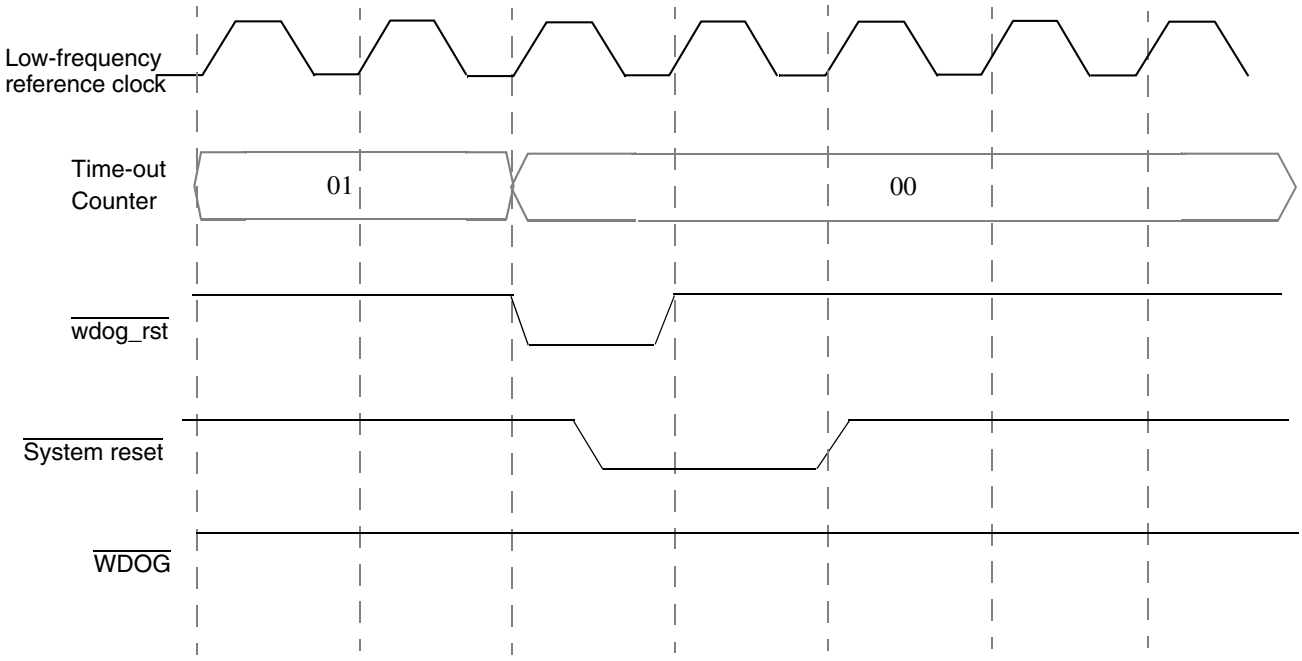


Figure 48-9. WDOG Time-Out Condition/WDT Bit Is Not Set

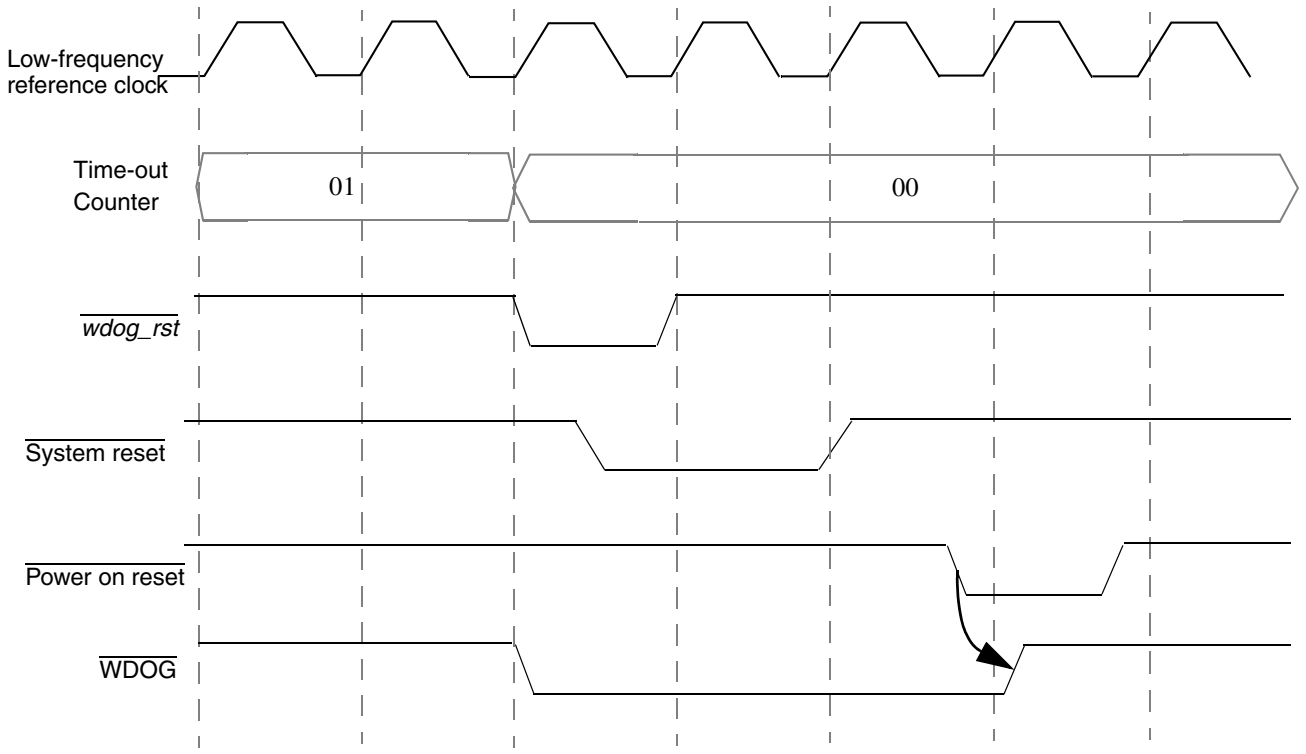


Figure 48-10. WDOG Time-Out Condition/WDT Bit Is Set

48.4.7 Clocks

This section describes clocks and special clocking requirements of the module.

The WDOG module uses the low-frequency reference clock for its counter and control operations. Peripheral Bus clock is used for register Read/Write operations.

The low-frequency reference clock is a free running clock and cannot be gated. The Peripheral Bus clock cannot be gated and is selectively switched ON whenever Read/Write operations take place.

48.4.8 Reset

This section describes how to reset the module and explains special requirements related to reset.

The module is reset by a system reset and has the following consequences:

- WDOG counter is disabled.
- The Power-Down counter is enabled and starts counting.

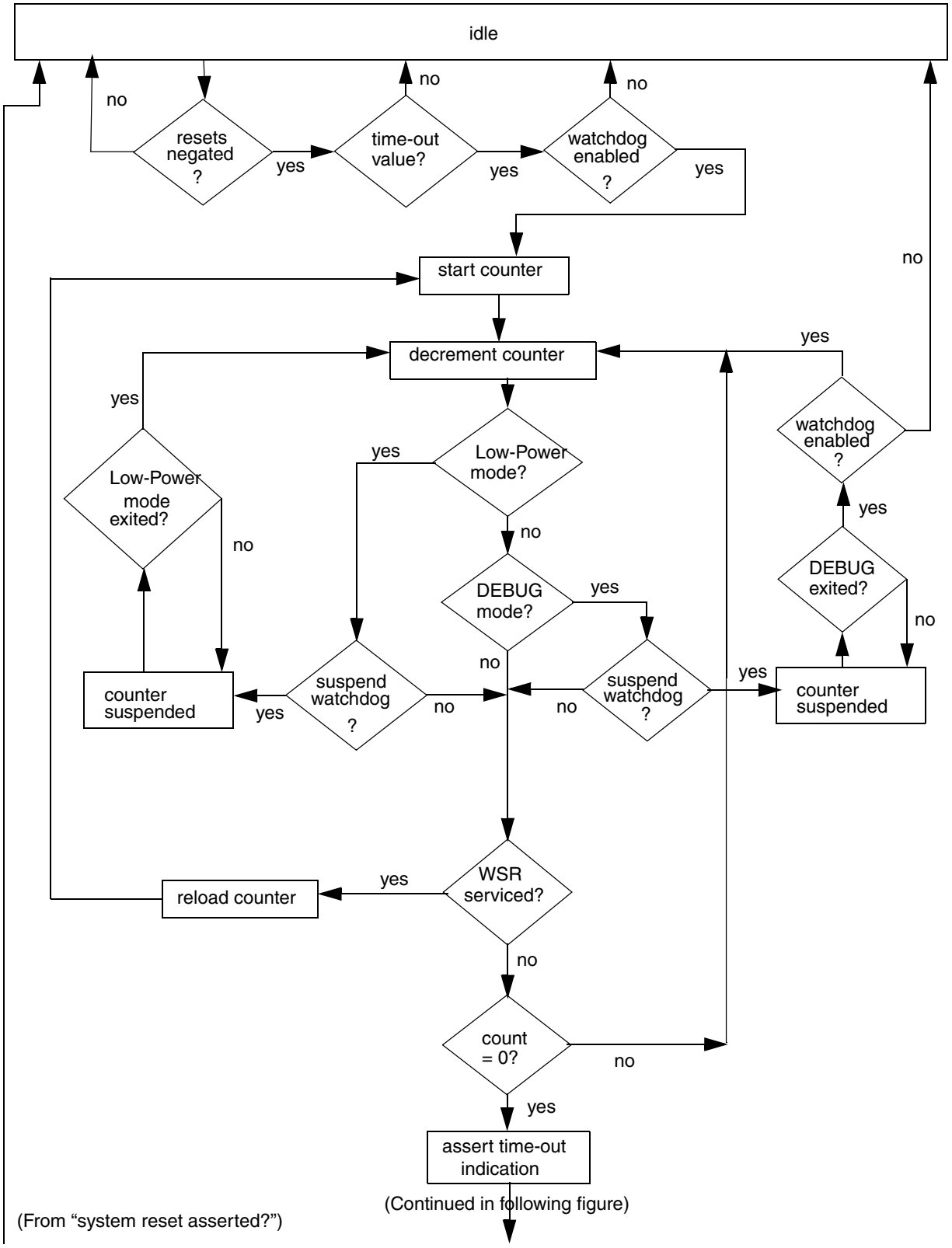
48.4.9 Interrupt

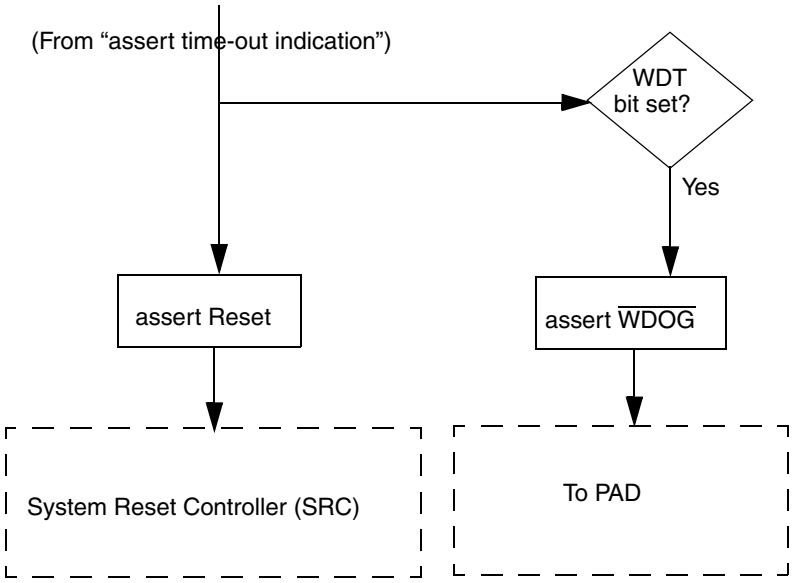
The WDOG module has the feature of Interrupt generation before time-out. The interrupt is generated only if the WIE bit in the Watchdog Interrupt Control Register (WICR) is set. The exact time at which the interrupt occurs prior to time-out depends on the value of the WICT field of the Watchdog Interrupt

Control Register (WICR). As an example, if the WICT field has a value of 0x04, then the interrupt is generated 2 seconds prior to time-out. After the interrupt is triggered, the WTIS bit in the Watchdog Interrupt Control Register (WICR) is set. The software need to clear this bit to de-assert the Interrupt. If the WDOG is serviced before the interrupt generation then the counter is reloaded with the time-out value WT[7:0] of Watchdog Control Register (WCR) and an interrupt is not be triggered.

48.4.10 Flow Diagrams

A flow diagram of watchdog operation is shown in Figure 48-11, Figure 48-12 and Figure 48-13.


idle
no
resets negated ?
yes
no
time-out value?
yes
no
watchdog enabled ?
yes
start counter
no
decrement counter
yes
yes
Low-Power mode exited?
no
yes
Low-Power mode?
no
DEBUG mode?
yes
no
suspend watchdog ?
yes
counter suspended
no
no
suspend watchdog ?
yes
counter suspended
yes
watchdog enabled ?
yes
DEBUG exited?
no
WSR serviced?
yes
reload counter
no
count = 0?
no
yes
assert time-out indication
(Continued in following figure)
(From “system reset asserted?”)



NOTE: A system reset forces the state machine to “idle” at any time during countdown.

Figure 48-11. Time-Out Counter Flow Diagram

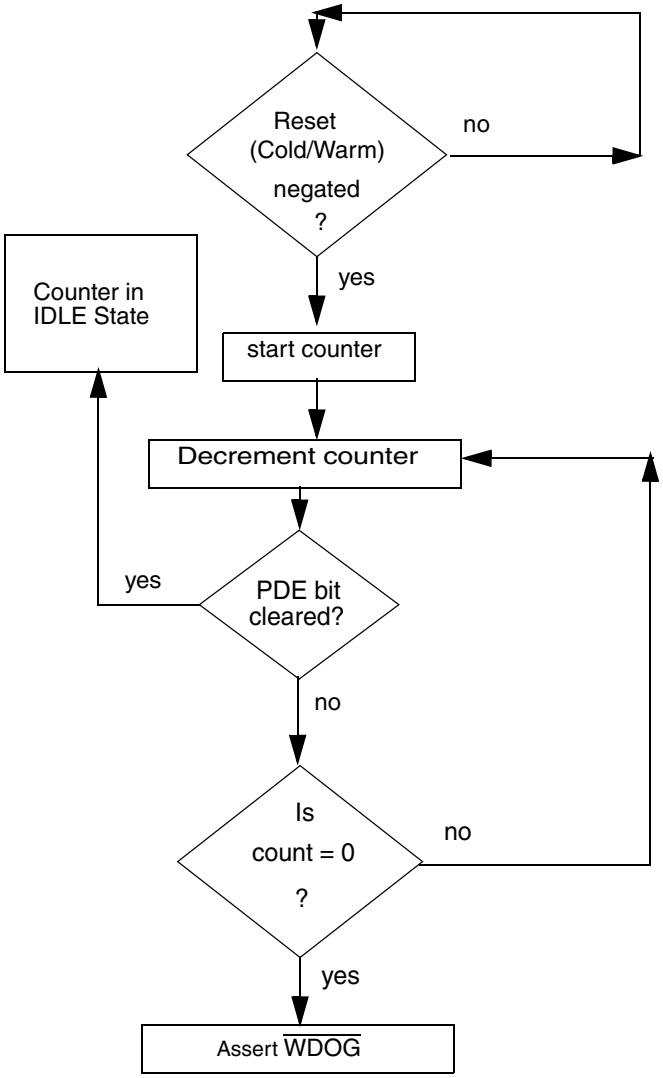


Figure 48-12. Power-Down Counter Flow Diagram

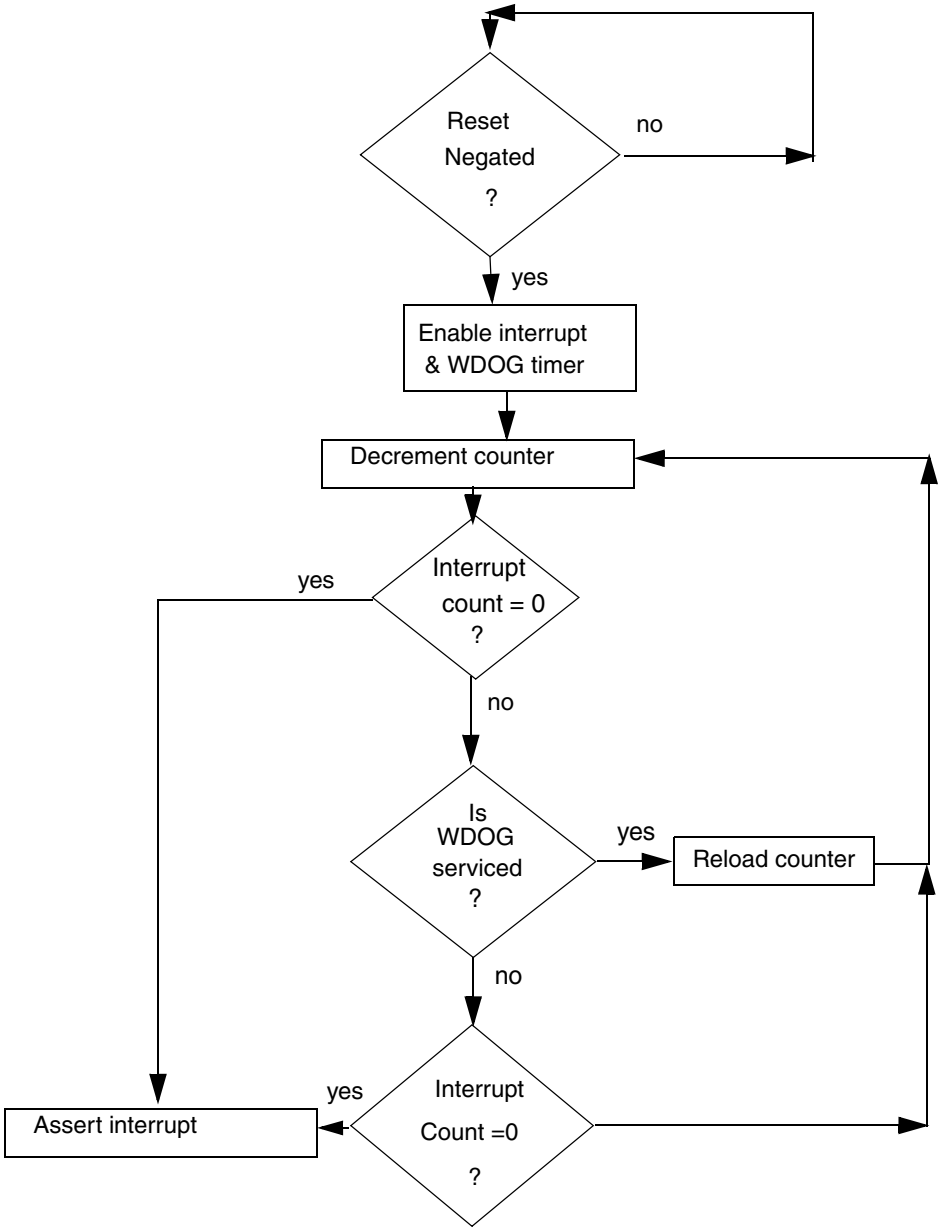


Figure 48-13. Interrupt Generation Flow Diagram

48.5 Initialization

The following programming sequence initializes the WDOG module:

- The WMCR[PDE] bit is cleared to disable the power-down counter.
- The WMCR[WDT] field is programmed for a sufficient time-out value.
- The WDOG module is enabled by setting the WMCR[WDE] bit so that the time-out counter loads the WMCR[WDT] field value and starts counting.

Chapter 49
Wireless External Interface Module (WEIM)

This chapter describes the wireless external interface module (WEIM) implemented on this device, and covers the following topics:

- Section 49.2, “External Signal Description”
- Section 49.3, “Memory Map and Register Definition”
- Section 49.4, “Functional Description”
- Section 49.5, “Initialization/Application Information”

49.1 Overview

The wireless external interface module (WEIM) handles the interface to devices external to the chip, including generation of chip selects, clock and control for external peripherals and memory. It provides asynchronous and synchronous access to devices with SRAM-like interface.

Figure 49-1 shows a top-level WEIM block diagram. The external signals shown in Figure 49-1 are described in Table 49-1.

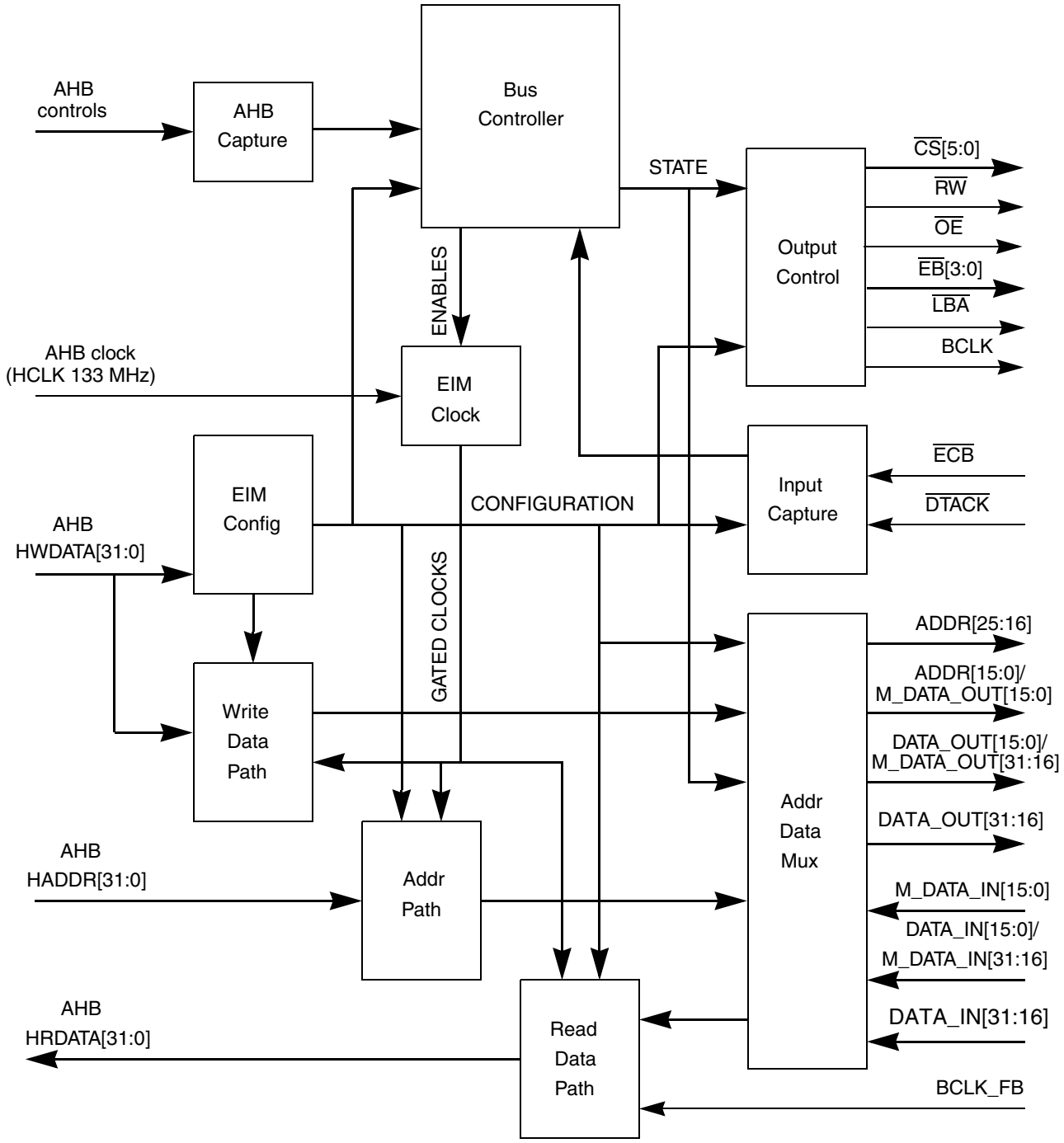


Figure 49-1. WEIM Block Diagram

49.1.1 Features

The WEIM includes the following features:

- Six chip selects for external devices, with $\overline{CS0}$ and $\overline{CS1}$ each covering a range of 128 Mbytes and $\overline{CS2}$–$\overline{CS5}$ each covering a range of 32 Mbytes
- $\overline{CS0}$ range can be increased to 256 Mbytes when collapsed with $\overline{CS1}$
- Selectable protection for each chip select
- Programmable data port size for each chip select
- Asynchronous accesses with programmable setup and hold times for control signals
- Synchronous memory burst read mode support for AMD, Intel, and Micron burst Flash memory
- Synchronous memory burst write mode support for PSRAM (CellularRAM™ from Micron, Infineon, and Cypress), fixed write latency support
- Support for multiplexed address / data bus operation
- External cycle termination/postpone with $\overline{DTACK}$ signal
- Programmable wait-state generator for each chip select
- Support for big-endian and little-endian modes of operation per access
- ARM AHB slave interface

49.1.2 Modes of Operation

The WEIM has six modes of operation. No dedicated low-power mode is included, because most of the clocks are gated when there are no accesses to the WEIM, providing maximum energy conservation.

The WEIM modes of operation are as follows:

- Asynchronous mode. This is a non-burst mode is used for NOR Flash and SRAM access. In this mode a single data is read/written with each access (asserted address). Timings are controlled by preset values in the chip select control registers. In this mode, the size of data writes should match the external data bus width. For example, only half-word writes should be performed if the external memory bus width is 16 bits.
- Synchronous read mode. This is a burst mode is used for reading Flash memory devices. In this mode, after address assertion a burst of sequential data can be read. Data exchange is carried out according to BCLK clock generated by WEIM. An access may be delayed by the assertion of the external $\overline{ECB}$ signal after the first word of data.
- Page mode.This mode is used for memory burst read, but the address is asserted for each data in the burst as $\overline{LBA}$ and BCLK operate asynchronously. In this mode the setup time is greater for the first data burst than for the remaining data in the burst (from the same page).
- Synchronous read/write mode for PSRAM synchronous mode and peripherals with write burst capability. In this mode both reads and writes are synchronous. Access may be additionally delayed according to the $\overline{ECB}$ state before the first piece of data arrives (refresh wait enable).
- DTACK mode. This is a non-burst mode used for PCMCIA accesses. In this mode WEIM waits for $\overline{DTACK}$ acknowledge until 1024 counts of the AHB clock have passed. In this mode $\overline{DTACK}$ can be used as either posedge or level sensitive, according to WSC field and EW bit settings.

- Multiplexed Address/Data mode. This mode supports multiplexing of addresses and data bits on the same pins for synchronous/asynchronous accesses to 32-bit data width memory devices.

49.2 External Signal Description

This section provides an overview and detailed description of WEIM external signals.

49.2.1 Overview

Table 49-1 lists input and output signals between the WEIM and the external devices.

Table 49-1. Signal Properties

Name	Port	Function	Direction	Reset State
ADDR[25:16]	—	Address bus MSB / $\overline{EB}$[3:2], CRE	Out	Low
ADDR[15:0]/ M_DATA_OUT[15:0]	—	Address bus LSB / Output data multiplexed bus LSB	Out	Low
BCLK	—	Burst Clock	Out	Low
BCLK_FB	—	Feedback burst clock	In	—
$\overline{CS}$[5:0]	—	Chip selects	Out	High
DATA_IN[31:16]	—	Input data bus MSB	In	—
DATA_IN[15:0]/ M_DATA_IN[31:16]	—	Input data bus LSB / Input data multiplexed bus MSB	In	—
DATA_OUT[31:16]	—	Output data bus MSB	Out	Low
DATA_OUT[15:0]/ M_DATA_OUT[31:16]	—	Output data bus LSB / Output data multiplexed bus MSB	Out	Low
$\overline{DTACK}$	—	Data transfer acknowledge / wait	In	—
$\overline{EB}$[3:0]	—	Enable Byte	Out	High
$\overline{ECB}$	—	End current burst / wait	In	—
$\overline{LBA}$	—	Load burst address	Out	High
M_DATA_IN[15:0]	—	Input data multiplexed bus LSB	In	—
$\overline{OE}$	—	Output enable	Out	High
$\overline{RW}$	—	Read/write	Out	High

49.2.2 Detailed Signal Descriptions

Table 49-2 gives a detailed description of the WEIM signals.

Table 49-2. WEIM Detailed Signal Descriptions

Signal	I/O	Description
ADDR[25:16]	IO	Address Bus MSB. These signals are used as address bits [25:16]. In multiplexed mode these signals do not change their state. If the corresponding AUS*x* bit is set in the WEIM control register, these signals reflect AHB address bits [25:16]. If the AUS*x* bit is not set, these signals represent AHB address bits [27:18] for word width memory, [26:17] for half-word width memory and [25:16] for byte-width memory. In the multiplexed mode (MUM = 1 in the chip select *x* additional control register), then ADDR[25:24] or ADDR[24:23] (depending on the CRER bit setting in the WEIM control register) are also used as $\overline{EB}$[3:2] outputs. ADDR[23] or ADDR[25] is also used as the CRE output in PSRAM mode (PSR=1).
ADDR[15:0]/ M_DATA_OUT[15:0]	IO	Multiplexed Address Bus LSB/Output Data Bus LSB. In non-multiplexed mode these signals are used as address bits [15:0]. If the corresponding AUS*x* bit is set in the WEIM control register, these signals reflect AHB address bits [15:0]. If the AUSx bit is not set, tThese signals reflect AHB address bits [17:2] for word width memory, [16:1] for half-word width memory and [15:0] for byte width memory. In multiplexed address/data mode these bits are multiplexed between address and output data bits [15:0]. The behavior in multiplexed address/data mode is affected by the LBA, LBN and LAH fields in the chip select control registers. In synchronous multiplexed mode the behavior is affected by the BCS, BCD and LAH fields. The bidirectional LSB address/data bus is made in IO PAD from M_DATA_IN[15:0] and ADDR[15:0]/M_DATA_OUT[15:0].
BCLK	O	Burst Clock. This active-high output signal BCLK is used to clock external, burst-capable devices to synchronize the loading and incrementing of addresses and delivery of burst read and write data to/from the WEIM. Its behavior is affected by the BCM bit in the WEIM configuration register and the SYNC, BCD and BCS fields in the chip select control registers. BCLK can start on both rising and falling edge of HCLK.
BCLK_FB	I	Burst Clock Feedback. This input is used to provide the input data sampling clock at high data speeds. It is an internal feedback from the I/O pin of the BCLK output signal that is used to align the clock used by the memory with the clock used to sample the read data.
$\overline{CS}$[5:0]	O	Chip selects. The $\overline{CS}$[5:0] signals are chip selects active-low output pins. The behavior is affected by the CSA and CSN fields in the chip select control registers. The $\overline{CS}$[0] address space range can be increased to 256 Mbytes by merging the $\overline{CS}$[0] and $\overline{CS}$[1] ranges. In this case the merged address space (MAS) bit is set in the WEIM configuration register, and the $\overline{CS}$[1] pin is used as address line A26. As shown in Table 49-5, the chip select signals are asserted based on a decode of address lines [31:24] (when enabled).
DATA_IN[31:16]	IO	Input Data Bus MSB. These signals are the MSB input data bus used to transfer data from external devices. Bidirectional MSB data bus are made in IO PAD from DATA_IN[31:16] and DATA_OUT[31:16].
DATA_IN[15:0]/M_DATA_IN[31:16]	IO	Input Data Bus LSB / Input Data Multiplexed Bus MSB. These signals are the input data bus used to transfer data from external devices. In a non multiplexed mode it is LSB, in a multiplexed mode it is MSB. The bidirectional LSB data bus are made in the IO PAD from DATA_IN[15:0]/M_DATA_IN[31:16] and DATA_OUT[15:0]/M_DATA_OUT[31:16].
DATA_OUT[31:16]	O	Output Data Bus MSB. Those signals are MSB output data bus used to transfer data to an external devices. Bidirectional MSB data bus are made in IO PAD from DATA_IN[31:16] and DATA_OUT[31:16].

Table 49-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
DATA_OUT[15:0]/M_DATA_OUT[31:16]	O	Output Data Bus LSB / Output Data Multiplexed Bus MSB. These signals are the output data bus used to transfer data to an external devices. In non-multiplexed mode it is LSB, and in multiplexed mode it is MSB. The bidirectional LSB data bus is made in IO PAD from DATA_IN[15:0]/M_DATA_IN[31:16] and DATA_OUT[15:0]/M_DATA_OUT[31:16].
$\overline{\text{DTACK}}$	I	Data Transfer Acknowledge. This input signal is used to externally terminate a data transfer when enabled. For $\overline{\text{DTACK}}$ enabled cycles, the bus time-out monitor generates a bus error if $\overline{\text{DTACK}}$ is asserted and is not negated before 1024 clocks have elapsed. This signal is used in two modes: with rising edge detection or with level detection with an insensitiveness time. Edge detection mode is used for devices like PCMCI cards. Level detection mode is used for asynchronous devices like ATI graphics controllers. $\overline{\text{DTACK}}$ control keeps backward compatibility with previous architectures that used it in the Application processors.
$\overline{\text{EB}}$[3:0]	O	Enable Byte. Those active-low output pins indicate active data bytes for the current access. They may be configured to assert for both read and write cycles or for write cycles only, as programmed in the chip select control registers. $\overline{\text{EB}}$[0] corresponds to DATA_OUT[7:0] and M_DATA_OUT[7:0]. $\overline{\text{EB}}$[1] corresponds to DATA_OUT[15:8] and M_DATA_OUT[15:8]. $\overline{\text{EB}}$[2] corresponds to DATA_OUT[23:16] and M_DATA_OUT[23:16]. $\overline{\text{EB}}$[3] corresponds to DATA_OUT[31:24] and M_DATA_OUT[31:24]. In the multiplexed mode (MUM = 1), $\overline{\text{EB}}$[3:2] also are multiplexed to the ADDR[25:24] bits (or ADDR[24:23], depending on the CRER bit setting in the WEIM control register). For write accesses the behavior of $\overline{\text{EB}}$ is affected by the EBWA, EBWN, and EBC fields in the chip select control registers. For read accesses the behavior of $\overline{\text{EB}}$ is affected by the EBRA and EBRN fields in the chip select control registers.
$\overline{\text{ECB}}$	I	End Current Burst (WAIT). This active-low input signal $\overline{\text{ECB}}$ is asserted by external burst capable devices. It is serviced in synchronous mode only (SYNC=1). This signal can be used in two different modes depending on the EW bit setting in the chip select control register, as follows: • In ECB mode (EW=0) $\overline{\text{ECB}}$ indicates the end of the current (continuous) burst sequence. Following assertion, the WEIM terminates the current burst sequence and initiate a new one. • In wait mode (EW=1) the memory device asserts this signal to insert wait states during refresh collisions or during a row boundary crossing. Following assertion, the WEIM does not terminate the current burst sequence and continues it once WAIT is negated. $\overline{\text{ECB}}$ has a pull-up resistor in I/O. For burst devices $\overline{\text{ECB}}$/WAIT output should be configured to change one cycle before data is ready (before delay).
$\overline{\text{LBA}}$	O	Load Burst Address. This active-low output signal is asserted during burst mode accesses to cause the external burst capable device to load a new starting burst address. Assertion of $\overline{\text{LBA}}$ indicates that a valid address is present on the address bus. Its behavior is affected by the SYNC bit, BCD, BCS, LBA and LBN fields in the Chip Select control registers. In asynchronous mode (SYNC=0) $\overline{\text{LBA}}$ length decreased by LBA and LBN fields. In the synchronous mode (SYNC=1) $\overline{\text{LBA}}$ length is equal BCD + BCS + LBN + 1 half AHB clock cycles.
M_DATA_IN[15:0]	I	Multiplexed Data Input Bus. These signals are the LSB input data bus used to transfer data from an external devices in multiplexed mode. The bidirectional LSB address/data bus is made in IO PAD from M_DATA_IN[15:0] and ADDR[15:0]/M_DATA_OUT[15:0].

Table 49-2. WEIM Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{OE}$	O	Output Enable. This active-low output signal $\overline{OE}$ indicates the bus access is a read, and enables slave devices to drive the data bus with read data. Its behavior is affected by the OEA and OEN fields in the Chip Select control registers.
$\overline{RW}$	O	Read/Write. The $\overline{RW}$ output signal indicates if the current bus access is a read or write cycle. A high (logic one) level indicates a read cycle and a low (logic zero) level indicates a write cycle. Its behavior is affected by the RWA and RWN fields in the Chip Select control registers.

Table 49-3. WEIM Out/In Data in Case AHB Out/In Data is 0xB3B2B1B0

Endian mode	AHB access	AHB address [1:0]	Port Size And Used Bits								
			Word Port				Half Word Port			Byte Port	
			[31:24]	[23:16]	[15:8]	[7:0]	External address [0]	[31:24] ([15:8])	[23:16] ([7:0])	External address [1:0]	[31:24] ([15:8], [23:16], [7:0])
Little	Word	0	**0xB3**	**0xB2**	**0xB1**	**0xB0**	0	**0xB1**	**0xB0**	0	**0xB0**
										1	**0xB1**
							1	**0xB3**	**0xB2**	2	**0xB2**
										3	**0xB3**
	Half Word	0	—	—	**0xB1**	**0xB0**	0	**0xB1**	**0xB0**	0	**0xB0**
										1	**0xB1**
		2	**0xB3**	**0xB2**	—	—	1	**0xB3**	**0xB2**	2	**0xB2**
										3	**0xB3**
	Byte	0	—	—	—	**0xB0**	0	—	**0xB0**	0	**0xB0**
		1	—	—	**0xB1**	—		**0xB1**	—	1	**0xB1**
		2	—	**0xB2**	—	—	1	—	**0xB2**	2	**0xB2**
		3	**0xB3**	—	—	—		**0xB3**	—	3	**0xB3**

49.3 Memory Map and Register Definition

The WEIM module includes 19 user-accessible 32-bit registers. These registers are accessible only in supervisor mode, with 32-bit reads and writes.

The WEIM registers can be summarized as follows:

- The WEIM configuration register (WCR) contains control bits that configure the WEIM for different modes of operation.

In addition, each chip select has three chip select control registers denoted as upper, lower, and additional respectively. These registers are identical for chip selects 1–5, but for chip select 0 the registers' reset states depend on the boot configuration inputs, for more details, refer Chapter 7, "System Boot."

Complete decoding is not performed, so shadowing can occur with these registers. The user should not attempt to address these registers at any other address location other than those listed in the Table 49-4 and Table 49-5.

49.3.1 Memory Map

The memory map for the WEIM is shown in the Table 49-4 and the memory map for the Chip Select memory ranges is shown in the Table 49-5.

Table 49-4. WEIM Memory Map

Address	Use	Access	Reset	Section/Page
0x0000 (CSCR0U)	Chip Select 0 Upper Control Register	R/W	0x0000_1E00	49.3.3.1/49-12
0x0004 (CSCR0L)	Chip Select 0 Lower Control Register	R/W	0xA000_0841[1]	49.3.3.2/49-18
0x0008 (CSCR0A)	Chip Select 0 Additional Control Register	R/W	0x0000_5000	49.3.3.3/49-22
0x0010 (CSCR1U)	Chip Select 1 Upper Control Register	R/W	0x0000_0000	49.3.3.1/49-12
0x0014 (CSCR1L)	Chip Select 1 Lower Control Register	R/W	0x0000_0000	49.3.3.2/49-18
0x0018 (CSCR1A)	Chip Select 1 Additional Control Register	R/W	0x0000_0000	49.3.3.3/49-22
0x0020 (CSCR2U)	Chip Select 2 Upper Control Register	R/W	0x0000_0000	49.3.3.1/49-12
0x0024 (CSCR2L)	Chip Select 2 Lower Control Register	R/W	0x0000_0000	49.3.3.2/49-18
0x0028 (CSCR2A)	Chip Select 2 Additional Control Register	R/W	0x0000_0000	49.3.3.3/49-22
0x0030 (CSCR3U)	Chip Select 3 Upper Control Register	R/W	0x0000_0000	49.3.3.1/49-12
0x0034 (CSCR3L)	Chip Select 3 Lower Control Register	R/W	0x0000_0000	49.3.3.2/49-18
0x0038 (CSCR3A)	Chip Select 3 Additional Control Register	R/W	0x0000_0000	49.3.3.3/49-22
0x0040 (CSCR4U)	Chip Select 4 Upper Control Register	R/W	0x0000_0000	49.3.3.1/49-12
0x0044 (CSCR4L)	Chip Select 4 Lower Control Register	R/W	0x0000_0000	49.3.3.2/49-18
0x0048 (CSCR4A)	Chip Select 4 Additional Control Register	R/W	0x0000_0000	49.3.3.3/49-22
0x0050 (CSCR5U)	Chip Select 5 Upper Control Register	R/W	0x0000_0000	49.3.3.1/49-12
0x0054 (CSCR5L)	Chip Select 5 Lower Control Register	R/W	0x0000_0000	49.3.3.2/49-18
0x0058 (CSCR5A)	Chip Select 5 Additional Control Register	R/W	0x0000_0000	49.3.3.3/49-22
0x0060 (WCR)	WEIM Configuration Register (WCR)	R/W	0x0000_0100	49.3.3.4/49-25

[1] Some bits are set according to boot configuration input

Table 49-5. Chip Selection Memory Map

Address	Use	Access
0xA0000000 ... 0xA7FF_FFFF	$\overline{CS0}$ memory region	R/W
0xA8000000 ... 0xAFFF_FFFF	$\overline{CS1}$ memory region	R/W
0xB0000000 ... 0xB1FF_FFFF	$\overline{CS2}$ memory region	R/W

Table 49-5. Chip Selection Memory Map (continued)

Address	Use	Access
0xB2000000 ... 0xB3FF_FFFF	$\overline{CS3}$ memory region	R/W
0xB4000000 ... 0xB5FF_FFFF	$\overline{CS4}$ memory region	R/W
0xB6000000 ... 0xB7FF_FFFF	$\overline{CS5}$ memory region	R/W

49.3.2 Register Summary

Figure 49-2 shows the key to the register fields and Table 49-6 shows the register figure conventions.

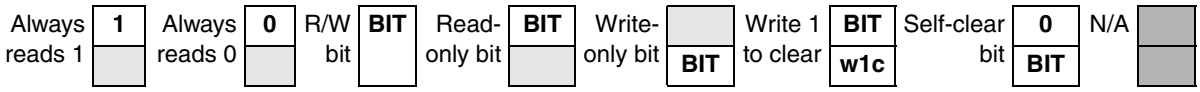


Figure 49-2. Key to Register Fields

Table 49-6. Register Figure Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writable.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.
Register Field Types	
R	Read-only. Writing this bit has no effect.
W	Write-only.
rw	Standard read/write bit. Only software can change the bit's value (other than a hardware reset).
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A status bit that can be read, and is cleared by writing a one.
slfclr	Self-clearing bit. Writing a one has some effect on the module, but it always reads as zero.
Reset Values	
0	Resets to zero.
1	Resets to one.
—	Undefined at reset.
u	Unaffected by reset.
[*signal_name*]	Reset value is determined by polarity of indicated signal.

49.3.3 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bits and field function follow the register diagrams, in bit order. The 96 bits used to control each chip select are divided into three registers: chip

select *x* upper control register (CSCR*x*U), chip select *x* lower control register (CSCR*x*L) and chip select *x* additional control register (CSCRxA).

- Bits [95:64] are located in the chip select *x* upper control register (see Figure 49-3).
- Bits [63:32] are located in the chip select *x* lower control register (see Figure 49-4).
- Bits [31:0] are located in the chip select *x* additional control register (see Figure 49-5).

Table 49-7 provides a summary of the registers in the WEIM module. For the base address of a particular module instantiation, see the system memory map.

Table 49-7. WEIM Register Summary

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 (CSCR0U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0x0004 (CSCR0L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0008 (CSCR0A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH		LBN			LBA		DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0x0010 (CSCR1U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0x0014 (CSCR1L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0018 (CSCR1A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH		LBN			LBA		DWW		DCT		WWU	AGE	CNC2	FCE
	W																

Table 49-7. WEIM Register Summary (continued)

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0020 (CSCR2U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0x0024 (CSCR2L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0028 (CSCR2A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0x0030 (CSCR3U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0x0034 (CSCR3L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0038 (CSCR3A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0x0040 (CSCR4U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																

Table 49-7. WEIM Register Summary (continued)

Base Address Offset (Name Abbreviation)		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0044 (CSCR4L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0048 (CSCR4A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0x0050 (CSCR5U)	R	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
	W																
	R	CNC		WSC						EW	WWS			EDC			
	W																
0x0054 (CSCR5L)	R	OEA				OEN				EBWA				EBWN			
	W																
	R	CSA				EBC	DSZ			CSN				PSR	CRE	WRAP	CS EN
	W																
0x0058 (CSCR5A)	R	EBRA				EBRN				RWA				RWN			
	W																
	R	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
	W																
0x0060 (WCR)	R													ECP5	ECP 4	ECP3	EC P2
	W																
	R	ECP1	EC P0	AUS 5	AUS 4	AUS 3	AUS 2	AUS 1	AUS 0						BC M		MA S
	W																

49.3.3.1 Chip Select *x* Upper Control Register (CSCR*x*U)

Offset 0x0000 (CSCR0U)
0x0010 (CSCR1U)
0x0020 (CSCR2U)
0x0030 (CSCR3U)
0x0040 (CSCR4U)
0x0050 (CSCR5U)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SP	WP	BCD		BCS				PSZ		PME	SYNC	DOL			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CNC		WSC[1]						EW	WWS			EDC			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 49-3. Chip Select *x* Upper Control Register

[1] The WSC field (bits 8 - 13) reset value is 011110 for CS0U register and is 0 for all others

Table 49-8. Chip Select *x* Upper Control Register Field Descriptions

Field	Description
31 SP	Supervisor Protect. This bit prevents accesses to the address range defined by the corresponding chip select when the access is attempted in the User mode. SP is cleared by a hardware reset. 0 User mode accesses are allowed in the memory range defined by chip select. 1 User mode accesses are prohibited. All attempts to access an address mapped by this chip select in User mode results in a error response on the AHB and no assertion of the chip select output.
30 WP	Write Protect. This bit prevents writes to the address range defined by the corresponding chip select. WP is cleared by a hardware reset. 0 Writes are allowed in the memory range defined by chip. 1 Writes are prohibited. All attempts to write to an address mapped by this chip select result in a error response on the AHB and no assertion of the chip select output.
29–28 BCD	Burst Clock Divisor. This bit field contains the value used to program the burst clock divisor for BCLK generation. It is used to divide the internal AHB bus frequency (HCLK 133 MHz). See 49.4.4/49-29 for more information on the burst clock divisors. An example is shown in Figure 49-41. When the BCM bit is set in the WEIM configuration register, BCD is ignored. BCD is cleared by a hardware reset. **Note:** When working with DOL=BCD=0, SYNC=1,FCE should be configured to 1 and EDC field should be configured to 4, because of tight timings in the controller. 00 Divide AHB clock by 1 01 Divide AHB clock by 2 10 Divide AHB clock by 3 11 Divide AHB clock by 4

Table 49-8. Chip Select *x* Upper Control Register Field Descriptions (continued)

Field	Description
27-24 BCS	Burst Clock Start. If SYNC = 1 this bit field determines the number of half cycles after address assertion before the first rising edge of BCLK is seen. See example on the Figure 49-41. A value of 0 results in a half clock delay, not an immediate assertion. When the BCM bit is set in the WEIM configuration register, this overrides the BCS bits. BCS is cleared by a hardware reset. **Note:** when working in synchronous mode when DOL=BCD=0/1 and FCE=1, BCS should be configured to an odd number (1,3,5,...). 0000 1 half AHB clock cycle. 0001 2 half AHB clock cycles. ... 1111 16 half AHB clock cycles.
23–22 PSZ	Page Size. If PME is clear the PSZ bit field indicates memory burst length in words (where *word* is defined by the DSZ field) and should be properly initialized for mixed wrap/increment AHB accesses support. Continuous PSZ value corresponds to continuous burst length setting of the external memory device. If PME is set (set to 1) the PSZ bit field indicates number of words (where “word” is defined by the port size in DSZ field) in a page in memory. This ensures that the WEIM does not burst past a page boundary at increment access when the PME bit is set. PSZ is cleared by a hardware reset. See Table 49-9 for PSZ Bit Field Combinations.
21 PME	Page Mode Emulation. This bit enables page mode emulation in burst mode. When PME is set (and SYNC equals 1), the external address is asserted for each piece of data requested. Additionally, the $\overline{\text{LBA}}$ and BCLK signals behave in the same way when an asynchronous access is performed (see Figure 49-25). PME is cleared by a hardware reset. 0 Disables page mode emulation 1 Enables page mode emulation
20 SYNC	Synchronous Burst Mode Enable. This bit enables synchronous burst mode if PME is clear (see also PME and PSR description). When enabled, the WEIM is capable of interfacing to burst Flash devices through additional burst control signals: BCLK, $\overline{\text{LBA}}$, and $\overline{\text{ECB}}$ (see example on the Figure 49-29). The sequencing of these additional I/Os is controlled by other WEIM configuration register bit settings as described in Section 49.4.3/49-28. If SYNC = 1 and PSR = 0, reads will be performed synchronously and writes will be performed asynchronously. If SYNC = 1 and PSR = 1, both reads and writes will be performed synchronously. SYNC =1, PSR = 1 and PME =1 is a reserved configuration forbidden for use. SYNC is cleared by a hardware reset. 0 Disables synchronous burst mode 1 Enables synchronous burst mode

Table 49-8. Chip Select *x* Upper Control Register Field Descriptions (continued)

Field	Description
19–16 DOL	Data Output Length. If the SYNC is set (equals 1) the DOL bit field specifies number of wait states during the burst access after the delay of the first data according to the settings shown below (see examples on the Figure 49-25 and Figure 49-40). The reset value 0 specifies that burst data is held for a single AHB clock period. As AHB clock frequencies increase, it may become necessary to delay sampling the data for multiple AHB clock periods in order to meet burst memory setup and/or frequency specifications and/or WEIM data setup time requirements. DOL has no effect on burst data length when SYNC = 0. DOL is cleared by a hardware reset. Bit 19, in addition, allows to write with fixed latency, to support this mode for PSRAM ver.1.5 and above. **Note:** When working with DOL=BCD=0, SYNC=1,FCE should be configured to 1 and EDC field should be configured to 4. 0000 1 AHB clock cycle data length. 0001 2 AHB clock cycles data length. ... 1111 16 AHB clock cycles data length. 0111 8 AHB clock cycle data length. 1000 1 AHB clock cycle data length, fixed write latency. 1001 2 AHB clock cycles data length, fixed write latency. ... 1111 8 AHB clock cycle data length, fixed write latency.
15–14 CNC	Chip Select Negation Clock Cycles. This bit field specifies the minimum number of clock cycles a chip select must remain negated after it is negated (but doesn't guarantee negation for back-to-back accesses, it requires EDC using) according to the settings shown below. See examples on the Figure 49-12, and Figure 49-13. CNC has no effect on write accesses when any CSA bit is set. CNC is cleared by a hardware reset. The number of clock cycles of this field can be increased using the CNC2 bit in the appropriate Chip Select Additional Control Register. The number of AHB clock cycles produced by both bit fields is shown in. 00 0 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 01 2 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 10 3 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 11 4 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated.
13–8 WSC	Wait State Control. This bit field programs the number of wait-states for an access to the external device connected to the chip select (see Figure 49-7). For SYNC = 1 WSC programs the number of AHB clock cycles required for the initial access (see Figure 49-25, and Figure 49-32) of a memory burst sequence initiated by the WEIM to an external burst device. For EW = 1 after the wait cycle count expires $\overline{ECB}$ is sampled and the cycle terminates when the $\overline{ECB}$ is negated. On other case WEIM special watch dog counter check that $\overline{ECB}$ will not be asserted for more than 1024 AHB clocks. Additionally in this case WEIM suppresses $\overline{LBA}$ generation after next $\overline{ECB}$ assertion (during increment burst at page boundary crossing). For write accesses the number of wait-states is increased according WWS value or decreased by DWW (see Table 49-10). WSC = 11_1111 indicates operation in a positive edge-sensitive DTACK mode. It selects $\overline{DTACK}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{DTACK}$ length. WSC is set to 01_1110 by a hardware reset for CSCR0. WSC is cleared by a hardware reset for CSCR1 – CSCR5. **Note:** WSC usage when SYNC=1, PSR=1 and FCE=1 is to mask the ECB_B signal for read accesses, and should be determined from the access time of the specific connected memory. **Note:** For SYNC=1, PSR=0 and DOL=0 the WSC value should be at least 4. **Note:** For SYNC=1, MUM =0, WSC≥ 2(BCD + 1) + (BCS + LBN + 2)/2. **Note:** For SYNC=1, MUM =1, WSCŠ 2(BCD + 1) + (BCS + LBN + 2 + LBH) + 2/2.

Table 49-8. Chip Select *x* Upper Control Register Field Descriptions (continued)

Field	Description
7 EW	$\overline{ECB}$/WAIT. This bit determines how WEIM supports the $\overline{ECB}$ input in the synchronous mode. In asynchronous mode this bit determines the operation of level-sensitive DTACK mode. (see Table 49-17 for EW effect on WEIM operation modes) 0 For SYNC = 1, if $\overline{ECB}$ goes to low state in the middle of memory burst access then the WEIM starts a new access by asserting the current AHB address to the ADDR pins and LBA assertion (ECB mode). If SYNC = 0 and WSC=111111, the WEIM waits for $\overline{DTACK}$ posedge for access termination. 1 if $\overline{ECB}$ goes to low state in the middle of a memory burst access then the WEIM waits until $\overline{ECB}$ goes high (WAIT mode) to continue current access; at the end of first access in burst it allows to wait $\overline{ECB}$ negation till 1024 clock. For SYNC = 0 WEIM begins access and after (2+DCT) clocks tests $\overline{DTACK}$ input. If $\overline{DTACK}$ is low WEIM waits $\overline{DTACK}$ high state then loads WSC value (see Figure 49-27 and Figure 49-28). **Note:** EW configuration does not influence the controller behavior when a non aligned burst access occur in page boundary. For example, INC16 memory configuration is used when burst start address is 0xF8 (non aligned, should be 0xE0), in that case the controller would made another access to the next page to complete the initial burst access.
6–4 WWS	Write Wait State. This bit field determines whether additional wait-states are required for write cycles (see Table 49-10). This is useful for writing to memories that require additional data setup time (see example on Figure 49-9). The DWW field should be zero when this field is in use. WWS is cleared by a hardware reset. **Note:** To decrease write wait states use the DWW bit field.
3–0 EDC	Extra Dead Cycles. This bit field determines whether idle cycles are inserted before a new access (see example in Figure 49-10). If the currently accessed CS EDC field is not empty then idle cycles are inserted before next access except for two conditions: current access is an asynchronous write (its SYNC = 0) or the next access is an asynchronous read from the same chip select. This field is used in two cases: • Slow memory or peripherals that use long CS or $\overline{OE}$ to output data hold times to prevent data bus contention on back-to-back external transfers. • Synchronous accesses (SYNC = 1) to provide $\overline{CS}$ high minimum pulse width (even for back-to-back accesses). The EDC field is cleared by a hardware reset. **Note:** When working with DOL=BCD=0, SYNC=1,FCE=1 EDC field should be configured to 4. 0000 0 No idle AHB clock cycles inserted. 0001 1 1 Idle AHB clock cycle inserted. ... 1111 15 AHB clock cycles inserted.

Table 49-9. PSZ Bit Field Values

PSZ	PME=0 Memory Burst Length	PME=1 Number of Words in Page
00	4	4
01	8	8
10	16	16
11	continuous	32

Table 49-10. WSC Bit Field Values

WSC	Number of Wait-States					
	Read Access	Write Access				
		WWS = 0, DWW = 0	WWS = 1, DWW = 0	WWS = 7, DWW = 0	WWS = 0, DWW = 1	WWS = 0, DWW = 2
000000	1	1	1	7	1	1
000001	1	1	2	8	1	1
000010	2	2	3	9	1	1
000011	3	3	4	10	2	1
000100	4	4	5	11	3	2
...	–	–	–	–	–	–
110111	55	55	56	62	54	53
111000	56	56	57	63	55	54
111001	57	57	58	64	56	55
111010	58	58	59	65	57	56
111011	59	59	60	66	58	57
111100	60	60	61	67	59	58
111101	61	61	62	68	60	59
111110	62	62	63	69	61	60
111111	Posedge sensitive DTACK mode					

49.3.3.2 Chip Select *x* Lower Control Register (CSCRxL)

Offset 0x0004 (CSCR0L)
0x0014 (CSCR1L)
0x0024 (CSCR2L)
0x0034 (CSCR3L)
0x0044 (CSCR4L)
0x0054 (CSCR5L)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OEA[1]				OEN				EBWA				EBWN			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CSA				EBC[2]	DSZ[3]			CSN[4]				PSR	CRE	WRAP	CSEN
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[5]

Figure 49-4. Chip Select *x* Lower Control Register

1 OEA (bits 28 - 31) reset value is 1010 for CSCR0L and 0 for CSCR*x*L with *x* > 0.

2 EBC (bit 11) reset value is 1 for CSCR0L and 0 for CSCR*x*L with *x* > 0.

3 DSZ (bits 8 - 10) reset value is determined by boot configuration inputs for CSCR0L. For CSCRxL with *x* > 0, the reset value is 0.

4 CSN (bits 4 - 7) reset value is 0100 for CSCR0L and 0 for CSCRxL with *x* > 0.

5 Bit 0 reset value is 1 for CSCR0L and 0 for CSCR*x*L with *x* > 0.

Table 49-11. Chip Select *x* Lower Control Register Field Descriptions

Field	Description
31–28 OEA	$\overline{OE}$ Assert. This bit field determines when $\overline{OE}$ is asserted during a read cycle. For SYNC = 0, OEA determines the number of half clocks before $\overline{OE}$ asserts during a read cycle. For SYNC = 1, after the initial memory burst access, $\overline{OE}$ is asserted continuously for subsequent memory burst accesses, and is not affected by OEA (see memory burst read timing diagrams for more detail); the behavior of $\overline{OE}$ on the initial memory burst access is the same as when SYNC = 0 (see example on the Figure 49-25). The OEA field do not affect the cycle length. OEA is set to 1010 by a hardware reset for CSCR0L register and is cleared for other registers. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ assertion and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ assertion and end of access ... 1111 15 Half AHB clock cycles between $\overline{OE}$ assertion and end of access
27–24 OEN	$\overline{OE}$ Negate. This bit field determines when $\overline{OE}$ is negated during a read cycle (see example on the Figure 49-20). Setting the SYNC bit (SYNC = 1) overrides OEN and $\overline{OE}$ negates at the end of a read access and no sooner. OEN does not affect the cycle length, except in posedge sensitive DTACK mode. OEN is cleared by a hardware reset. **Note:** The term “end of a read access” is the nearest possible address, data or control signal change by another access (an own next access or an access from others pin shared controller). **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{OE}$ negation and end of access 0001 1 Half AHB clock cycle between $\overline{OE}$ negation and end of access ... 1111 15 Half AHB clock cycles between $\overline{OE}$ negation and end of access
23–20 EBWA	$\overline{EB}$ Write Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during write cycles (see example on the Figure 49-8). This is useful to meet data setup time requirements for slow memories. EBWA does not affect the cycle length. EBWA is cleared by a hardware reset. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles before $\overline{EB}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ is asserted. ... 1111 15 Half AHB clock cycles before $\overline{EB}$ is asserted.
19–16 EBWN	$\overline{EB}$ Write Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a write cycle (see example on the Figure 49-8). This is useful to meet data hold time requirements for slow memories. EBWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBWN and $\overline{EB}$ negates at the end of a write access and according AHB hbstrb[3:0]. EBWN is cleared by a hardware reset. **Note:** Minimum time $\overline{OE}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. ... 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.

Table 49-11. Chip Select *x* Lower Control Register Field Descriptions (continued)

Field	Description
15–12 CSA	$\overline{CS}$ Assert. This bit field determines when chip select is asserted for devices that require additional address setup time (see example on the Figure 49-11). It does not affect the cycle length. CSA is cleared by a hardware reset. **Note:** The CSA bit setting affects both reads and writes for all WEIM modes. 0000 0 Half AHB clock cycles before $\overline{CS}$ is asserted. 0001 1 Half AHB clock cycle before $\overline{CS}$ is asserted. ... 1111 15 Half AHB clock cycles before $\overline{CS}$ is asserted.
11 EBC	Enable Byte Control. This bit indicates the types of access that assert Enable Byte outputs $\overline{EB}$[3:0] (see example on Figure 49-7). The $\overline{EB}$[3:0] outputs can be configured as byte write enables. EBC is set by a hardware reset for CSCR0L register and is cleared for other registers. 0 Both read and write accesses assert the $\overline{EB}$[3:0] 1 Only write accesses assert the $\overline{EB}$[3:0], thus configuring as byte write enables
10–8 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown in the Table 49-12. DSZ is set by a hardware reset for CSCR0L by the value of the boot configuration pins. DSZ and MUM (multiplexed mode) affected on data port location as shown in the Table 49-12. DSZ is cleared by a hardware reset for CSCR1L-CSCR5L.
7–4 CSN	$\overline{CS}$ Negate. This bit field determines when chip select is negated for devices that require additional address/data hold times (see example on the Figure 49-11). CSN affects only asynchronous (read and write) accesses (SYNC=0), and is ignored on synchronous (SYNC=1). CSN does not affect the cycle length, except in positive edge sensitive DTACK mode. CSN is set to 0100 by a hardware reset for CSCR0L register and is cleared by a hardware reset for other registers. 0000 0 Half AHB clock cycles between $\overline{CS}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{CS}$ negation and end of access. ... 1111 15 Half AHB clock cycles between $\overline{CS}$ negation and end of access.
3 PSR	Pseudo SRAM Enable (Burst Write Enable). This bit enables four functions for Pseudo SRAM (for example CellularRAM™) or any other device that support these modes: a burst write, a write wrap disable, a read wait state increase and a memory control register accessibility. If PSR=1then a memory burst write is enable (with SYNC = 1). In this mode the WRAP bit is masked on write time, unless WWU bit is set in the CSCRxA register, and wait state on read is automatically increased to WSC +1 (see Figure 49-40 and Figure 49-41). With PSR = 1 and SYNC = 1, during a write access the RWA field is ignored and RW_B is asserted at the start of the write cycle. An asynchronous access (SYNC=0) should be used with PSR=1 and CRE=1 to write to the memory control register. With PSR = 1, A23 is set to zero during read cycles and reflects the CRE bit value during write accesses. This limits the contiguous memory address range to that provided by A0 to A22 when PSR is set. PSR is cleared by a hardware reset. **Note:** This bit is also used when working with NOR Flash memory type for burst read accesses only. 0 PSRAM mode is disabled 1 PSRAM mode is enabled

Table 49-11. Chip Select *x* Lower Control Register Field Descriptions (continued)

Field	Description
2 CRE	Control Register Enable. This bit indicates CRE memory pin state while writing to CS address space, for PSRAM control register write. For PSR=1 the CRE bit will be driven on pin ADDR[23] in a write access time. CRE is cleared by a hardware reset. **Note:** SYNC = 0 should be used to access to PSRAM control register. 0 CRE pin 0 1 CRE pin 1
1 WRAP	Wrap Memory Mode. This bit indicates that memory is in the wrap mode. The size of wrap is set using the PSZ field. In case not matching wrap boundaries in both the memory (PSZ field) and AHB access on the current address WEIM puts address on address bus and generates $\overline{\text{LBA}}$ signal (see example on the Figure 49-37). WRAP is cleared by a hardware reset. 0 Memory is in not in wrap mode. 1 Memory is in wrap mode.
0 CSEN	$\overline{\text{CS}}$ Enable. This bit controls the operation of the chip select pin. CSEN is set by a hardware reset for CSCR0L to allow CSCR0L to select from an external boot ROM. CSEN is cleared by a hardware reset to CSCR1L–CSCR5L. 0 Chip select function is disabled; attempts to access an address mapped by this chip select results in a error respond on the AHB and no assertion of the chip select output 1 Chip select is enabled, and is asserted when presented with a valid AHB access.

Table 49-12. DSZ Bit Field Values

DSZ	Data Port Size	
	MUM=0	MUM=1
000	Reserved	Reserved
001	Reserved	Reserved
010	8-bit port, resides on DATA_IN/OUT [15:8] pins	Reserved
011	8-bit port, resides on DATA_IN/OUT [7:0] pins	Reserved
100	Reserved	Reserved
101	16-bit port, resides on DATA_IN/OUT [15:0] pins	Reserved
110	Reserved	32-bit port, resides on ADDR / M_DATA_IN/OUT [15:0] and M_DATA_IN/OUT [31:16] pins
111	Reserved	Reserved

49.3.3.3 Chip Select *x* Additional Control Register (CSCR*x*A)

Offset 0x0008 (CSCR0A)
0x0018 (CSCR1A)
0x0028 (CSCR2A)
0x0038 (CSCR3A)
0x0048 (CSCR4A)
0x0058 (CSCR5A)

Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EBRA				EBRN				RWA				RWN			
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MUM	LAH			LBN			LBA	DWW		DCT		WWU	AGE	CNC2	FCE
RESET	1[1]/0	0[2]		0	0[3]	0	0	0	0	0	0	0	0	0	0	0

Figure 49-5. Chip Select *x* Additional Control Register

1 MUM (bit 15) reset value is determined by the settings of the boot configuration inputs (refer to Chapter 7, “System Boot” for details on boot configuration signals).

2 LAH (bits 13, 14) reset value is 10 for CSCR0A and 0 for other registers.

3 LBN (bits 10 -12) reset value is 100 for CSCR0A and 0 for other registers.

Table 49-13. Chip Select *x* Additional Control Register Field Descriptions

Field	Description
31–28 EBRA	$\overline{EB}$ Read Assert. This bit field determines when $\overline{EB}$ [3:0] is asserted during read cycles (see example on the Figure 49-25). EBRA does not affect the cycle length. EBRA is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles before $\overline{EB}$ asserted. 0001 1 Half AHB clock cycle before $\overline{EB}$ asserted. ... 1111 15 Half AHB clock cycles before $\overline{EB}$ asserted.
27–24 EBRN	$\overline{EB}$ Read Negate. This bit field determines when $\overline{EB}$ [3:0] outputs are negated during a read cycle (see example on the Figure 49-20). EBRN does not affect the cycle length, except when in positive edge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides EBRN and $\overline{EB}$ negates at the end of a read access but not sooner. EBRN is cleared by a hardware reset. **Note:** Minimum time $\overline{EB}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{EB}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{EB}$ negation and end of access. ... 1111 15 Half AHB clock cycles between $\overline{EB}$ negation and end of access.

Table 49-13. Chip Select *x* Additional Control Register Field Descriptions (continued)

Field	Description
23–20 RWA	$\overline{RW}$ Assertion. This bit field determines when $\overline{RW}$ is asserted during write cycles. (see example on the Figure 49-28). RWA is cleared by a hardware reset. **Note:** Minimum time $\overline{RW}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles $\overline{RW}$ delay 0001 1 Half AHB clock cycle $\overline{RW}$ delay. ... 1111 15 Half AHB clock cycles $\overline{RW}$ delay.
19–16 RWN	$\overline{RW}$ Negation. This bit field determines when $\overline{RW}$ is negated during a write cycle (see example on the Figure 49-28). RWN does not affect the cycle length, except in posedge sensitive DTACK mode. Setting the SYNC bit (SYNC = 1) overrides RWN and $\overline{RW}$ negates at the end of a write access and no sooner. RWN is cleared by a hardware reset. **Note:** Minimum time $\overline{RW}$ is asserted is one clock cycle. 0000 0 Half AHB clock cycles between $\overline{RW}$ negation and end of access. 0001 1 Half AHB clock cycle between $\overline{RW}$ negation and end of access. ... 1111 15 Half AHB clock cycles between $\overline{RW}$ negation and end of access.
15 MUM	Multiplexed Mode. This bit determines the address/data multiplexed mode for asynchronous and synchronous accesses (see examples in Figure 49-43 - Figure 49-46). Port mapping is defined in the Table 49-12. MUM is cleared by a hardware reset for CSCR1A - CSCR5A. For CSCR0A, MUM is configured at reset time with the boot configuration inputs (refer to Chapter 7, “System Boot” for more details). 0 Non-multiplexed mode 1 Multiplexed mode
14–13 LAH	$\overline{LBA}$ to Address Hold. This bit field determines address hold time after $\overline{LBA}$ negation for MUM = 1 only. See example on the Figure 49-43 and Figure 49-44. LAH is cleared by a hardware reset for CSCR1A - CSCR5A. For CSCR0A, LAH is set to 10 by hardware reset. 00 2 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 01 3 AHB half clock cycle between $\overline{LBA}$ negation and address invalid. 10 4 AHB half clock cycles between $\overline{LBA}$ negation and address invalid. 11 5 AHB half clock cycles between $\overline{LBA}$ negation and address invalid.
12–10 LBN	$\overline{LBA}$ Negation. This bit field determines when $\overline{LBA}$ is negated. For SYNC=0 and MUM =0 LBN determines how many half AHB clock cycle will be between $\overline{LBA}$ negation and end of access (see example on the Figure 49-8). For SYNC=0 and MUM =1 this field determines $\overline{LBA}$ length (see example on the Figure 49-44). Negation time and $\overline{LBA}$ lengths are listed in Table 49-14. For SYNC=1 (MUM=0 and MUM=1) $\overline{LBA}$ negation occurs (LBN+BCD+1) half AHB clock cycles after first BCLK posedge detection. LBN does not affect the cycle length, except in positive edge sensitive DTACK mode. LBN is cleared by a hardware reset for CSCR1A - CSCR5A. For CSCR0A LBN is set to 100 by hardware reset. **Note:** Minimum of time $\overline{LBA}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1.
9–8 LBA	$\overline{LBA}$ Assertion. This bit field determines when $\overline{LBA}$ is asserted according the settings shown below (see example on the Figure 49-11). LBA is cleared by a hardware reset. **Note:** LBA field affects all modes. **Note:** Minimum of time $\overline{LBA}$ to be asserted is a one clock for MUM = 0 and two clocks for MUM = 1. 00 0 AHB half clock cycles between beginning of access and $\overline{LBA}$ assertion. 01 1 AHB half clock cycle between beginning of access and $\overline{LBA}$ assertion. 10 2 AHB half clock cycles between beginning of access and $\overline{LBA}$ assertion. 11 3 AHB half clock cycles between beginning of access and $\overline{LBA}$ assertion.
7–6 DWW	Decrease Write Wait State. This bit field in combination with WWS determines whether write cycles are shorter than the read cycles (see Table 49-10). The WWS field should be zero when this field is in use. DWW is cleared by a hardware reset.

Table 49-13. Chip Select *x* Additional Control Register Field Descriptions (continued)

Field	Description
5–4 DCT	DTACK Check Time. This bit field determines time of insensitivity at the beginning of access for SYNC=0 and EW=1 according to the settings shown below (see example on the Figure 49-27). DCT is a number of clock cycles between CS assertion and first DTACK check. DCT is cleared by a hardware reset. 00 2 AHB clock cycles between CS assertion and first DTACK check. 01 6 AHB clock cycles between CS assertion and first DTACK check. 10 8 AHB clock cycles between CS assertion and first DTACK check 11 12 AHB clock cycles between CS assertion and first DTACK check.
3 WWU	Write Wrap Unmask. This bit allow unmask WRAP bit in case PSR = 1 and write access. WWU is cleared by a hardware reset. 0 Prevents Wrap during write access 1 Allow wrap on write
2 AGE	Acknowledge Glue Enable. This bit is used to enable/disable glue logic between external DTACK and internal control logic. The glue logic is a flip-flop that is reset by CS assertion, Its data input is a constant 1 and DTACK goes to its clock input. This glue logic is used to synchronize posedge of the external DTACK in worst-noise or slow edge growth conditions. AGE is cleared by a hardware reset. 0 Disable glue logic 1 Enable glue logic
1 CNC2	Chip Select Negation Clock Cycles, Bit [2]. This bit is used to increase the CNC field to a 3-bit field. See description CNC field in the Chip Select *x* Upper Control Register. CNC2 is cleared by a hardware reset. The number of AHB clock cycles produced by both bit fields is shown in Table 49-15
0 FCE	Feedback Clock Enable. This bit is used to enable / disable data capture by BCLK_FB. If FCE=1 WEIM used addition one clock to synchronize feedback clock captured data to AHB clock, so read access is slow then FCE=0. FCE is cleared by a hardware reset. **Note:** when working in synchronous mode DOL=BCD=1, FCE bit should be configured to 1 (FCE=0 in this specific mode is not supported) 0 Data captured using AHB clock 1 Data captured using BCLK_FB

Table 49-14. LBN Bit Field Values

LBN	Half AHB Clock Cycle Between LBA Negation and End of Access	LBA Length, Half AHB Clock Cycle
	MUM = 0	MUM = 1
000	0	2
001	1	3
...	–	–
111	7	9

Table 49-15. CNC/CNC2 Bit Values

CNC2	CNC	Minimum CS Negation, in AHB clock cycle
0	00	0
0	01	2

Table 49-15. CNC/CNC2 Bit Values (continued)

CNC2	CNC	Minimum $\overline{CS}$ Negation, in AHB clock cycle
0	10	3
0	11	4
1	00	5
1	01	6
1	10	7
1	11	8

49.3.3.4 WEIM Configuration Register (WCR)

The WCR contains control bits for the configuration and operation of the WEIM.

0x0060 (WCR) Access: User read-write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	ECP5	ECP4	ECP3	ECP2
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECP1	ECP0	AUS5	AUS4	AUS3	AUS2	AUS1	AUS0	0	0	0	0	0	BCM	CRE R	MAS
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 49-6. WCR Register

Table 49-16. WEIM Control Register Field Descriptions

Field	Description
31–20	Reserved
19 ECP5	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS5 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
18 ECP4	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS4 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of $\overline{ECB}$ in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.

Table 49-16. WEIM Control Register Field Descriptions (continued)

Field	Description
17 ECP3	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS3 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
16 ECP2	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS2 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
15 ECP1	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS1 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
14 ECP0	$\overline{ECB}$ Capture Phase. This bit indicates in which phase of HCLK the BCLK is generated to the memory for synchronous CS0 write accesses. This bit is XORed with BCS[0] bit in write accesses to determine the BCLK starting phase to memory and it also influence on capturing of ECB in the WEIM design. 0 BCLK starting phase is as BCS[0] bit indicates. 1 BCLK starting phase is the opposite of BCS[0] bit indicates for write accesses only.
13 AUS5	Address Unshifted for CS5. This bit indicates an unshifted mode for address assertion for CS5 accesses. This bit is cleared by a hardware reset. 0 Address shifted according CS5 port size. 1 Address unshifted
12 AUS4	Address Unshifted for CS4. This bit indicates an unshifted mode for address assertion for CS4 accesses. This bit is cleared by a hardware reset. 0 Address shifted according CS4 port size. 1 Address unshifted
11 AUS3	Address Unshifted for CS3. This bit indicates an unshifted mode for address assertion for CS3 accesses. This bit is cleared by a hardware reset. 0 Address shifted according CS3 port size. 1 Address unshifted
10 AUS2	Address Unshifted for CS2. This bit indicates an unshifted mode for address assertion for CS2 accesses. This bit is cleared by a hardware reset. 0 Address shifted according CS2 port size. 1 Address unshifted
9 AUS1	Address Unshifted for CS1. This bit indicates an unshifted mode for address assertion for CS1 accesses. This bit is cleared by a hardware reset. 0 Address shifted according CS1 port size. 1 Address unshifted
8 AUS0	Address Unshifted for CS0. This bit indicates an unshifted mode for address assertion for CS0 accesses. In the unshifted mode AHB address is transmitted to address pins without port size independent shift. This mode is used when working with different data widths in the multi-chip package. By default this bit comes up low (set to 0) after initial power up. 0 Address shifted according CS0 port size. 1 Address unshifted
7–3	Reserved

Table 49-16. WEIM Control Register Field Descriptions (continued)

Field	Description
2 BCM	Burst Clock Mode. This bit selects the burst clock mode of operation. It is used for system debug mode. Don't use BCM = 1 in work configuration. BCM is cleared by a hardware reset. 0 The burst clock runs only when accessing a chip select range with the SYNC bit set; when the burst clock is not running it remains in a logic 0 state; when the burst clock is running it is configured by the BCD and BCS fields in the chip select control register. 1 The burst clock runs on every memory access (independent of chip select configuration)
1 CRER	CRE Output Relocation. 0 CRE on ADDR[23] (ADDR[25:24] are used as EB[3:2] in the 32-bit multiplexed mode MUM=1). 1 Invalid setting.
0 MAS	Merged Address Space. This bit indicates merged address space mode. If MAS is set the $\overline{CS1}$ address space is merged with $\overline{CS0}$ for a total of 256 (for half-word width port and 512 for word width port) Mbytes. CS1 output is used as A26. By default this bit comes up low (set to 0) after initial power up. 0 Standard address space. 1 Merged address space.

49.4 Functional Description

This section provides the functional description for the WEIM module.

49.4.1 Configurable Bus Sizing

The WEIM supports byte, halfword, and word operands allowing access to 8-bit, 16-bit, and multiplexed 32-bit (multiplexed or not) ports. The port size is programmable using the DSZ field in the corresponding Chip Select control register. In addition, the portion of the data bus used for transfer to or from an 8-bit and 16-bit ports is programmable using the same DSZ field. An 8-bit port can reside on DATA_IN/OUT bus bits [15:8] or [7:0]. A 16-bit port can resides on DATA_IN/OUT bus bits [15:0]. A 32-bit multiplexed port resides on M_DATA_IN/OUT bus bits [31:0]. A 32-bit not multiplexed port resides on DATA_IN/OUT bus bits [31:0].

NOTE

Misaligned transfers are not supported.

A word access to or from an 8-bit port requires four external bus cycles to complete the transfer. A word access to or from a 16-bit port requires two external bus cycles to complete the transfer. A halfword access to or from an 8-bit port requires two external bus cycles to complete the transfer.

In the case of a multi-cycle transfer, the lower two address bits (ADDR[1:0]) are incremented appropriately. The WEIM address bus is configured according to DSZ field and AUSx bits, too. There is either one or two bits right shift of AHB address bits for half-word or word width port accordingly.

The WEIM has a data multiplexer which takes the four bytes of the AHB interface data bus and routes them to their required positions to properly interface to memory and peripherals.

49.4.2 WEIM Operational Modes

WEIM has 9 main operational modes selected by control fields settings as described in the Table 49-17. For details see corresponding bit fields descriptions.

Table 49-17. WEIM Operation Modes Field Settings

Control fields					Brief mode description
SYNC	PME	MUM	EW	WSC	
0	0	0	0	< 11_1111	Asynchronous
		1	0		Asynchronous multiplexed
		0	1		Asynchronous level sensitive DTACK mode
		0	0	11_1111	Asynchronous posedge sensitive DTACK mode
1	1	0	0	< 11_1111	Page mode emulation
	0	0	0		Synchronous burst with restart on $\overline{ECB}$ negation
		1	0		Synchronous multiplexed burst with restart on $\overline{ECB}$ negation
		0	1		Synchronous burst with wait on $\overline{ECB}$ negation
		1	1		Synchronous multiplexed burst with wait on $\overline{ECB}$ negation

49.4.3 Burst Mode Memory Operation

With memory burst mode enabled (SYNC = 1), the WEIM attempts to translate AHB burst accesses to memory burst accesses, being limited by the memory burst length, predefined PSZ value, or memory and AHB WRAP/INCR boundary crossing non-matching. The WEIM only displays the first address accessed in a memory burst sequence unless the page mode emulation (PME) bit is set.

WEIM may translate from some AHB sequential accesses to one or few memory bursts, but not from two AHB nonsequential accesses to one memory burst.

For the first access in a memory burst sequence, the WEIM asserts $\overline{LBA}$ - causing the external burst device to latch the starting burst address - and then toggle the burst clock (BCLK) a predefined number of cycles in order to latch the first unit of data. Subsequent accessed data units can then be burst in fewer clock cycles, realizing an overall increase in bus bandwidth.

Memory burst accesses are terminated by the WEIM whenever it detects that:

- the next AHB access is not sequential,
- next sequential access crosses boundary with unequal condition (wrap/increment, burst length) on the AHB and memory,
- current memory burst length reached,
- by the external burst device request if it needs additional cycles to retrieve the next requested memory location.

In last case, the burst memory device provides an $\overline{ECB}$ (or WAIT) feedback signal to the WEIM whenever it is necessary to terminate/postponed the on-going burst sequence. If EW = 0 WEIM initiate a new (with

long first access) memory burst sequence, if EW = 1 WEIM only waits $\overline{\text{ECB}}$ negation to continue current memory burst sequence. Additionally EW = 1 allows wait states insertion after wait state counter expired, but $\overline{\text{ECB}}$ still asserted. Over this a new memory burst sequence should be generated.

The synchronous mode is also used for burst Cellular RAM, which supports memory burst writes, that is enabled by PSR = 1.

49.4.4 Burst Clock Divisor

In some cases it may be necessary to slow the external bus in relation to the internal bus to allow accesses to burst devices that have a maximum operating frequency which is less than the operating frequency of the internal bus. The internal AHB bus frequency (HCLK 133 MHz) can be divided by two, three, or four for presentation on the external bus in burst mode operation.

By programming the BCD field to various values, two signals on the external bus are affected; $\overline{\text{LBA}}$ and BCLK. The $\overline{\text{LBA}}$ signal is asserted according to LBA field programming and remain asserted until the first falling edge of the BCLK signal. The BCLK signal runs with a 50% duty cycle until a non-sequential internal request is received or an external $\overline{\text{ECB}}$ signal is recognized.

Caution should be exercised when programming these fields to ensure the WSC and DOL fields are coordinated to provide the desired external bus waveforms. BCD and DOL fields should always get the same value when configured. As an example, if the BCD field is programmed to 01, the DOL field should be programmed to 0001. If the BCD field is programmed to 10, the DOL field should be programmed to 0010.

The BCM bit in the WEIM configuration register has priority over the BCD field. If BCM = 1, the BCLK runs at full frequency on every memory access (both with SYNC=1 and with SYNC=0). the BCM bit is used mainly for system debug mode. it has no functional use of the WEIM.

49.4.5 Burst Clock Start

In an effort to allow greater flexibility in achieving the minimum number of wait states on bursted accesses, the user can determine when they want the BCLK to start toggling. This allows the BCLK to be skewed from point of data capture on the AHB clock by any number of AHB clock phases. Care must be exercised when setting BCS field in conjunction with the BCD, WSC, and DOL fields. See the external timing diagrams from Section 49.6.4, “Burst Memory Accesses Timing Diagrams,” for some examples of how to use the BCS, BCD, WSC, and DOL fields together.

49.4.6 Page Mode Emulation

Setting the PME and SYNC bits causes the WEIM to perform memory bursted accesses by emulating page mode operation. The $\overline{\text{LBA}}$ signal remains asserted for the entire access, the burst clock does not send a signal, and the external address asserts for each access are made. The initial access timing is dictated by the WSC field, and the page mode access timing is dictated by the DOL field.

See the external timing diagrams from Section 49.6.2, “Page Mode Timing Diagrams,” for some examples.

The WEIM can take advantage of improved page timing for sequential accesses only. Accesses that are on the page but are not sequential in nature have their timing dictated by the WSC field. The page size can be set using the PSZ field to 4, 8, 16, or 32 words (the word size is determined by the data width of the external memory, such as the DSZ field).

49.4.7 PSRAM Mode Operation

A control bit PSR is provided to enable PSRAM operation. For SYNC = 1 this bit enable a memory burst write. In this mode WRAP bit on write time is automatic masked (according CellularRAM™ SPEC wrap supports only for read accesses) unless WWU bit is set. Initial wait state value is automatic incremented on read access (see Figure 49-40 and Figure 49-41).

Bit EW determines how WEIM support $\overline{\text{ECB}}$ input. For SYNC = 1 if $\overline{\text{ECB}}$ goes to low state in the middle of memory burst access then WEIM only waits it'll go high (WAIT mode) to continue current access; at the end of first access in memory burst it allows to wait $\overline{\text{ECB}}$ negation during PSRAM refresh insertion.

Bit CRE and an unused address line can be used to drive the control register enable (CRE) memory input to load the PSRAM configuration registers. For PSR = 1 the CRE bit will be driven on pin ADDR[23] in a write access time.

NOTE

SYNC = 0 should be used to access to PSRAM control register.

49.4.8 Multiplexed Address/Data Mode

A control bit MUM allows support memory with multiplexed address/data bus both in the asynchronous and in the synchronous modes. The LBN and LBH bit fields should be used for properly bus timing setup (see Figure 49-43 - Figure 49-46).

49.4.9 Mixed AHB/Memory Burst Modes Support

To provide mixed sequential/wrap accesses with different length WEIM interprets burst signal and generate additional $\overline{\text{LBA}}$ signals whenever there appear unequal address or burst boundary crossing condition (see Section 49.4.3, "Burst Mode Memory Operation"). PSZ field and WRAP bit should be used to notify WEIM about current memory burst and wrap condition for properly external address generation. In case of non matching boundaries in both the memory and AHB access WEIM starts a new memory burst access by putting address from AHB on address bus and generating $\overline{\text{LBA}}$ signal. For example, Table 49-19 shows how WEIM interprets various types of AHB access in case memory configured to 8 beat burst with wrap.

49.4.10 AHB Bus Cycles Support

The WEIM uses an ARM AHB slave interface. It has a 32-bit bus and supports the four transfer types defined in the AHB specification (IDLE, BUSY, NONSEQ, and SEQ).

NOTE

Only 32-bit accesses are supported for SEQ mode.

It also supports the AHB transfers shown on Table 49-18. These AHB cycles will be translated into the necessary cycles on the memory side. For example for optimal operation in case ARM cache is configured to 8 beat burst with wrap, a synchronous Flash and cellular RAM memory should be configured in 16 word wrap burst mode when using a 16-bit data port, and in 8 word wrap burst mode when using a 32-bit data port. WEIM uses WRAP bit and PSZ field for support different memory configurations. The controller splits the transaction when needed in some cases (see Section 49.4.3, "Burst Mode Memory Operation," on page 49-28).

Table 49-18. AHB Burst Cycles Supported

HBURST	TYPE	Supported	Description
000	SINGLE	Yes	Single transfer
001	INCR	Yes	Incrementing burst
010	WRAP4	Yes	4-beat wrapping burst
011	INCR4	Yes	4-beat incrementing burst
100	WRAP8	Yes	8-beat wrapping burst
101	INCR8	Yes	8-beat incrementing burst
110	WRAP16	Yes	16-beat wrapping burst
111	INCR16	Yes	16-beat incrementing burst

For example Table 49-19 shows AHB bus sequential accesses breaking in to external memory bursts for memory configured to 8 beat burst with wrap and for some different AHB burst types and start addresses. $\overline{\text{LBA}}$(X) means start memory burst access ($\overline{\text{LBA}}$ generation) from address X (all addresses in hex form).

Table 49-19. External Memory Bursts Start Addresses for Some AHB Burst Accesses

AHB burst type	Memory data port width	AHB burst start address							
		0	4	8	C	10	14	18	1C
WRAP8	16-bit	$\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
INCR8	16-bit	$\overline{\text{LBA}}$(0) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20) $\overline{\text{LBA}}$(30)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(10) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)
WRAP4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14)	$\overline{\text{LBA}}$(18)	$\overline{\text{LBA}}$(1C)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(10)

Table 49-19. External Memory Bursts Start Addresses for Some AHB Burst Accesses (continued)

AHB burst type	Memory data port width	AHB burst start address							
		0	4	8	C	10	14	18	1C
INCR4	16-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(8) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(C) $\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)
	32-bit	$\overline{\text{LBA}}$(0)	$\overline{\text{LBA}}$(4)	$\overline{\text{LBA}}$(8)	$\overline{\text{LBA}}$(C)	$\overline{\text{LBA}}$(10)	$\overline{\text{LBA}}$(14) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(18) $\overline{\text{LBA}}$(20)	$\overline{\text{LBA}}$(1C) $\overline{\text{LBA}}$(20)

Some examples are shown in Figure 49-32 through Figure 49-39.

49.4.11 DTACK Mode

In DTACK mode, the WEIM timing depends on $\overline{\text{DTACK}}$ input signal. This signal may be used in two ways: by posedge sensitive or by level sensitive (with an initial insensitiveness time).

Posedge sensitive mode is set by WSC=111111 (imply EW=0) and selects $\overline{\text{DTACK}}$ input as access length control sign (instead of default WSC counter). It means that access length is determined by $\overline{\text{DTACK}}$ length. WEIM begins deasserting control signals after approximately 1.5 clock (synchronization delay) in the sequence according negation control fields.

NOTE

It may be required to program CSA and/or CSN fields for a correct word access to 16 or 8-bit port in this mode if the corresponding module is CS-sensitive. In this case, the maximum value of CSN is 6.

Level sensitive mode is set by EW=1 (imply WSC < 111111). There access length is controlled by WSC. In this case WEIM begins access (by CS assertion) and after some clocks (according DCT field) checks $\overline{\text{DTACK}}$ input. If $\overline{\text{DTACK}}$ is low WEIM waits $\overline{\text{DTACK}}$ high state and reload wait state counter (see Figure 49-27 and Figure 49-28). For sequential AHB accesses where CS does not negate during the burst, $\overline{\text{DTACK}}$ is checked on the first access only.

Glue logic enabled by the AGE bit of the CSCR*x*A register can be used for noisy or slow-rising $\overline{\text{DTACK}}$. See the AGE bit description for more details.

49.4.12 Internal Input Data Capture

In the typical case the input data is not sampled by the WEIM, but it is sampled by the AHB master on the rising edge of HCLK when HREADY is high. WEIM asserts the HREADY signal to the AHB master (according to the WSC or DOL counting correspondingly). This enables better performance on the data path.

There are two cases when input data are sampled inside the WEIM:

- The first case is when an access size is larger then a port size. In this case, WEIM samples all the data coming from the memory device except the last one of that transaction. For example if there is a word access to the byte wide memory, the WEIM captures the first three input bytes internally

and drives them together with the last byte to the AHB master (the last byte is not sampled in the WEIM). The WEIM captures data by the rising edge of HCLK when WSC (or DOL if it is part of a burst) time is expired and (if it depends) suitable $\overline{\text{ECB}}$ or $\overline{\text{DTACK}}$ input condition.

- The second case is when a feedback clock is used (FCE=1) for synchronous burst. Data will be sampled on the rising edge of the feedback clock (when the WSC or DOL time expired and input conditions are kept), and then the captured data is again sampled by HCLK before it is driven to the AHB master.

49.4.13 Error Conditions

The following conditions cause an error signal:

- Access to a disabled chip select (access to a mapped chip select address space where the CSEN bit in the corresponding chip select control register is clear)
- Write access to a write-protected chip select address space (the WP bit in the corresponding chip select control register is set)
- User access to a supervisor-protected chip select address space (the SP bit in the corresponding chip select control register is set)
- User read or write access to a chip select control register or the WEIM configuration register
- Byte or halfword access to a chip select control register or the WEIM configuration register
- $\overline{\text{DTACK}}$ acknowledge is absent more then 1024 clocks
- WAIT negation more then 1024 clocks.

49.5 Initialization/Application Information

WEIM is ready to work with $\overline{\text{CS0}}$ after hardware reset, but it has been configured for very slow access (for booting) without additional setup and hold time. The other $\overline{\text{CS}}$ are disabled by hardware reset, and must be initialized before use by writing to the high and low configuration registers.

Example 49-1 shows how to prepare WEIM and 16-bit Flash memory to work in the synchronous mode.

```
@; config WEIM to Async access with EDC, OEA, RWA, RWN, EBC, 16 bit port and PSR
        WRITE WEIM_CSCR2U, 0x12020802
        WRITE WEIM_CSCR2L, 0x80330d03

@ ; config Flash to WRAP 8 mode (by half word accesses)
        WRITE_H (CS2_BASE_ADDR+0x2384), 0x60    @ ; offset = 0x11c2 << 1 for 16 bit port
        WRITE_H (CS2_BASE_ADDR+0x2384), 0x03

        WRITE_H (CS2_BASE_ADDR+0x0), 0xff       @ ; Flash to read mode

@ ; config to WEIM Sync access with WRAP8, 16 bit port
        WRITE WEIM_CSCR2U, 0x13510802
        WRITE WEIM_CSCR2L, 0x80330d03
```

Example 49-1. WEIM and Flash Memory Initialization for Working in Synchronous Mode

49.6 External Bus Timing Diagrams

The following timing diagrams show the timing of accesses to memory or a peripheral with different timing parameters. All examples done for $\overline{CS0}$, but are valid for any others chip select. $\overline{EB}$ means one from current used $\overline{EB}$[3:0]

For asynchronous mode:

- Figure 49-7 through Figure 49-13 show half-word read and write accesses to half-word-width memory.
- Figure 49-14 through Figure 49-24 shows word read and write accesses to half-word-width memory.
- Figure 49-25 shows page mode timing diagrams. Figure 49-26 through Figure 49-28 shows two kinds of $\overline{DTACK}$ accesses.
- Figure 49-43 and Figure 49-44 shows asynchronous data exchange in multiplexed address/data mode with word-width memory.

For synchronous (burst) mode:

- Figure 49-29 shows synchronous word read accesses to half-word-width memory.
- Figure 49-30 shows recommended parameter setting using for synchronous accesses: BCLK clock has a short pulse at the end of access.
- Different cases of wrap/increment states on AHB and memory are shown in the Figure 49-32 through Figure 49-39 for burs size 4. AHB increment/wrap accesses with another length are made like this.
- PSRAM read and write accesses are shown in the Figure 49-40 and Figure 49-41.
- Figure 49-45 and Figure 49-46 show synchronous data exchange in multiplexed address/data mode with word width memory.

49.6.1 Asynchronous Memory Accesses Timing Diagrams

49.6.1.1 AHB Half-Word Access to Half-Word Width Memory

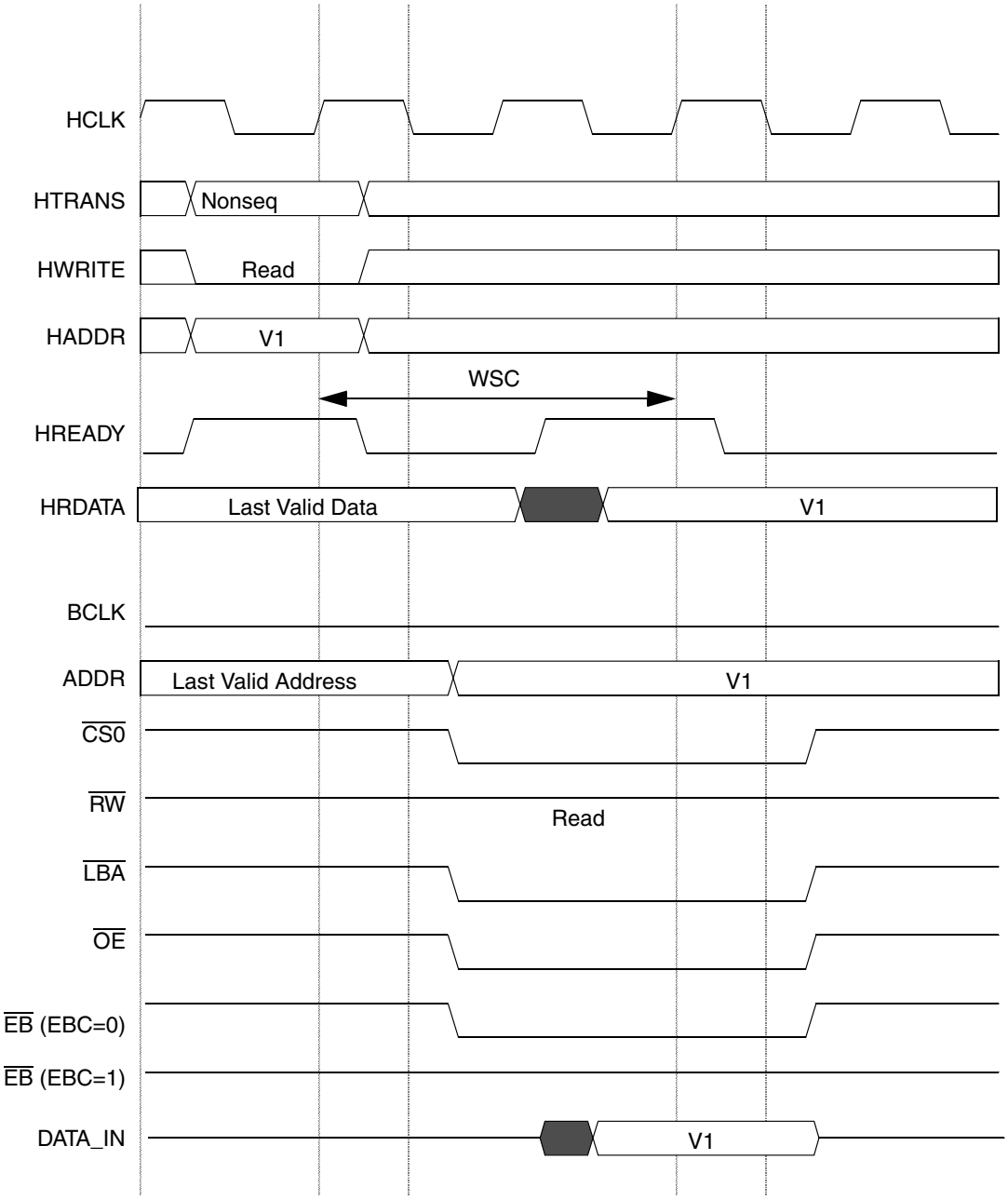


Figure 49-7. Read Access, WSC=1

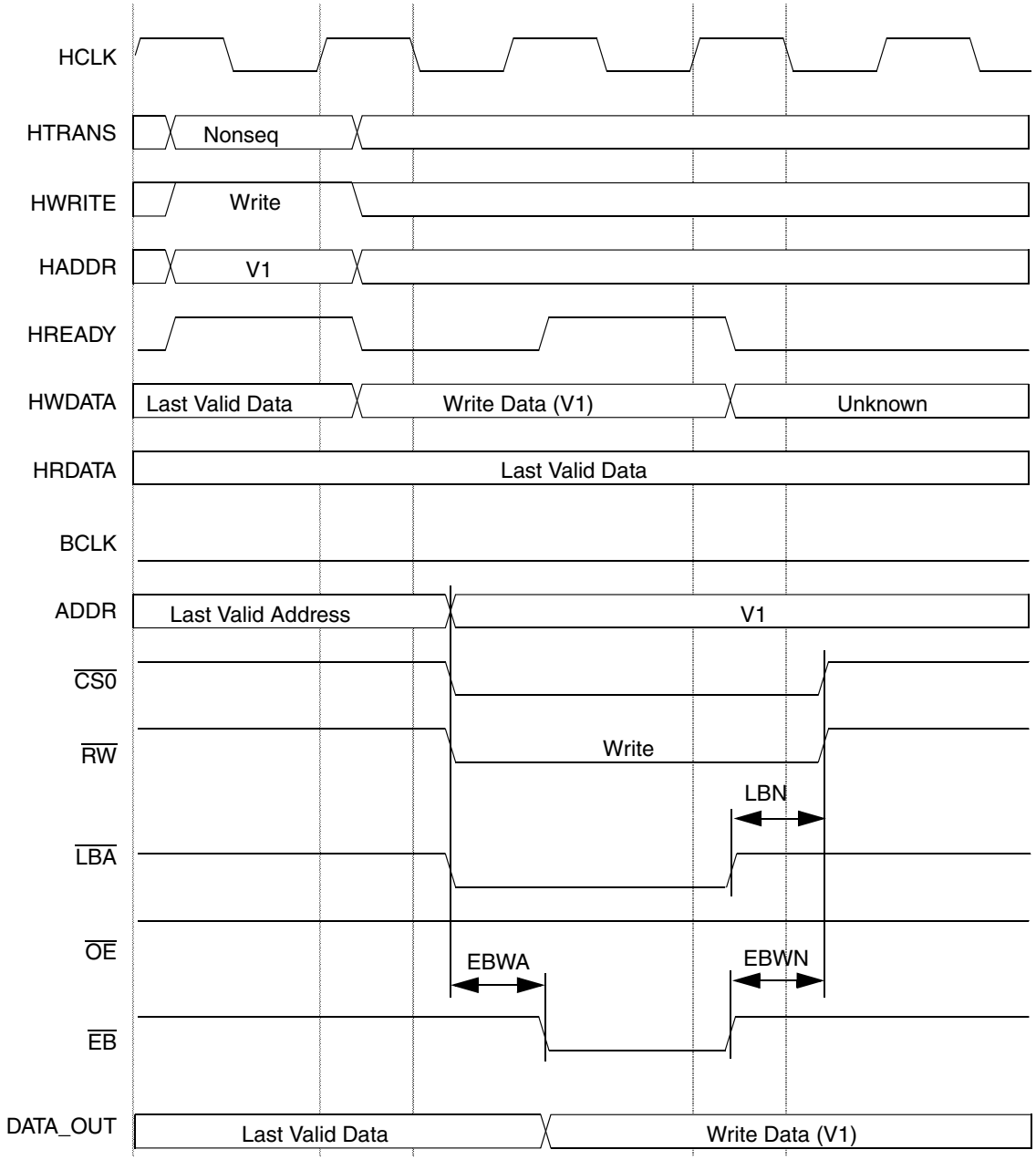


Figure 49-8. Write Access, WSC=1, EBWA=1, EBWN=1, LBN=1

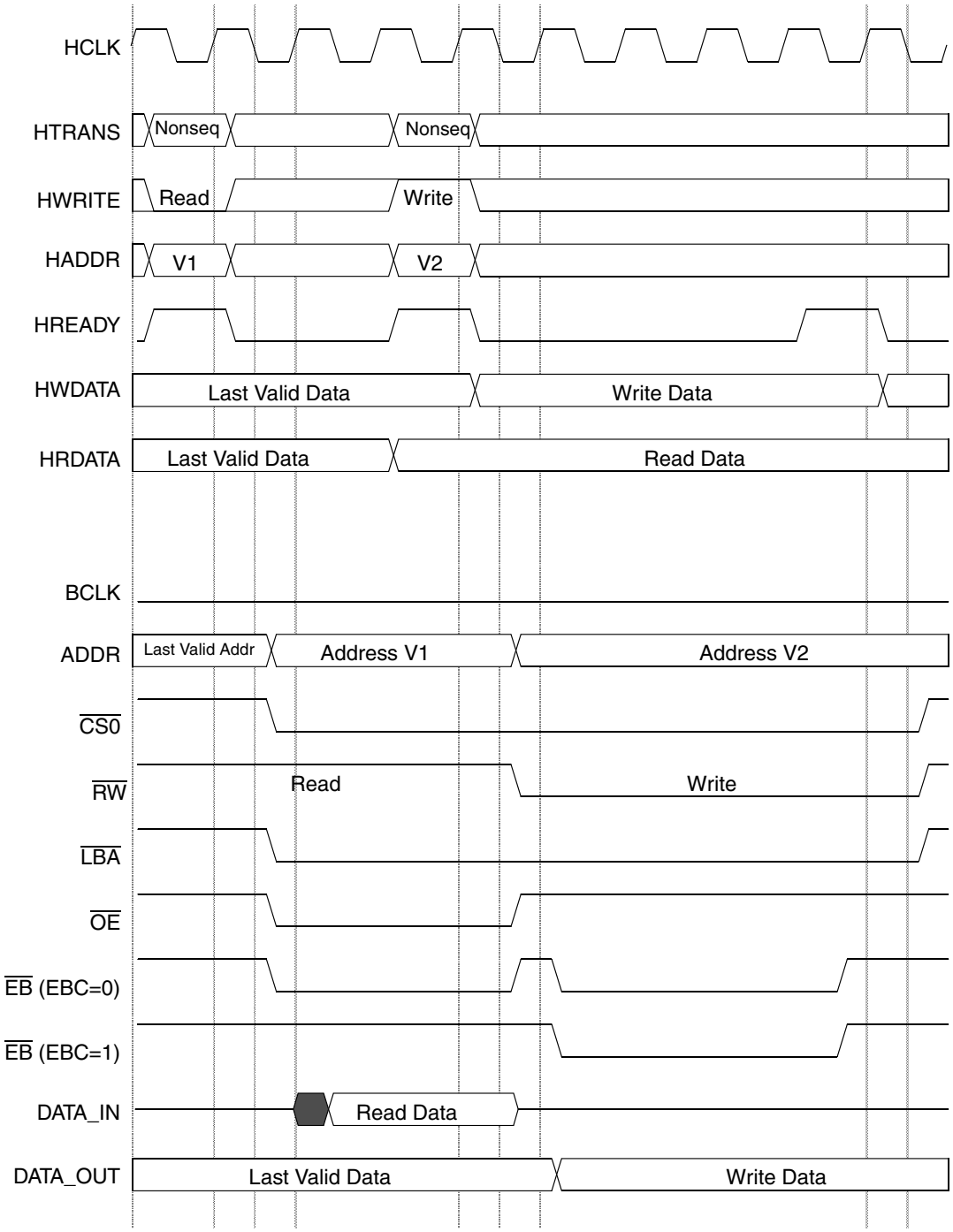


Figure 49-9. Read and Write Accesses, WSC=2, WWS=2, EBWA=1, EBWN=2

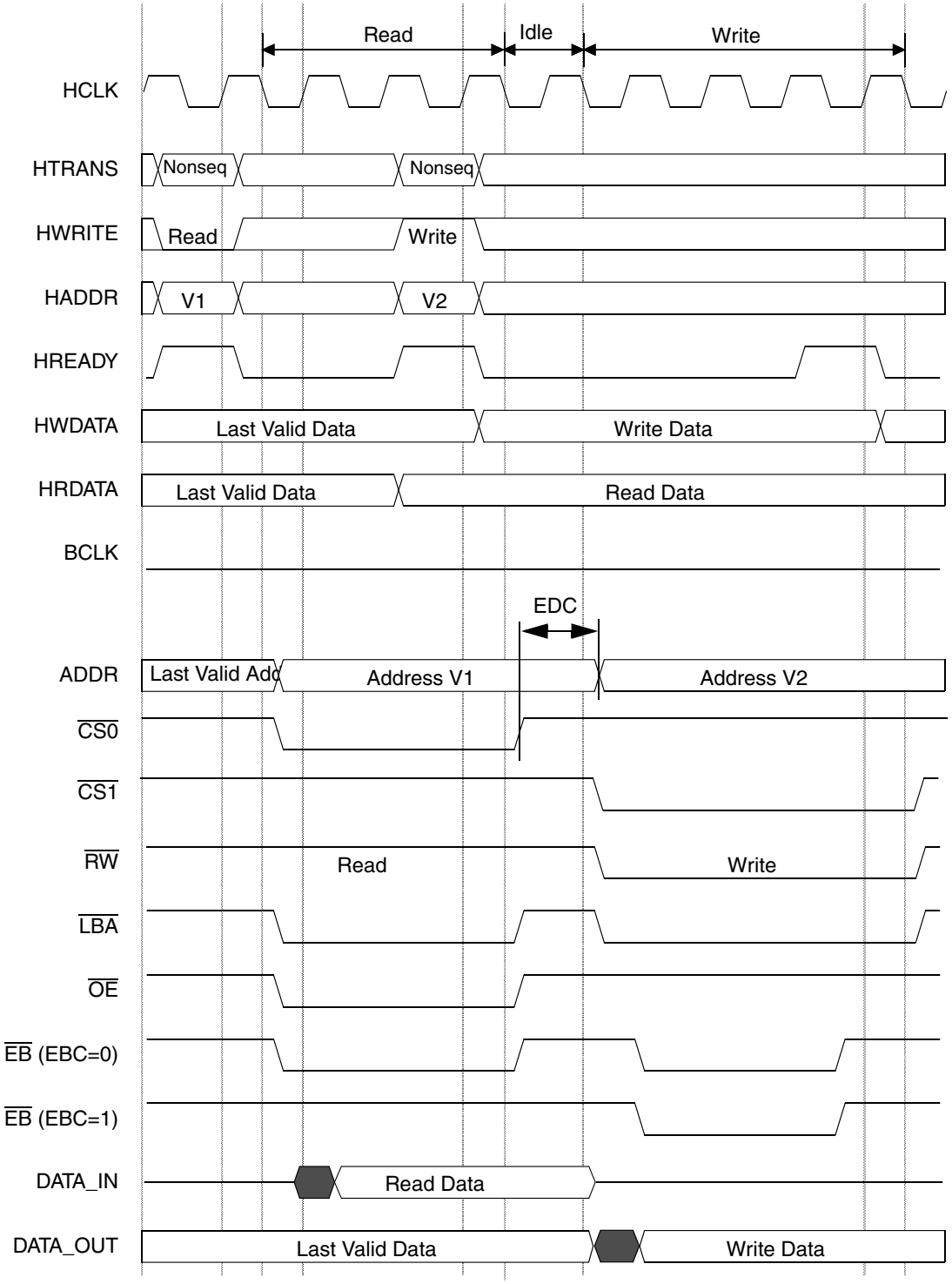


Figure 49-10. Read and Write Accesses, WSC=2, WWS=1, EBWA=1, EBWN=2, EDC=1

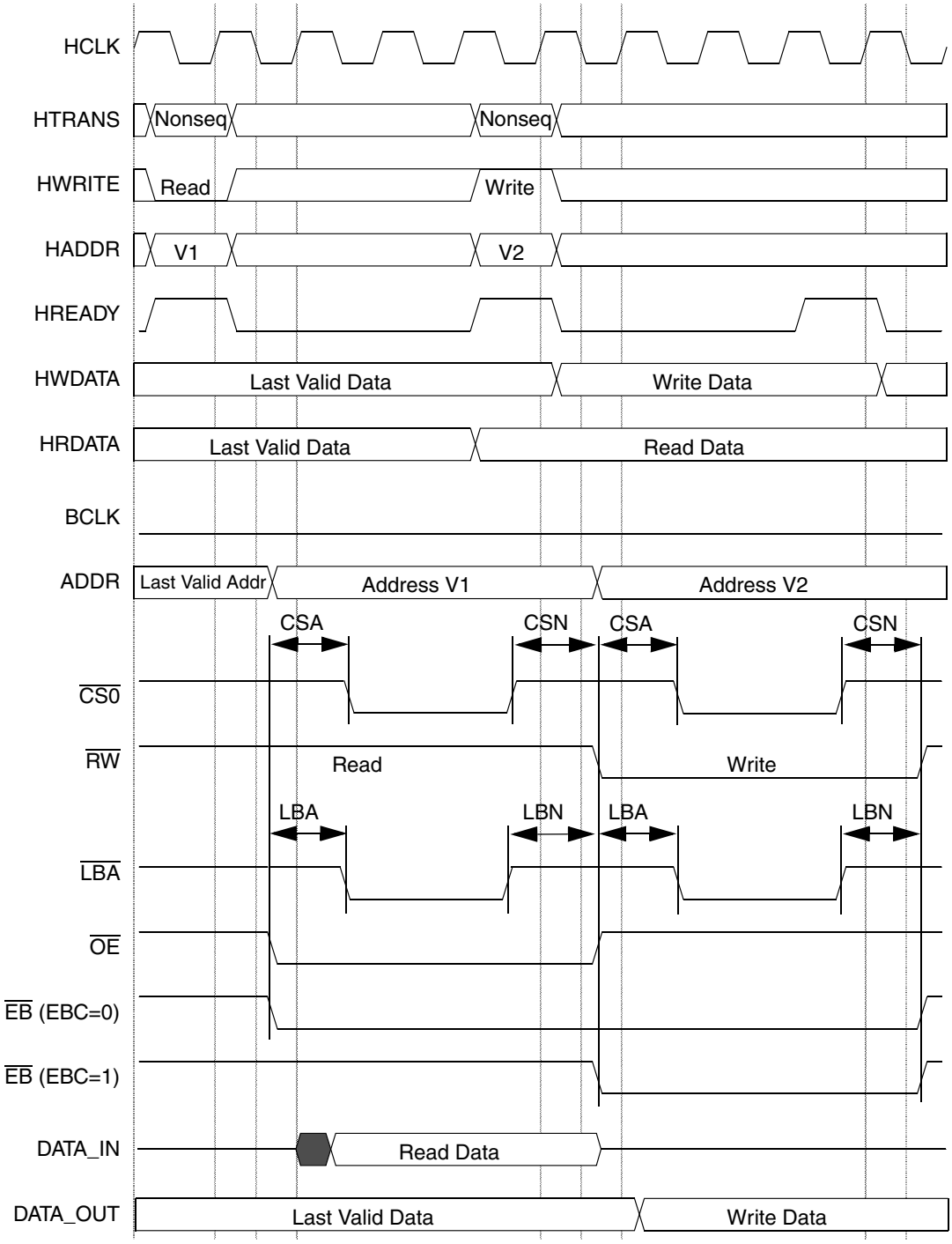


Figure 49-11. Read and Write Accesses, WSC=3, CSA=1, CSN=1, LBA=1, LBN=1

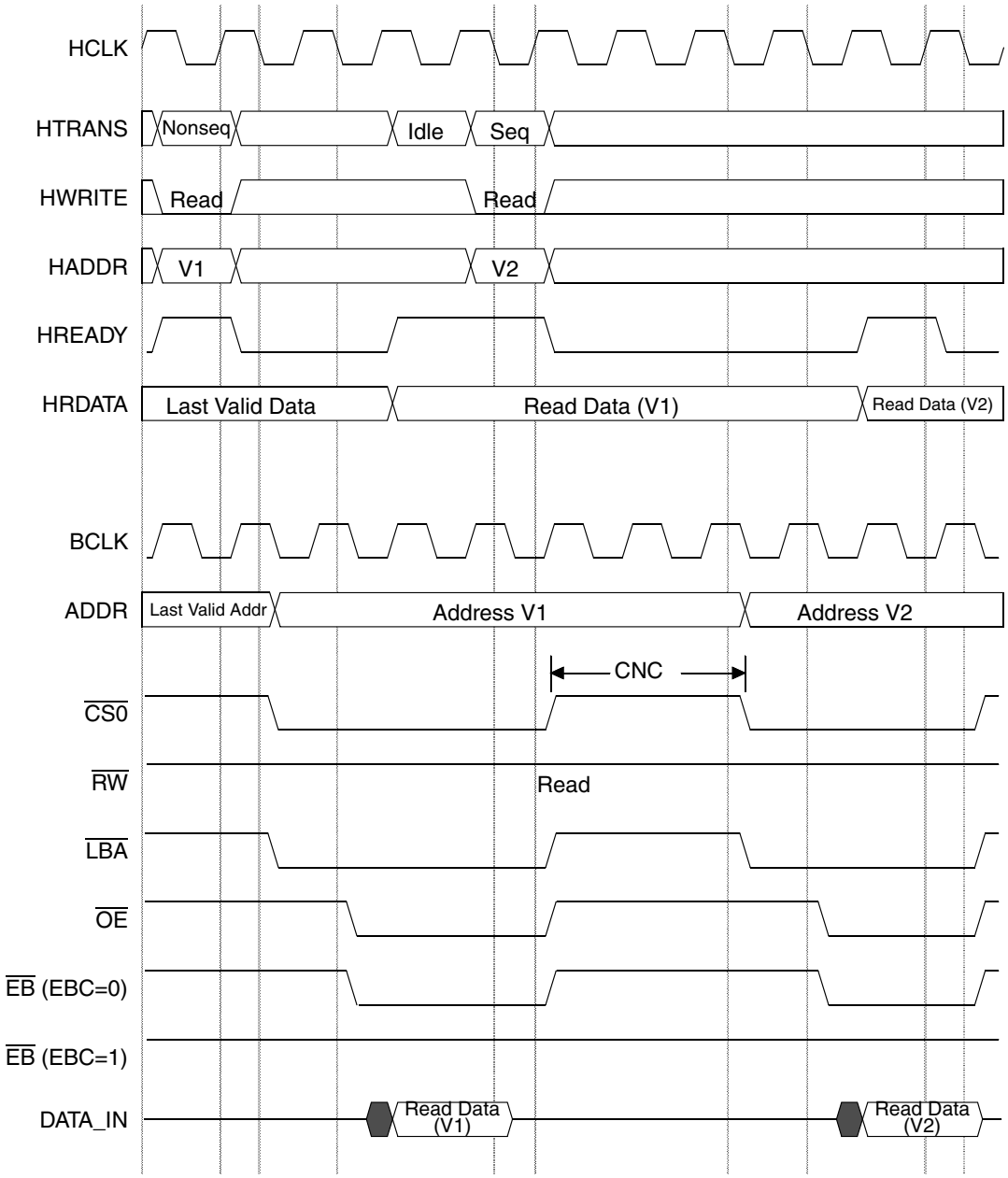


Figure 49-12. Read Accesses, WSC=2, OEA=2, CNC=2, BCM=1, EBRA=2

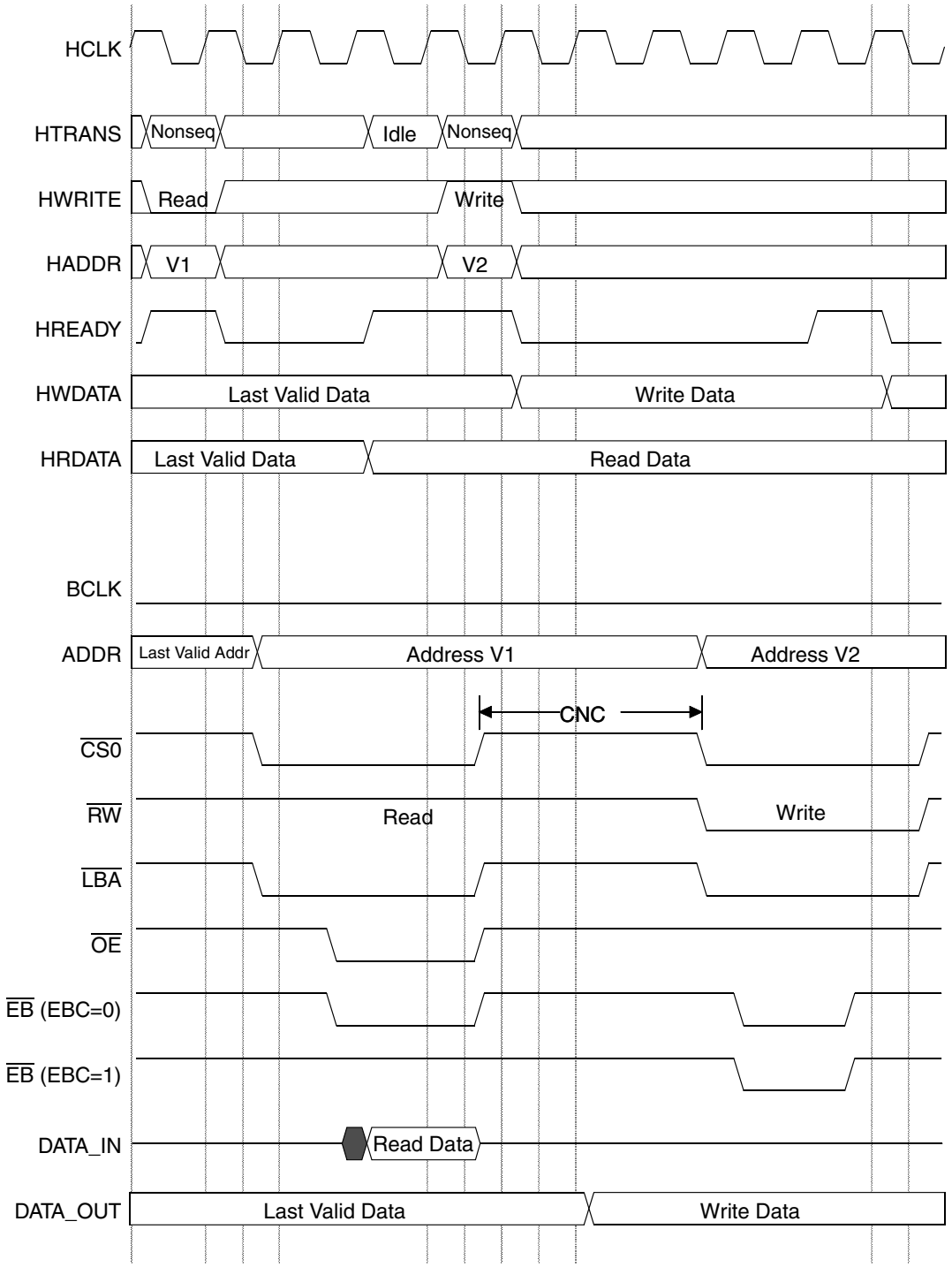


Figure 49-13. Read and Write Accesses, WSC=2, OEA=2, EBWA=1, EBWN=2, CNC=2, EBRA=2

49.6.1.2 AHB Word Access to Half-word Width Memory

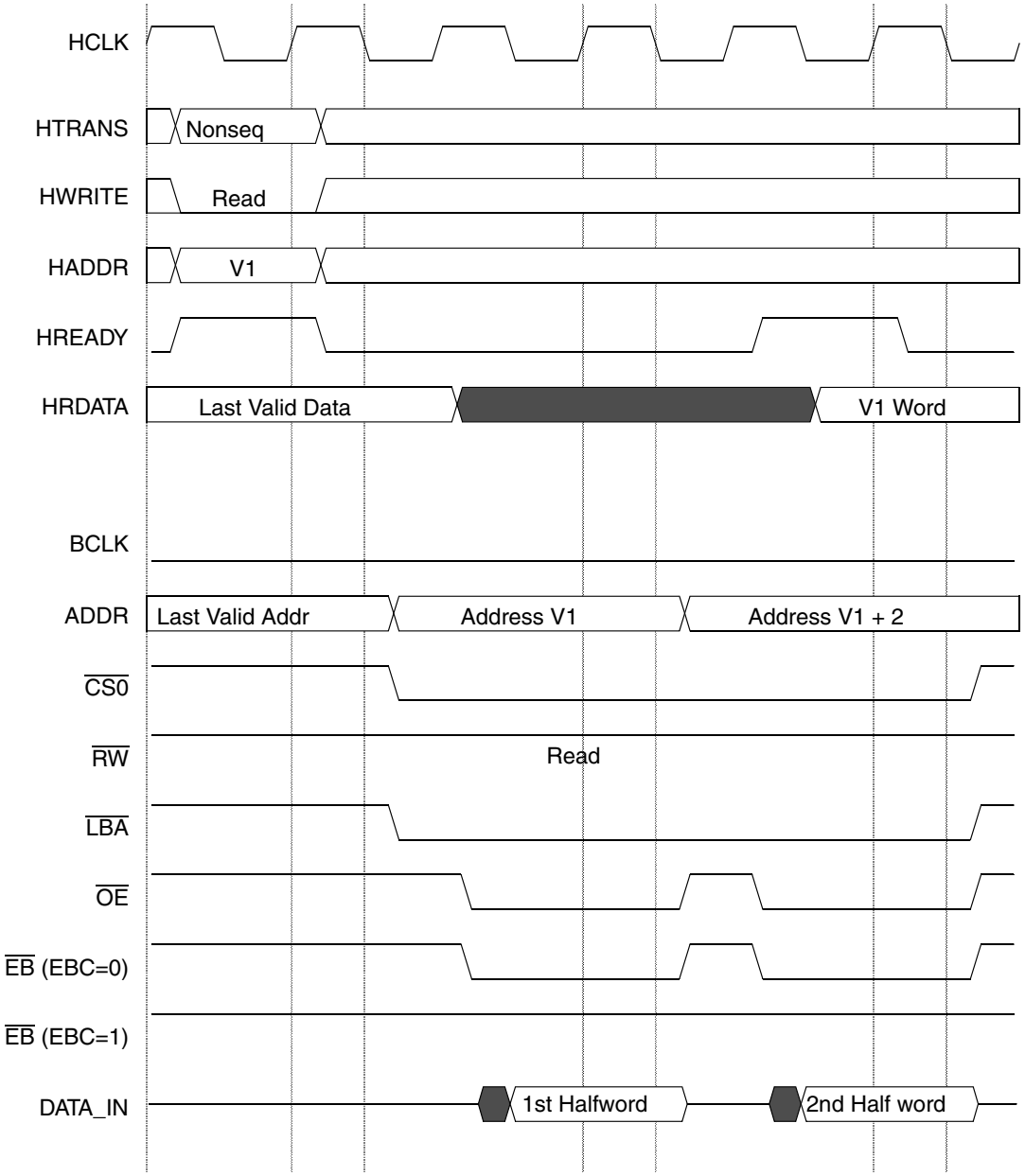


Figure 49-14. Read Access, WSC=1, OEA=1, EBRA=1

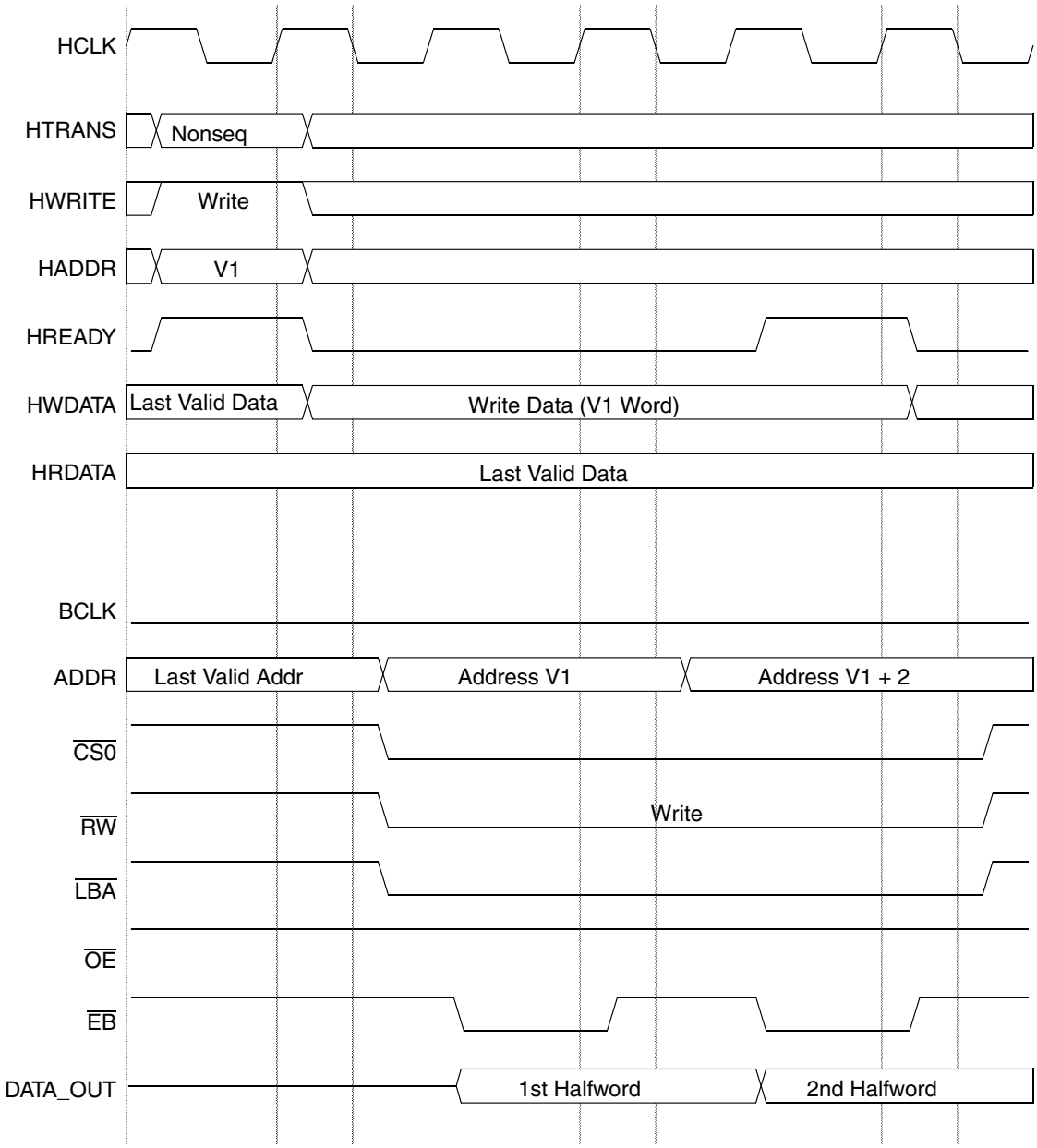


Figure 49-15. Write Access, WSC=1, EBWA=1, EBWN=1

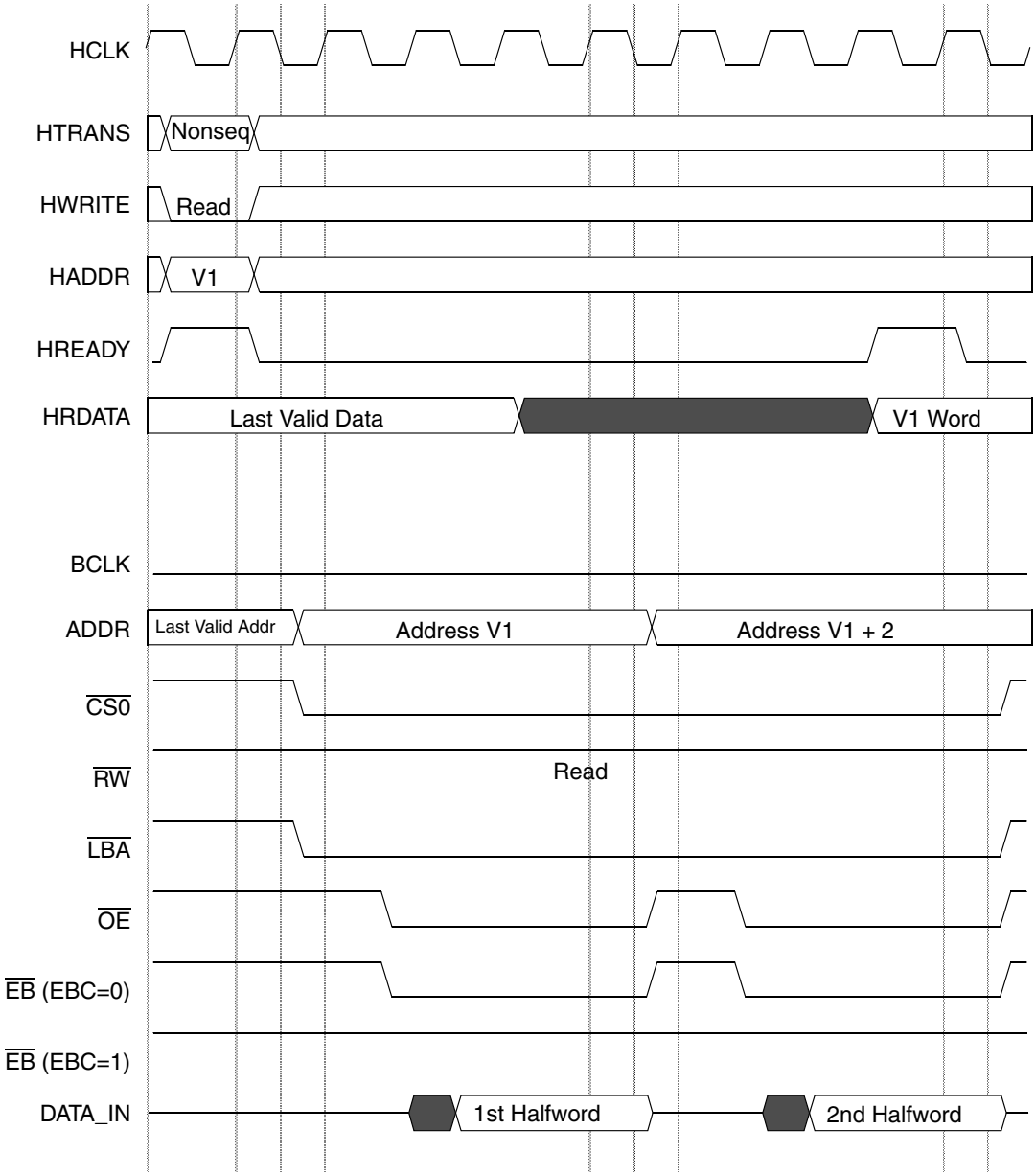


Figure 49-16. Read Access, WSC=3, OEA=2, EBRA=2

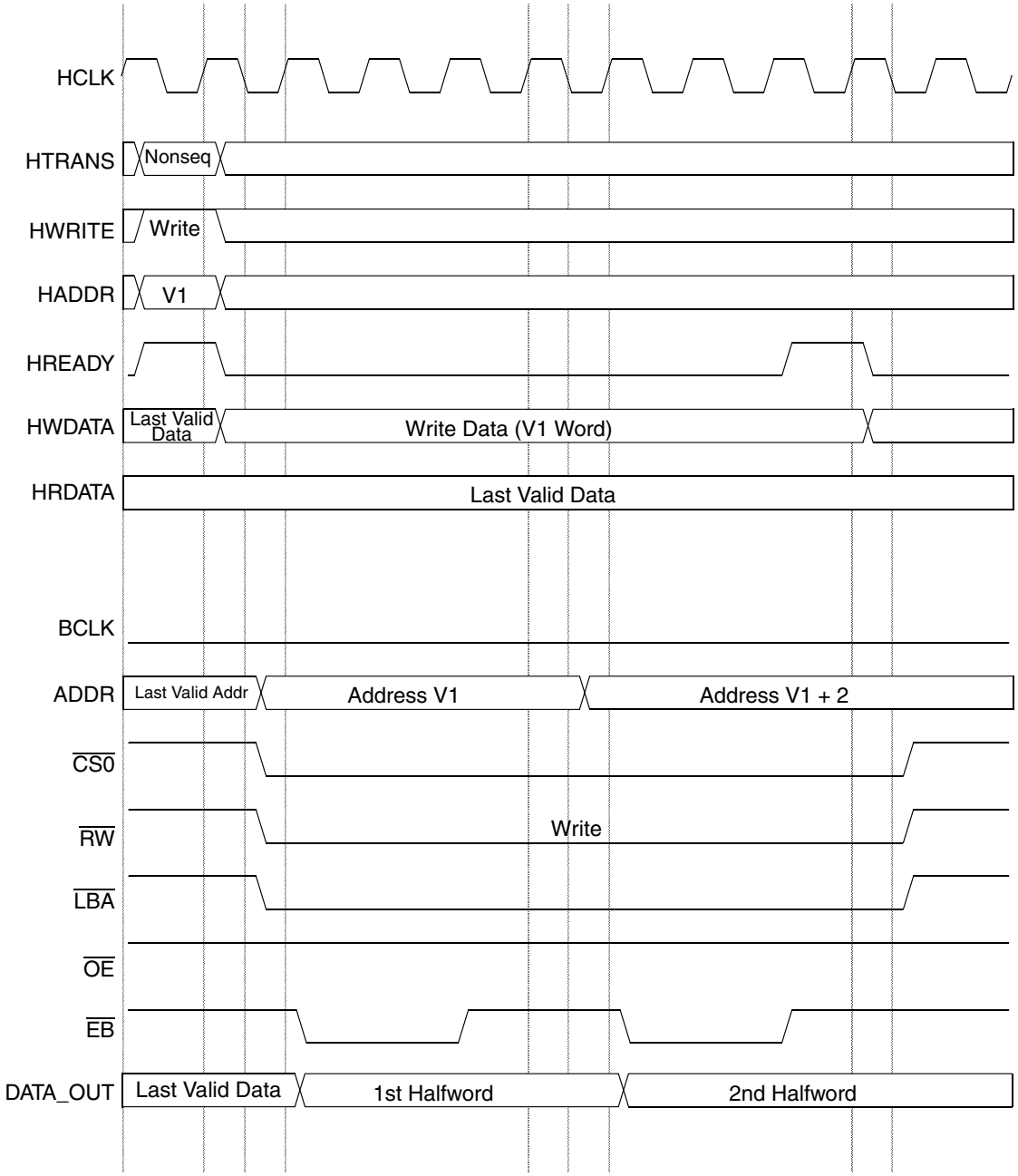


Figure 49-17. Write Access, WSC=3, EBWA=1, EBWN=3

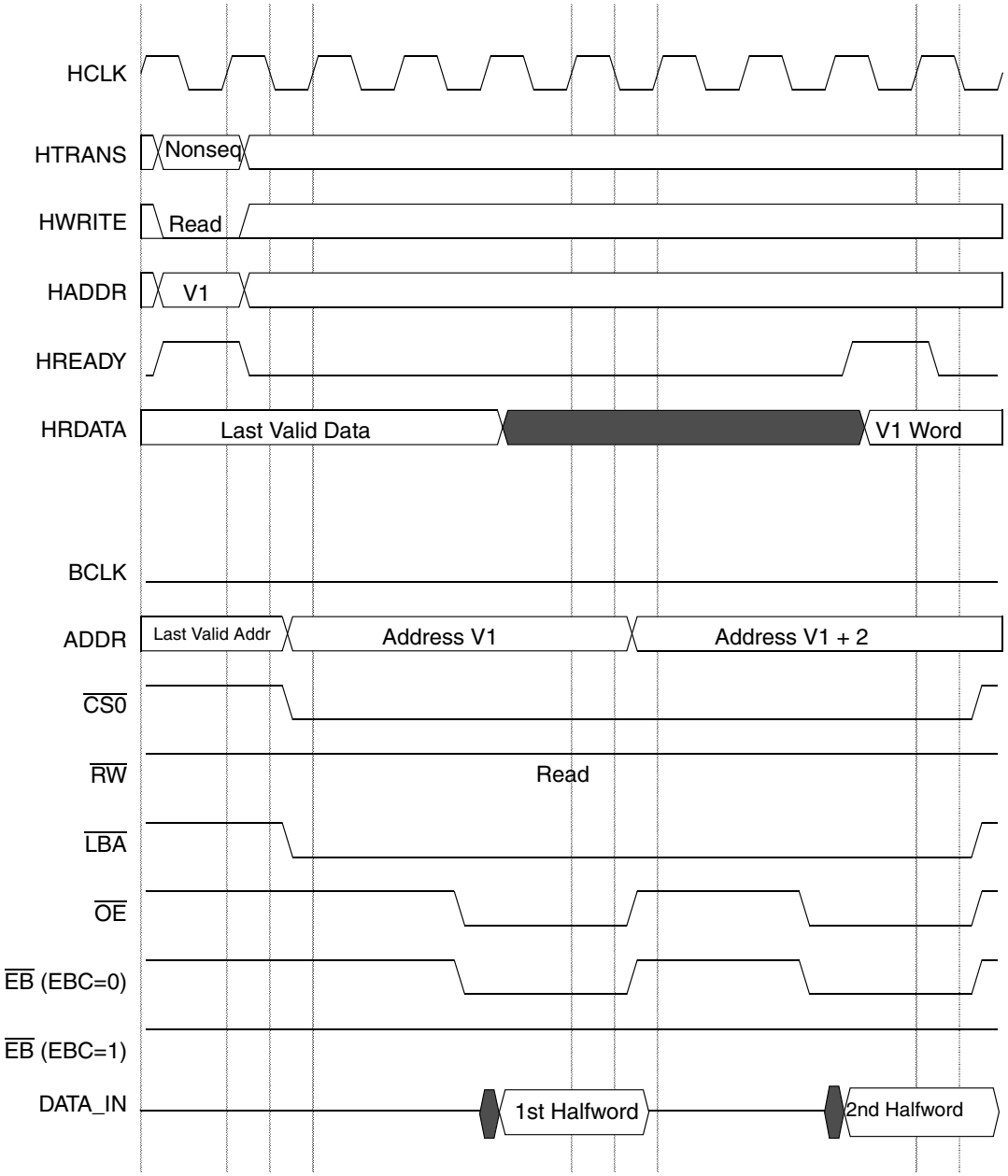


Figure 49-18. Read Access, WSC=3, OEA=4, EBRA=4

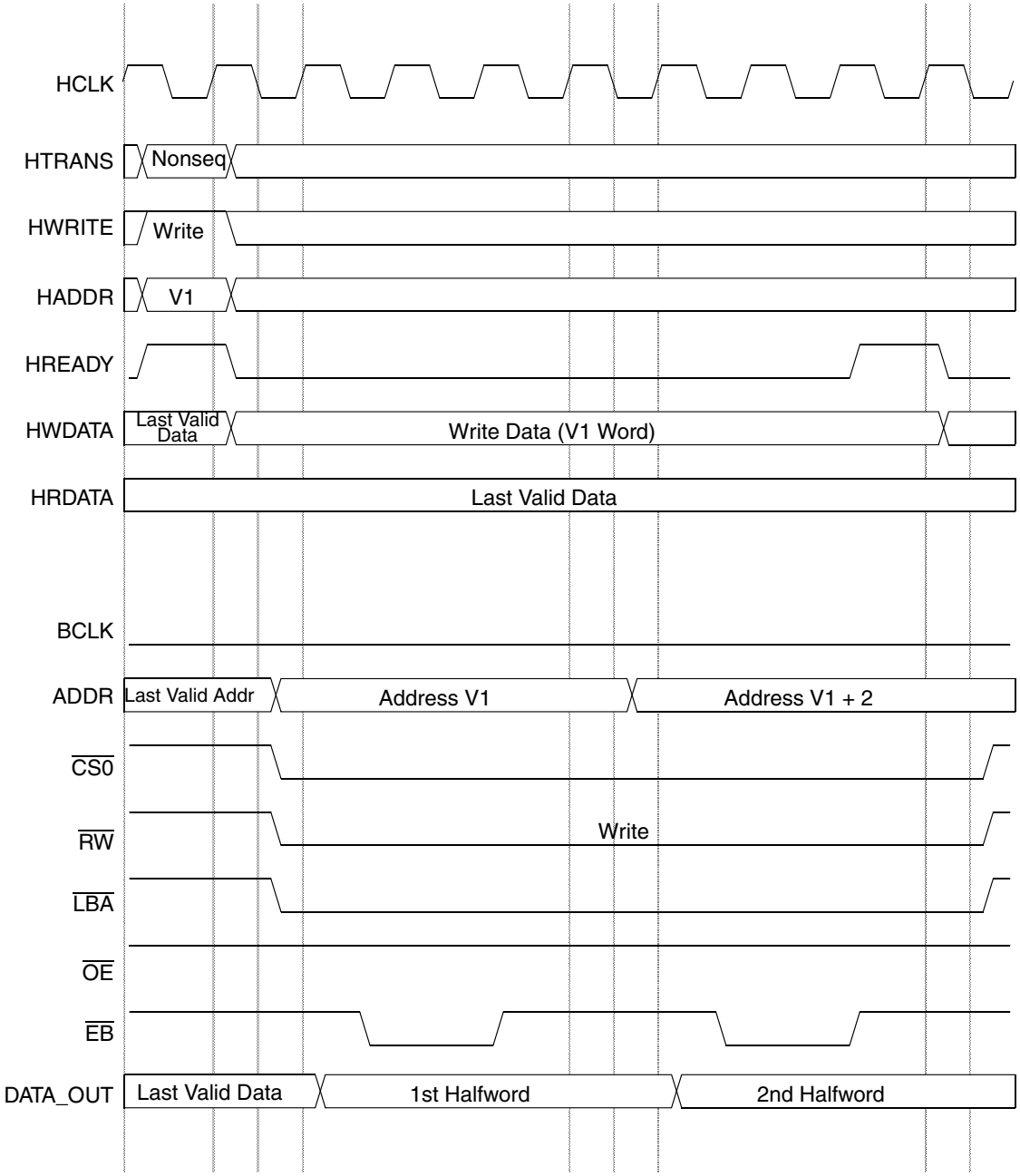


Figure 49-19. Write Access, WSC=3, EBWA2, EBWN=3

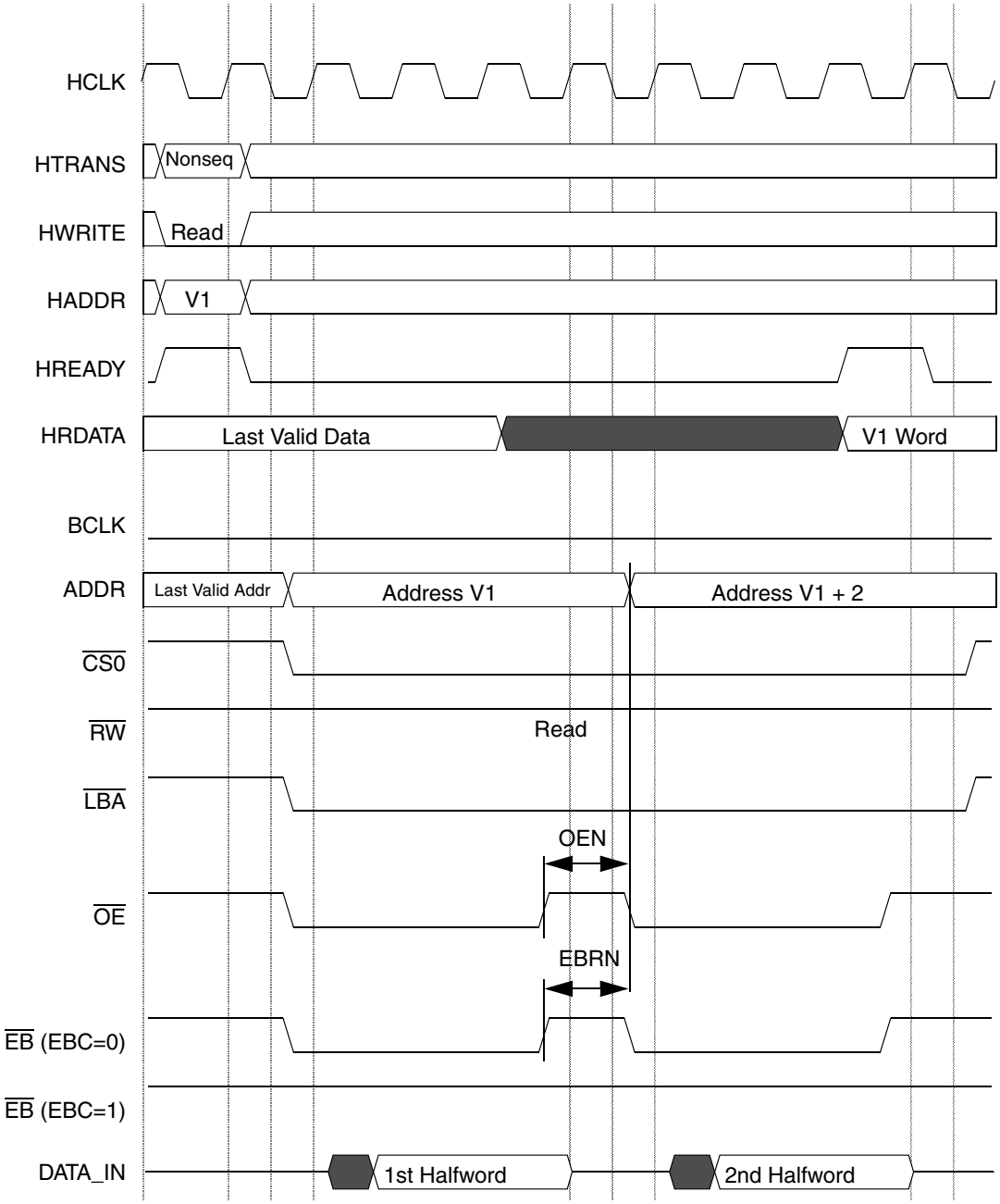


Figure 49-20. Read Access, WSC=3, OEN=2, EBRN=2

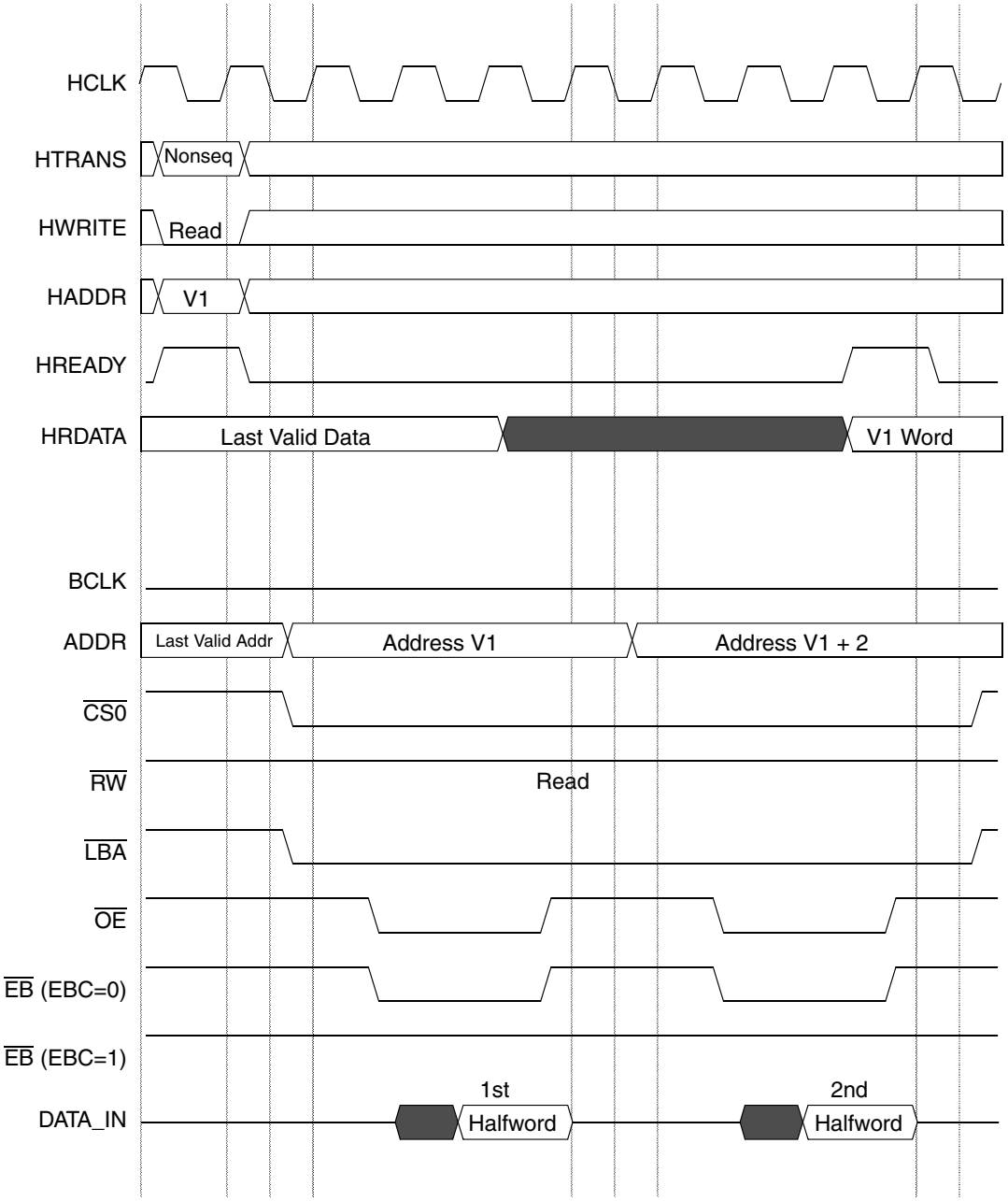


Figure 49-21. Read Access, WSC=3, OEA=2, OEN=2, EBRA=2, EBRN=2

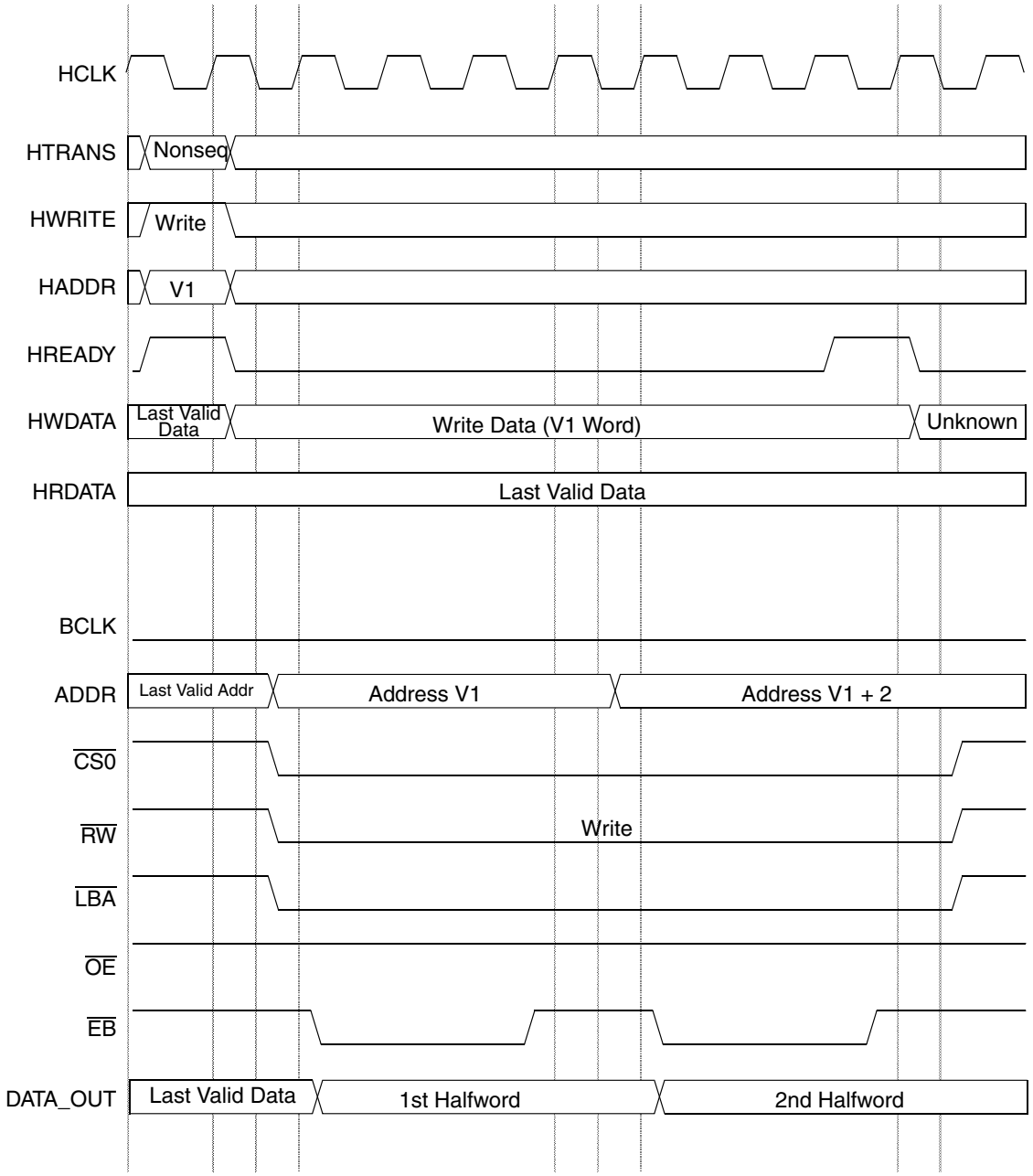


Figure 49-22. Write Access, WSC=2, WWS=1, EBWA=1, EBWN=2

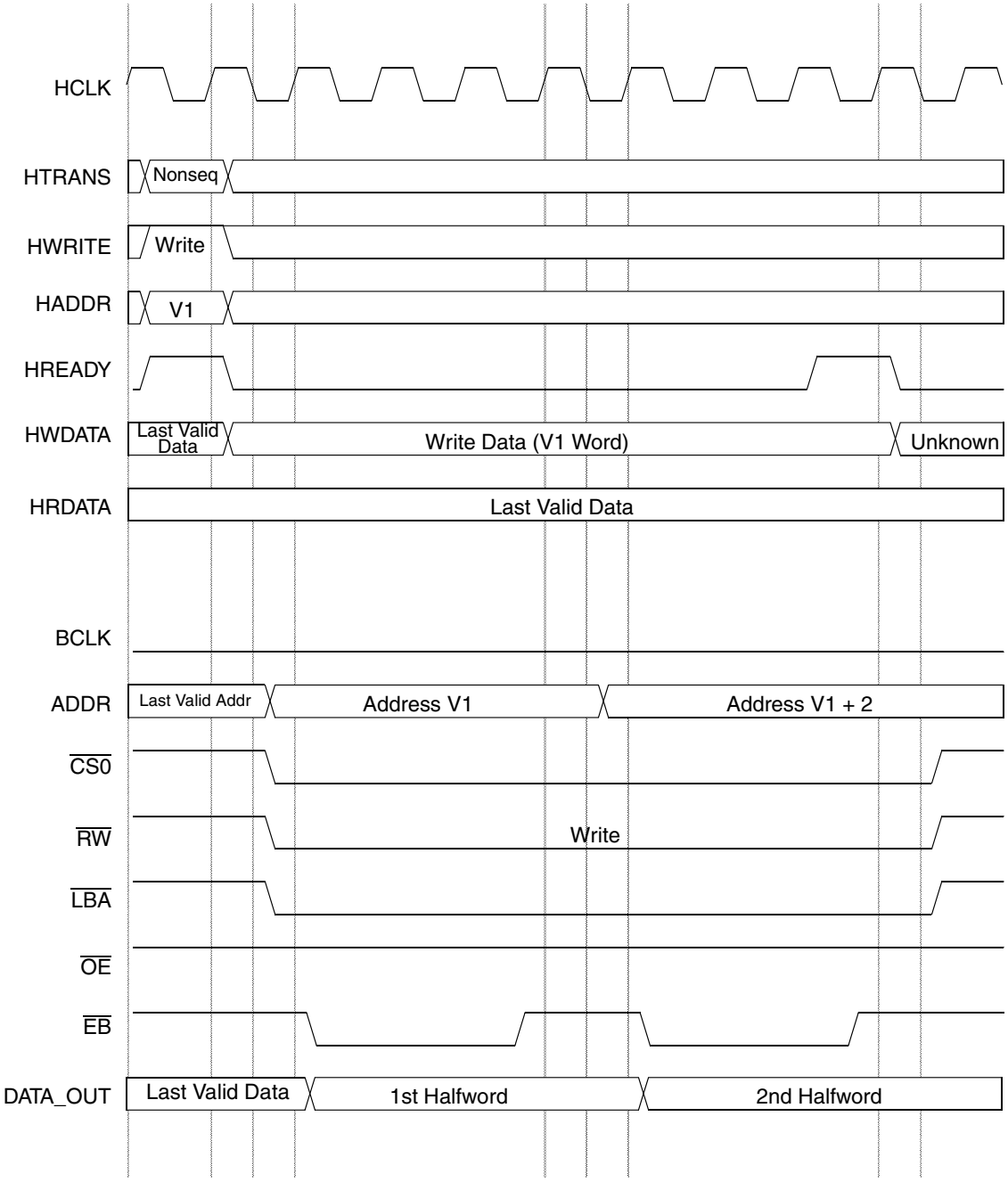


Figure 49-23. Write Access, WSC=1, WWS=2, EBWA=1, EBWN=2

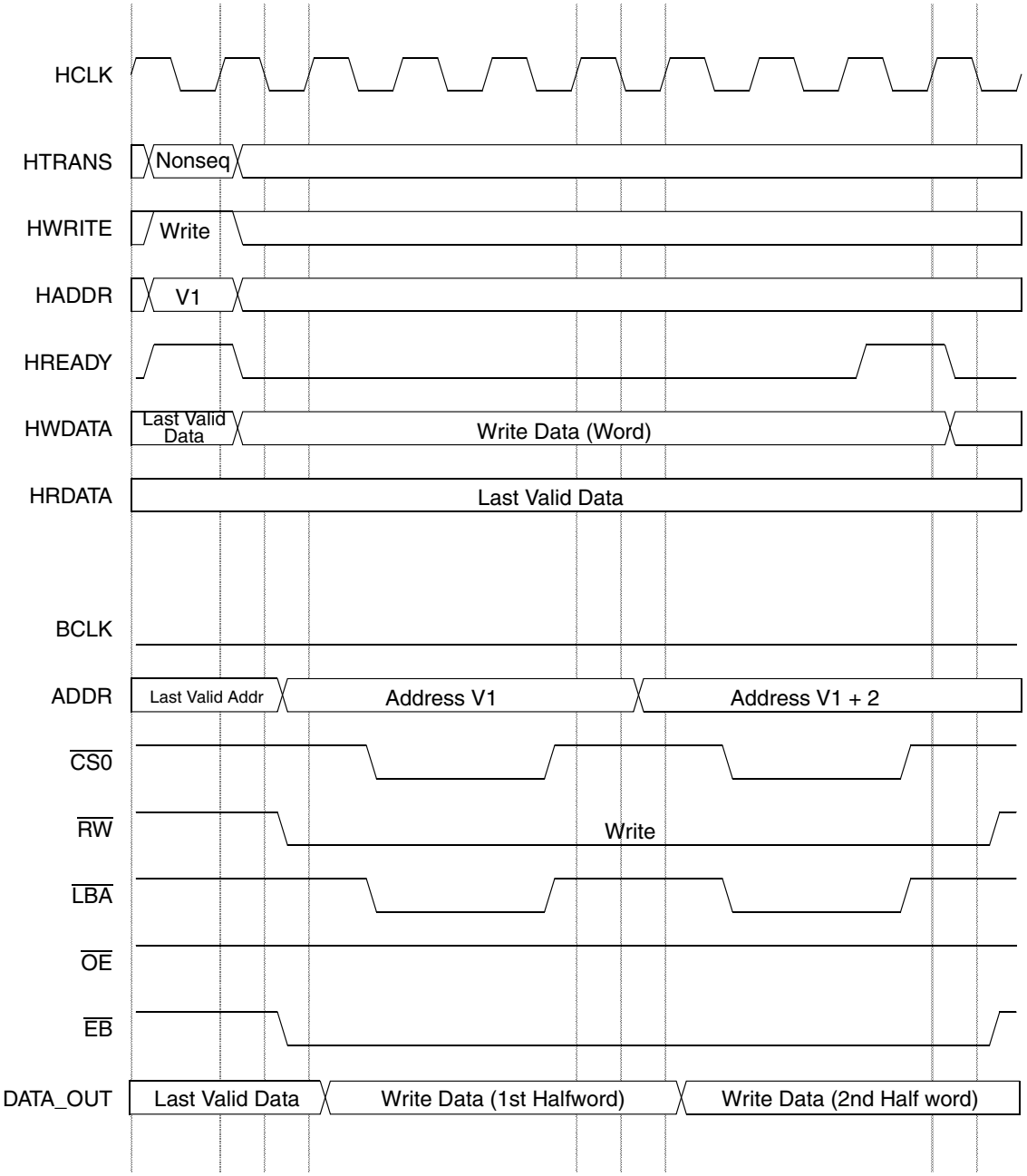


Figure 49-24. Write Access, WSC=2, CSA=1, WWS=1, CSN=1

49.6.2 Page Mode Timing Diagrams

49.6.2.1 AHB Word Accesses to Half-word Width Memory

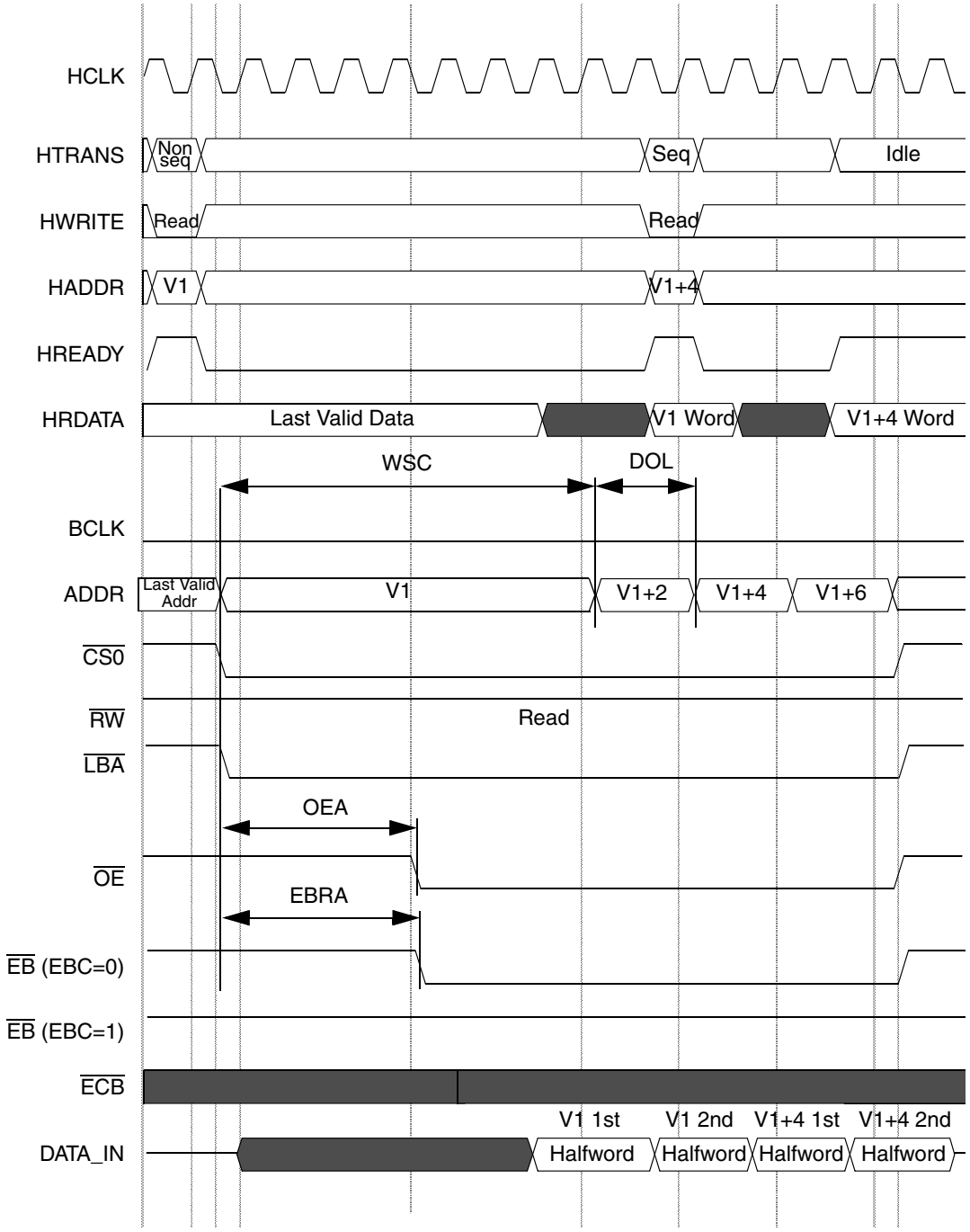


Figure 49-25. Sequential Read Access, WSC=7, OEA=8, PME=1, SYNC=1, DOL=1, EBRA=8

49.6.3 DTACK Mode Memory Accesses Timing Diagrams

49.6.3.1 AHB Word Accesses to Word-width Memory

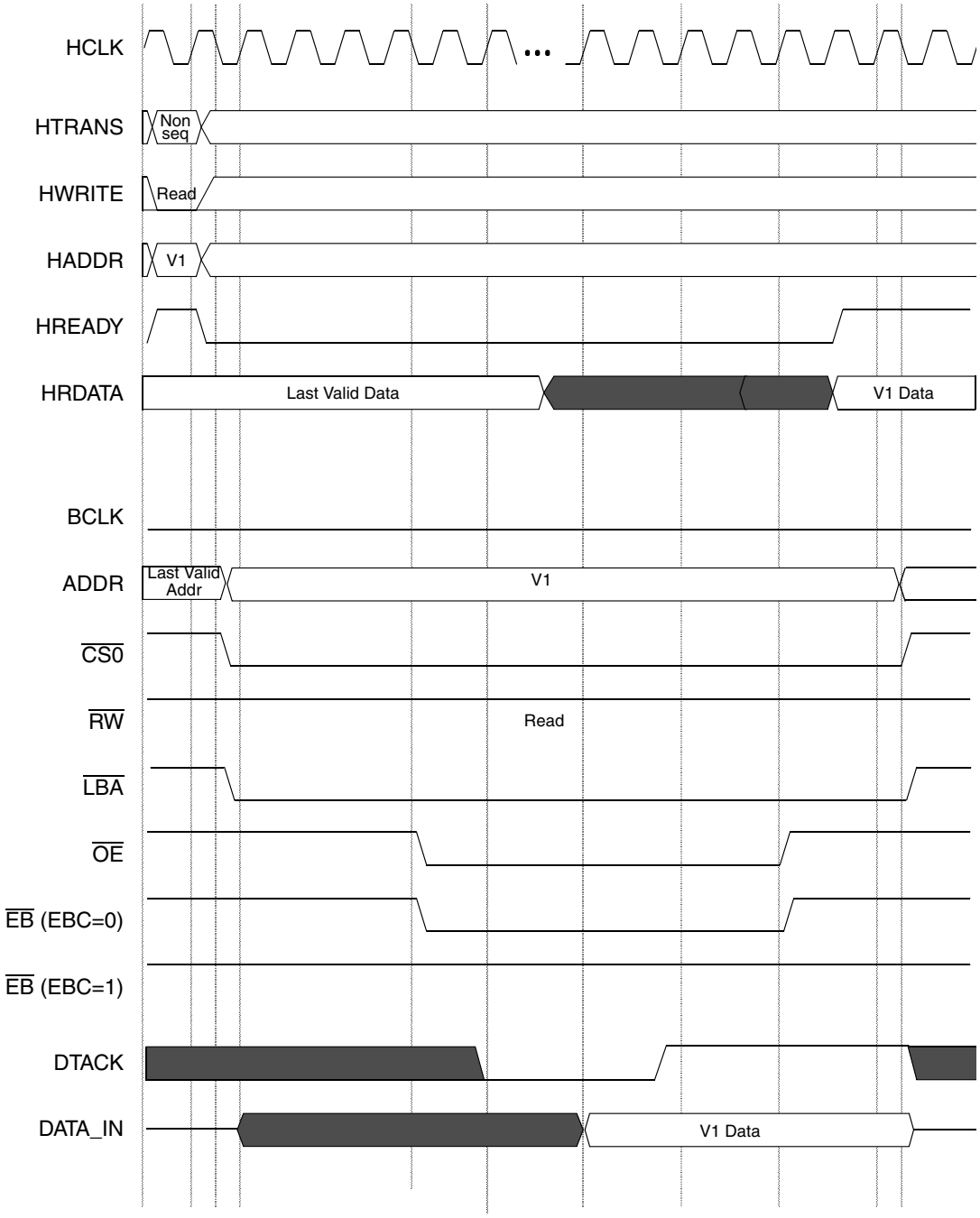


Figure 49-26. Read Access, WSC=3F, OEA=8, OEN=5, EBRA=8, EBRN=5

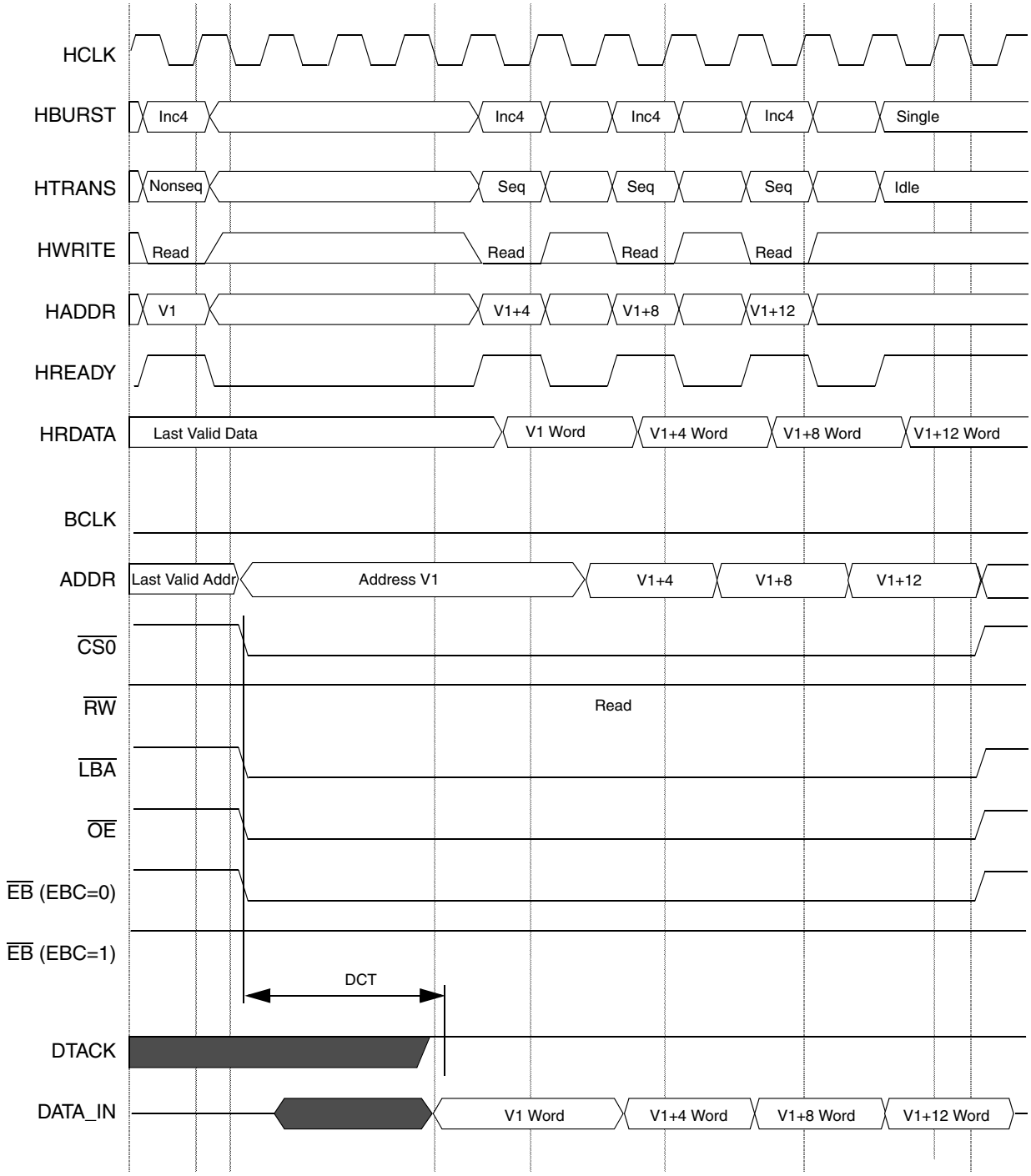


Figure 49-27. Sequential Read Accesses, WSC=1, EW=1, DCT=1

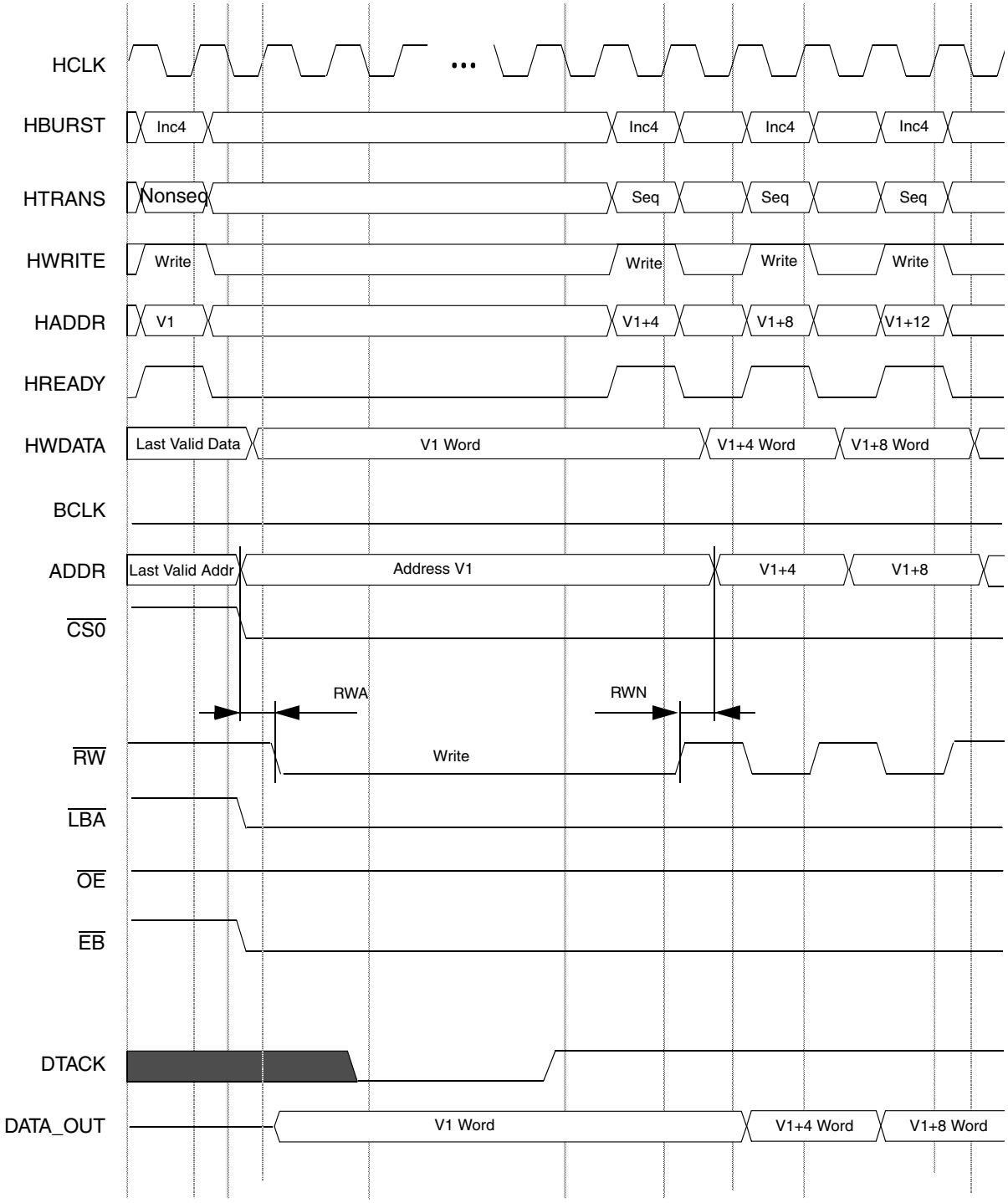


Figure 49-28. Sequential Write Accesses, WSC=1, EW=1, RWA=1, RWN=1

49.6.4 Burst Memory Accesses Timing Diagrams

49.6.4.1 AHB Word Accesses to Half-word Width Memory

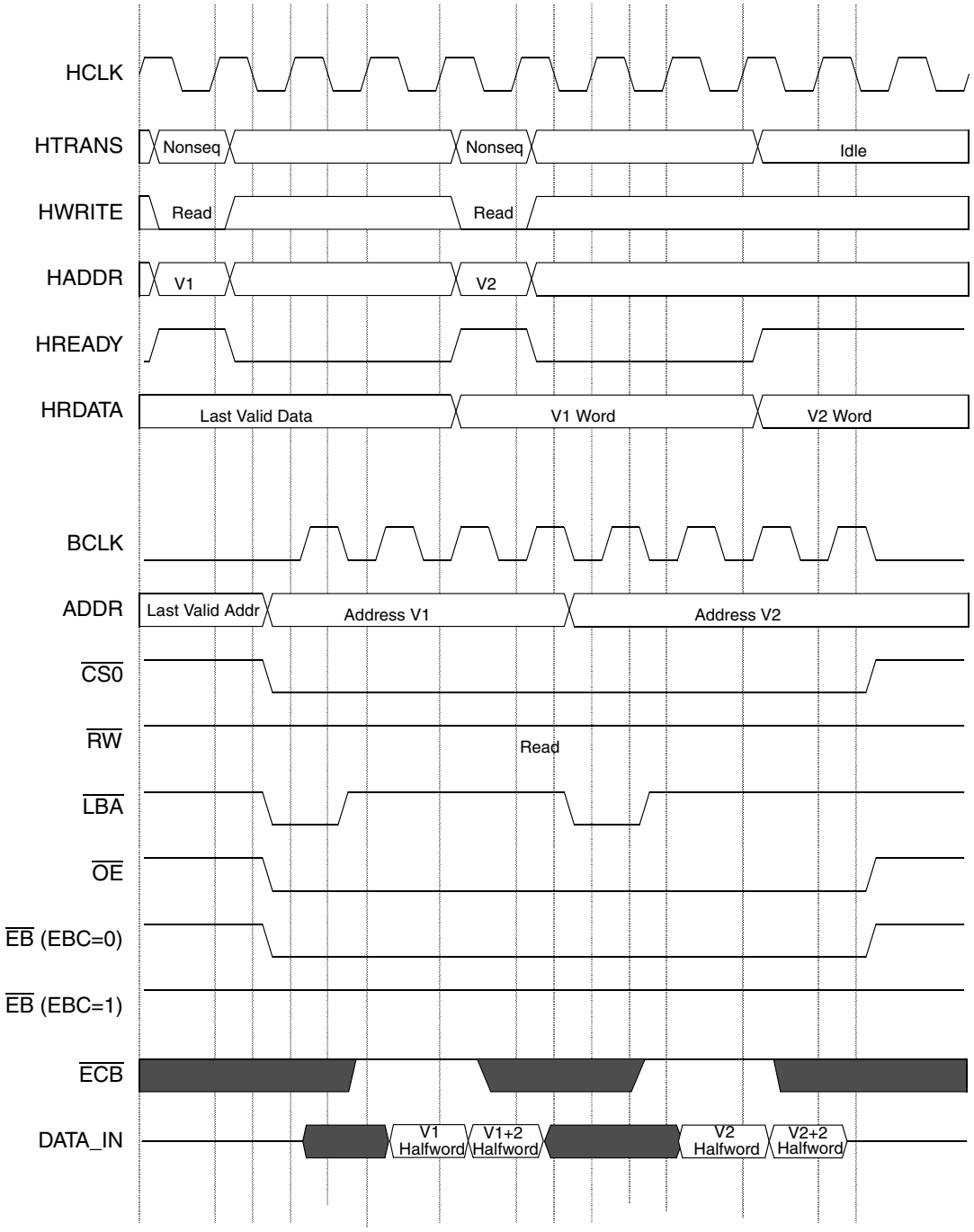


Figure 49-29. Non-sequential Read Accesses, WSC=2, SYNC=1, DOL=0

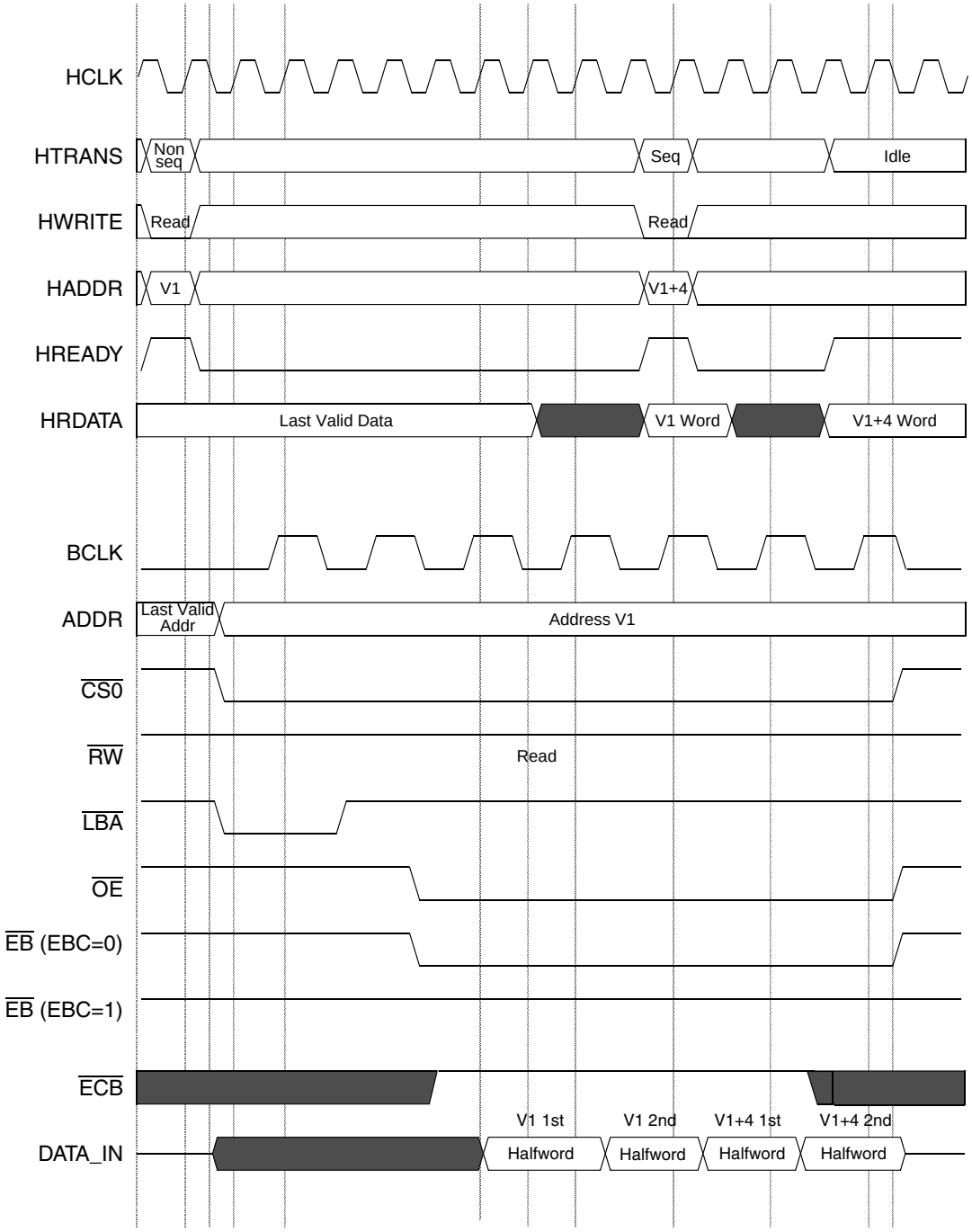


Figure 49-30. Sequential Read Access, WSC=7, OEA=8, SYNC=1, DOL=1, BCD=1, BCS=1, EBRA=8

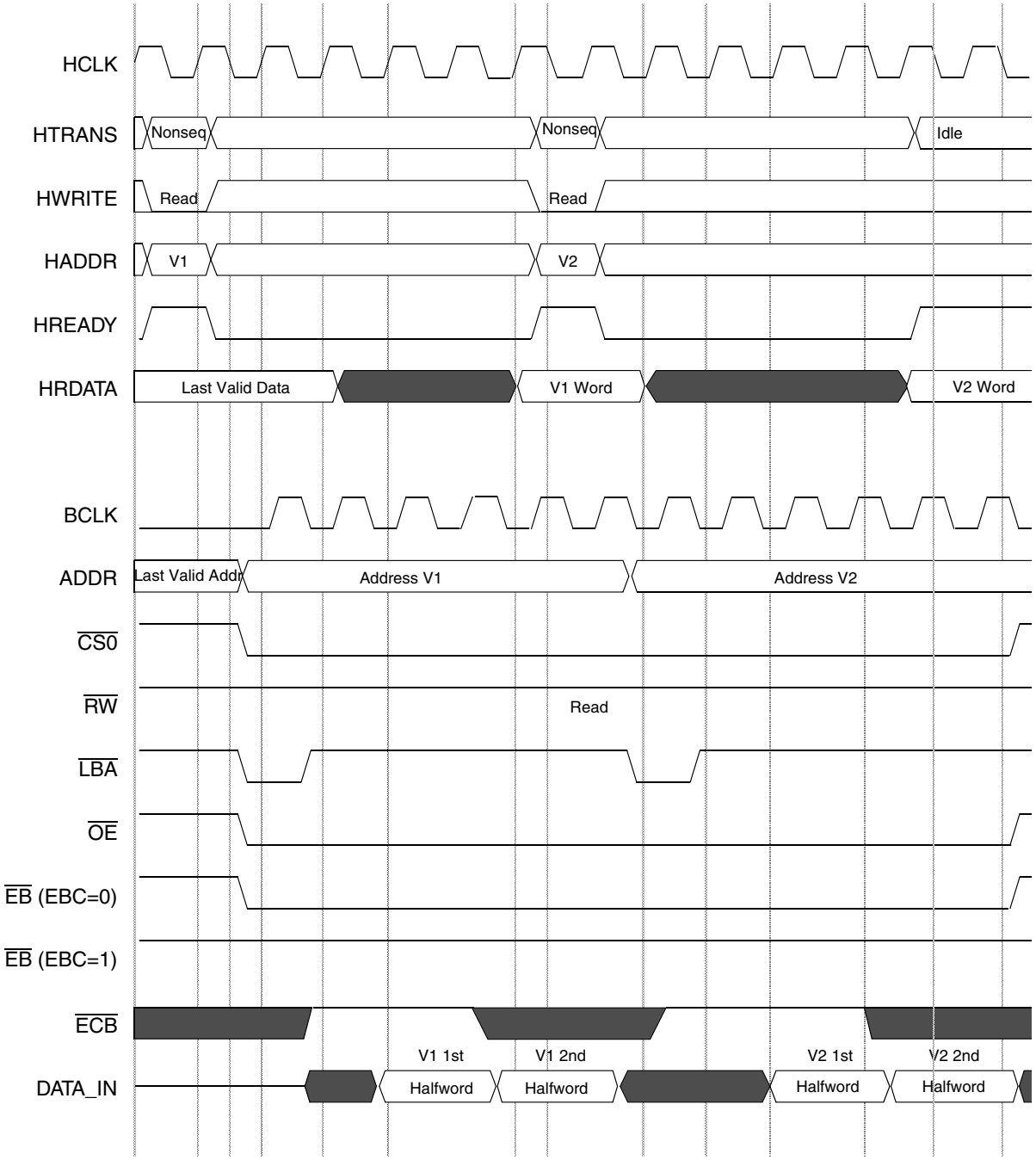


Figure 49-31. Non-sequential Read Accesses, WSC=3, SYNC=1, DOL=1

49.6.4.2 AHB Accesses to Word-width Burst Memory

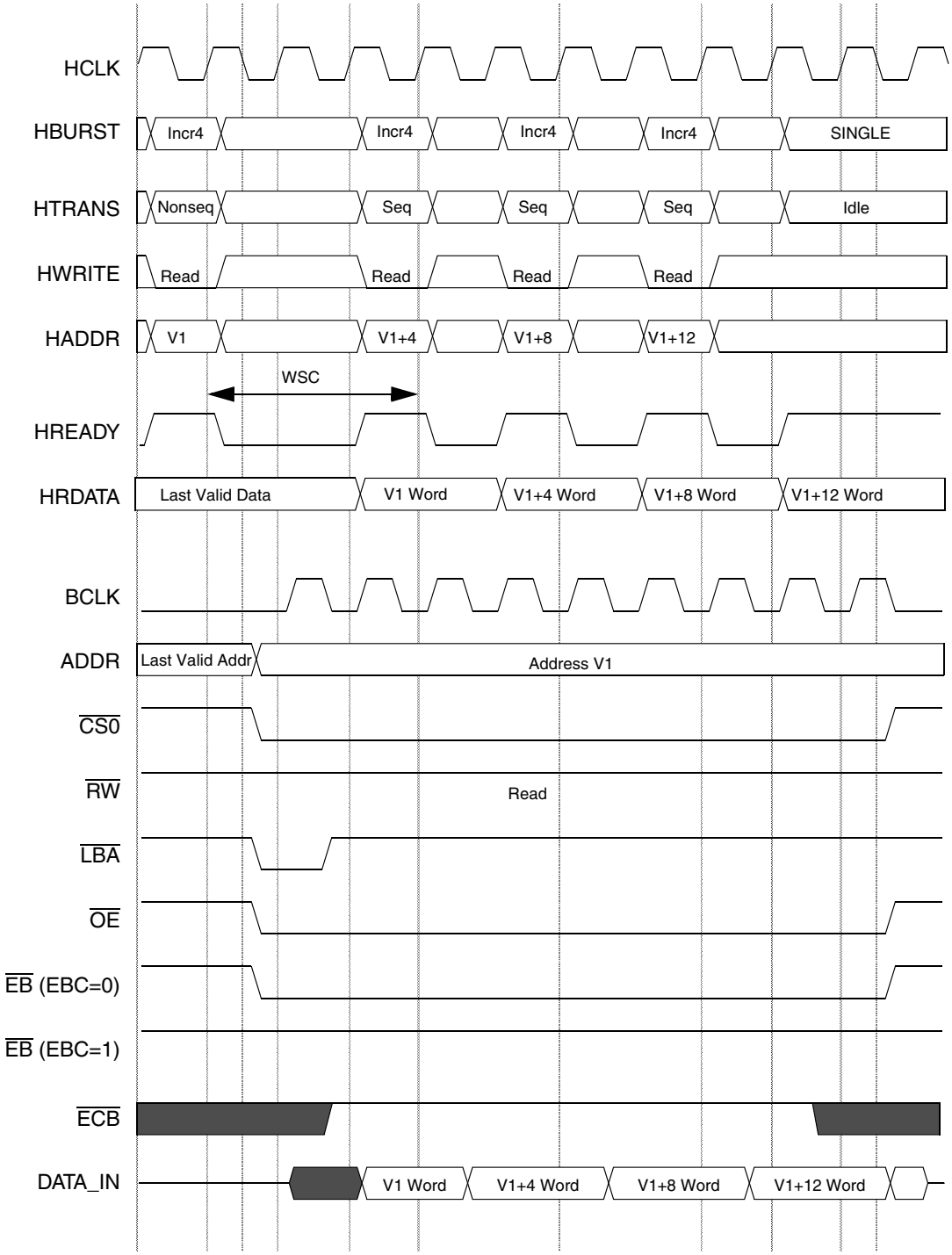


Figure 49-32. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 49-32 any address may be a four word boundary address, but not a memory boundary address.

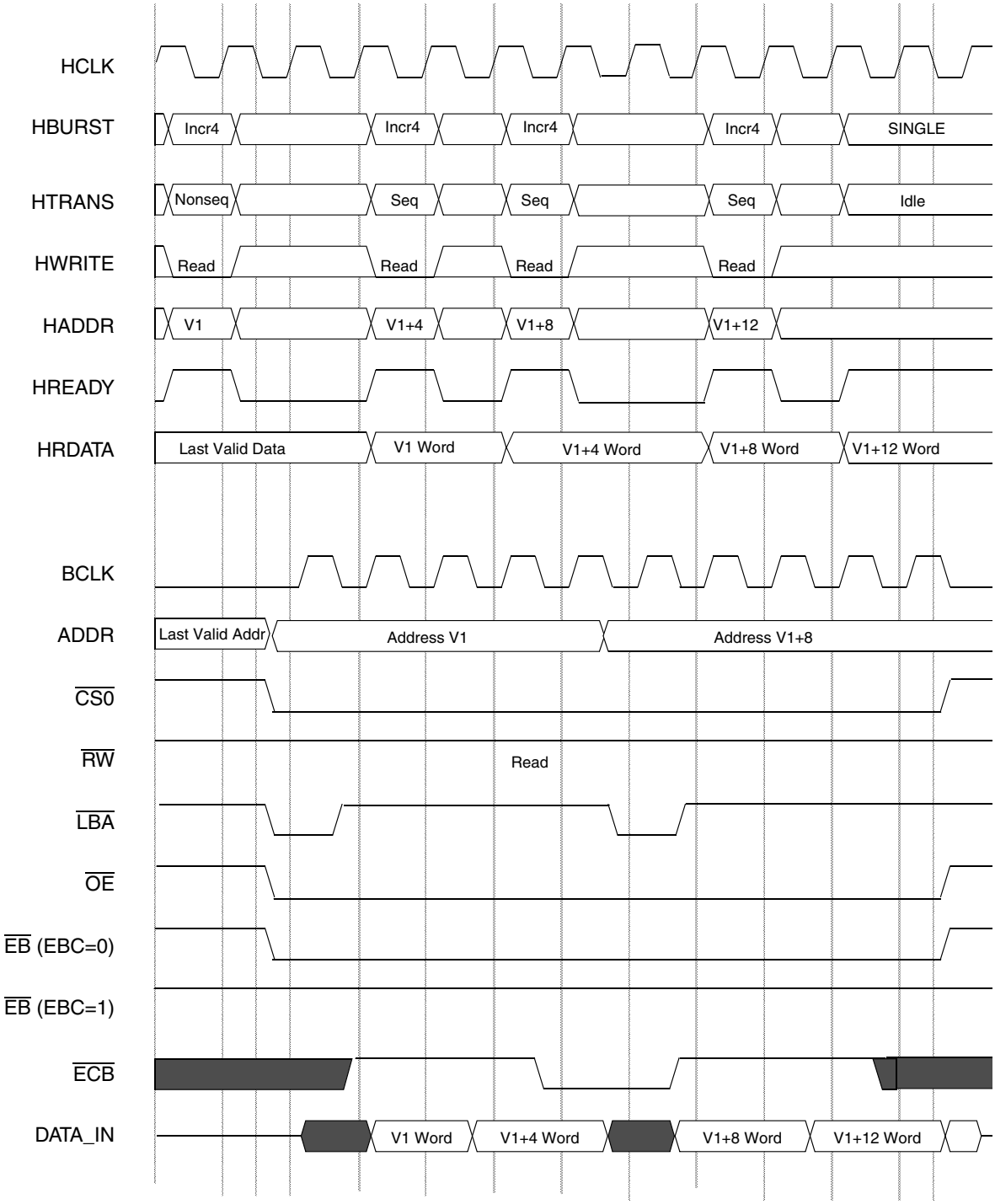


Figure 49-33. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 49-33 (V1+8) is a memory boundary address and may be a four word boundary address.

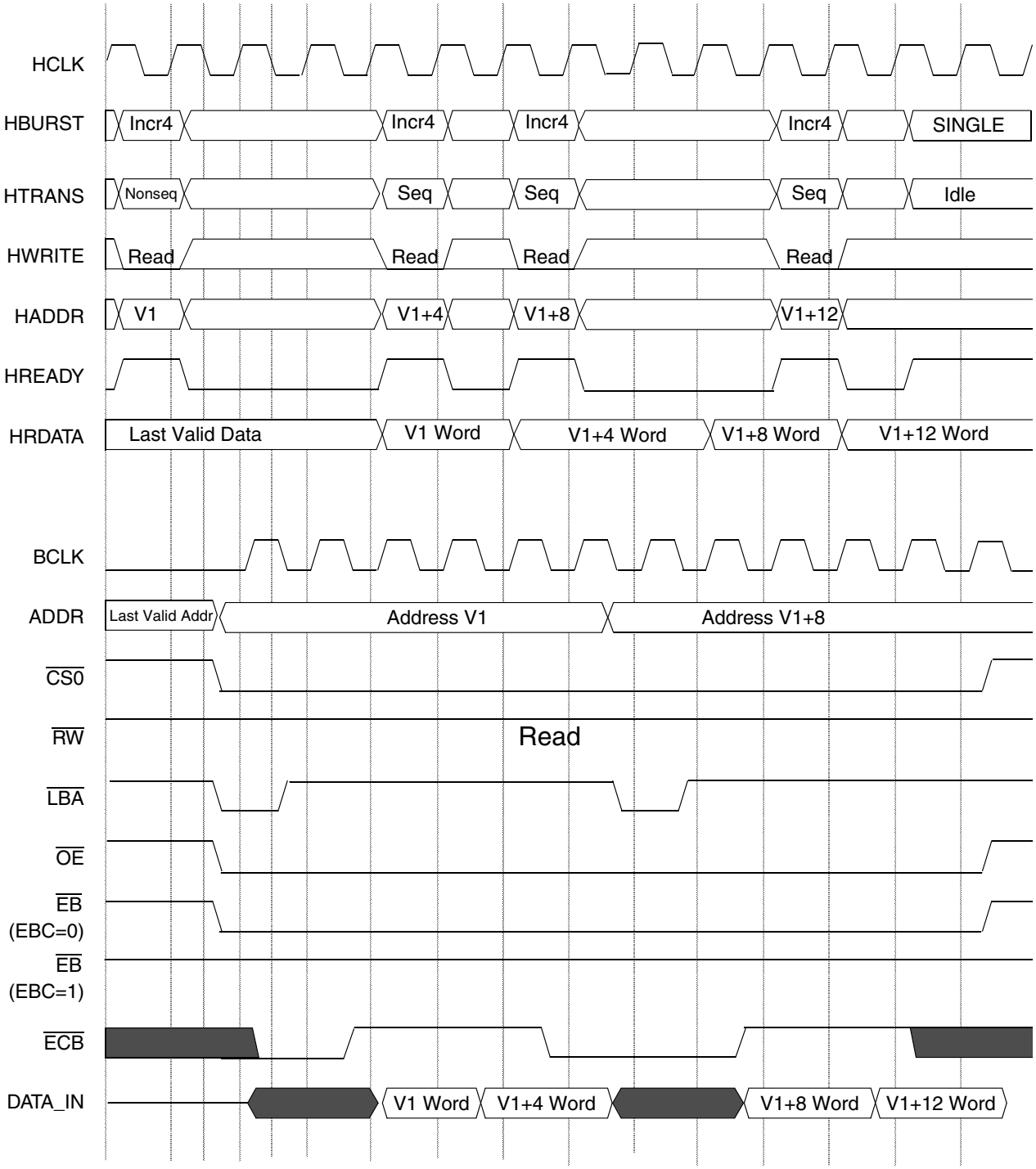


Figure 49-34. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=0

In the accesses on Figure 49-34 (V1+8) is a memory boundary address and may be a four word boundary address.

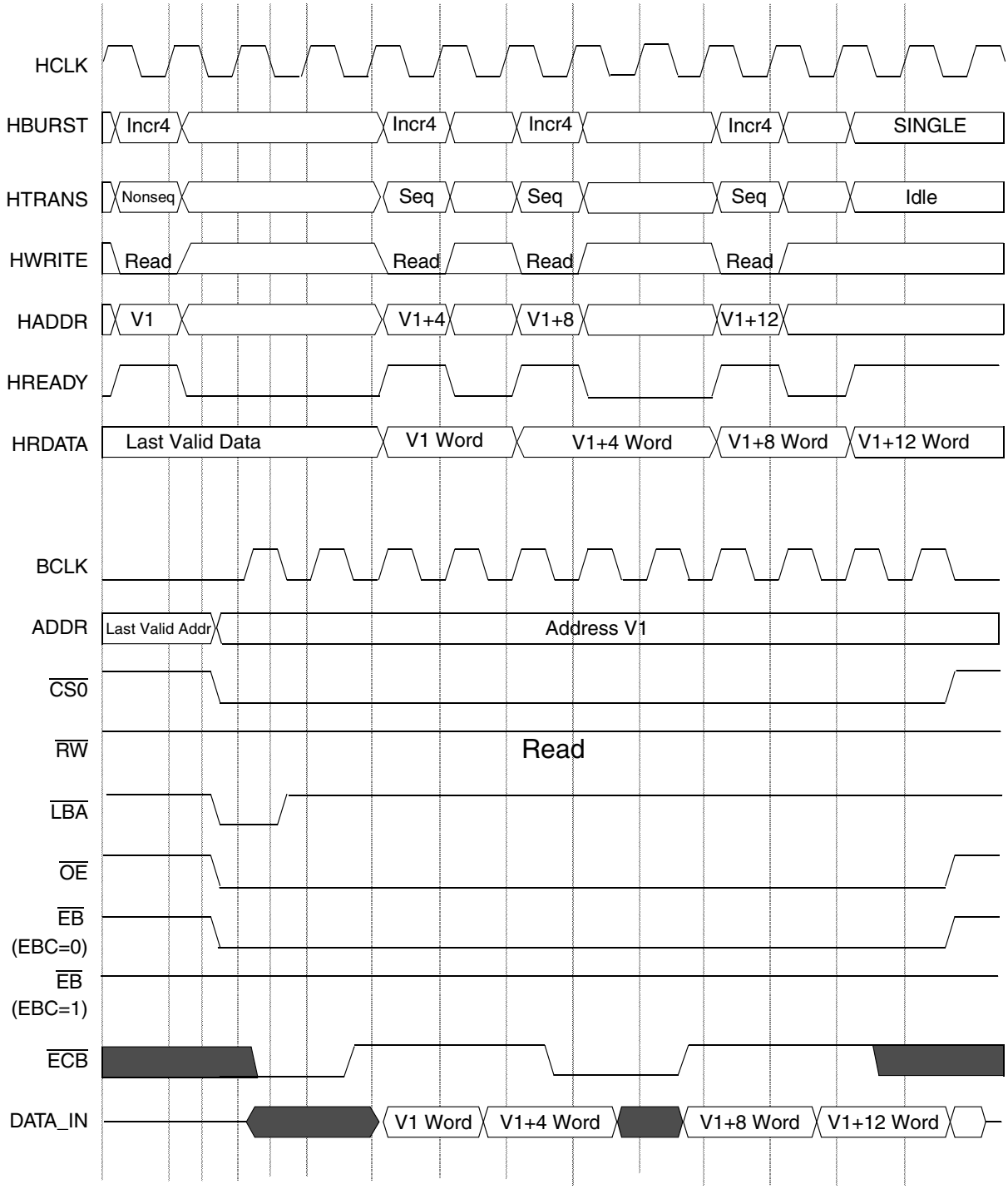


Figure 49-35. Increment 4 AHB Read Access, WSC=3, SYNC=1, DOL=1, WRAP=0, EW=1

In the accesses on Figure 49-35 and Figure 49-36(V1+8) is a memory boundary address and may be a four word boundary address.

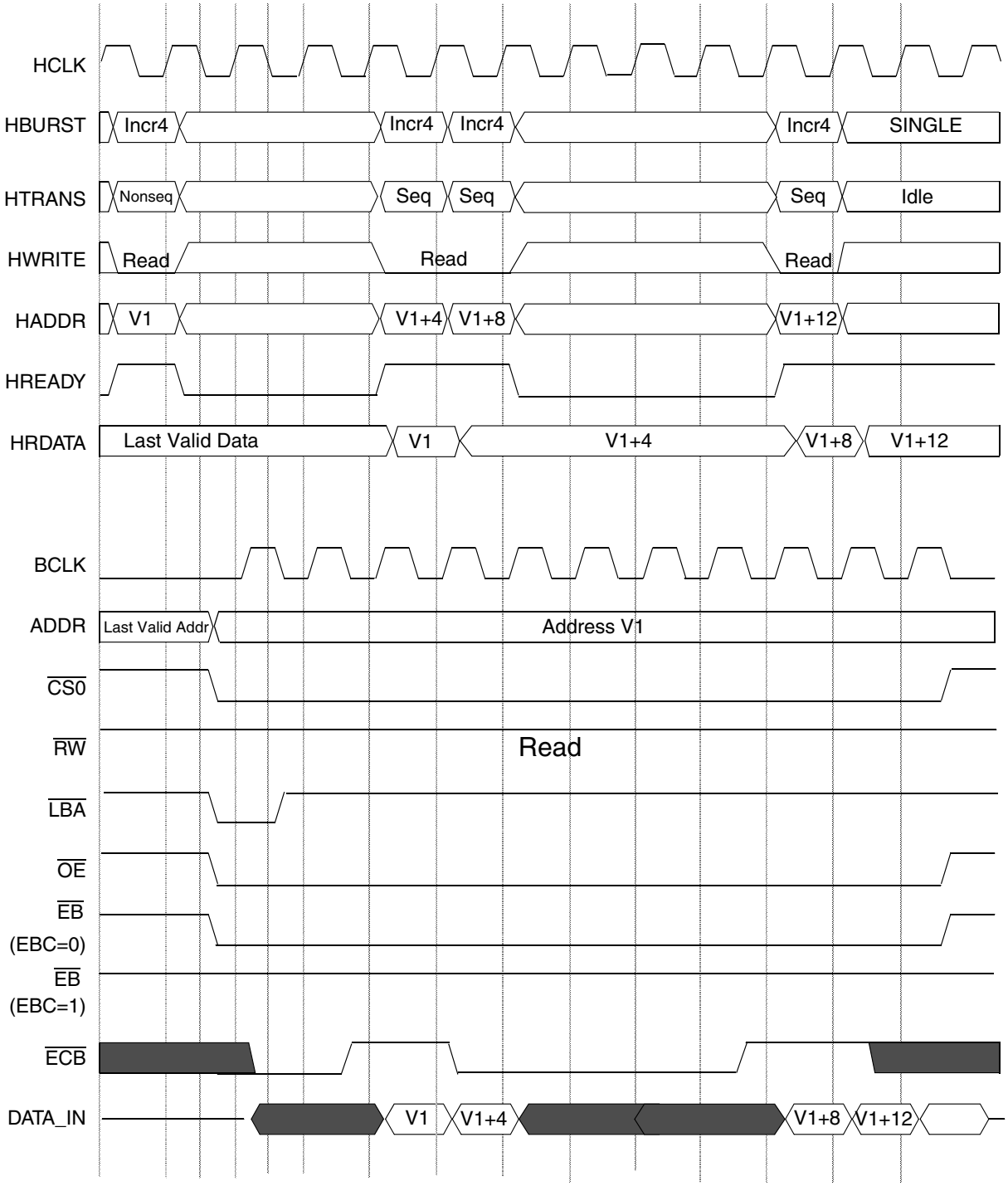


Figure 49-36. Increment 4 AHB Read Access, WSC=3, SYNC=1, WRAP=0, EW=1

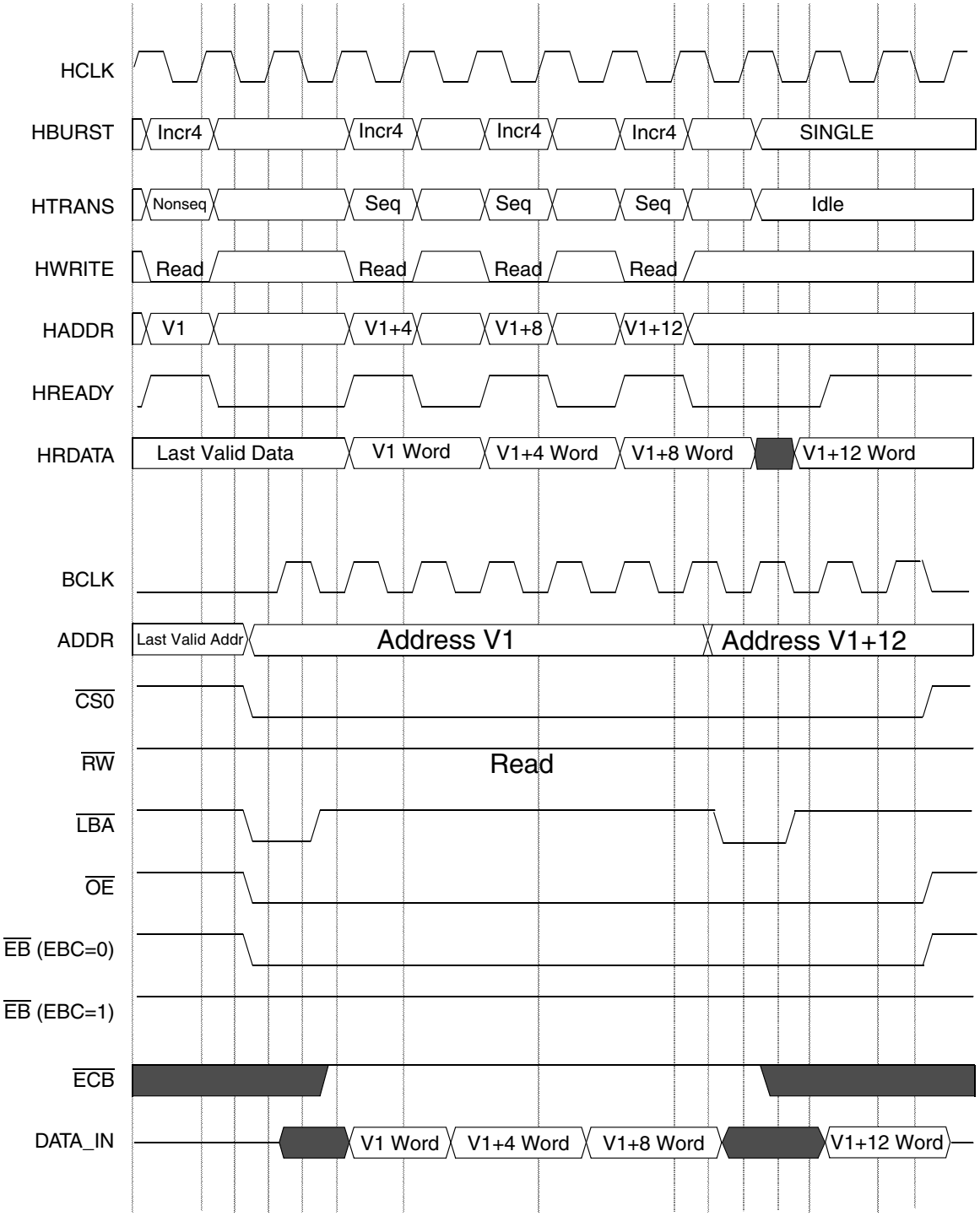


Figure 49-37. Increment 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In the accesses on Figure 49-37 (V1+12) is a four-word boundary address.

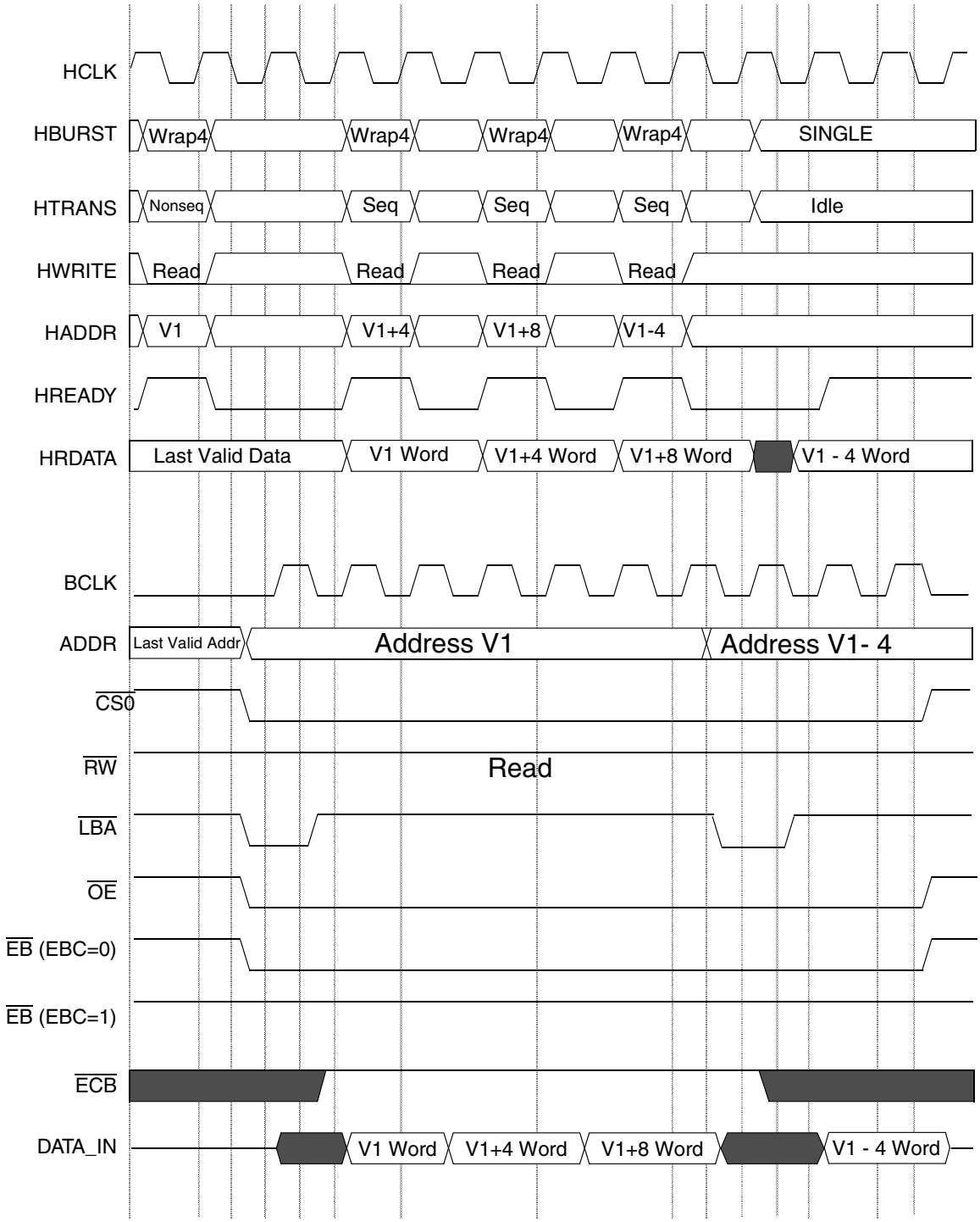


Figure 49-38. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=0

In the accesses on Figure 49-38 (V1-4) is a four-word boundary address.

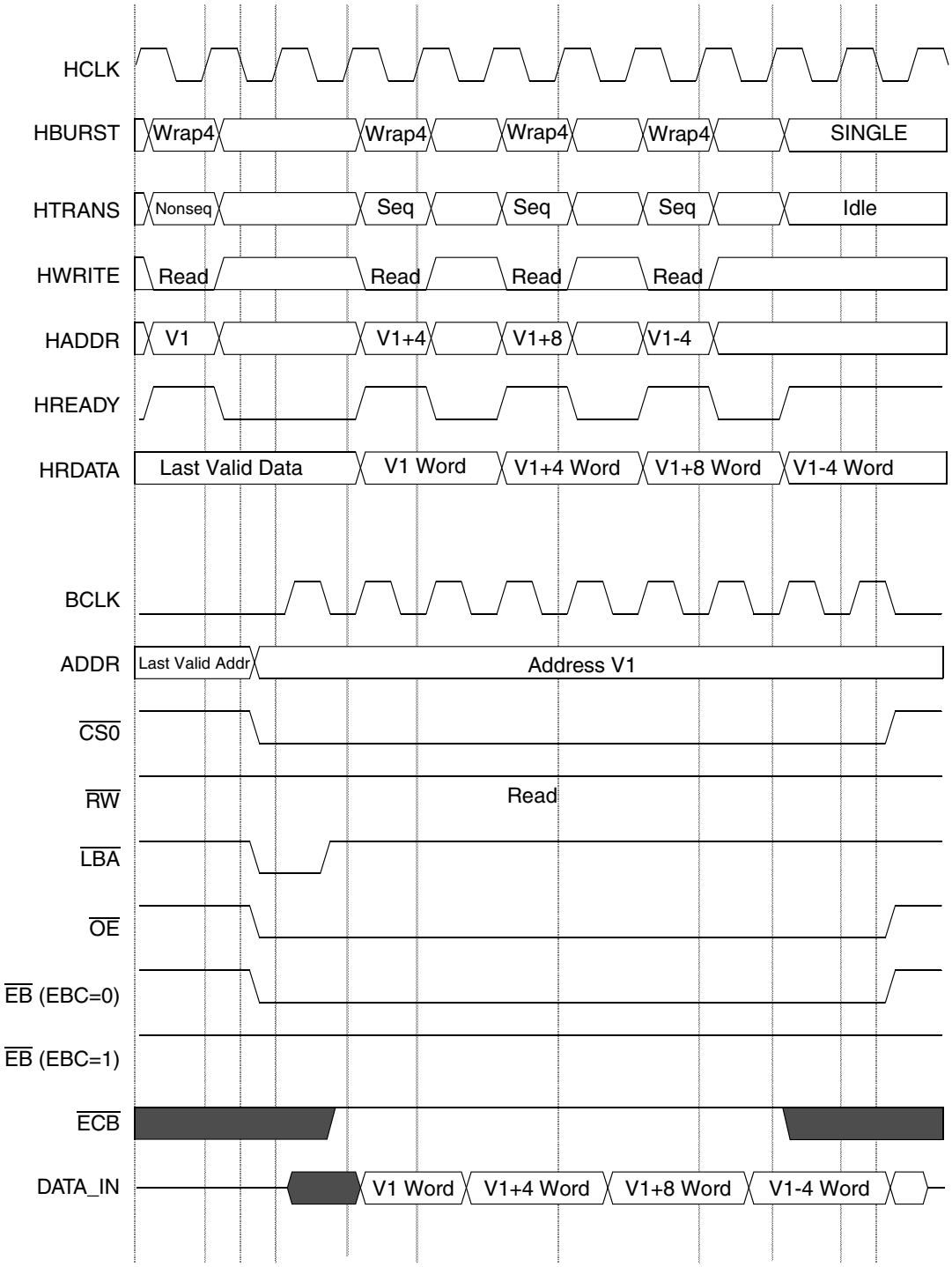


Figure 49-39. Wrap 4 AHB Read Access, WSC=2, SYNC=1, DOL=1, WRAP=1, PSZ=0

In the accesses on Figure 49-39 (V1-4) is a four-word boundary address.

49.6.5 Synchronous Accesses Timing Diagrams with PSRAM

49.6.5.1 AHB Sequential Accesses to Half-word Width PSRAM Memory

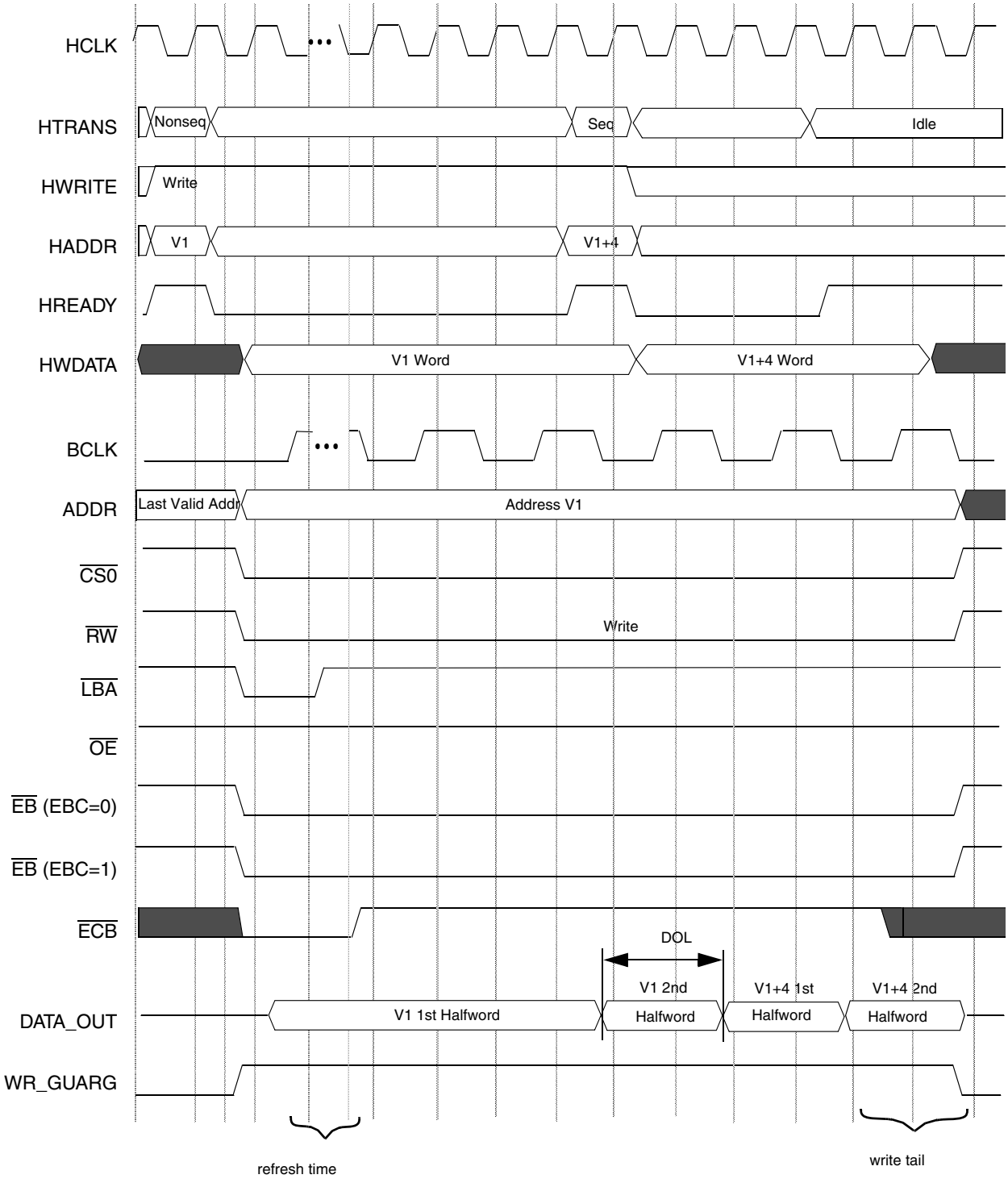


Figure 49-40. Write Access, BCD=1, BCS=1, WSC=5, SYNC=1, DOL=1, EW=1, PSR=1

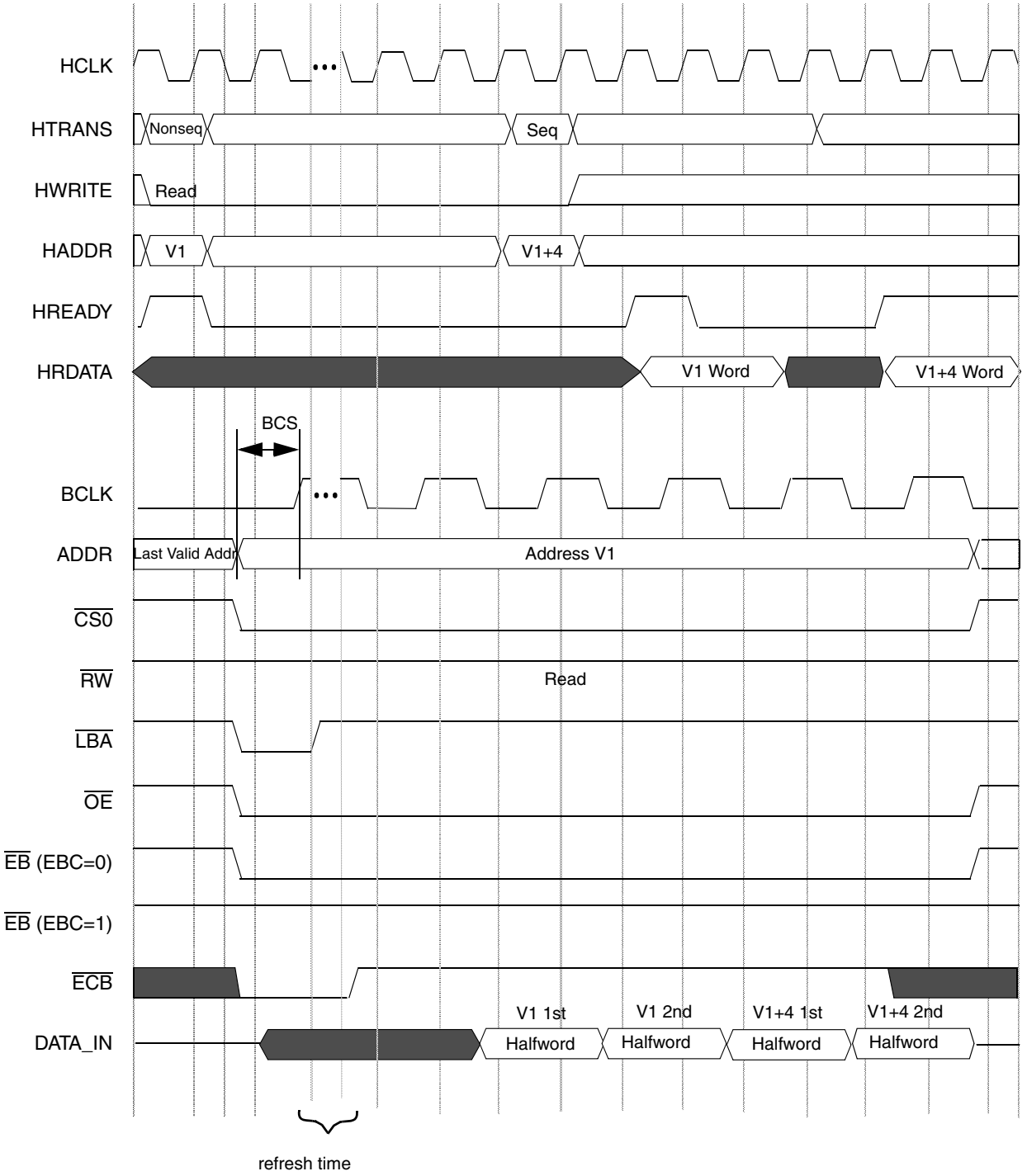


Figure 49-41. Read Access, BCD=1, BCS=1, WSC=5, SYNC=1, DOL=1, EW=1, PSR=1

49.6.5.2 AHB Sequential Accesses to Word-width PSRAM Memory

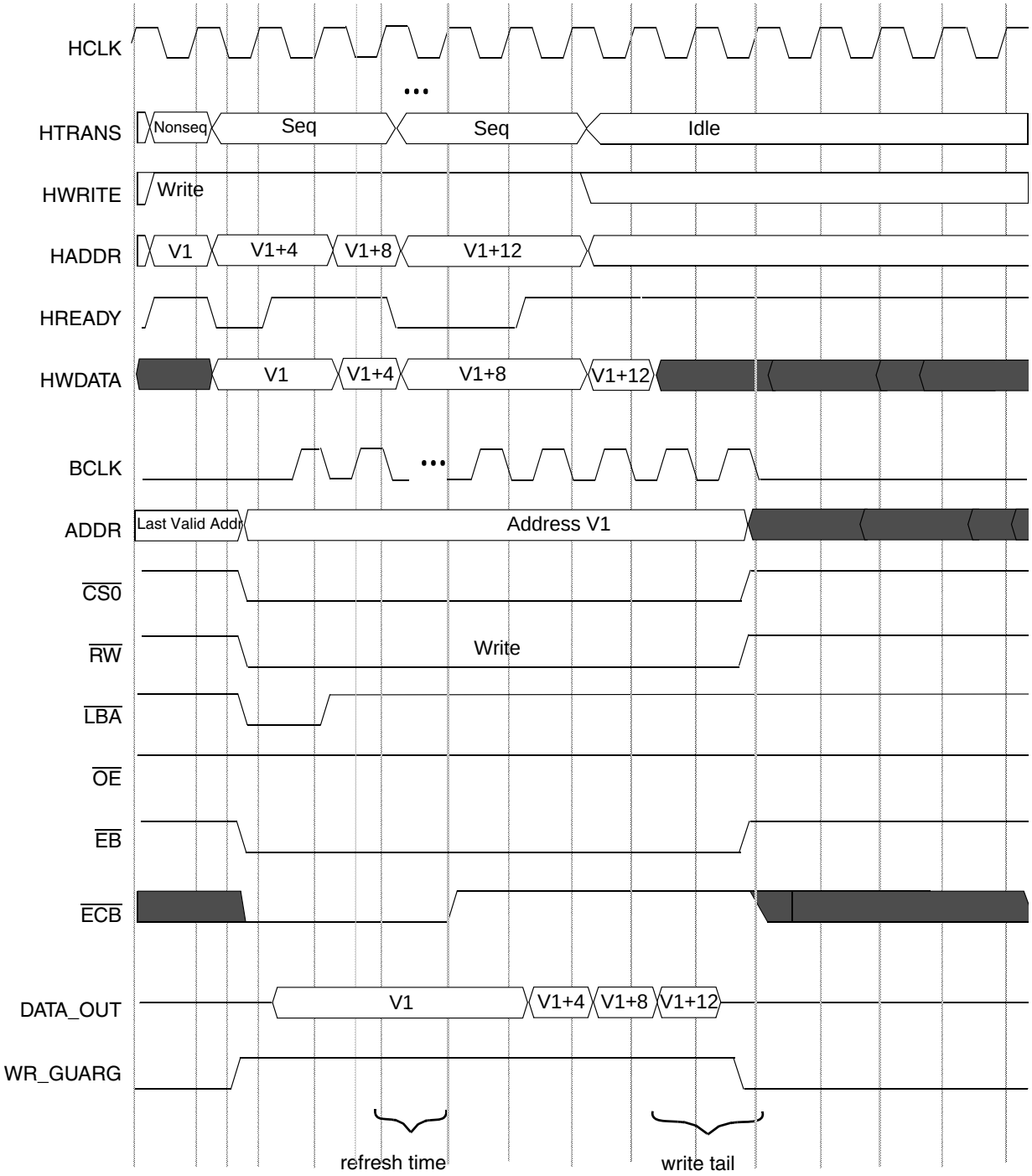


Figure 49-42. Write Access, BCS=1, WSC=4, SYNC=1, PSR=1

49.6.6 Multiplexed A/D Mode

49.6.6.1 Asynchronous Word Accesses to Word-width Memory

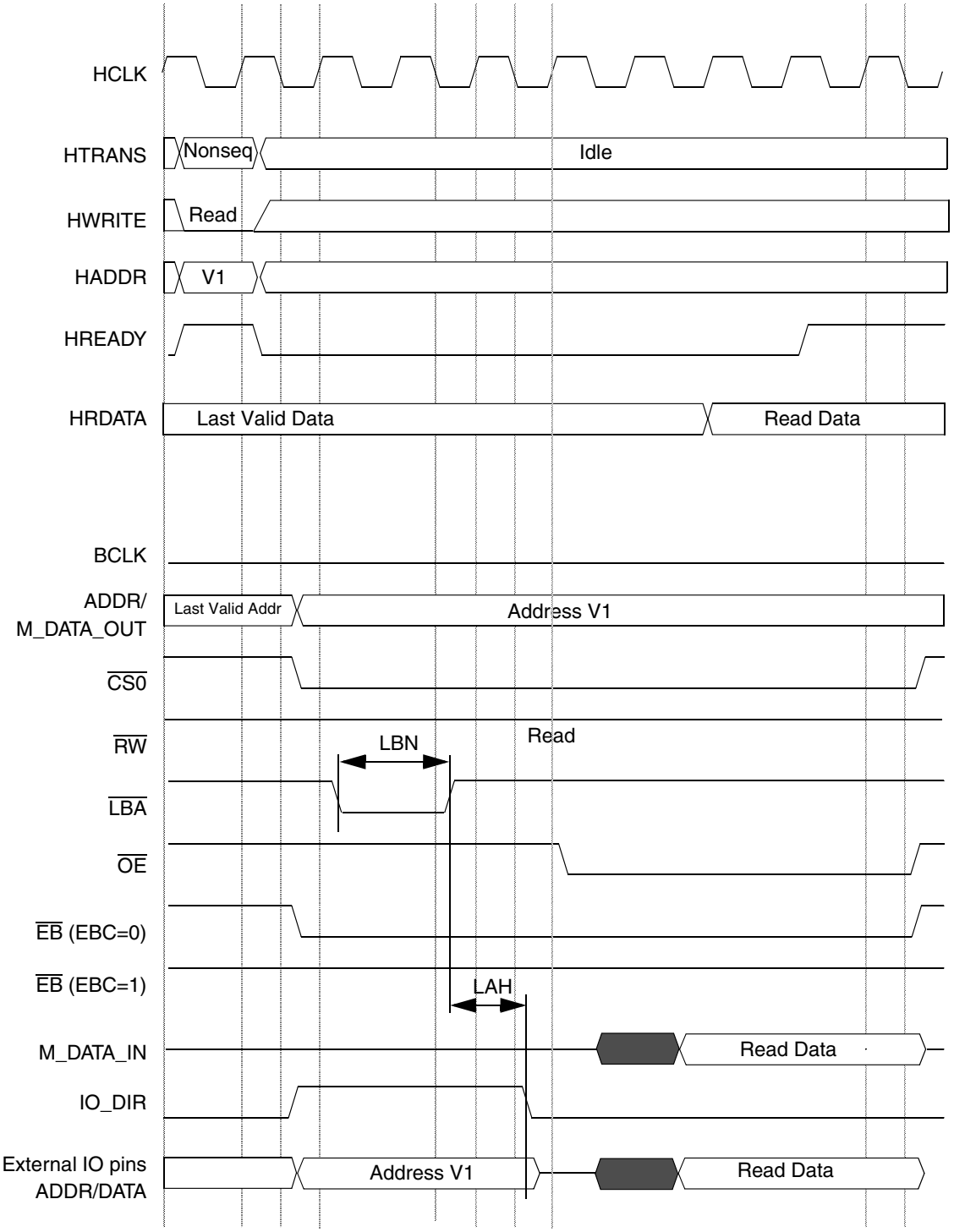


Figure 49-43. Read Access, WSC=7, LBA=1, LBN=1, LAH=1, OEA=7

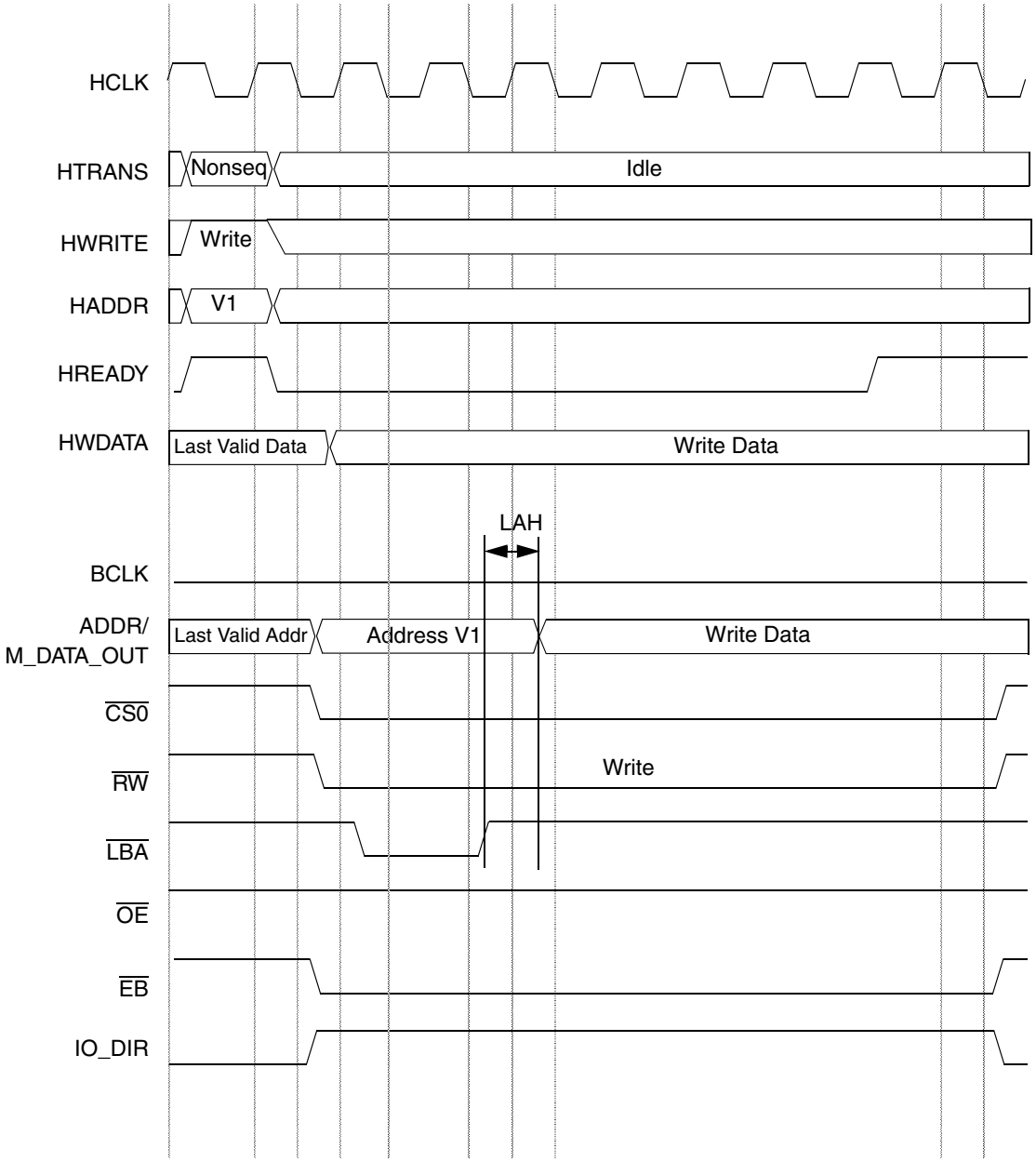


Figure 49-44. Write Access, WSC=7, LBA=1, LBN=1, LAH=1

49.6.6.2 Synchronous Accesses with Word-width Memory

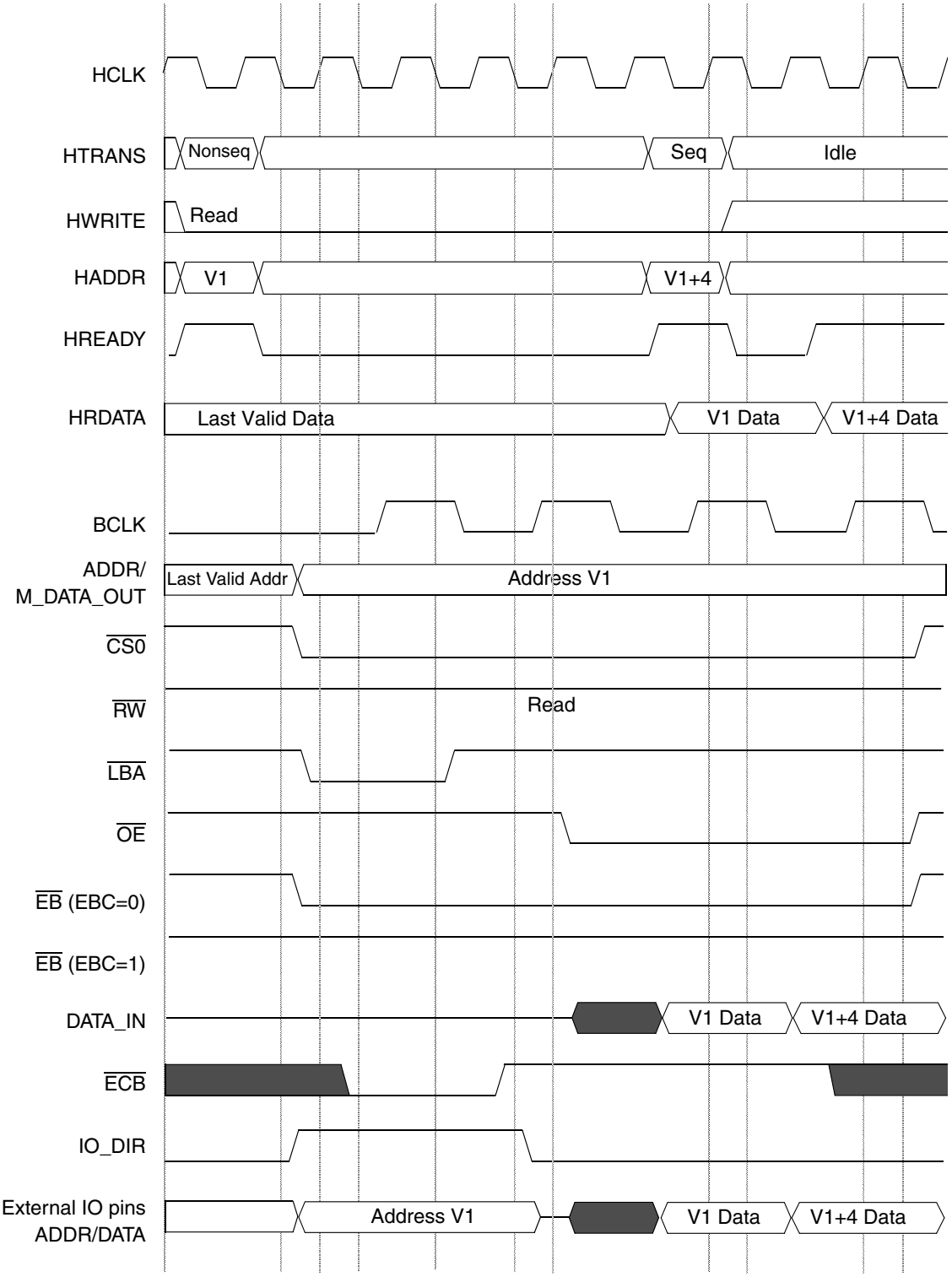


Figure 49-45. Read Access, BCD=1, SYNC=1, WCS=4, DOL=1, LBN=2, LAH=1, PSR=1

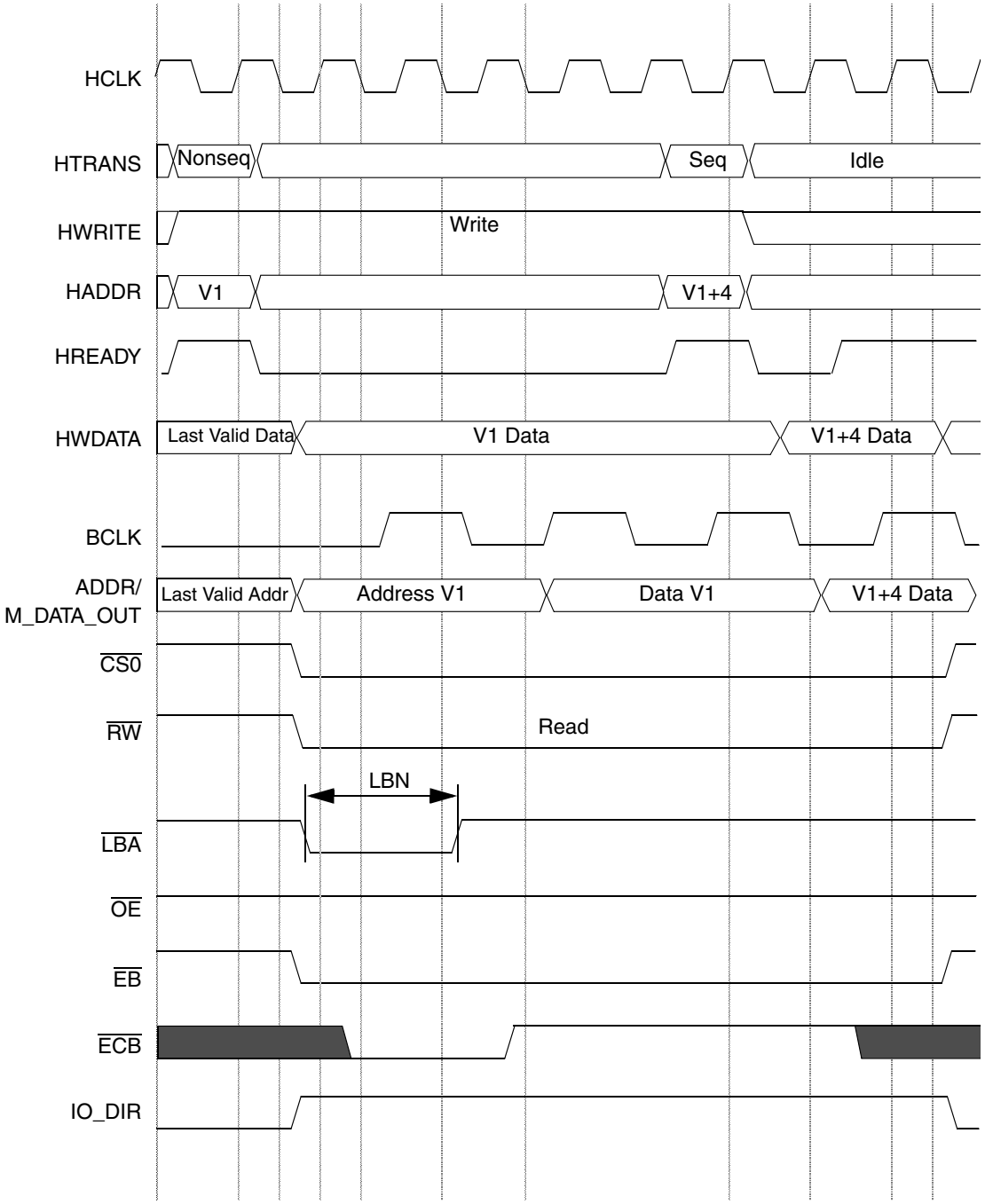


Figure 49-46. Write Access, BCD=1, SYNC=1, WCS=5, DOL=1, LBN=2, LAH=1, PSR=1

Appendix A
IOMUX Registers

Table 1: Register: IOMUXC_GPR1

Offset 0x0000 (IOMUXC_GPR1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0		SDCTL_CSD1_SEL_B	SDCTL_CSD0_SEL_B
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: Register IOMUXC_GPR1 Bits Description

Field	Description
31-2	Reserved. Not used in SoC integration
1 SDCTL_CSD1_SEL_B	EMI Chip Select Control Bit for SDCTL over WEIM. Select multiplexed chip select (CS3) between SDRAM/DDR and WEIM in EMI. 0 SDCTL chip select 1 WEIM chip select
0 SDCTL_CSD0_SEL_B	EMI Chip Select Control Bit for SDCTL over WEIM. Select multiplexed chip select (CS2) between SDRAM/DDR and WEIM in EMI. 0 SDCTL chip select 1 WEIM chip select

Table 3: Register: IOMUXC_OBSERVE_INT_MUX

Offset 0x0004 (IOMUXC_OBSERVE_INT_MUX) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	OBSRV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4: Register IOMUXC_OBSERVE_INT_MUX Bits Description

Field	Description
31-7	Reserved

Table 4: Register IOMUXC_OBSERVE_INT_MUX Bits Description

Field	Description
6-0 OBSRV	Select Instance Pin for Observability IOMUXC_OBSERVE_INT_MUX 0000000: Select Instance cspi3, Pin ipi_int_cspi_b 0000001: Select Instance gpt4_ipi_int_gpt_inv, Pin x 0000010: Select Instance owire_x_ipi_owire_int_inv, Pin x 0000011: Select Instance i2c1, Pin ipi_int_b 0000100: Select Instance i2c2, Pin ipi_int_b 0000101: Select Instance uart4, Pin ipi_uart_anded_b 0000110: Select Instance rtic_x_ipi_rtic_done_int_inv, Pin x 0000111: Select Instance esai, Pin ipi_esai_b 0001000: Select Instance esdhc1, Pin ipi_esdhcv2_irq_b 0001001: Select Instance esdhc2, Pin ipi_esdhcv2_irq_b 0001010: Select Instance i2c3, Pin ipi_int_b 0001011: Select Instance ssi2, Pin ipi_int_b 0001100: Select Instance ssi1, Pin ipi_int_b 0001101: Select Instance cspi2, Pin ipi_int_cspi_b 0001110: Select Instance cspi1, Pin ipi_int_cspi_b 0001111: Select Instance ata_x_ipbus_int_inv, Pin x 0010000: Select Instance gpio3_x_ipi_gpio_int_inv, Pin x 0010001: Select Instance csi, Pin ipi_csi_int_b 0010010: Select Instance uart3, Pin ipi_uart_anded_b 0010011: Select Instance iim, Pin iim_irq_b 0010100: Select Instance sim1_nor2_int_b, Pin x 0010101: Select Instance sim2_nor2_int_b, Pin x 0010110: Select Instance rngb_ipi_rngb_int_inv, Pin x 0010111: Select Instance gpio4_x_ipi_gpio_int_inv, Pin x 0011000: Select Instance kpp, Pin ipi_int_kpp_b 0011001: Select Instance dryice2ips, Pin ipi_dryice_normal_b 0011010: Select Instance pwm1_ipi_int_pwm_inv, Pin x 0011011: Select Instance epit2_x_ipi_int_epit_oc_inv, Pin x 0011100: Select Instance epit1_x_ipi_int_epit_oc_inv, Pin x 0011101: Select Instance gpt3_ipi_int_gpt_inv, Pin x 0011110: Select Instance ioring_ipp_ind_powerfail_inv, Pin x 0011111: Select Instance crm, Pin ipi_crm_irq_b 0100000: Select Instance uart2, Pin ipi_uart_anded_b 0100001: Select Instance emi, Pin ipi_int_nfc_b 0100010: Select Instance sdma, Pin ipi_host_intr_b 0100011: Select Instance usb_top_x_ipi_int_uh2_inv, Pin x 0100100: Select Instance pwm2_ipi_int_pwm_inv, Pin x 0100101: Select Instance usb_top_x_ipi_int_uotg_inv, Pin x 0100110: Select Instance slcdc, Pin slcdc_irq_b 0100111: Select Instance lcdc, Pin ipi_irq_b 0101000: Select Instance uart5, Pin ipi_uart_anded_b 0101001: Select Instance pwm3_ipi_int_pwm_inv, Pin x 0101010: Select Instance pwm4_ipi_int_pwm_inv, Pin x 0101011: Select Instance can1_nor4_int_b, Pin x 0101100: Select Instance can2_nor4_int_b, Pin x 0101101: Select Instance uart1, Pin ipi_uart_anded_b 0101110: Select Instance tsc, Pin ipi_tsc_int_b 0101111: Select Instance GND, Pin GND 0110000: Select Instance ect, Pin cti_trig_out_0[7] 0110001: Select Instance scc_x_ipi_int_scm_inv, Pin x 0110010: Select Instance scc_x_ipi_int_smn_inv, Pin x 0110011: Select Instance gpio2_x_ipi_gpio_int_inv, Pin x 0110100: Select Instance gpio1_x_ipi_gpio_int_inv, Pin x 0110101: Select Instance gpt2_ipi_int_gpt_inv, Pin x 0110110: Select Instance gpt1_ipi_int_gpt_inv, Pin x 0110111: Select Instance wdog, Pin ipi_wdog_int_b 0111000: Select Instance dryice2ips, Pin ipi_dryice_security_b 0111001: Select Instance fec_x_fec_ipi_int_inv, Pin x 0111010: Select Instance gpio1_ipi_gpio_int32_5_inv, Pin x 0111011: Select Instance gpio1_ipi_gpio_int32_4_inv, Pin x 0111100: Select Instance gpio1_ipi_gpio_int32_3_inv, Pin x 0111101: Select Instance gpio1_ipi_gpio_int32_2_inv, Pin x 0111110: Select Instance gpio1_ipi_gpio_int32_1_inv, Pin x 0111111: Select Instance gpio1_ipi_gpio_int32_0_inv, Pin x 1000000: Select Instance sdma, Pin debug_yield 1000001: Select Instance sdma, Pin debug_core_run 1000010: Select Instance sdma, Pin debug_mode 1000011: Select Instance sdma, Pin debug_bus_error 1000100: Select Instance sdma, Pin debug_bus_device[0] 1000101: Select Instance sdma, Pin debug_bus_device[1] 1000110: Select Instance sdma, Pin debug_bus_device[2] 1000111: Select Instance sdma, Pin debug_bus_device[3] 1001000: Select Instance sdma, Pin debug_bus_device[4] 1001001: Select Instance sdma, Pin debug_bus_rwb 1001010: Select Instance sdma, Pin debug_matched_dmbus 1001011: Select Instance sdma, Pin debug_rtbuffer_write 1001100: Select Instance sdma, Pin debug_evt_chn_lines[0] 1001101: Select Instance sdma, Pin debug_evt_chn_lines[1] 1001110: Select Instance sdma, Pin debug_evt_chn_lines[2] 1001111: Select Instance sdma, Pin debug_evt_chn_lines[3] 1010000: Select Instance sdma, Pin debug_evt_chn_lines[4] 1010001: Select Instance sdma, Pin debug_evt_chn_lines[5] 1010010: Select Instance sdma, Pin debug_evt_chn_lines[6] 1010011: Select Instance sdma, Pin debug_evt_chn_lines[7] 1010100: Select Instance arm926p_platform, Pin etm_portsize[0] 1010101: Select Instance arm926p_platform, Pin etm_portsize[1] 1010110: Select Instance arm926p_platform, Pin etm_portsize[2] 1010111: Select Instance fec, Pin miigsk_int_col 1011000: Select Instance fec, Pin miigsk_int_crs 1011001: Select Instance fec, Pin miigsk_int_rx_clk 1011010: Select Instance fec, Pin miigsk_int_rx_dv 1011011: Select Instance fec, Pin miigsk_int_tx_clk 1011100: Select Instance fec, Pin miigsk_mux_cnt_mii_mode 1011101: Select Instance fec, Pin miigsk_slow_en

Table 5: Register: IOMUXC_SW_MUX_CTL_PAD_A10

Offset 0x0008 (IOMUXC_SW_MUX_CTL_PAD_A10) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6: Register IOMUXC_SW_MUX_CTL_PAD_A10 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: A10. 000: Select mux mode: ALT0 mux port: EIM_DA_H[10] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[0] of instance: gpio4.

Table 7: Register: IOMUXC_SW_MUX_CTL_PAD_A13

Offset 0x000c (IOMUXC_SW_MUX_CTL_PAD_A13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8: Register IOMUXC_SW_MUX_CTL_PAD_A13 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: A13. 000: Select mux mode: ALT0 mux port: EIM_DA_H[13] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[1] of instance: gpio4. 111: Select mux mode: ALT7 mux port: LCDC_CLS of instance: lcdc.

Table 9: Register: IOMUXC_SW_MUX_CTL_PAD_A14

Offset 0x0010 (IOMUXC_SW_MUX_CTL_PAD_A14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10: Register IOMUXC_SW_MUX_CTL_PAD_A14 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A14. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A14. 000: Select mux mode: ALT0 mux port: EIM_DA_H2[14] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[0] of instance: gpio2. 110: Select mux mode: ALT6 mux port: CLK1 of instance: sim1. 111: Select mux mode: ALT7 mux port: LCDC_SPL of instance: lcdc.

Table 11: Register: IOMUXC_SW_MUX_CTL_PAD_A15

Offset 0x0014 (IOMUXC_SW_MUX_CTL_PAD_A15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12: Register IOMUXC_SW_MUX_CTL_PAD_A15 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A15. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A15. 000: Select mux mode: ALT0 mux port: EIM_DA_H2[15] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[1] of instance: gpio2. 110: Select mux mode: ALT6 mux port: RST1 of instance: sim1. 111: Select mux mode: ALT7 mux port: LCDC_PS of instance: lcdc.

Table 13: Register: IOMUXC_SW_MUX_CTL_PAD_A16

Offset 0x0018 (IOMUXC_SW_MUX_CTL_PAD_A16) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14: Register IOMUXC_SW_MUX_CTL_PAD_A16 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A16. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A16. 000: Select mux mode: ALT0 mux port: EIM_A[16] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[2] of instance: gpio2. 110: Select mux mode: ALT6 mux port: VEN1 of instance: sim1. 111: Select mux mode: ALT7 mux port: LCDC_REV of instance: lcdc.

Table 15: Register: IOMUXC_SW_MUX_CTL_PAD_A17

Offset 0x001c (IOMUXC_SW_MUX_CTL_PAD_A17) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16: Register IOMUXC_SW_MUX_CTL_PAD_A17 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A17. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 16: Register IOMUXC_SW_MUX_CTL_PAD_A17 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A17. 000: Select mux mode: ALT0 mux port: EIM_A[17] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[3] of instance: gpio2. 110: Select mux mode: ALT6 mux port: TX1 of instance: sim1. 111: Select mux mode: ALT7 mux port: TX_ERR of instance: fec. NOTE: Pad A17 is involved in Daisy Chain. - Config Register IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT for mode ALT6.

Table 17: Register: IOMUXC_SW_MUX_CTL_PAD_A18

Offset 0x0020 (IOMUXC_SW_MUX_CTL_PAD_A18) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18: Register IOMUXC_SW_MUX_CTL_PAD_A18 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A18. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 18: Register IOMUXC_SW_MUX_CTL_PAD_A18 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A18. 000: Select mux mode: ALT0 mux port: EIM_A[18] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[4] of instance: gpio2. 110: Select mux mode: ALT6 mux port: PD1 of instance: sim1. 111: Select mux mode: ALT7 mux port: COL of instance: fec. NOTE: Pad A18 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_COL_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT for mode ALT6.

Table 19: Register: IOMUXC_SW_MUX_CTL_PAD_A19

Offset 0x0024 (IOMUXC_SW_MUX_CTL_PAD_A19) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20: Register IOMUXC_SW_MUX_CTL_PAD_A19 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A19. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 20: Register IOMUXC_SW_MUX_CTL_PAD_A19 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A19. 000: Select mux mode: ALT0 mux port: EIM_A[19] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[5] of instance: gpio2. 110: Select mux mode: ALT6 mux port: RX1 of instance: sim1. 111: Select mux mode: ALT7 mux port: RX_ERR of instance: fec. NOTE: Pad A19 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT for mode ALT6.

Table 21: Register: IOMUXC_SW_MUX_CTL_PAD_A20

Offset 0x0028 (IOMUXC_SW_MUX_CTL_PAD_A20) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22: Register IOMUXC_SW_MUX_CTL_PAD_A20 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A20. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 22: Register IOMUXC_SW_MUX_CTL_PAD_A20 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A20. 000: Select mux mode: ALT0 mux port: EIM_A[20] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[6] of instance: gpio2. 110: Select mux mode: ALT6 mux port: CLK1 of instance: sim2. 111: Select mux mode: ALT7 mux port: RDATA[2] of instance: fec. NOTE: Pad A20 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT for mode ALT7.

Table 23: Register: IOMUXC_SW_MUX_CTL_PAD_A21

Offset 0x002c (IOMUXC_SW_MUX_CTL_PAD_A21) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24: Register IOMUXC_SW_MUX_CTL_PAD_A21 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A21. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 24: Register IOMUXC_SW_MUX_CTL_PAD_A21 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A21. 000: Select mux mode: ALT0 mux port: EIM_A[21] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[7] of instance: gpio2. 110: Select mux mode: ALT6 mux port: RST1 of instance: sim2. 111: Select mux mode: ALT7 mux port: RDATA[3] of instance: fec. NOTE: Pad A21 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT for mode ALT7.

Table 25: Register: IOMUXC_SW_MUX_CTL_PAD_A22

Offset 0x0030 (IOMUXC_SW_MUX_CTL_PAD_A22) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26: Register IOMUXC_SW_MUX_CTL_PAD_A22 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A22. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A22. 000: Select mux mode: ALT0 mux port: EIM_A[22] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[8] of instance: gpio2. 110: Select mux mode: ALT6 mux port: VEN1 of instance: sim2. 111: Select mux mode: ALT7 mux port: TDATA[2] of instance: fec.

Table 27: Register: IOMUXC_SW_MUX_CTL_PAD_A23

Offset 0x0034 (IOMUXC_SW_MUX_CTL_PAD_A23) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28: Register IOMUXC_SW_MUX_CTL_PAD_A23 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A23. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 28: Register IOMUXC_SW_MUX_CTL_PAD_A23 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A23. 000: Select mux mode: ALT0 mux port: EIM_A[23] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[9] of instance: gpio2. 110: Select mux mode: ALT6 mux port: TX1 of instance: sim2. 111: Select mux mode: ALT7 mux port: TDATA[3] of instance: fec. NOTE: Pad A23 is involved in Daisy Chain. - Config Register IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT for mode ALT6.

Table 29: Register: IOMUXC_SW_MUX_CTL_PAD_A24

Offset 0x0038 (IOMUXC_SW_MUX_CTL_PAD_A24) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30: Register IOMUXC_SW_MUX_CTL_PAD_A24 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A24. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 30: Register IOMUXC_SW_MUX_CTL_PAD_A24 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A24. 000: Select mux mode: ALT0 mux port: EIM_A[24] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[10] of instance: gpio2. 110: Select mux mode: ALT6 mux port: PD1 of instance: sim2. 111: Select mux mode: ALT7 mux port: RX_CLK of instance: fec. NOTE: Pad A24 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT for mode ALT6.

Table 31: Register: IOMUXC_SW_MUX_CTL_PAD_A25

Offset 0x003c (IOMUXC_SW_MUX_CTL_PAD_A25) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32: Register IOMUXC_SW_MUX_CTL_PAD_A25 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad A25. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 32: Register IOMUXC_SW_MUX_CTL_PAD_A25 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: A25. 000: Select mux mode: ALT0 mux port: EIM_A[25] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[11] of instance: gpio2. 110: Select mux mode: ALT6 mux port: RX1 of instance: sim2. 111: Select mux mode: ALT7 mux port: CRS of instance: fec. NOTE: Pad A25 is involved in Daisy Chain. - Config Register IOMUXC_FEC_FEC_CRS_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT for mode ALT6.

Table 33: Register: IOMUXC_SW_MUX_CTL_PAD_EB0

Offset 0x0040 (IOMUXC_SW_MUX_CTL_PAD_EB0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 34: Register IOMUXC_SW_MUX_CTL_PAD_EB0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad EB0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 34: Register IOMUXC_SW_MUX_CTL_PAD_EB0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: EB0. 100: Select mux mode: ALT4 mux port: AUD4_TXD of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[12] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SS0 of instance: cspi3. 000: Select mux mode: ALT0 mux port: EIM_EB0_B of instance: emi. NOTE: Pad EB0 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT for mode ALT6.

Table 35: Register: IOMUXC_SW_MUX_CTL_PAD_EB1

Offset 0x0044 (IOMUXC_SW_MUX_CTL_PAD_EB1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 36: Register IOMUXC_SW_MUX_CTL_PAD_EB1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad EB1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 36: Register IOMUXC_SW_MUX_CTL_PAD_EB1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: EB1. 000: Select mux mode: ALT0 mux port: EIM_EB1_B of instance: emi. 100: Select mux mode: ALT4 mux port: AUD4_RXD of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[13] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SS1 of instance: cspi3. NOTE: Pad EB1 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT for mode ALT6.

Table 37: Register: IOMUXC_SW_MUX_CTL_PAD_OE

Offset 0x0048 (IOMUXC_SW_MUX_CTL_PAD_OE) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 38: Register IOMUXC_SW_MUX_CTL_PAD_OE Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad OE. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 38: Register IOMUXC_SW_MUX_CTL_PAD_OE Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: OE. 000: Select mux mode: ALT0 mux port: EIM_OE of instance: emi. 100: Select mux mode: ALT4 mux port: AUD4_TXC of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[14] of instance: gpio2. NOTE: Pad OE is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT for mode ALT4.

Table 39: Register: IOMUXC_SW_MUX_CTL_PAD_CS0

Offset 0x004c (IOMUXC_SW_MUX_CTL_PAD_CS0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 40: Register IOMUXC_SW_MUX_CTL_PAD_CS0 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CS0. 000: Select mux mode: ALT0 mux port: EIM_CS0 of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[2] of instance: gpio4.

Table 41: Register: IOMUXC_SW_MUX_CTL_PAD_CS1

Offset 0x0050 (IOMUXC_SW_MUX_CTL_PAD_CS1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 42: Register IOMUXC_SW_MUX_CTL_PAD_CS1 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: CS1. 000: Select mux mode: ALT0 mux port: EIM_CS1 of instance: emi. 001: Select mux mode: ALT1 mux port: NANDF_CE3 of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[3] of instance: gpio4.

Table 43: Register: IOMUXC_SW_MUX_CTL_PAD_CS4

Offset 0x0054 (IOMUXC_SW_MUX_CTL_PAD_CS4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44: Register IOMUXC_SW_MUX_CTL_PAD_CS4 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CS4. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 44: Register IOMUXC_SW_MUX_CTL_PAD_CS4 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: CS4. 000: Select mux mode: ALT0 mux port: EIM_CS4 of instance: emi. 001: Select mux mode: ALT1 mux port: NANDF_CE1 of instance: emi. 011: Select mux mode: ALT3 mux port: CTS of instance: uart5. 100: Select mux mode: ALT4 mux port: AUD4_RXC of instance: aud-mux. 101: Select mux mode: ALT5 mux port: GPIO[20] of instance: gpio3. 110: Select mux mode: ALT6 mux port: MOSI of instance: cspi3. 111: Select mux mode: ALT7 mux port: TRSYNC of instance: arm926p_platform. NOTE: Pad CS4 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT for mode ALT6.

Table 45: Register: IOMUXC_SW_MUX_CTL_PAD_CS5

Offset 0x0058 (IOMUXC_SW_MUX_CTL_PAD_CS5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 46: Register IOMUXC_SW_MUX_CTL_PAD_CS5 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CS5. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 46: Register IOMUXC_SW_MUX_CTL_PAD_CS5 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CS5. 000: Select mux mode: ALT0 mux port: EIM_CS5 of instance: emi. 001: Select mux mode: ALT1 mux port: NANDF_CE2 of instance: emi. 010: Select mux mode: ALT2 mux port: DTACK_B of instance: emi. 011: Select mux mode: ALT3 mux port: RTS of instance: uart5. 100: Select mux mode: ALT4 mux port: AUD4_RXFS of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[21] of instance: gpio3. 110: Select mux mode: ALT6 mux port: MISO of instance: cspi3. 111: Select mux mode: ALT7 mux port: TRCLK of instance: arm926p_platform. NOTE: Pad CS5 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT for mode ALT3.

Table 47: Register: IOMUXC_SW_MUX_CTL_PAD_NF_CE0

Offset 0x005c (IOMUXC_SW_MUX_CTL_PAD_NF_CE0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 48: Register IOMUXC_SW_MUX_CTL_PAD_NF_CE0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NF_CE0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 48: Register IOMUXC_SW_MUX_CTL_PAD_NF_CE0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: NF_CE0. 000: Select mux mode: ALT0 mux port: NANDF_CE0 of instance: emi. 001: Select mux mode: ALT1 mux port: SS3 of instance: cspi1. 101: Select mux mode: ALT5 mux port: GPIO[22] of instance: gpio3. 111: Select mux mode: ALT7 mux port: TRACE[3] of instance: arm926p_platform. NOTE: Pad NF_CE0 is involved in Daisy Chain. - Config Register IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT for mode ALT1.

Table 49: Register: IOMUXC_SW_MUX_CTL_PAD_ECB

Offset 0x0060 (IOMUXC_SW_MUX_CTL_PAD_ECB) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 50: Register IOMUXC_SW_MUX_CTL_PAD_ECB Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad ECB. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 50: Register IOMUXC_SW_MUX_CTL_PAD_ECB Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: ECB. 000: Select mux mode: ALT0 mux port: EIM_ECB of instance: emi. 011: Select mux mode: ALT3 mux port: TXD_MUX of instance: uart5. 101: Select mux mode: ALT5 mux port: GPIO[23] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SCLK of instance: cspi3. NOTE: Pad ECB is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT for mode ALT6.

Table 51: Register: IOMUXC_SW_MUX_CTL_PAD_LBA

Offset 0x0064 (IOMUXC_SW_MUX_CTL_PAD_LBA) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 52: Register IOMUXC_SW_MUX_CTL_PAD_LBA Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LBA. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 52: Register IOMUXC_SW_MUX_CTL_PAD_LBA Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: LBA. 000: Select mux mode: ALT0 mux port: EIM_LBA of instance: emi. 011: Select mux mode: ALT3 mux port: RXD_MUX of instance: uart5. 101: Select mux mode: ALT5 mux port: GPIO[24] of instance: gpio3. 110: Select mux mode: ALT6 mux port: RDY of instance: cspi3. NOTE: Pad LBA is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT3.

Table 53: Register: IOMUXC_SW_MUX_CTL_PAD_BCLK

Offset 0x0068 (IOMUXC_SW_MUX_CTL_PAD_BCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 54: Register IOMUXC_SW_MUX_CTL_PAD_BCLK Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: BCLK. 000: Select mux mode: ALT0 mux port: EIM_BCLK of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[4] of instance: gpio4.

Table 55: Register: IOMUXC_SW_MUX_CTL_PAD_RW

Offset 0x006c (IOMUXC_SW_MUX_CTL_PAD_RW) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 56: Register IOMUXC_SW_MUX_CTL_PAD_RW Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad RW. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 56: Register IOMUXC_SW_MUX_CTL_PAD_RW Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: RW. 000: Select mux mode: ALT0 mux port: EIM_RW of instance: emi. 100: Select mux mode: ALT4 mux port: AUD4_TXFS of instance: aud-mux. 101: Select mux mode: ALT5 mux port: GPIO[25] of instance: gpio3. NOTE: Pad RW is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT4.

Table 57: Register: IOMUXC_SW_MUX_CTL_PAD_NFWE_B

Offset 0x0070 (IOMUXC_SW_MUX_CTL_PAD_NFWE_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 58: Register IOMUXC_SW_MUX_CTL_PAD_NFWE_B Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFWE_B. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFWE_B. 000: Select mux mode: ALT0 mux port: NANDF_WE_B of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[26] of instance: gpio3. 111: Select mux mode: ALT7 mux port: PIPESTAT[2] of instance: arm926p_platform.

Table 59: Register: IOMUXC_SW_MUX_CTL_PAD_NFRE_B

Offset 0x0074 (IOMUXC_SW_MUX_CTL_PAD_NFRE_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 60: Register IOMUXC_SW_MUX_CTL_PAD_NFRE_B Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFRE_B. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFRE_B. 000: Select mux mode: ALT0 mux port: NANDF_RE_B of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[27] of instance: gpio3. 111: Select mux mode: ALT7 mux port: PIPESTAT[1] of instance: arm926p_platform.

Table 61: Register: IOMUXC_SW_MUX_CTL_PAD_NFALE

Offset 0x0078 (IOMUXC_SW_MUX_CTL_PAD_NFALE) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 62: Register IOMUXC_SW_MUX_CTL_PAD_NFALE Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFALE. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFALE. 000: Select mux mode: ALT0 mux port: NANDF_ALE of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[28] of instance: gpio3. 111: Select mux mode: ALT7 mux port: PIPESTAT[0] of instance: arm926p_platform.

Table 63: Register: IOMUXC_SW_MUX_CTL_PAD_NFCLE

Offset 0x007c (IOMUXC_SW_MUX_CTL_PAD_NFCLE) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 64: Register IOMUXC_SW_MUX_CTL_PAD_NFCLE Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFCLE. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFCLE. 000: Select mux mode: ALT0 mux port: NANDF_CLE of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[29] of instance: gpio3. 111: Select mux mode: ALT7 mux port: TRACE[0] of instance: arm926p_platform.

Table 65: Register: IOMUXC_SW_MUX_CTL_PAD_NFWP_B

Offset 0x0080 (IOMUXC_SW_MUX_CTL_PAD_NFWP_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 66: Register IOMUXC_SW_MUX_CTL_PAD_NFWP_B Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFWP_B. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFWP_B. 000: Select mux mode: ALT0 mux port: NANDF_WP_B of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[30] of instance: gpio3. 111: Select mux mode: ALT7 mux port: TRACE[1] of instance: arm926p_platform.

Table 67: Register: IOMUXC_SW_MUX_CTL_PAD_NFRB

Offset 0x0084 (IOMUXC_SW_MUX_CTL_PAD_NFRB) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 68: Register IOMUXC_SW_MUX_CTL_PAD_NFRB Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad NFRB. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: NFRB. 000: Select mux mode: ALT0 mux port: NANDF_RB of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[31] of instance: gpio3. 111: Select mux mode: ALT7 mux port: TRACE[2] of instance: arm926p_platform.

Table 69: Register: IOMUXC_SW_MUX_CTL_PAD_D15

Offset 0x0088 (IOMUXC_SW_MUX_CTL_PAD_D15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 70: Register IOMUXC_SW_MUX_CTL_PAD_D15 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D15. 000: Select mux mode: ALT0 mux port: EIM_D[15] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[16] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[5] of instance: gpio4. 110: Select mux mode: ALT6 mux port: DAT7 of instance: esdhc1. NOTE: Pad D15 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT for mode ALT6.

Table 71: Register: IOMUXC_SW_MUX_CTL_PAD_D14

Offset 0x008c (IOMUXC_SW_MUX_CTL_PAD_D14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 72: Register IOMUXC_SW_MUX_CTL_PAD_D14 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D14. 000: Select mux mode: ALT0 mux port: EIM_D[14] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[17] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[6] of instance: gpio4. 110: Select mux mode: ALT6 mux port: DAT6 of instance: esdhc1. NOTE: Pad D14 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT for mode ALT6.

Table 73: Register: IOMUXC_SW_MUX_CTL_PAD_D13

Offset 0x0090 (IOMUXC_SW_MUX_CTL_PAD_D13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 74: Register IOMUXC_SW_MUX_CTL_PAD_D13 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D13. 000: Select mux mode: ALT0 mux port: EIM_D[13] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[18] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[7] of instance: gpio4. 110: Select mux mode: ALT6 mux port: DAT5 of instance: esdhc1. NOTE: Pad D13 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT for mode ALT6.

Table 75: Register: IOMUXC_SW_MUX_CTL_PAD_D12

Offset 0x0094 (IOMUXC_SW_MUX_CTL_PAD_D12) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 76: Register IOMUXC_SW_MUX_CTL_PAD_D12 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D12. 000: Select mux mode: ALT0 mux port: EIM_D[12] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[19] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[8] of instance: gpio4. 110: Select mux mode: ALT6 mux port: DAT4 of instance: esdhc1. NOTE: Pad D12 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT for mode ALT6.

Table 77: Register: IOMUXC_SW_MUX_CTL_PAD_D11

Offset 0x0098 (IOMUXC_SW_MUX_CTL_PAD_D11) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 78: Register IOMUXC_SW_MUX_CTL_PAD_D11 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D11. 000: Select mux mode: ALT0 mux port: EIM_D[11] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[20] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[9] of instance: gpio4. 110: Select mux mode: ALT6 mux port: USBOTG_PWR of instance: usb_top.

Table 79: Register: IOMUXC_SW_MUX_CTL_PAD_D10

Offset 0x009c (IOMUXC_SW_MUX_CTL_PAD_D10) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 80: Register IOMUXC_SW_MUX_CTL_PAD_D10 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D10. 000: Select mux mode: ALT0 mux port: EIM_D[10] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[21] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[10] of instance: gpio4. 110: Select mux mode: ALT6 mux port: USBOTG_OC of instance: usb_top. NOTE: Pad D10 is involved in Daisy Chain. - Config Register IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT for mode ALT6.

Table 81: Register: IOMUXC_SW_MUX_CTL_PAD_D9

Offset 0x00a0 (IOMUXC_SW_MUX_CTL_PAD_D9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 82: Register IOMUXC_SW_MUX_CTL_PAD_D9 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D9. 000: Select mux mode: ALT0 mux port: EIM_D[9] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[22] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[11] of instance: gpio4. 110: Select mux mode: ALT6 mux port: USBH2_PWR of instance: usb_top.

Table 83: Register: IOMUXC_SW_MUX_CTL_PAD_D8

Offset 0x00a4 (IOMUXC_SW_MUX_CTL_PAD_D8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 84: Register IOMUXC_SW_MUX_CTL_PAD_D8 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: D8. 000: Select mux mode: ALT0 mux port: EIM_D[8] of instance: emi. 001: Select mux mode: ALT1 mux port: LCDC_LD[23] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[12] of instance: gpio4. 110: Select mux mode: ALT6 mux port: USBH2_OC of instance: usb_top. NOTE: Pad D8 is involved in Daisy Chain. - Config Register IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT for mode ALT6.

Table 85: Register: IOMUXC_SW_MUX_CTL_PAD_D7

Offset 0x00a8 (IOMUXC_SW_MUX_CTL_PAD_D7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 86: Register IOMUXC_SW_MUX_CTL_PAD_D7 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D7. 000: Select mux mode: ALT0 mux port: EIM_D[7] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[13] of instance: gpio4.

Table 87: Register: IOMUXC_SW_MUX_CTL_PAD_D6

Offset 0x00ac (IOMUXC_SW_MUX_CTL_PAD_D6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 88: Register IOMUXC_SW_MUX_CTL_PAD_D6 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D6. 000: Select mux mode: ALT0 mux port: EIM_D[6] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[14] of instance: gpio4.

Table 89: Register: IOMUXC_SW_MUX_CTL_PAD_D5

Offset 0x00b0 (IOMUXC_SW_MUX_CTL_PAD_D5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 90: Register IOMUXC_SW_MUX_CTL_PAD_D5 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D5. 000: Select mux mode: ALT0 mux port: EIM_D[5] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[15] of instance: gpio4.

Table 91: Register: IOMUXC_SW_MUX_CTL_PAD_D4

Offset 0x00b4 (IOMUXC_SW_MUX_CTL_PAD_D4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 92: Register IOMUXC_SW_MUX_CTL_PAD_D4 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D4. 000: Select mux mode: ALT0 mux port: EIM_D[4] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[16] of instance: gpio4.

Table 93: Register: IOMUXC_SW_MUX_CTL_PAD_D3

Offset 0x00b8 (IOMUXC_SW_MUX_CTL_PAD_D3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 94: Register IOMUXC_SW_MUX_CTL_PAD_D3 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D3. 000: Select mux mode: ALT0 mux port: EIM_D[3] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[17] of instance: gpio4.

Table 95: Register: IOMUXC_SW_MUX_CTL_PAD_D2

Offset 0x00bc (IOMUXC_SW_MUX_CTL_PAD_D2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 96: Register IOMUXC_SW_MUX_CTL_PAD_D2 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D2. 000: Select mux mode: ALT0 mux port: EIM_D[2] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[18] of instance: gpio4.

Table 97: Register: IOMUXC_SW_MUX_CTL_PAD_D1

Offset 0x00c0 (IOMUXC_SW_MUX_CTL_PAD_D1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 98: Register IOMUXC_SW_MUX_CTL_PAD_D1 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D1. 000: Select mux mode: ALT0 mux port: EIM_D[1] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[19] of instance: gpio4.

Table 99: Register: IOMUXC_SW_MUX_CTL_PAD_D0

Offset 0x00c4 (IOMUXC_SW_MUX_CTL_PAD_D0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 100: Register IOMUXC_SW_MUX_CTL_PAD_D0 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: D0. 000: Select mux mode: ALT0 mux port: EIM_D[0] of instance: emi. 101: Select mux mode: ALT5 mux port: GPIO[20] of instance: gpio4.

Table 101: Register: IOMUXC_SW_MUX_CTL_PAD_LD0

Offset 0x00c8 (IOMUXC_SW_MUX_CTL_PAD_LD0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 102: Register IOMUXC_SW_MUX_CTL_PAD_LD0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 102: Register IOMUXC_SW_MUX_CTL_PAD_LD0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD0. 000: Select mux mode: ALT0 mux port: LCDC_LD[0] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[0] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[0] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[0] of instance: ata. 100: Select mux mode: ALT4 mux port: CLK1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[15] of instance: gpio2. 110: Select mux mode: ALT6 mux port: USBH2_CLK of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_MEM_CTRL[0] of instance: crm. NOTE: Pad LD0 is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT for mode ALT2.

Table 103: Register: IOMUXC_SW_MUX_CTL_PAD_LD1

Offset 0x00cc (IOMUXC_SW_MUX_CTL_PAD_LD1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 104: Register IOMUXC_SW_MUX_CTL_PAD_LD1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 104: Register IOMUXC_SW_MUX_CTL_PAD_LD1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD1. 000: Select mux mode: ALT0 mux port: LCDC_LD[1] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[1] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[1] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[1] of instance: ata. 100: Select mux mode: ALT4 mux port: RST1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[16] of instance: gpio2. 110: Select mux mode: ALT6 mux port: USBH2_DIR of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_MEM_CTRL[1] of instance: crm. NOTE: Pad LD1 is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT for mode ALT2.

Table 105: Register: IOMUXC_SW_MUX_CTL_PAD_LD2

Offset 0x00d0 (IOMUXC_SW_MUX_CTL_PAD_LD2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 106: Register IOMUXC_SW_MUX_CTL_PAD_LD2 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD2. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 106: Register IOMUXC_SW_MUX_CTL_PAD_LD2 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD2. 000: Select mux mode: ALT0 mux port: LCDC_LD[2] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[2] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[15] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[2] of instance: ata. 100: Select mux mode: ALT4 mux port: VEN1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[17] of instance: gpio2. 110: Select mux mode: ALT6 mux port: USBH2_STP of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_MEM_TYPE[0] of instance: crm.

Table 107: Register: IOMUXC_SW_MUX_CTL_PAD_LD3

Offset 0x00d4 (IOMUXC_SW_MUX_CTL_PAD_LD3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 108: Register IOMUXC_SW_MUX_CTL_PAD_LD3 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD3. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 108: Register IOMUXC_SW_MUX_CTL_PAD_LD3 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD3. 000: Select mux mode: ALT0 mux port: LCDC_LD[3] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[3] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[14] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[3] of instance: ata. 100: Select mux mode: ALT4 mux port: TX1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[18] of instance: gpio2. 110: Select mux mode: ALT6 mux port: USBH2_NXT of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_MEM_TYPE[1] of instance: crm. NOTE: Pad LD3 is involved in Daisy Chain. - Config Register IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT for mode ALT4.

Table 109: Register: IOMUXC_SW_MUX_CTL_PAD_LD4

Offset 0x00d8 (IOMUXC_SW_MUX_CTL_PAD_LD4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 110: Register IOMUXC_SW_MUX_CTL_PAD_LD4 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD4. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 110: Register IOMUXC_SW_MUX_CTL_PAD_LD4 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD4. 000: Select mux mode: ALT0 mux port: LCDC_LD[4] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[4] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[13] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[4] of instance: ata. 100: Select mux mode: ALT4 mux port: PD1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[19] of instance: gpio2. 110: Select mux mode: ALT6 mux port: USBH2_DATA[0] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_PAGE_SIZE[0] of instance: crm. NOTE: Pad LD4 is involved in Daisy Chain. - Config Register IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT for mode ALT4.

Table 111: Register: IOMUXC_SW_MUX_CTL_PAD_LD5

Offset 0x00dc (IOMUXC_SW_MUX_CTL_PAD_LD5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 112: Register IOMUXC_SW_MUX_CTL_PAD_LD5 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD5. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 112: Register IOMUXC_SW_MUX_CTL_PAD_LD5 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD5. 000: Select mux mode: ALT0 mux port: LCDC_LD[5] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[5] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[12] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[5] of instance: ata. 100: Select mux mode: ALT4 mux port: RX1 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[19] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[1] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_PAGE_SIZE[1] of instance: crm. NOTE: Pad LD5 is involved in Daisy Chain. - Config Register IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT for mode ALT4.

Table 113: Register: IOMUXC_SW_MUX_CTL_PAD_LD6

Offset 0x00e0 (IOMUXC_SW_MUX_CTL_PAD_LD6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 114: Register IOMUXC_SW_MUX_CTL_PAD_LD6 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD6. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 114: Register IOMUXC_SW_MUX_CTL_PAD_LD6 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD6. 000: Select mux mode: ALT0 mux port: LCDC_LD[6] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[6] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[11] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[6] of instance: ata. 100: Select mux mode: ALT4 mux port: CLK1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[20] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[2] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_BUS_WIDTH[0] of instance: crm.

Table 115: Register: IOMUXC_SW_MUX_CTL_PAD_LD7

Offset 0x00e4 (IOMUXC_SW_MUX_CTL_PAD_LD7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 116: Register IOMUXC_SW_MUX_CTL_PAD_LD7 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD7. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 116: Register IOMUXC_SW_MUX_CTL_PAD_LD7 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD7. 000: Select mux mode: ALT0 mux port: LCDC_LD[7] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[7] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CSI_D[10] of instance: csi. 011: Select mux mode: ALT3 mux port: DATA[7] of instance: ata. 100: Select mux mode: ALT4 mux port: RST1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[21] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[3] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_BUS_WIDTH[1] of instance: crm.

Table 117: Register: IOMUXC_SW_MUX_CTL_PAD_LD8

Offset 0x00e8 (IOMUXC_SW_MUX_CTL_PAD_LD8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 118: Register IOMUXC_SW_MUX_CTL_PAD_LD8 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD8. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 118: Register IOMUXC_SW_MUX_CTL_PAD_LD8 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD8. 000: Select mux mode: ALT0 mux port: LCDC_LD[8] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[8] of instance: slcdc. 010: Select mux mode: ALT2 mux port: RXD_MUX of instance: uart4. 011: Select mux mode: ALT3 mux port: DATA[8] of instance: ata. 100: Select mux mode: ALT4 mux port: AUD3_TXD of instance: audmux. 101: Select mux mode: ALT5 mux port: TX_ERR of instance: fec. 110: Select mux mode: ALT6 mux port: CMD of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_USB_SRC[0] of instance: crm. NOTE: Pad LD8 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2.

Table 119: Register: IOMUXC_SW_MUX_CTL_PAD_LD9

Offset 0x00ec (IOMUXC_SW_MUX_CTL_PAD_LD9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 120: Register IOMUXC_SW_MUX_CTL_PAD_LD9 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD9. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 120: Register IOMUXC_SW_MUX_CTL_PAD_LD9 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD9. 000: Select mux mode: ALT0 mux port: LCDC_LD[9] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[9] of instance: slcdc. 010: Select mux mode: ALT2 mux port: TXD_MUX of instance: uart4. 011: Select mux mode: ALT3 mux port: DATA[9] of instance: ata. 100: Select mux mode: ALT4 mux port: AUD3_RXD of instance: audmux. 101: Select mux mode: ALT5 mux port: COL of instance: fec. 110: Select mux mode: ALT6 mux port: CLK of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_USB_SRC[1] of instance: crm. NOTE: Pad LD9 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_FEC_FEC_COL_SELECT_INPUT for mode ALT5.

Table 121: Register: IOMUXC_SW_MUX_CTL_PAD_LD10

Offset 0x00f0 (IOMUXC_SW_MUX_CTL_PAD_LD10) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 122: Register IOMUXC_SW_MUX_CTL_PAD_LD10 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD10. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 122: Register IOMUXC_SW_MUX_CTL_PAD_LD10 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD10. 000: Select mux mode: ALT0 mux port: LCDC_LD[10] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[10] of instance: slcdc. 010: Select mux mode: ALT2 mux port: RTS of instance: uart4. 011: Select mux mode: ALT3 mux port: DATA[10] of instance: ata. 100: Select mux mode: ALT4 mux port: AUD3_TXC of instance: aud-mux. 101: Select mux mode: ALT5 mux port: RX_ERR of instance: fec. 110: Select mux mode: ALT6 mux port: DAT0 of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_MLC_SEL of instance: crm. NOTE: Pad LD10 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT for mode ALT5. - Config Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2.

Table 123: Register: IOMUXC_SW_MUX_CTL_PAD_LD11

Offset 0x00f4 (IOMUXC_SW_MUX_CTL_PAD_LD11) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 124: Register IOMUXC_SW_MUX_CTL_PAD_LD11 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD11. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 124: Register IOMUXC_SW_MUX_CTL_PAD_LD11 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD11. 000: Select mux mode: ALT0 mux port: LCDC_LD[11] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[11] of instance: slcdc. 010: Select mux mode: ALT2 mux port: CTS of instance: uart4. 011: Select mux mode: ALT3 mux port: DATA[11] of instance: ata. 100: Select mux mode: ALT4 mux port: AUD3_TXFS of instance: audmux. 101: Select mux mode: ALT5 mux port: RDATA[2] of instance: fec. 110: Select mux mode: ALT6 mux port: DAT1 of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_SPARE_SIZE of instance: crm. NOTE: Pad LD11 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT for mode ALT5.

Table 125: Register: IOMUXC_SW_MUX_CTL_PAD_LD12

Offset 0x00f8 (IOMUXC_SW_MUX_CTL_PAD_LD12) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 126: Register IOMUXC_SW_MUX_CTL_PAD_LD12 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD12. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 126: Register IOMUXC_SW_MUX_CTL_PAD_LD12 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD12. 000: Select mux mode: ALT0 mux port: LCDC_LD[12] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[12] of instance: slcdc. 010: Select mux mode: ALT2 mux port: MOSI of instance: cspi2. 011: Select mux mode: ALT3 mux port: DATA[12] of instance: ata. 100: Select mux mode: ALT4 mux port: ROW[6] of instance: kpp. 101: Select mux mode: ALT5 mux port: RDATA[3] of instance: fec. 110: Select mux mode: ALT6 mux port: DAT2 of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_SRC[0] of instance: crm. NOTE: Pad LD12 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT for mode ALT5. - Config Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT for mode ALT4.

Table 127: Register: IOMUXC_SW_MUX_CTL_PAD_LD13

Offset 0x00fc (IOMUXC_SW_MUX_CTL_PAD_LD13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 128: Register IOMUXC_SW_MUX_CTL_PAD_LD13 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD13. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 128: Register IOMUXC_SW_MUX_CTL_PAD_LD13 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD13. 000: Select mux mode: ALT0 mux port: LCDC_LD[13] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[13] of instance: slcdc. 010: Select mux mode: ALT2 mux port: MISO of instance: cspi2. 011: Select mux mode: ALT3 mux port: DATA[13] of instance: ata. 100: Select mux mode: ALT4 mux port: ROW[7] of instance: kpp. 101: Select mux mode: ALT5 mux port: TDATA[2] of instance: fec. 110: Select mux mode: ALT6 mux port: DAT3 of instance: esdhc2. 111: Select mux mode: ALT7 mux port: BT_SRC[1] of instance: crm. NOTE: Pad LD13 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT for mode ALT4.

Table 129: Register: IOMUXC_SW_MUX_CTL_PAD_LD14

Offset 0x0100 (IOMUXC_SW_MUX_CTL_PAD_LD14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 130: Register IOMUXC_SW_MUX_CTL_PAD_LD14 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD14. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 130: Register IOMUXC_SW_MUX_CTL_PAD_LD14 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD14. 000: Select mux mode: ALT0 mux port: LCDC_LD[14] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[14] of instance: slcdc. 010: Select mux mode: ALT2 mux port: SCLK of instance: cspi2. 011: Select mux mode: ALT3 mux port: DATA[14] of instance: ata. 100: Select mux mode: ALT4 mux port: COL[6] of instance: kpp. 101: Select mux mode: ALT5 mux port: TDATA[3] of instance: fec. 110: Select mux mode: ALT6 mux port: AUD3_RXC of instance: audmux. 111: Select mux mode: ALT7 mux port: BT_EEPROM_CFG of instance: crm. NOTE: Pad LD14 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT for mode ALT4.

Table 131: Register: IOMUXC_SW_MUX_CTL_PAD_LD15

Offset 0x0104 (IOMUXC_SW_MUX_CTL_PAD_LD15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 132: Register IOMUXC_SW_MUX_CTL_PAD_LD15 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LD15. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 132: Register IOMUXC_SW_MUX_CTL_PAD_LD15 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: LD15. 000: Select mux mode: ALT0 mux port: LCDC_LD[15] of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_DATA[15] of instance: slcdc. 010: Select mux mode: ALT2 mux port: RDY of instance: cspi2. 011: Select mux mode: ALT3 mux port: DATA[15] of instance: ata. 100: Select mux mode: ALT4 mux port: COL[7] of instance: kpp. 101: Select mux mode: ALT5 mux port: RX_CLK of instance: fec. 110: Select mux mode: ALT6 mux port: AUD3_RXFS of instance: aud-mux. 111: Select mux mode: ALT7 mux port: BT_UART_SRC[0] of instance: crm. NOTE: Pad LD15 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT for mode ALT5. - Config Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT for mode ALT4.

Table 133: Register: IOMUXC_SW_MUX_CTL_PAD_HSYNC

Offset 0x0108 (IOMUXC_SW_MUX_CTL_PAD_HSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 134: Register IOMUXC_SW_MUX_CTL_PAD_HSYNC Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad HSYNC. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 134: Register IOMUXC_SW_MUX_CTL_PAD_HSYNC Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: HSYNC. 000: Select mux mode: ALT0 mux port: LCDC_HSYN of instance: lcdc. 010: Select mux mode: ALT2 mux port: SCL of instance: i2c3. 011: Select mux mode: ALT3 mux port: BUFFER_EN of instance: ata. 100: Select mux mode: ALT4 mux port: VEN1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[22] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[4] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_UART_SRC[1] of instance: crm. NOTE: Pad HSYNC is involved in Daisy Chain. - Config Register IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT for mode ALT2.

Table 135: Register: IOMUXC_SW_MUX_CTL_PAD_VSYNC

Offset 0x010c (IOMUXC_SW_MUX_CTL_PAD_VSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 136: Register IOMUXC_SW_MUX_CTL_PAD_VSYNC Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad VSYNC. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 136: Register IOMUXC_SW_MUX_CTL_PAD_VSYNC Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: VSYNC. 000: Select mux mode: ALT0 mux port: LCDC_VSYN of instance: lcdc. 010: Select mux mode: ALT2 mux port: SDA of instance: i2c3. 011: Select mux mode: ALT3 mux port: DMARQ of instance: ata. 100: Select mux mode: ALT4 mux port: TX1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[23] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[5] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_UART_SRC[2] of instance: crm. NOTE: Pad VSYNC is involved in Daisy Chain. - Config Register IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT for mode ALT4.

Table 137: Register: IOMUXC_SW_MUX_CTL_PAD_LSCLK

Offset 0x0110 (IOMUXC_SW_MUX_CTL_PAD_LSCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 138: Register IOMUXC_SW_MUX_CTL_PAD_LSCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad LSCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 138: Register IOMUXC_SW_MUX_CTL_PAD_LSCLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: LSCLK. 000: Select mux mode: ALT0 mux port: LCDC_LSCLK of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_CS of instance: slcdc. 011: Select mux mode: ALT3 mux port: DA_0 of instance: ata. 100: Select mux mode: ALT4 mux port: PD1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[24] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[6] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_LPB_FREQ[0] of instance: crm. NOTE: Pad LSCLK is involved in Daisy Chain. - Config Register IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT for mode ALT4.

Table 139: Register: IOMUXC_SW_MUX_CTL_PAD_OE_ACD

Offset 0x0114 (IOMUXC_SW_MUX_CTL_PAD_OE_ACD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 140: Register IOMUXC_SW_MUX_CTL_PAD_OE_ACD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad OE_ACD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 140: Register IOMUXC_SW_MUX_CTL_PAD_OE_ACD Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: OE_ACD. 000: Select mux mode: ALT0 mux port: LCDC_OE_ACD of instance: lcdc. 001: Select mux mode: ALT1 mux port: SLCDC_RS of instance: slcdc. 010: Select mux mode: ALT2 mux port: SS0 of instance: cspi2. 011: Select mux mode: ALT3 mux port: DA_1 of instance: ata. 100: Select mux mode: ALT4 mux port: RX1 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[25] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_DATA[7] of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_LPB_FREQ[1] of instance: crm. NOTE: Pad OE_ACD is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT for mode ALT4.

Table 141: Register: IOMUXC_SW_MUX_CTL_PAD_CONTRAST

Offset 0x0118 (IOMUXC_SW_MUX_CTL_PAD_CONTRAST) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 142: Register IOMUXC_SW_MUX_CTL_PAD_CONTRAST Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CONTRAST. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 142: Register IOMUXC_SW_MUX_CTL_PAD_CONTRAST Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CONTRAST. 000: Select mux mode: ALT0 mux port: LCDC_CONTRAST of instance: lcdc. 001: Select mux mode: ALT1 mux port: CAPIN1 of instance: gpt4. 010: Select mux mode: ALT2 mux port: SS1 of instance: cspi2. 011: Select mux mode: ALT3 mux port: DA_2 of instance: ata. 100: Select mux mode: ALT4 mux port: PWMO of instance: pwm4. 101: Select mux mode: ALT5 mux port: CRS of instance: fec. 110: Select mux mode: ALT6 mux port: USBH2_PWR of instance: usb_top. 111: Select mux mode: ALT7 mux port: WDOG_B of instance: wdog. NOTE: Pad CONTRAST is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_FEC_FEC_CRS_SELECT_INPUT for mode ALT5.

Table 143: Register: IOMUXC_SW_MUX_CTL_PAD_PWM

Offset 0x011c (IOMUXC_SW_MUX_CTL_PAD_PWM) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 144: Register IOMUXC_SW_MUX_CTL_PAD_PWM Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad PWM. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 144: Register IOMUXC_SW_MUX_CTL_PAD_PWM Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: PWM. 000: Select mux mode: ALT0 mux port: PWMO of instance: pwm1. 001: Select mux mode: ALT1 mux port: CMPOUT1 of instance: gpt4. 101: Select mux mode: ALT5 mux port: GPIO[26] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBH2_OC of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_LPB_FREQ[2] of instance: crm. NOTE: Pad PWM is involved in Daisy Chain. - Config Register IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT for mode ALT6.

Table 145: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D2

Offset 0x0120 (IOMUXC_SW_MUX_CTL_PAD_CSI_D2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 146: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D2 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D2. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 146: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D2 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D2. 000: Select mux mode: ALT0 mux port: CSI_D[2] of instance: csi. 001: Select mux mode: ALT1 mux port: RXD_MUX of instance: uart5. 010: Select mux mode: ALT2 mux port: DAT4 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: SCKR of instance: esai. 100: Select mux mode: ALT4 mux port: CLK0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[27] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[0] of instance: usb_top. 111: Select mux mode: ALT7 mux port: MOSI of instance: cspi3. NOTE: Pad CSI_D2 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT1.

Table 147: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D3

Offset 0x0124 (IOMUXC_SW_MUX_CTL_PAD_CSI_D3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 148: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D3 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D3. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 148: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D3 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D3. 000: Select mux mode: ALT0 mux port: CSI_D[3] of instance: csi. 001: Select mux mode: ALT1 mux port: TXD_MUX of instance: uart5. 010: Select mux mode: ALT2 mux port: DAT5 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: FSR of instance: esai. 100: Select mux mode: ALT4 mux port: RST0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[28] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[1] of instance: usb_top. 111: Select mux mode: ALT7 mux port: MISO of instance: cspi3. NOTE: Pad CSI_D3 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT for mode ALT2.

Table 149: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D4

Offset 0x0128 (IOMUXC_SW_MUX_CTL_PAD_CSI_D4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 150: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D4 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D4. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 150: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D4 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D4. 000: Select mux mode: ALT0 mux port: CSI_D[4] of instance: csi. 001: Select mux mode: ALT1 mux port: RTS of instance: uart5. 010: Select mux mode: ALT2 mux port: DAT6 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: HCKR of instance: esai. 100: Select mux mode: ALT4 mux port: VEN0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[29] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[2] of instance: usb_top. 111: Select mux mode: ALT7 mux port: SCLK of instance: cspi3. NOTE: Pad CSI_D4 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT for mode ALT1.

Table 151: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D5

Offset 0x012c (IOMUXC_SW_MUX_CTL_PAD_CSI_D5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 152: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D5 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D5. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 152: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D5 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D5. 000: Select mux mode: ALT0 mux port: CSI_D[5] of instance: csi. 001: Select mux mode: ALT1 mux port: CTS of instance: uart5. 010: Select mux mode: ALT2 mux port: DAT7 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: SCKT of instance: esai. 100: Select mux mode: ALT4 mux port: TX0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[30] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[3] of instance: usb_top. 111: Select mux mode: ALT7 mux port: RDY of instance: cspi3. NOTE: Pad CSI_D5 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT for mode ALT2.

Table 153: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D6

Offset 0x0130 (IOMUXC_SW_MUX_CTL_PAD_CSI_D6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 154: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D6 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D6. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 154: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D6 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D6. 000: Select mux mode: ALT0 mux port: CSI_D[6] of instance: csi. 001: Select mux mode: ALT1 mux port: ROW[6] of instance: kpp. 010: Select mux mode: ALT2 mux port: CMD of instance: esdhc2. 011: Select mux mode: ALT3 mux port: FST of instance: esai. 100: Select mux mode: ALT4 mux port: PD0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[31] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[4] of instance: usb_top. 111: Select mux mode: ALT7 mux port: SS0 of instance: cspi3. NOTE: Pad CSI_D6 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT for mode ALT1.

Table 155: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D7

Offset 0x0134 (IOMUXC_SW_MUX_CTL_PAD_CSI_D7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 156: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D7 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D7. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 156: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D7 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D7. 000: Select mux mode: ALT0 mux port: CSI_D[7] of instance: csi. 001: Select mux mode: ALT1 mux port: ROW[7] of instance: kpp. 010: Select mux mode: ALT2 mux port: CLK of instance: esdhc2. 011: Select mux mode: ALT3 mux port: HCKT of instance: esai. 100: Select mux mode: ALT4 mux port: RX0 of instance: sim1. 101: Select mux mode: ALT5 mux port: GPIO[6] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[5] of instance: usb_top. 111: Select mux mode: ALT7 mux port: SS1 of instance: cspi3. NOTE: Pad CSI_D7 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT for mode ALT1.

Table 157: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D8

Offset 0x0138 (IOMUXC_SW_MUX_CTL_PAD_CSI_D8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 158: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D8 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D8. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 158: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D8 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D8. 000: Select mux mode: ALT0 mux port: CSI_D[8] of instance: csi. 001: Select mux mode: ALT1 mux port: COL[6] of instance: kpp. 010: Select mux mode: ALT2 mux port: AUD6_RXC of instance: audmux. 011: Select mux mode: ALT3 mux port: TX5_RX0 of instance: esai. 100: Select mux mode: ALT4 mux port: CLK0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[7] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[6] of instance: usb_top. 111: Select mux mode: ALT7 mux port: SS2 of instance: cspi3. NOTE: Pad CSI_D8 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT for mode ALT1.

Table 159: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_D9

Offset 0x013c (IOMUXC_SW_MUX_CTL_PAD_CSI_D9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 160: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D9 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_D9. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 160: Register IOMUXC_SW_MUX_CTL_PAD_CSI_D9 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_D9. 000: Select mux mode: ALT0 mux port: CSI_D[9] of instance: csi. 001: Select mux mode: ALT1 mux port: COL[7] of instance: kpp. 010: Select mux mode: ALT2 mux port: AUD6_RXFS of instance: audmux. 011: Select mux mode: ALT3 mux port: TX4_RX1 of instance: esai. 100: Select mux mode: ALT4 mux port: RST0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[21] of instance: gpio4. 110: Select mux mode: ALT6 mux port: USBOTG_DATA[7] of instance: usb_top. 111: Select mux mode: ALT7 mux port: SS3 of instance: cspi3. NOTE: Pad CSI_D9 is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT for mode ALT1.

Table 161: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK

Offset 0x0140 (IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162: Register IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_MCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 162: Register IOMUXC_SW_MUX_CTL_PAD_CSI_MCLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_MCLK. 000: Select mux mode: ALT0 mux port: CSI_MCLK of instance: csi. 001: Select mux mode: ALT1 mux port: AUD6_TXD of instance: audmux. 010: Select mux mode: ALT2 mux port: DAT0 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: TX3_RX2 of instance: esai. 100: Select mux mode: ALT4 mux port: VEN0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[8] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_DIR of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_RES[0] of instance: crm. NOTE: Pad CSI_MCLK is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT for mode ALT2.

Table 163: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC

Offset 0x0144 (IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 164: Register IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_VSYNC. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 164: Register IOMUXC_SW_MUX_CTL_PAD_CSI_VSYNC Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_VSYNC. 000: Select mux mode: ALT0 mux port: CSI_VSYNC of instance: csi. 001: Select mux mode: ALT1 mux port: AUD6_RXD of instance: audmux. 010: Select mux mode: ALT2 mux port: DAT1 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: TX2_RX3 of instance: esai. 100: Select mux mode: ALT4 mux port: TX0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[9] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_STP of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_RES[1] of instance: crm. NOTE: Pad CSI_VSYNC is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT for mode ALT2.

Table 165: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC

Offset 0x0148 (IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 166: Register IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_HSYNC. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 166: Register IOMUXC_SW_MUX_CTL_PAD_CSI_HSYNC Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_HSYNC. 000: Select mux mode: ALT0 mux port: CSI_HSYNC of instance: csi. 001: Select mux mode: ALT1 mux port: AUD6_TXC of instance: audmux. 010: Select mux mode: ALT2 mux port: DAT2 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: TX1 of instance: esai. 100: Select mux mode: ALT4 mux port: PD0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[10] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_NXT of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_RES[2] of instance: crm. NOTE: Pad CSI_HSYNC is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT for mode ALT2.

Table 167: Register: IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK

Offset 0x014c (IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 168: Register IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSI_PIXCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 168: Register IOMUXC_SW_MUX_CTL_PAD_CSI_PIXCLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: CSI_PIXCLK. 000: Select mux mode: ALT0 mux port: CSI_PIXCLK of instance: csi. 001: Select mux mode: ALT1 mux port: AUD6_TXFS of instance: audmux. 010: Select mux mode: ALT2 mux port: DAT3 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: TX0 of instance: esai. 100: Select mux mode: ALT4 mux port: RX0 of instance: sim2. 101: Select mux mode: ALT5 mux port: GPIO[11] of instance: gpio1. 110: Select mux mode: ALT6 mux port: USBOTG_CLK of instance: usb_top. 111: Select mux mode: ALT7 mux port: BT_RES[3] of instance: crm. NOTE: Pad CSI_PIXCLK is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT for mode ALT2.

Table 169: Register: IOMUXC_SW_MUX_CTL_PAD_I2C1_CLK

Offset 0x0150 (IOMUXC_SW_MUX_CTL_PAD_I2C1_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 170: Register IOMUXC_SW_MUX_CTL_PAD_I2C1_CLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad I2C1_CLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C1_CLK. 000: Select mux mode: ALT0 mux port: SCL of instance: i2c1. 101: Select mux mode: ALT5 mux port: GPIO[12] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[6] of instance: slcdc.

Table 171: Register: IOMUXC_SW_MUX_CTL_PAD_I2C1_DAT

Offset 0x0154 (IOMUXC_SW_MUX_CTL_PAD_I2C1_DAT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 172: Register IOMUXC_SW_MUX_CTL_PAD_I2C1_DAT Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad I2C1_DAT. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: I2C1_DAT. 000: Select mux mode: ALT0 mux port: SDA of instance: i2c1. 101: Select mux mode: ALT5 mux port: GPIO[13] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[7] of instance: slcdc.

Table 173: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_MOSI

Offset 0x0158 (IOMUXC_SW_MUX_CTL_PAD_CSPI1_MOSI) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 174: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_MOSI Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_MOSI. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 174: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_MOSI Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: CSPI1_MOSI. 000: Select mux mode: ALT0 mux port: MOSI of instance: cspi1. 010: Select mux mode: ALT2 mux port: RXD_MUX of instance: uart3. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_0 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[14] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[12] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[4] of instance: arm926p_platform. NOTE: Pad CSPI1_MOSI is involved in Daisy Chain. - Config Register IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT2.

Table 175: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_MISO

Offset 0x015c (IOMUXC_SW_MUX_CTL_PAD_CSPI1_MISO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 176: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_MISO Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_MISO. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 176: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_MISO Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: CSPI1_MISO. 000: Select mux mode: ALT0 mux port: MISO of instance: cspi1. 010: Select mux mode: ALT2 mux port: TXD_MUX of instance: uart3. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_1 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[15] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[13] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[5] of instance: arm926p_platform.

Table 177: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS0

Offset 0x0160 (IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 178: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_SS0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 178: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: CSPI1_SS0. 000: Select mux mode: ALT0 mux port: SS0 of instance: cspi1. 001: Select mux mode: ALT1 mux port: LCDC_LD[16] of instance: lcdc. 010: Select mux mode: ALT2 mux port: PWMO of instance: pwm2. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_2 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[16] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_CS of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[6] of instance: arm926p_platform.

Table 179: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS1

Offset 0x0164 (IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 180: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_SS1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 180: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SS1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: CSPI1_SS1. 000: Select mux mode: ALT0 mux port: SS1 of instance: cspi1. 001: Select mux mode: ALT1 mux port: SDA of instance: i2c3. 010: Select mux mode: ALT2 mux port: RTS of instance: uart3. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_3 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[17] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_RS of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[7] of instance: arm926p_platform. NOTE: Pad CSPI1_SS1 is involved in Daisy Chain. - Config Register IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT2.

Table 181: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_SCLK

Offset 0x0168 (IOMUXC_SW_MUX_CTL_PAD_CSPI1_SCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 182: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_SCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 182: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_SCLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: CSPI1_SCLK. 000: Select mux mode: ALT0 mux port: SCLK of instance: cspi1. 010: Select mux mode: ALT2 mux port: CTS of instance: uart3. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_4 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[18] of instance: gpio1. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[14] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[8] of instance: arm926p_platform.

Table 183: Register: IOMUXC_SW_MUX_CTL_PAD_CSPI1_RDY

Offset 0x016c (IOMUXC_SW_MUX_CTL_PAD_CSPI1_RDY) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 184: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_RDY Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CSPI1_RDY. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 184: Register IOMUXC_SW_MUX_CTL_PAD_CSPI1_RDY Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: CSPI1_RDY. 000: Select mux mode: ALT0 mux port: RDY of instance: cspi1. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_5 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[22] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[15] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[9] of instance: arm926p_platform.

Table 185: Register: IOMUXC_SW_MUX_CTL_PAD_UART1_RXD

Offset 0x0170 (IOMUXC_SW_MUX_CTL_PAD_UART1_RXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 186: Register IOMUXC_SW_MUX_CTL_PAD_UART1_RXD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART1_RXD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART1_RXD. 000: Select mux mode: ALT0 mux port: RXD_MUX of instance: uart1. 011: Select mux mode: ALT3 mux port: DTR of instance: uart2. 100: Select mux mode: ALT4 mux port: LCDC_CLS of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[22] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[8] of instance: slcdc.

Table 187: Register: IOMUXC_SW_MUX_CTL_PAD_UART1_TXD

Offset 0x0174 (IOMUXC_SW_MUX_CTL_PAD_UART1_TXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 188: Register IOMUXC_SW_MUX_CTL_PAD_UART1_TXD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART1_TXD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART1_TXD. 000: Select mux mode: ALT0 mux port: TXD_MUX of instance: uart1. 011: Select mux mode: ALT3 mux port: DSR of instance: uart2. 100: Select mux mode: ALT4 mux port: LCDC_SPL of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[23] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[9] of instance: slcdc.

Table 189: Register: IOMUXC_SW_MUX_CTL_PAD_UART1_RTS

Offset 0x0178 (IOMUXC_SW_MUX_CTL_PAD_UART1_RTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 190: Register IOMUXC_SW_MUX_CTL_PAD_UART1_RTS Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART1_RTS. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 190: Register IOMUXC_SW_MUX_CTL_PAD_UART1_RTS Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART1_RTS. 000: Select mux mode: ALT0 mux port: RTS of instance: uart1. 001: Select mux mode: ALT1 mux port: CSI_D[0] of instance: csi. 010: Select mux mode: ALT2 mux port: CAPIN1 of instance: gpt3. 011: Select mux mode: ALT3 mux port: DCD of instance: uart2. 100: Select mux mode: ALT4 mux port: LCDC_PS of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[24] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[10] of instance: slcdc. NOTE: Pad UART1_RTS is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT for mode ALT1.

Table 191: Register: IOMUXC_SW_MUX_CTL_PAD_UART1_CTS

Offset 0x017c (IOMUXC_SW_MUX_CTL_PAD_UART1_CTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 192: Register IOMUXC_SW_MUX_CTL_PAD_UART1_CTS Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART1_CTS. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 192: Register IOMUXC_SW_MUX_CTL_PAD_UART1_CTS Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART1_CTS. 000: Select mux mode: ALT0 mux port: CTS of instance: uart1. 001: Select mux mode: ALT1 mux port: CSI_D[1] of instance: csi. 010: Select mux mode: ALT2 mux port: CMPOUT1 of instance: gpt3. 011: Select mux mode: ALT3 mux port: RI of instance: uart2. 100: Select mux mode: ALT4 mux port: LCDC_REV of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[25] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[11] of instance: slcdc. NOTE: Pad UART1_CTS is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT for mode ALT1.

Table 193: Register: IOMUXC_SW_MUX_CTL_PAD_UART2_RXD

Offset 0x0180 (IOMUXC_SW_MUX_CTL_PAD_UART2_RXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 194: Register IOMUXC_SW_MUX_CTL_PAD_UART2_RXD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART2_RXD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 194: Register IOMUXC_SW_MUX_CTL_PAD_UART2_RXD Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 3 iomux modes to be used for pad: UART2_RXD. 000: Select mux mode: ALT0 mux port: RXD_MUX of instance: uart2. 001: Select mux mode: ALT1 mux port: DAT7 of instance: esdhc1. 101: Select mux mode: ALT5 mux port: GPIO[26] of instance: gpio4. NOTE: Pad UART2_RXD is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT for mode ALT1.

Table 195: Register: IOMUXC_SW_MUX_CTL_PAD_UART2_TXD

Offset 0x0184 (IOMUXC_SW_MUX_CTL_PAD_UART2_TXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 196: Register IOMUXC_SW_MUX_CTL_PAD_UART2_TXD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART2_TXD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 196: Register IOMUXC_SW_MUX_CTL_PAD_UART2_TXD Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: UART2_TXD. 000: Select mux mode: ALT0 mux port: TXD_MUX of instance: uart2. 001: Select mux mode: ALT1 mux port: DAT6 of instance: esdhc1. 010: Select mux mode: ALT2 mux port: TX_ERR of instance: fec. 101: Select mux mode: ALT5 mux port: GPIO[27] of instance: gpio4. 111: Select mux mode: ALT7 mux port: EXTDMA_0 of instance: sdma. NOTE: Pad UART2_TXD is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT for mode ALT1.

Table 197: Register: IOMUXC_SW_MUX_CTL_PAD_UART2_RTS

Offset 0x0188 (IOMUXC_SW_MUX_CTL_PAD_UART2_RTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 198: Register IOMUXC_SW_MUX_CTL_PAD_UART2_RTS Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART2_RTS. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 198: Register IOMUXC_SW_MUX_CTL_PAD_UART2_RTS Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: UART2_RTS. 000: Select mux mode: ALT0 mux port: RTS of instance: uart2. 001: Select mux mode: ALT1 mux port: DAT5 of instance: esdhc1. 010: Select mux mode: ALT2 mux port: COL of instance: fec. 011: Select mux mode: ALT3 mux port: CAPIN1 of instance: gpt1. 100: Select mux mode: ALT4 mux port: EPITO of instance: epit2. 101: Select mux mode: ALT5 mux port: GPIO[28] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SS3 of instance: cspi2. 111: Select mux mode: ALT7 mux port: EXTDMA_1 of instance: sdma. NOTE: Pad UART2_RTS is involved in Daisy Chain. - Config Register IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_COL_SELECT_INPUT for mode ALT2.

Table 199: Register: IOMUXC_SW_MUX_CTL_PAD_UART2_CTS

Offset 0x018c (IOMUXC_SW_MUX_CTL_PAD_UART2_CTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 200: Register IOMUXC_SW_MUX_CTL_PAD_UART2_CTS Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UART2_CTS. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 200: Register IOMUXC_SW_MUX_CTL_PAD_UART2_CTS Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: UART2_CTS. 000: Select mux mode: ALT0 mux port: CTS of instance: uart2. 001: Select mux mode: ALT1 mux port: DAT4 of instance: esdhc1. 010: Select mux mode: ALT2 mux port: RX_ERR of instance: fec. 011: Select mux mode: ALT3 mux port: CMPOUT1 of instance: gpt1. 101: Select mux mode: ALT5 mux port: GPIO[29] of instance: gpio4. 110: Select mux mode: ALT6 mux port: SS3 of instance: cspi3. 111: Select mux mode: ALT7 mux port: EXTDMA_2 of instance: sdma. NOTE: Pad UART2_CTS is involved in Daisy Chain. - Config Register IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT for mode ALT2.

Table 201: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_CMD

Offset 0x0190 (IOMUXC_SW_MUX_CTL_PAD_SD1_CMD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 202: Register IOMUXC_SW_MUX_CTL_PAD_SD1_CMD Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_CMD. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 202: Register IOMUXC_SW_MUX_CTL_PAD_SD1_CMD Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_CMD. 000: Select mux mode: ALT0 mux port: CMD of instance: esdhc1. 001: Select mux mode: ALT1 mux port: MOSI of instance: cspi2. 010: Select mux mode: ALT2 mux port: RDATA[2] of instance: fec. 100: Select mux mode: ALT4 mux port: SDMA_DBG_EVT_SEL of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[23] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[0] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[10] of instance: arm926p_platform. NOTE: Pad SD1_CMD is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT for mode ALT2.

Table 203: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_CLK

Offset 0x0194 (IOMUXC_SW_MUX_CTL_PAD_SD1_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 204: Register IOMUXC_SW_MUX_CTL_PAD_SD1_CLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_CLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 204: Register IOMUXC_SW_MUX_CTL_PAD_SD1_CLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_CLK. 000: Select mux mode: ALT0 mux port: CLK of instance: esdhc1. 001: Select mux mode: ALT1 mux port: MISO of instance: cspi2. 010: Select mux mode: ALT2 mux port: RDATA[3] of instance: fec. 100: Select mux mode: ALT4 mux port: SDMA_DBG_STAT_0 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[24] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[1] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[11] of instance: arm926p_platform. NOTE: Pad SD1_CLK is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT for mode ALT2.

Table 205: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0

Offset 0x0198 (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 206: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_DATA0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 206: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA0. 000: Select mux mode: ALT0 mux port: DAT0 of instance: esdhc1. 001: Select mux mode: ALT1 mux port: SCLK of instance: cspi2. 010: Select mux mode: ALT2 mux port: TDATA[2] of instance: fec. 011: Select mux mode: ALT3 mux port: AUD7_TXFS of instance: audmux. 100: Select mux mode: ALT4 mux port: SDMA_DBG_STAT_1 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[25] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[2] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[12] of instance: arm926p_platform. NOTE: Pad SD1_DATA0 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT3. - Config Register IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT for mode ALT1.

Table 207: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1

Offset 0x019c (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 208: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_DATA1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 208: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA1. 000: Select mux mode: ALT0 mux port: DAT1 of instance: esdhc1. 001: Select mux mode: ALT1 mux port: RDY of instance: cspi2. 010: Select mux mode: ALT2 mux port: TDATA[3] of instance: fec. 011: Select mux mode: ALT3 mux port: AUD7_RXD of instance: audmux. 100: Select mux mode: ALT4 mux port: SDMA_DBG_STAT_2 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[26] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[3] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[13] of instance: arm926p_platform. NOTE: Pad SD1_DATA1 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT for mode ALT3. - Config Register IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT for mode ALT1.

Table 209: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2

Offset 0x01a0 (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 210: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_DATA2. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 210: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA2 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: SD1_DATA2. 000: Select mux mode: ALT0 mux port: DAT2 of instance: esdhc1. 001: Select mux mode: ALT1 mux port: SS0 of instance: cspi2. 010: Select mux mode: ALT2 mux port: RX_CLK of instance: fec. 011: Select mux mode: ALT3 mux port: AUD7_RXC of instance: audmux. 100: Select mux mode: ALT4 mux port: SDMA_DBG_STAT_3 of instance: sdma. 101: Select mux mode: ALT5 mux port: GPIO[27] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[4] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[14] of instance: arm926p_platform. NOTE: Pad SD1_DATA2 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT for mode ALT2.

Table 211: Register: IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3

Offset 0x01a4 (IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 212: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad SD1_DATA3. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 212: Register IOMUXC_SW_MUX_CTL_PAD_SD1_DATA3 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: SD1_DATA3. 000: Select mux mode: ALT0 mux port: DAT3 of instance: esdhc1. 001: Select mux mode: ALT1 mux port: SS1 of instance: cspi2. 010: Select mux mode: ALT2 mux port: CRS of instance: fec. 011: Select mux mode: ALT3 mux port: AUD7_RXFS of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[28] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SLCDC_DATA[5] of instance: slcdc. 111: Select mux mode: ALT7 mux port: TRACE[15] of instance: arm926p_platform. NOTE: Pad SD1_DATA3 is involved in Daisy Chain. - Config Register IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_FEC_FEC_CRS_SELECT_INPUT for mode ALT2.

Table 213: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_ROW0

Offset 0x01a8 (IOMUXC_SW_MUX_CTL_PAD_KPP_ROW0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 214: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_ROW0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 214: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KPP_ROW0. 000: Select mux mode: ALT0 mux port: ROW[0] of instance: kpp. 001: Select mux mode: ALT1 mux port: RXD_MUX of instance: uart3. 100: Select mux mode: ALT4 mux port: DTR of instance: uart1. 101: Select mux mode: ALT5 mux port: GPIO[29] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_0 of instance: sdma. NOTE: Pad KPP_ROW0 is involved in Daisy Chain. - Config Register IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT1.

Table 215: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_ROW1

Offset 0x01ac (IOMUXC_SW_MUX_CTL_PAD_KPP_ROW1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 216: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_ROW1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KPP_ROW1. 000: Select mux mode: ALT0 mux port: ROW[1] of instance: kpp. 001: Select mux mode: ALT1 mux port: TXD_MUX of instance: uart3. 100: Select mux mode: ALT4 mux port: DSR of instance: uart1. 101: Select mux mode: ALT5 mux port: GPIO[30] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_1 of instance: sdma.

Table 217: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_ROW2

Offset 0x01b0 (IOMUXC_SW_MUX_CTL_PAD_KPP_ROW2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 218: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW2 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_ROW2. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 218: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW2 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: KPP_ROW2. 000: Select mux mode: ALT0 mux port: ROW[2] of instance: kpp. 001: Select mux mode: ALT1 mux port: RTS of instance: uart3. 010: Select mux mode: ALT2 mux port: AUD5_RXC of instance: audmux. 011: Select mux mode: ALT3 mux port: CSI_D[0] of instance: csi. 100: Select mux mode: ALT4 mux port: DCD of instance: uart1. 101: Select mux mode: ALT5 mux port: GPIO[31] of instance: gpio2. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_2 of instance: sdma. NOTE: Pad KPP_ROW2 is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT for mode ALT3. - Config Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT for mode ALT1.

Table 219: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_ROW3

Offset 0x01b4 (IOMUXC_SW_MUX_CTL_PAD_KPP_ROW3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 220: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW3 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_ROW3. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 220: Register IOMUXC_SW_MUX_CTL_PAD_KPP_ROW3 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: KPP_ROW3. 000: Select mux mode: ALT0 mux port: ROW[3] of instance: kpp. 001: Select mux mode: ALT1 mux port: CTS of instance: uart3. 010: Select mux mode: ALT2 mux port: AUD5_RXFS of instance: aud-mux. 011: Select mux mode: ALT3 mux port: CSI_D[1] of instance: csi. 100: Select mux mode: ALT4 mux port: RI of instance: uart1. 101: Select mux mode: ALT5 mux port: GPIO[0] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_3 of instance: sdma. NOTE: Pad KPP_ROW3 is involved in Daisy Chain. - Config Register IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT for mode ALT3.

Table 221: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_COL0

Offset 0x01b8 (IOMUXC_SW_MUX_CTL_PAD_KPP_COL0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 222: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_COL0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 222: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KPP_COL0. 000: Select mux mode: ALT0 mux port: COL[0] of instance: kpp. 001: Select mux mode: ALT1 mux port: RXD_MUX of instance: uart4. 010: Select mux mode: ALT2 mux port: AUD5_TXD of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[1] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_4 of instance: sdma. NOTE: Pad KPP_COL0 is involved in Daisy Chain. - Config Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT1.

Table 223: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_COL1

Offset 0x01bc (IOMUXC_SW_MUX_CTL_PAD_KPP_COL1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 224: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_COL1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: KPP_COL1. 000: Select mux mode: ALT0 mux port: COL[1] of instance: kpp. 001: Select mux mode: ALT1 mux port: TXD_MUX of instance: uart4. 010: Select mux mode: ALT2 mux port: AUD5_RXD of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[2] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_5 of instance: sdma.

Table 225: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_COL2

Offset 0x01c0 (IOMUXC_SW_MUX_CTL_PAD_KPP_COL2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 226: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL2 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_COL2. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 226: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL2 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: KPP_COL2. 000: Select mux mode: ALT0 mux port: COL[2] of instance: kpp. 001: Select mux mode: ALT1 mux port: RTS of instance: uart4. 010: Select mux mode: ALT2 mux port: AUD5_TXC of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[3] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_6 of instance: sdma. 111: Select mux mode: ALT7 mux port: M3IF_CHOSEN_MASTER_1 of instance: emi. NOTE: Pad KPP_COL2 is involved in Daisy Chain. - Config Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT for mode ALT1.

Table 227: Register: IOMUXC_SW_MUX_CTL_PAD_KPP_COL3

Offset 0x01c4 (IOMUXC_SW_MUX_CTL_PAD_KPP_COL3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 228: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL3 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad KPP_COL3. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 228: Register IOMUXC_SW_MUX_CTL_PAD_KPP_COL3 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: KPP_COL3. 000: Select mux mode: ALT0 mux port: COL[3] of instance: kpp. 001: Select mux mode: ALT1 mux port: CTS of instance: uart4. 010: Select mux mode: ALT2 mux port: AUD5_TXFS of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[4] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_7 of instance: sdma. 111: Select mux mode: ALT7 mux port: M3IF_CHOSEN_MASTER_2 of instance: emi.

Table 229: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_MDC

Offset 0x01c8 (IOMUXC_SW_MUX_CTL_PAD_FEC_MDC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 230: Register IOMUXC_SW_MUX_CTL_PAD_FEC_MDC Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_MDC. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 230: Register IOMUXC_SW_MUX_CTL_PAD_FEC_MDC Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: FEC_MDC. 000: Select mux mode: ALT0 mux port: MDC of instance: fec. 001: Select mux mode: ALT1 mux port: CMD of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_TXD of instance: audmux. 011: Select mux mode: ALT3 mux port: DIOR of instance: ata. 101: Select mux mode: ALT5 mux port: GPIO[5] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_8 of instance: sdma. 111: Select mux mode: ALT7 mux port: LCDC_LD[16] of instance: lcdc. NOTE: Pad FEC_MDC is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT for mode ALT1.

Table 231: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_MDIO

Offset 0x01cc (IOMUXC_SW_MUX_CTL_PAD_FEC_MDIO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 232: Register IOMUXC_SW_MUX_CTL_PAD_FEC_MDIO Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_MDIO. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 232: Register IOMUXC_SW_MUX_CTL_PAD_FEC_MDIO Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: FEC_MDIO. 000: Select mux mode: ALT0 mux port: MDIO of instance: fec. 001: Select mux mode: ALT1 mux port: CLK of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_RXD of instance: audmux. 011: Select mux mode: ALT3 mux port: DIOW of instance: ata. 101: Select mux mode: ALT5 mux port: GPIO[6] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_9 of instance: sdma. 111: Select mux mode: ALT7 mux port: LCDC_LD[17] of instance: lcdc. NOTE: Pad FEC_MDIO is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT for mode ALT1.

Table 233: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA0

Offset 0x01d0 (IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 234: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_TDATA0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 234: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: FEC_TDATA0. 111: Select mux mode: ALT7 mux port: LCDC_LD[18] of instance: lcdc. 000: Select mux mode: ALT0 mux port: TDATA[0] of instance: fec. 001: Select mux mode: ALT1 mux port: DAT0 of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_TXC of instance: audmux. 011: Select mux mode: ALT3 mux port: DMACK of instance: ata. 101: Select mux mode: ALT5 mux port: GPIO[7] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_10 of instance: sdma. NOTE: Pad FEC_TDATA0 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT for mode ALT1.

Table 235: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA1

Offset 0x01d4 (IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 236: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_TDATA1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 236: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TDATA1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: FEC_TDATA1. 000: Select mux mode: ALT0 mux port: TDATA[1] of instance: fec. 001: Select mux mode: ALT1 mux port: DAT1 of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_TXFS of instance: audmux. 011: Select mux mode: ALT3 mux port: RESET_B of instance: ata. 101: Select mux mode: ALT5 mux port: GPIO[8] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_11 of instance: sdma. 111: Select mux mode: ALT7 mux port: LCDC_LD[19] of instance: lcdc. NOTE: Pad FEC_TDATA1 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT for mode ALT1.

Table 237: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_TX_EN

Offset 0x01d8 (IOMUXC_SW_MUX_CTL_PAD_FEC_TX_EN) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 238: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TX_EN Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_TX_EN. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 238: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TX_EN Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: FEC_TX_EN. 000: Select mux mode: ALT0 mux port: TX_EN of instance: fec. 001: Select mux mode: ALT1 mux port: DAT2 of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_RXC of instance: audmux. 011: Select mux mode: ALT3 mux port: IORDY of instance: ata. 100: Select mux mode: ALT4 mux port: TXCAN of instance: can2. 101: Select mux mode: ALT5 mux port: GPIO[9] of instance: gpio3. 110: Select mux mode: ALT6 mux port: ROW[4] of instance: kpp. 111: Select mux mode: ALT7 mux port: LCDC_LD[20] of instance: lcdc. NOTE: Pad FEC_TX_EN is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT for mode ALT6.

Table 239: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA0

Offset 0x01dc (IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 240: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA0 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_RDATA0. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 240: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA0 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: FEC_RDATA0. 000: Select mux mode: ALT0 mux port: RDATA[0] of instance: fec. 001: Select mux mode: ALT1 mux port: DAT3 of instance: esdhc2. 010: Select mux mode: ALT2 mux port: AUD4_RXFS of instance: audmux. 011: Select mux mode: ALT3 mux port: INTRQ of instance: ata. 100: Select mux mode: ALT4 mux port: RXCAN of instance: can2. 101: Select mux mode: ALT5 mux port: GPIO[10] of instance: gpio3. 110: Select mux mode: ALT6 mux port: ROW[5] of instance: kpp. 111: Select mux mode: ALT7 mux port: LCDC_LD[21] of instance: lcdc. NOTE: Pad FEC_RDATA0 is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT for mode ALT6.

Table 241: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA1

Offset 0x01e0 (IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 242: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA1 Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_RDATA1. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 242: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RDATA1 Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: FEC_RDATA1. 000: Select mux mode: ALT0 mux port: RDATA[1] of instance: fec. 001: Select mux mode: ALT1 mux port: SCL of instance: i2c2. 010: Select mux mode: ALT2 mux port: DAT4 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: CS0 of instance: ata. 100: Select mux mode: ALT4 mux port: TXCAN of instance: can1. 101: Select mux mode: ALT5 mux port: GPIO[11] of instance: gpio3. 110: Select mux mode: ALT6 mux port: COL[4] of instance: kpp. 111: Select mux mode: ALT7 mux port: LCDC_LD[22] of instance: lcdc. NOTE: Pad FEC_RDATA1 is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT for mode ALT6.

Table 243: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_RX_DV

Offset 0x01e4 (IOMUXC_SW_MUX_CTL_PAD_FEC_RX_DV) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 244: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RX_DV Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_RX_DV. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 244: Register IOMUXC_SW_MUX_CTL_PAD_FEC_RX_DV Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: FEC_RX_DV. 000: Select mux mode: ALT0 mux port: RX_DV of instance: fec. 001: Select mux mode: ALT1 mux port: SDA of instance: i2c2. 010: Select mux mode: ALT2 mux port: DAT5 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: CS1 of instance: ata. 100: Select mux mode: ALT4 mux port: RXCAN of instance: can1. 101: Select mux mode: ALT5 mux port: GPIO[12] of instance: gpio3. 110: Select mux mode: ALT6 mux port: COL[5] of instance: kpp. 111: Select mux mode: ALT7 mux port: LCDC_LD[23] of instance: lcdc. NOTE: Pad FEC_RX_DV is involved in Daisy Chain. - Config Register IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT for mode ALT6.

Table 245: Register: IOMUXC_SW_MUX_CTL_PAD_FEC_TX_CLK

Offset 0x01e8 (IOMUXC_SW_MUX_CTL_PAD_FEC_TX_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 246: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TX_CLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad FEC_TX_CLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 246: Register IOMUXC_SW_MUX_CTL_PAD_FEC_TX_CLK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: FEC_TX_CLK. 000: Select mux mode: ALT0 mux port: TX_CLK of instance: fec. 001: Select mux mode: ALT1 mux port: PWMO of instance: pwm3. 010: Select mux mode: ALT2 mux port: DAT6 of instance: esdhc2. 011: Select mux mode: ALT3 mux port: LCDC_LD[16] of instance: lcdc. 101: Select mux mode: ALT5 mux port: GPIO[13] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_12 of instance: sdma. 111: Select mux mode: ALT7 mux port: M3IF_CHOSEN_MASTER_0 of instance: emi. NOTE: Pad FEC_TX_CLK is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT for mode ALT2.

Table 247: Register: IOMUXC_SW_MUX_CTL_PAD_RTCK

Offset 0x01ec (IOMUXC_SW_MUX_CTL_PAD_RTCK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 248: Register IOMUXC_SW_MUX_CTL_PAD_RTCK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad RTCK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 248: Register IOMUXC_SW_MUX_CTL_PAD_RTCK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 5 iomux modes to be used for pad: RTCK. 000: Select mux mode: ALT0 mux port: RTCK of instance: arm926p_platform. 001: Select mux mode: ALT1 mux port: LINE of instance: owire. 010: Select mux mode: ALT2 mux port: DAT7 of instance: esdhc2. 101: Select mux mode: ALT5 mux port: GPIO[14] of instance: gpio3. 110: Select mux mode: ALT6 mux port: SDMA_DBG_PC_13 of instance: sdma. NOTE: Pad RTCK is involved in Daisy Chain. - Config Register IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT for mode ALT2.

Table 249: Register: IOMUXC_SW_MUX_CTL_PAD_DE_B

Offset 0x01f0 (IOMUXC_SW_MUX_CTL_PAD_DE_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 250: Register IOMUXC_SW_MUX_CTL_PAD_DE_B Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad DE_B. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: DE_B. 000: Select mux mode: ALT0 mux port: DE_B of instance: sjc. 101: Select mux mode: ALT5 mux port: GPIO[20] of instance: gpio2.

Table 251: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_A

Offset 0x01f4 (IOMUXC_SW_MUX_CTL_PAD_GPIO_A) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 252: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_A Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_A. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 252: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_A Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO_A. 000: Select mux mode: ALT0 mux port: GPIO[0] of instance: gpio1. 001: Select mux mode: ALT1 mux port: PWMO of instance: pwm2. 010: Select mux mode: ALT2 mux port: USBOTG_PWR of instance: usb_top. 011: Select mux mode: ALT3 mux port: ROW[4] of instance: kpp. 100: Select mux mode: ALT4 mux port: SCL of instance: i2c3. 110: Select mux mode: ALT6 mux port: TXCAN of instance: can1. 111: Select mux mode: ALT7 mux port: INT_MUX_OUT of instance: obsrv_module. NOTE: Pad GPIO_A is involved in Daisy Chain. - Config Register IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT for mode ALT3.

Table 253: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_B

Offset 0x01f8 (IOMUXC_SW_MUX_CTL_PAD_GPIO_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 254: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_B Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_B. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 254: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_B Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: GPIO_B. 000: Select mux mode: ALT0 mux port: GPIO[1] of instance: gpio1. 001: Select mux mode: ALT1 mux port: PWMO of instance: pwm3. 010: Select mux mode: ALT2 mux port: USBOTG_OC of instance: usb_top. 011: Select mux mode: ALT3 mux port: ROW[5] of instance: kpp. 100: Select mux mode: ALT4 mux port: SDA of instance: i2c3. 110: Select mux mode: ALT6 mux port: RXCAN of instance: can1. NOTE: Pad GPIO_B is involved in Daisy Chain. - Config Register IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT for mode ALT3. - Config Register IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT for mode ALT2.

Table 255: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_C

Offset 0x01fc (IOMUXC_SW_MUX_CTL_PAD_GPIO_C) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 256: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_C Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_C. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 256: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_C Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_C. 000: Select mux mode: ALT0 mux port: GPIO[2] of instance: gpio1. 001: Select mux mode: ALT1 mux port: PWMO of instance: pwm4. 010: Select mux mode: ALT2 mux port: SCL of instance: i2c2. 011: Select mux mode: ALT3 mux port: COL[4] of instance: kpp. 100: Select mux mode: ALT4 mux port: CAPIN1 of instance: gpt2. 101: Select mux mode: ALT5 mux port: SS2 of instance: cspi1. 110: Select mux mode: ALT6 mux port: TXCAN of instance: can2. 111: Select mux mode: ALT7 mux port: SS2 of instance: cspi2. NOTE: Pad GPIO_C is involved in Daisy Chain. - Config Register IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT for mode ALT3.

Table 257: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_D

Offset 0x0200 (IOMUXC_SW_MUX_CTL_PAD_GPIO_D) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 258: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_D Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_D. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 258: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_D Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 7 iomux modes to be used for pad: GPIO_D. 000: Select mux mode: ALT0 mux port: GPIO[3] of instance: gpio1. 001: Select mux mode: ALT1 mux port: WDOG_B of instance: wdog. 010: Select mux mode: ALT2 mux port: SDA of instance: i2c2. 011: Select mux mode: ALT3 mux port: COL[5] of instance: kpp. 100: Select mux mode: ALT4 mux port: CMPOUT1 of instance: gpt2. 110: Select mux mode: ALT6 mux port: RXCAN of instance: can2. 111: Select mux mode: ALT7 mux port: SS2 of instance: cspi3. NOTE: Pad GPIO_D is involved in Daisy Chain. - Config Register IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT for mode ALT6. - Config Register IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT for mode ALT7. - Config Register IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT for mode ALT2. - Config Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT for mode ALT3.

Table 259: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_E

Offset 0x0204 (IOMUXC_SW_MUX_CTL_PAD_GPIO_E) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 260: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_E Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_E. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 260: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_E Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: GPIO_E. 000: Select mux mode: ALT0 mux port: GPIO[4] of instance: gpio1. 001: Select mux mode: ALT1 mux port: SCL of instance: i2c3. 010: Select mux mode: ALT2 mux port: LCDC_LD[16] of instance: lcdc. 100: Select mux mode: ALT4 mux port: AUD7_TXD of instance: audmux. 110: Select mux mode: ALT6 mux port: RXD_MUX of instance: uart4. 111: Select mux mode: ALT7 mux port: CTI_TRIG_IN0_6 of instance: ect. NOTE: Pad GPIO_E is involved in Daisy Chain. - Config Register IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT for mode ALT1. - Config Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT6.

Table 261: Register: IOMUXC_SW_MUX_CTL_PAD_GPIO_F

Offset 0x0208 (IOMUXC_SW_MUX_CTL_PAD_GPIO_F) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 262: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_F Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad GPIO_F. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 262: Register IOMUXC_SW_MUX_CTL_PAD_GPIO_F Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 6 iomux modes to be used for pad: GPIO_F. 000: Select mux mode: ALT0 mux port: GPIO[5] of instance: gpio1. 010: Select mux mode: ALT2 mux port: LCDC_LD[17] of instance: lcdc. 011: Select mux mode: ALT3 mux port: EPITO of instance: epit1. 100: Select mux mode: ALT4 mux port: AUD7_TXC of instance: audmux. 110: Select mux mode: ALT6 mux port: TXD_MUX of instance: uart4. 111: Select mux mode: ALT7 mux port: CTI_TRIG_OUT0_6 of instance: ect.

Table 263: Register: IOMUXC_SW_MUX_CTL_PAD_EXT_ARMCLK

Offset 0x020c (IOMUXC_SW_MUX_CTL_PAD_EXT_ARMCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 264: Register IOMUXC_SW_MUX_CTL_PAD_EXT_ARMCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad EXT_ARMCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: EXT_ARMCLK. 000: Select mux mode: ALT0 mux port: EXT_ARMCLK of instance: crm. 101: Select mux mode: ALT5 mux port: GPIO[15] of instance: gpio3.

Table 265: Register: IOMUXC_SW_MUX_CTL_PAD_UPLL_BYPCLK

Offset: 0x0210 (IOMUXC_SW_MUX_CTL_PAD_UPLL_BYPCLK) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 266: Register IOMUXC_SW_MUX_CTL_PAD_UPLL_BYPCLK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad UPLL_BYPCLK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: UPLL_BYPCLK. 000: Select mux mode: ALT0 mux port: UPLL_BYPCLK of instance: crm. 101: Select mux mode: ALT5 mux port: GPIO[16] of instance: gpio3.

Table 267: Register: IOMUXC_SW_MUX_CTL_PAD_VSTBY_REQ

Offset 0x0214 (IOMUXC_SW_MUX_CTL_PAD_VSTBY_REQ) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 268: Register IOMUXC_SW_MUX_CTL_PAD_VSTBY_REQ Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad VSTBY_REQ. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 268: Register IOMUXC_SW_MUX_CTL_PAD_VSTBY_REQ Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: VSTBY_REQ. 000: Select mux mode: ALT0 mux port: VSTBY_REQ of instance: crm. 100: Select mux mode: ALT4 mux port: AUD7_TXFS of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[17] of instance: gpio3. 110: Select mux mode: ALT6 mux port: RTS of instance: uart4. NOTE: Pad VSTBY_REQ is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT for mode ALT4. - Config Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT for mode ALT6.

Table 269: Register: IOMUXC_SW_MUX_CTL_PAD_VSTBY_ACK

Offset 0x0218 (IOMUXC_SW_MUX_CTL_PAD_VSTBY_ACK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 270: Register IOMUXC_SW_MUX_CTL_PAD_VSTBY_ACK Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad VSTBY_ACK. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 270: Register IOMUXC_SW_MUX_CTL_PAD_VSTBY_ACK Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: VSTBY_ACK. 000: Select mux mode: ALT0 mux port: VSTBY_ACK of instance: crm. 010: Select mux mode: ALT2 mux port: SS3 of instance: cspi1. 011: Select mux mode: ALT3 mux port: EPITO of instance: epit1. 101: Select mux mode: ALT5 mux port: GPIO[18] of instance: gpio3. NOTE: Pad VSTBY_ACK is involved in Daisy Chain. - Config Register IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT for mode ALT2.

Table 271: Register: IOMUXC_SW_MUX_CTL_PAD_POWER_FAIL

Offset 0x021c (IOMUXC_SW_MUX_CTL_PAD_POWER_FAIL) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 272: Register IOMUXC_SW_MUX_CTL_PAD_POWER_FAIL Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad POWER_FAIL. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved

Table 272: Register IOMUXC_SW_MUX_CTL_PAD_POWER_FAIL Bits Description

Field	Description
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 4 iomux modes to be used for pad: POWER_FAIL. 000: Select mux mode: ALT0 mux port: POWER_FAIL_INT of instance: ioring_ipp_ind_powerfail_inv. 100: Select mux mode: ALT4 mux port: AUD7_RXD of instance: audmux. 101: Select mux mode: ALT5 mux port: GPIO[19] of instance: gpio3. 110: Select mux mode: ALT6 mux port: CTS of instance: uart4. NOTE: Pad POWER_FAIL is involved in Daisy Chain. - Config Register IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT for mode ALT4.

Table 273: Register: IOMUXC_SW_MUX_CTL_PAD_CLKO

Offset 0x0220 (IOMUXC_SW_MUX_CTL_PAD_CLKO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	SION	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 274: Register IOMUXC_SW_MUX_CTL_PAD_CLKO Bits Description

Field	Description
31-5	Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1: Force input path of pad CLKO. 0: Input Path is determined by functionality of the selected mux mode (regular).
3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: CLKO. 000: Select mux mode: ALT0 mux port: CLKO of instance: crm. 101: Select mux mode: ALT5 mux port: GPIO[21] of instance: gpio2.

Table 275: Register: IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0

Offset: 0x0224 (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 276: Register IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE0 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: BOOT_MODE0. 000: Select mux mode: ALT0 mux port: BOOT_MODE[0] of instance: crm. 101: Select mux mode: ALT5 mux port: GPIO[30] of instance: gpio4.

Table 277: Register: IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1

Offset: 0x0228 (IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	MUX_MODE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 278: Register IOMUXC_SW_MUX_CTL_PAD_BOOT_MODE1 Bits Description

Field	Description
31-3	Reserved
2-0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: BOOT_MODE1. 000: Select mux mode: ALT0 mux port: BOOT_MODE[1] of instance: crm. 101: Select mux mode: ALT5 mux port: GPIO[31] of instance: gpio4.

Table 279: Register: IOMUXC_SW_PAD_CTL_PAD_A13

Offset 0x022c (IOMUXC_SW_PAD_CTL_PAD_A13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 280: Register IOMUXC_SW_PAD_CTL_PAD_A13 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A13. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 281: Register: IOMUXC_SW_PAD_CTL_PAD_A14

Offset 0x0230 (IOMUXC_SW_PAD_CTL_PAD_A14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 282: Register IOMUXC_SW_PAD_CTL_PAD_A14 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A14. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 283: Register: IOMUXC_SW_PAD_CTL_PAD_A15

Offset 0x0234 (IOMUXC_SW_PAD_CTL_PAD_A15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 284: Register IOMUXC_SW_PAD_CTL_PAD_A15 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A15. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 285: Register: IOMUXC_SW_PAD_CTL_PAD_A17

Offset 0x0238 (IOMUXC_SW_PAD_CTL_PAD_A17) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 286: Register IOMUXC_SW_PAD_CTL_PAD_A17 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A17. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 287: Register: IOMUXC_SW_PAD_CTL_PAD_A18

Offset 0x023c (IOMUXC_SW_PAD_CTL_PAD_A18) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 288: Register IOMUXC_SW_PAD_CTL_PAD_A18 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A18. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 289: Register: IOMUXC_SW_PAD_CTL_PAD_A19

Offset 0x0240 (IOMUXC_SW_PAD_CTL_PAD_A19) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 290: Register IOMUXC_SW_PAD_CTL_PAD_A19 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A19. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 291: Register: IOMUXC_SW_PAD_CTL_PAD_A20

Offset 0x0244 (IOMUXC_SW_PAD_CTL_PAD_A20) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 292: Register IOMUXC_SW_PAD_CTL_PAD_A20 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A20. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 293: Register: IOMUXC_SW_PAD_CTL_PAD_A21

Offset 0x0248 (IOMUXC_SW_PAD_CTL_PAD_A21) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 294: Register IOMUXC_SW_PAD_CTL_PAD_A21 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A21. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 295: Register: IOMUXC_SW_PAD_CTL_PAD_A23

Offset 0x024c (IOMUXC_SW_PAD_CTL_PAD_A23) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 296: Register IOMUXC_SW_PAD_CTL_PAD_A23 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A23. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 297: Register: IOMUXC_SW_PAD_CTL_PAD_A24

Offset 0x0250 (IOMUXC_SW_PAD_CTL_PAD_A24) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 298: Register IOMUXC_SW_PAD_CTL_PAD_A24 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A24. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 299: Register: IOMUXC_SW_PAD_CTL_PAD_A25

Offset 0x0254 (IOMUXC_SW_PAD_CTL_PAD_A25) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 300: Register IOMUXC_SW_PAD_CTL_PAD_A25 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: A25. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 301: Register: IOMUXC_SW_PAD_CTL_PAD_EB0

Offset 0x0258 (IOMUXC_SW_PAD_CTL_PAD_EB0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 302: Register IOMUXC_SW_PAD_CTL_PAD_EB0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EB0. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 303: Register: IOMUXC_SW_PAD_CTL_PAD_EB1

Offset 0x025c (IOMUXC_SW_PAD_CTL_PAD_EB1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 304: Register IOMUXC_SW_PAD_CTL_PAD_EB1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: EB1. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 305: Register: IOMUXC_SW_PAD_CTL_PAD_OE

Offset 0x0260 (IOMUXC_SW_PAD_CTL_PAD_OE) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 306: Register IOMUXC_SW_PAD_CTL_PAD_OE Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: OE. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 307: Register: IOMUXC_SW_PAD_CTL_PAD_CS4

Offset 0x0264 (IOMUXC_SW_PAD_CTL_PAD_CS4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 308: Register IOMUXC_SW_PAD_CTL_PAD_CS4 Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for pad: CS4. 0: 3.3v Drive 1: 1.8v Drive
12-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CS4. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved

Table 308: Register IOMUXC_SW_PAD_CTL_PAD_CS4 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CS4. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CS4. 0: Slow Slew Rate 1: Fast Slew Rate

Table 309: Register: IOMUXC_SW_PAD_CTL_PAD_CS5

Offset 0x0268 (IOMUXC_SW_PAD_CTL_PAD_CS5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	HYS	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 310: Register IOMUXC_SW_PAD_CTL_PAD_CS5 Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for pad: CS5. 0: 3.3v Drive 1: 1.8v Drive
12-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: CS5. 0: Hysteresis Disabled 1: Hysteresis Enabled

Table 310: Register IOMUXC_SW_PAD_CTL_PAD_CS5 Bits Description

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CS5. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CS5. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CS5. 0: Slow Slew Rate 1: Fast Slew Rate

Table 311: Register: IOMUXC_SW_PAD_CTL_PAD_NF_CE0

Offset 0x026c (IOMUXC_SW_PAD_CTL_PAD_NF_CE0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 312: Register IOMUXC_SW_PAD_CTL_PAD_NF_CE0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: NF_CE0. 0: Pull/Kepper Disabled 1: Enabled
6-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: NF_CE0. 0: Slow Slew Rate 1: Fast Slew Rate

Table 313: Register: IOMUXC_SW_PAD_CTL_PAD_ECB

Offset 0x0270 (IOMUXC_SW_PAD_CTL_PAD_ECB) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	HYS	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 314: Register IOMUXC_SW_PAD_CTL_PAD_ECB Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for pad: ECB. 0: 3.3v Drive 1: 1.8v Drive
12-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: ECB. 0: Hysteresis Disabled 1: Hysteresis Enabled

Table 314: Register IOMUXC_SW_PAD_CTL_PAD_ECB Bits Description

Field	Description
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: ECB. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 315: Register: IOMUXC_SW_PAD_CTL_PAD_LBA

Offset 0x0274 (IOMUXC_SW_PAD_CTL_PAD_LBA) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 316: Register IOMUXC_SW_PAD_CTL_PAD_LBA Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LBA. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 317: Register: IOMUXC_SW_PAD_CTL_PAD_RW

Offset 0x0278 (IOMUXC_SW_PAD_CTL_PAD_RW) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 318: Register IOMUXC_SW_PAD_CTL_PAD_RW Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: RW. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 319: Register: IOMUXC_SW_PAD_CTL_PAD_NFRB

Offset 0x027c (IOMUXC_SW_PAD_CTL_PAD_NFRB) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 320: Register IOMUXC_SW_PAD_CTL_PAD_NFRB Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: NFRB. 0: Pull/Kepper Disabled 1: Enabled
6-0	Reserved

Table 321: Register: IOMUXC_SW_PAD_CTL_PAD_D15

Offset 0x0280 (IOMUXC_SW_PAD_CTL_PAD_D15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 322: Register IOMUXC_SW_PAD_CTL_PAD_D15 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D15. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D15. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D15. 0: Keeper 1: Pull

Table 322: Register IOMUXC_SW_PAD_CTL_PAD_D15 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D15. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D15. 0: Slow Slew Rate 1: Fast Slew Rate

Table 323: Register: IOMUXC_SW_PAD_CTL_PAD_D14

Offset 0x0284 (IOMUXC_SW_PAD_CTL_PAD_D14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 324: Register IOMUXC_SW_PAD_CTL_PAD_D14 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D14. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D14. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D14. 0: Keeper 1: Pull

Table 324: Register IOMUXC_SW_PAD_CTL_PAD_D14 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D14. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D14. 0: Slow Slew Rate 1: Fast Slew Rate

Table 325: Register: IOMUXC_SW_PAD_CTL_PAD_D13

Offset 0x0288 (IOMUXC_SW_PAD_CTL_PAD_D13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 326: Register IOMUXC_SW_PAD_CTL_PAD_D13 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D13. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D13. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D13. 0: Keeper 1: Pull

Table 326: Register IOMUXC_SW_PAD_CTL_PAD_D13 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D13. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D13. 0: Slow Slew Rate 1: Fast Slew Rate

Table 327: Register: IOMUXC_SW_PAD_CTL_PAD_D12

Offset 0x028c (IOMUXC_SW_PAD_CTL_PAD_D12) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 328: Register IOMUXC_SW_PAD_CTL_PAD_D12 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D12. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D12. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D12. 0: Keeper 1: Pull

Table 328: Register IOMUXC_SW_PAD_CTL_PAD_D12 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D12. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D12. 0: Slow Slew Rate 1: Fast Slew Rate

Table 329: Register: IOMUXC_SW_PAD_CTL_PAD_D11

Offset 0x0290 (IOMUXC_SW_PAD_CTL_PAD_D11) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 330: Register IOMUXC_SW_PAD_CTL_PAD_D11 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D11. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D11. 0: Keeper 1: Pull

Table 330: Register IOMUXC_SW_PAD_CTL_PAD_D11 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D11. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D11. 0: Slow Slew Rate 1: Fast Slew Rate

Table 331: Register: IOMUXC_SW_PAD_CTL_PAD_D10

Offset 0x0294 (IOMUXC_SW_PAD_CTL_PAD_D10) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 332: Register IOMUXC_SW_PAD_CTL_PAD_D10 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D10. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D10. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D10. 0: Keeper 1: Pull

Table 332: Register IOMUXC_SW_PAD_CTL_PAD_D10 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D10. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D10. 0: Slow Slew Rate 1: Fast Slew Rate

Table 333: Register: IOMUXC_SW_PAD_CTL_PAD_D9

Offset 0x0298 (IOMUXC_SW_PAD_CTL_PAD_D9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 334: Register IOMUXC_SW_PAD_CTL_PAD_D9 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D9. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D9. 0: Keeper 1: Pull

Table 334: Register IOMUXC_SW_PAD_CTL_PAD_D9 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D9. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D9. 0: Slow Slew Rate 1: Fast Slew Rate

Table 335: Register: IOMUXC_SW_PAD_CTL_PAD_D8

Offset 0x029c (IOMUXC_SW_PAD_CTL_PAD_D8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 336: Register IOMUXC_SW_PAD_CTL_PAD_D8 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: D8. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: D8. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D8. 0: Keeper 1: Pull

Table 336: Register IOMUXC_SW_PAD_CTL_PAD_D8 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: D8. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: D8. 0: Slow Slew Rate 1: Fast Slew Rate

Table 337: Register: IOMUXC_SW_PAD_CTL_PAD_D7

Offset 0x02a0 (IOMUXC_SW_PAD_CTL_PAD_D7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 338: Register IOMUXC_SW_PAD_CTL_PAD_D7 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D7. 0: Keeper 1: Pull
5-0	Reserved

Table 339: Register: IOMUXC_SW_PAD_CTL_PAD_D6

Offset 0x02a4 (IOMUXC_SW_PAD_CTL_PAD_D6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 340: Register IOMUXC_SW_PAD_CTL_PAD_D6 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D6. 0: Keeper 1: Pull
5-0	Reserved

Table 341: Register: IOMUXC_SW_PAD_CTL_PAD_D5

Offset 0x02a8 (IOMUXC_SW_PAD_CTL_PAD_D5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 342: Register IOMUXC_SW_PAD_CTL_PAD_D5 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D5. 0: Keeper 1: Pull
5-0	Reserved

Table 343: Register: IOMUXC_SW_PAD_CTL_PAD_D4

Offset 0x02ac (IOMUXC_SW_PAD_CTL_PAD_D4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 344: Register IOMUXC_SW_PAD_CTL_PAD_D4 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D4. 0: Keeper 1: Pull
5-0	Reserved

Table 345: Register: IOMUXC_SW_PAD_CTL_PAD_D3

Offset 0x02b0 (IOMUXC_SW_PAD_CTL_PAD_D3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 346: Register IOMUXC_SW_PAD_CTL_PAD_D3 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D3. 0: Keeper 1: Pull
5-0	Reserved

Table 347: Register: IOMUXC_SW_PAD_CTL_PAD_D2

Offset 0x02b4 (IOMUXC_SW_PAD_CTL_PAD_D2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 348: Register IOMUXC_SW_PAD_CTL_PAD_D2 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D2. 0: Keeper 1: Pull
5-0	Reserved

Table 349: Register: IOMUXC_SW_PAD_CTL_PAD_D1

Offset 0x02b8 (IOMUXC_SW_PAD_CTL_PAD_D1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 350: Register IOMUXC_SW_PAD_CTL_PAD_D1 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D1. 0: Keeper 1: Pull
5-0	Reserved

Table 351: Register: IOMUXC_SW_PAD_CTL_PAD_D0

Offset 0x02bc (IOMUXC_SW_PAD_CTL_PAD_D0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	PUE	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 352: Register IOMUXC_SW_PAD_CTL_PAD_D0 Bits Description

Field	Description
31-7	Reserved
6 PUE	Pull / Keep Select Field Select one out of next values for pad: D0. 0: Keeper 1: Pull
5-0	Reserved

Table 353: Register: IOMUXC_SW_PAD_CTL_PAD_LD0

Offset 0x02c0 (IOMUXC_SW_PAD_CTL_PAD_LD0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 354: Register IOMUXC_SW_PAD_CTL_PAD_LD0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD0. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD0. 0: Keeper 1: Pull

Table 354: Register IOMUXC_SW_PAD_CTL_PAD_LD0 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 355: Register: IOMUXC_SW_PAD_CTL_PAD_LD1

Offset 0x02c4 (IOMUXC_SW_PAD_CTL_PAD_LD1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 356: Register IOMUXC_SW_PAD_CTL_PAD_LD1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD1. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD1. 0: Keeper 1: Pull

Table 356: Register IOMUXC_SW_PAD_CTL_PAD_LD1 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 357: Register: IOMUXC_SW_PAD_CTL_PAD_LD2

Offset 0x02c8 (IOMUXC_SW_PAD_CTL_PAD_LD2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 358: Register IOMUXC_SW_PAD_CTL_PAD_LD2 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD2. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD2. 0: Keeper 1: Pull

Table 358: Register IOMUXC_SW_PAD_CTL_PAD_LD2 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD2. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD2. 0: Slow Slew Rate 1: Fast Slew Rate

Table 359: Register: IOMUXC_SW_PAD_CTL_PAD_LD3

Offset 0x02cc (IOMUXC_SW_PAD_CTL_PAD_LD3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 360: Register IOMUXC_SW_PAD_CTL_PAD_LD3 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD3. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD3. 0: Keeper 1: Pull

Table 360: Register IOMUXC_SW_PAD_CTL_PAD_LD3 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD3. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 361: Register: IOMUXC_SW_PAD_CTL_PAD_LD4

Offset 0x02d0 (IOMUXC_SW_PAD_CTL_PAD_LD4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 362: Register IOMUXC_SW_PAD_CTL_PAD_LD4 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD4. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD4. 0: Keeper 1: Pull

Table 362: Register IOMUXC_SW_PAD_CTL_PAD_LD4 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD4. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD4. 0: Slow Slew Rate 1: Fast Slew Rate

Table 363: Register: IOMUXC_SW_PAD_CTL_PAD_LD5

Offset 0x02d4 (IOMUXC_SW_PAD_CTL_PAD_LD5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 364: Register IOMUXC_SW_PAD_CTL_PAD_LD5 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD5. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD5. 0: Keeper 1: Pull

Table 364: Register IOMUXC_SW_PAD_CTL_PAD_LD5 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD5. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD5. 0: Slow Slew Rate 1: Fast Slew Rate

Table 365: Register: IOMUXC_SW_PAD_CTL_PAD_LD6

Offset 0x02d8 (IOMUXC_SW_PAD_CTL_PAD_LD6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 366: Register IOMUXC_SW_PAD_CTL_PAD_LD6 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD6. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD6. 0: Keeper 1: Pull

Table 366: Register IOMUXC_SW_PAD_CTL_PAD_LD6 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD6. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD6. 0: Slow Slew Rate 1: Fast Slew Rate

Table 367: Register: IOMUXC_SW_PAD_CTL_PAD_LD7

Offset 0x02dc (IOMUXC_SW_PAD_CTL_PAD_LD7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 368: Register IOMUXC_SW_PAD_CTL_PAD_LD7 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD7. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD7. 0: Keeper 1: Pull

Table 368: Register IOMUXC_SW_PAD_CTL_PAD_LD7 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD7. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD7. 0: Slow Slew Rate 1: Fast Slew Rate

Table 369: Register: IOMUXC_SW_PAD_CTL_PAD_LD8

Offset 0x02e0 (IOMUXC_SW_PAD_CTL_PAD_LD8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 370: Register IOMUXC_SW_PAD_CTL_PAD_LD8 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD8. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD8. 0: Keeper 1: Pull

Table 370: Register IOMUXC_SW_PAD_CTL_PAD_LD8 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD8. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD8. 0: Slow Slew Rate 1: Fast Slew Rate

Table 371: Register: IOMUXC_SW_PAD_CTL_PAD_LD9

Offset 0x02e4 (IOMUXC_SW_PAD_CTL_PAD_LD9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0

Table 372: Register IOMUXC_SW_PAD_CTL_PAD_LD9 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: LD9. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD9. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD9. 0: Keeper 1: Pull

Table 372: Register IOMUXC_SW_PAD_CTL_PAD_LD9 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD9. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD9. 0: Slow Slew Rate 1: Fast Slew Rate

Table 373: Register: IOMUXC_SW_PAD_CTL_PAD_LD10

Offset 0x02e8 (IOMUXC_SW_PAD_CTL_PAD_LD10) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 374: Register IOMUXC_SW_PAD_CTL_PAD_LD10 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD10. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD10. 0: Keeper 1: Pull

Table 374: Register IOMUXC_SW_PAD_CTL_PAD_LD10 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD10. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD10. 0: Slow Slew Rate 1: Fast Slew Rate

Table 375: Register: IOMUXC_SW_PAD_CTL_PAD_LD11

Offset 0x02ec (IOMUXC_SW_PAD_CTL_PAD_LD11) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 376: Register IOMUXC_SW_PAD_CTL_PAD_LD11 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD11. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD11. 0: Keeper 1: Pull

Table 376: Register IOMUXC_SW_PAD_CTL_PAD_LD11 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD11. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD11. 0: Slow Slew Rate 1: Fast Slew Rate

Table 377: Register: IOMUXC_SW_PAD_CTL_PAD_LD12

Offset 0x02f0 (IOMUXC_SW_PAD_CTL_PAD_LD12) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 378: Register IOMUXC_SW_PAD_CTL_PAD_LD12 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD12. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD12. 0: Keeper 1: Pull

Table 378: Register IOMUXC_SW_PAD_CTL_PAD_LD12 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD12. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: LD12. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD12. 0: Slow Slew Rate 1: Fast Slew Rate

Table 379: Register: IOMUXC_SW_PAD_CTL_PAD_LD13

Offset 0x02f4 (IOMUXC_SW_PAD_CTL_PAD_LD13) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 380: Register IOMUXC_SW_PAD_CTL_PAD_LD13 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD13. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD13. 0: Keeper 1: Pull

Table 380: Register IOMUXC_SW_PAD_CTL_PAD_LD13 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD13. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: LD13. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LD13. 0: Slow Slew Rate 1: Fast Slew Rate

Table 381: Register: IOMUXC_SW_PAD_CTL_PAD_LD14

Offset 0x02f8 (IOMUXC_SW_PAD_CTL_PAD_LD14) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 382: Register IOMUXC_SW_PAD_CTL_PAD_LD14 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD14. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD14. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 382: Register IOMUXC_SW_PAD_CTL_PAD_LD14 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: LD14. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 383: Register: IOMUXC_SW_PAD_CTL_PAD_LD15

Offset 0x02fc (IOMUXC_SW_PAD_CTL_PAD_LD15) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 384: Register IOMUXC_SW_PAD_CTL_PAD_LD15 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LD15. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LD15. 0: Keeper 1: Pull

Table 384: Register IOMUXC_SW_PAD_CTL_PAD_LD15 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LD15. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: LD15. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 385: Register: IOMUXC_SW_PAD_CTL_PAD_HSYNC

Offset 0x0300 (IOMUXC_SW_PAD_CTL_PAD_HSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 386: Register IOMUXC_SW_PAD_CTL_PAD_HSYNC Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: HSYNC. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: HSYNC. 0: Keeper 1: Pull

Table 386: Register IOMUXC_SW_PAD_CTL_PAD_HSYNC Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: HSYNC. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: HSYNC. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: HSYNC. 0: Slow Slew Rate 1: Fast Slew Rate

Table 387: Register: IOMUXC_SW_PAD_CTL_PAD_VSYNC

Offset 0x0304 (IOMUXC_SW_PAD_CTL_PAD_VSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 388: Register IOMUXC_SW_PAD_CTL_PAD_VSYNC Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: VSYNC. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: VSYNC. 0: Keeper 1: Pull

Table 388: Register IOMUXC_SW_PAD_CTL_PAD_VSYNC Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: VSYNC. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: VSYNC. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: VSYNC. 0: Slow Slew Rate 1: Fast Slew Rate

Table 389: Register: IOMUXC_SW_PAD_CTL_PAD_LSCLK

Offset 0x0308 (IOMUXC_SW_PAD_CTL_PAD_LSCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Table 390: Register IOMUXC_SW_PAD_CTL_PAD_LSCLK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: LSCLK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: LSCLK. 0: Keeper 1: Pull

Table 390: Register IOMUXC_SW_PAD_CTL_PAD_LSCLK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: LSCLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: LSCLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 391: Register: IOMUXC_SW_PAD_CTL_PAD_OE_ACD

Offset 0x030c (IOMUXC_SW_PAD_CTL_PAD_OE_ACD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 392: Register IOMUXC_SW_PAD_CTL_PAD_OE_ACD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: OE_ACD. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: OE_ACD. 0: Keeper 1: Pull

Table 392: Register IOMUXC_SW_PAD_CTL_PAD_OE_ACD Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: OE_ACD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: OE_ACD. 0: Slow Slew Rate 1: Fast Slew Rate

Table 393: Register: IOMUXC_SW_PAD_CTL_PAD_CONTRAST

Offset 0x0310 (IOMUXC_SW_PAD_CTL_PAD_CONTRAST) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 394: Register IOMUXC_SW_PAD_CTL_PAD_CONTRAST Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CONTRAST. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CONTRAST. 0: Keeper 1: Pull

Table 394: Register IOMUXC_SW_PAD_CTL_PAD_CONTRAST Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CONTRAST. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 395: Register: IOMUXC_SW_PAD_CTL_PAD_PWM

Offset 0x0314 (IOMUXC_SW_PAD_CTL_PAD_PWM) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 396: Register IOMUXC_SW_PAD_CTL_PAD_PWM Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: PWM. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: PWM. 0: Keeper 1: Pull

Table 396: Register IOMUXC_SW_PAD_CTL_PAD_PWM Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: PWM. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for pad: PWM. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 397: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D2

Offset 0x0318 (IOMUXC_SW_PAD_CTL_PAD_CSI_D2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

Table 398: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D2 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D2. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D2. 0: Keeper 1: Pull

Table 398: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D2 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D2. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D2. 0: Slow Slew Rate 1: Fast Slew Rate

Table 399: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D3

Offset 0x031c (IOMUXC_SW_PAD_CTL_PAD_CSI_D3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 400: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D3 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D3. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D3. 0: Keeper 1: Pull

Table 400: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D3 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D3. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D3. 0: Slow Slew Rate 1: Fast Slew Rate

Table 401: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D4

Offset 0x0320 (IOMUXC_SW_PAD_CTL_PAD_CSI_D4) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1

Table 402: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D4 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_D4. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D4. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D4. 0: Keeper 1: Pull

Table 402: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D4 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D4. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D4. 0: Slow Slew Rate 1: Fast Slew Rate

Table 403: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D5

Offset 0x0324 (IOMUXC_SW_PAD_CTL_PAD_CSI_D5) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 404: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D5 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D5. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D5. 0: Keeper 1: Pull

Table 404: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D5 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D5. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D5. 0: Slow Slew Rate 1: Fast Slew Rate

Table 405: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D6

Offset 0x0328 (IOMUXC_SW_PAD_CTL_PAD_CSI_D6) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 406: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D6 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D6. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D6. 0: Keeper 1: Pull

Table 406: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D6 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D6. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_D6. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D6. 0: Slow Slew Rate 1: Fast Slew Rate

Table 407: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D7

Offset 0x032c (IOMUXC_SW_PAD_CTL_PAD_CSI_D7) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Table 408: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D7 Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_D7. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D7. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D7. 0: Keeper 1: Pull

Table 408: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D7 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D7. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_D7. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D7. 0: Slow Slew Rate 1: Fast Slew Rate

Table 409: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D8

Offset 0x0330 (IOMUXC_SW_PAD_CTL_PAD_CSI_D8) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 410: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D8 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D8. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D8. 0: Keeper 1: Pull

Table 410: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D8 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D8. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_D8. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D8. 0: Slow Slew Rate 1: Fast Slew Rate

Table 411: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_D9

Offset 0x0334 (IOMUXC_SW_PAD_CTL_PAD_CSI_D9) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 412: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D9 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_D9. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_D9. 0: Keeper 1: Pull

Table 412: Register IOMUXC_SW_PAD_CTL_PAD_CSI_D9 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_D9. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSI_D9. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_D9. 0: Slow Slew Rate 1: Fast Slew Rate

Table 413: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK

Offset 0x0338 (IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Table 414: Register IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_MCLK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_MCLK. 0: Keeper 1: Pull

Table 414: Register IOMUXC_SW_PAD_CTL_PAD_CSI_MCLK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_MCLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_MCLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 415: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC

Offset 0x033c (IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 416: Register IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_VSYNC. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_VSYNC. 0: Keeper 1: Pull

Table 416: Register IOMUXC_SW_PAD_CTL_PAD_CSI_VSYNC Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_VSYNC. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_VSYNC. 0: Slow Slew Rate 1: Fast Slew Rate

Table 417: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC

Offset 0x0340 (IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 418: Register IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_HSYNC. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_HSYNC. 0: Keeper 1: Pull

Table 418: Register IOMUXC_SW_PAD_CTL_PAD_CSI_HSYNC Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_HSYNC. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_HSYNC. 0: Slow Slew Rate 1: Fast Slew Rate

Table 419: Register: IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK

Offset 0x0344 (IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

Table 420: Register IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: CSI_PIXCLK. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSI_PIXCLK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSI_PIXCLK. 0: Keeper 1: Pull

Table 420: Register IOMUXC_SW_PAD_CTL_PAD_CSI_PIXCLK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSI_PIXCLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSI_PIXCLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 421: Register: IOMUXC_SW_PAD_CTL_PAD_I2C1_CLK

Offset 0x0348 (IOMUXC_SW_PAD_CTL_PAD_I2C1_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 422: Register IOMUXC_SW_PAD_CTL_PAD_I2C1_CLK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C1_CLK. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C1_CLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 422: Register IOMUXC_SW_PAD_CTL_PAD_I2C1_CLK Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C1_CLK. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: I2C1_CLK. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 423: Register: IOMUXC_SW_PAD_CTL_PAD_I2C1_DAT

Offset 0x034c (IOMUXC_SW_PAD_CTL_PAD_I2C1_DAT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 424: Register IOMUXC_SW_PAD_CTL_PAD_I2C1_DAT Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: I2C1_DAT. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: I2C1_DAT. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 424: Register IOMUXC_SW_PAD_CTL_PAD_I2C1_DAT Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: I2C1_DAT. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: I2C1_DAT. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 425: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_MOSI

Offset 0x0350 (IOMUXC_SW_PAD_CTL_PAD_CSPI1_MOSI) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 426: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_MOSI Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_MOSI. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_MOSI. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved

Table 426: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_MOSI Bits Description

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_MOSI. 0: Slow Slew Rate 1: Fast Slew Rate

Table 427: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_MISO

Offset 0x0354 (IOMUXC_SW_PAD_CTL_PAD_CSPI1_MISO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 428: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_MISO Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_MISO. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_MISO. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved

Table 428: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_MISO Bits Description

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_MISO. 0: Slow Slew Rate 1: Fast Slew Rate

Table 429: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS0

Offset 0x0358 (IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Table 430: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_SS0. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: CSPI1_SS0. 0: Keeper 1: Pull

Table 430: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS0 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_SS0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_SS0. 0: Slow Slew Rate 1: Fast Slew Rate

Table 431: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS1

Offset 0x035c (IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 432: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_SS1. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_SS1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 432: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SS1 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: CSPI1_SS1. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_SS1. 0: Slow Slew Rate 1: Fast Slew Rate

Table 433: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_SCLK

Offset 0x0360 (IOMUXC_SW_PAD_CTL_PAD_CSPI1_SCLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 434: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SCLK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_SCLK. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_SCLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved

Table 434: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_SCLK Bits Description

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_SCLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 435: Register: IOMUXC_SW_PAD_CTL_PAD_CSPI1_RDY

Offset 0x0364 (IOMUXC_SW_PAD_CTL_PAD_CSPI1_RDY) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 436: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_RDY Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: CSPI1_RDY. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: CSPI1_RDY. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved

Table 436: Register IOMUXC_SW_PAD_CTL_PAD_CSPI1_RDY Bits Description

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: CSPI1_RDY. 0: Slow Slew Rate 1: Fast Slew Rate

Table 437: Register: IOMUXC_SW_PAD_CTL_PAD_UART1_RXD

Offset 0x0368 (IOMUXC_SW_PAD_CTL_PAD_UART1_RXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 438: Register IOMUXC_SW_PAD_CTL_PAD_UART1_RXD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RXD. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RXD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 439: Register: IOMUXC_SW_PAD_CTL_PAD_UART1_TXD

Offset 0x036c (IOMUXC_SW_PAD_CTL_PAD_UART1_TXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 440: Register IOMUXC_SW_PAD_CTL_PAD_UART1_TXD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_TXD. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_TXD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 441: Register: IOMUXC_SW_PAD_CTL_PAD_UART1_RTS

Offset 0x0370 (IOMUXC_SW_PAD_CTL_PAD_UART1_RTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Table 442: Register IOMUXC_SW_PAD_CTL_PAD_UART1_RTS Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_RTS. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_RTS. 0: Keeper 1: Pull

Table 442: Register IOMUXC_SW_PAD_CTL_PAD_UART1_RTS Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_RTS. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 443: Register: IOMUXC_SW_PAD_CTL_PAD_UART1_CTS

Offset 0x0374 (IOMUXC_SW_PAD_CTL_PAD_UART1_CTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 444: Register IOMUXC_SW_PAD_CTL_PAD_UART1_CTS Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART1_CTS. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART1_CTS. 0: Keeper 1: Pull

Table 444: Register IOMUXC_SW_PAD_CTL_PAD_UART1_CTS Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART1_CTS. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 445: Register: IOMUXC_SW_PAD_CTL_PAD_UART2_RXD

Offset 0x0378 (IOMUXC_SW_PAD_CTL_PAD_UART2_RXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Table 446: Register IOMUXC_SW_PAD_CTL_PAD_UART2_RXD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RXD. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RXD. 0: Keeper 1: Pull

Table 446: Register IOMUXC_SW_PAD_CTL_PAD_UART2_RXD Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RXD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-0	Reserved

Table 447: Register: IOMUXC_SW_PAD_CTL_PAD_UART2_TXD

Offset 0x037c (IOMUXC_SW_PAD_CTL_PAD_UART2_TXD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 448: Register IOMUXC_SW_PAD_CTL_PAD_UART2_TXD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_TXD. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_TXD. 0: Keeper 1: Pull

Table 448: Register IOMUXC_SW_PAD_CTL_PAD_UART2_TXD Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_TXD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_TXD. 0: Slow Slew Rate 1: Fast Slew Rate

Table 449: Register: IOMUXC_SW_PAD_CTL_PAD_UART2_RTS

Offset 0x0380 (IOMUXC_SW_PAD_CTL_PAD_UART2_RTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1

Table 450: Register IOMUXC_SW_PAD_CTL_PAD_UART2_RTS Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_RTS. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_RTS. 0: Keeper 1: Pull

Table 450: Register IOMUXC_SW_PAD_CTL_PAD_UART2_RTS Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_RTS. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_RTS. 0: Slow Slew Rate 1: Fast Slew Rate

Table 451: Register: IOMUXC_SW_PAD_CTL_PAD_UART2_CTS

Offset 0x0384 (IOMUXC_SW_PAD_CTL_PAD_UART2_CTS) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 452: Register IOMUXC_SW_PAD_CTL_PAD_UART2_CTS Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: UART2_CTS. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: UART2_CTS. 0: Keeper 1: Pull

Table 452: Register IOMUXC_SW_PAD_CTL_PAD_UART2_CTS Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: UART2_CTS. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: UART2_CTS. 0: Slow Slew Rate 1: Fast Slew Rate

Table 453: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_CMD

Offset 0x0388 (IOMUXC_SW_PAD_CTL_PAD_SD1_CMD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 454: Register IOMUXC_SW_PAD_CTL_PAD_SD1_CMD Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CMD. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CMD. 0: Keeper 1: Pull

Table 454: Register IOMUXC_SW_PAD_CTL_PAD_SD1_CMD Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CMD. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_CMD. 0: Slow Slew Rate 1: Fast Slew Rate

Table 455: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_CLK

Offset 0x038c (IOMUXC_SW_PAD_CTL_PAD_SD1_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 456: Register IOMUXC_SW_PAD_CTL_PAD_SD1_CLK Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: SD1_CLK. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_CLK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_CLK. 0: Keeper 1: Pull

Table 456: Register IOMUXC_SW_PAD_CTL_PAD_SD1_CLK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_CLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_CLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 457: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0

Offset 0x0390 (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 458: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA0. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA0. 0: Keeper 1: Pull

Table 458: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA0 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA0. 0: Slow Slew Rate 1: Fast Slew Rate

Table 459: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1

Offset 0x0394 (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 460: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA1. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA1. 0: Keeper 1: Pull

Table 460: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA1 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA1. 0: Slow Slew Rate 1: Fast Slew Rate

Table 461: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2

Offset 0x0398 (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 462: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA2. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA2. 0: Keeper 1: Pull

Table 462: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA2 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA2. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA2. 0: Slow Slew Rate 1: Fast Slew Rate

Table 463: Register: IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3

Offset 0x039c (IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1

Table 464: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: SD1_DATA3. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: SD1_DATA3. 0: Keeper 1: Pull

Table 464: Register IOMUXC_SW_PAD_CTL_PAD_SD1_DATA3 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: SD1_DATA3. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: SD1_DATA3. 0: Slow Slew Rate 1: Fast Slew Rate

Table 465: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_ROW0

Offset 0x03a0 (IOMUXC_SW_PAD_CTL_PAD_KPP_ROW0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 466: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_ROW0. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_ROW0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 466: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW0 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_ROW0. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 467: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_ROW1

Offset 0x03a4 (IOMUXC_SW_PAD_CTL_PAD_KPP_ROW1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 468: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_ROW1. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_ROW1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 468: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW1 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_ROW1. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 469: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_ROW2

Offset 0x03a8 (IOMUXC_SW_PAD_CTL_PAD_KPP_ROW2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Table 470: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW2 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_ROW2. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KPP_ROW2. 0: Keeper 1: Pull

Table 470: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW2 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_ROW2. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_ROW2. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 471: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_ROW3

Offset 0x03ac (IOMUXC_SW_PAD_CTL_PAD_KPP_ROW3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Table 472: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW3 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_ROW3. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: KPP_ROW3. 0: Keeper 1: Pull

Table 472: Register IOMUXC_SW_PAD_CTL_PAD_KPP_ROW3 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_ROW3. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_ROW3. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 473: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_COL0

Offset 0x03b0 (IOMUXC_SW_PAD_CTL_PAD_KPP_COL0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 474: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_COL0. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_COL0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 474: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL0 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_COL0. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 475: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_COL1

Offset 0x03b4 (IOMUXC_SW_PAD_CTL_PAD_KPP_COL1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 476: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_COL1. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_COL1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 476: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL1 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_COL1. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 477: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_COL2

Offset 0x03b8 (IOMUXC_SW_PAD_CTL_PAD_KPP_COL2) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 478: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL2 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_COL2. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_COL2. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 478: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL2 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_COL2. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 479: Register: IOMUXC_SW_PAD_CTL_PAD_KPP_COL3

Offset 0x03bc (IOMUXC_SW_PAD_CTL_PAD_KPP_COL3) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 480: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL3 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: KPP_COL3. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: KPP_COL3. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 480: Register IOMUXC_SW_PAD_CTL_PAD_KPP_COL3 Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: KPP_COL3. 0: Open Drain Disabled 1: Open Drain Enabled
2-0	Reserved

Table 481: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_MDC

Offset 0x03c0 (IOMUXC_SW_PAD_CTL_PAD_FEC_MDC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 482: Register IOMUXC_SW_PAD_CTL_PAD_FEC_MDC Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_MDC. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_MDC. 0: Keeper 1: Pull

Table 482: Register IOMUXC_SW_PAD_CTL_PAD_FEC_MDC Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_MDC. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_MDC. 0: Slow Slew Rate 1: Fast Slew Rate

Table 483: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_MDIO

Offset 0x03c4 (IOMUXC_SW_PAD_CTL_PAD_FEC_MDIO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0

Table 484: Register IOMUXC_SW_PAD_CTL_PAD_FEC_MDIO Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: FEC_MDIO. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_MDIO. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_MDIO. 0: Keeper 1: Pull

Table 484: Register IOMUXC_SW_PAD_CTL_PAD_FEC_MDIO Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_MDIO. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_MDIO. 0: Slow Slew Rate 1: Fast Slew Rate

Table 485: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA0

Offset 0x03c8 (IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 486: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_TDATA0. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_TDATA0. 0: Keeper 1: Pull

Table 486: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA0 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_TDATA0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_TDATA0. 0: Slow Slew Rate 1: Fast Slew Rate

Table 487: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA1

Offset 0x03cc (IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 488: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_TDATA1. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_TDATA1. 0: Keeper 1: Pull

Table 488: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TDATA1 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_TDATA1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_TDATA1. 0: Slow Slew Rate 1: Fast Slew Rate

Table 489: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_TX_EN

Offset 0x03d0 (IOMUXC_SW_PAD_CTL_PAD_FEC_TX_EN) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Table 490: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TX_EN Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_TX_EN. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_TX_EN. 0: Keeper 1: Pull

Table 490: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TX_EN Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_TX_EN. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: FEC_TX_EN. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_TX_EN. 0: Slow Slew Rate 1: Fast Slew Rate

Table 491: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA0

Offset 0x03d4 (IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA0) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

Table 492: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA0 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_RDATA0. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_RDATA0. 0: Keeper 1: Pull

Table 492: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA0 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_RDATA0. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: FEC_RDATA0. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_RDATA0. 0: Slow Slew Rate 1: Fast Slew Rate

Table 493: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA1

Offset 0x03d8 (IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 494: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA1 Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_RDATA1. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_RDATA1. 0: Keeper 1: Pull

Table 494: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RDATA1 Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_RDATA1. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: FEC_RDATA1. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_RDATA1. 0: Slow Slew Rate 1: Fast Slew Rate

Table 495: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_RX_DV

Offset 0x03dc (IOMUXC_SW_PAD_CTL_PAD_FEC_RX_DV) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 496: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RX_DV Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_RX_DV. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_RX_DV. 0: Keeper 1: Pull

Table 496: Register IOMUXC_SW_PAD_CTL_PAD_FEC_RX_DV Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_RX_DV. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: FEC_RX_DV. 0: Open Drain Disabled 1: Open Drain Enabled
2-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_RX_DV. 0: Slow Slew Rate 1: Fast Slew Rate

Table 497: Register: IOMUXC_SW_PAD_CTL_PAD_FEC_TX_CLK

Offset 0x03e0 (IOMUXC_SW_PAD_CTL_PAD_FEC_TX_CLK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	HYS	PKE	PUE	PUS		0	0	0	SRE
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 498: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TX_CLK Bits Description

Field	Description
31-9	Reserved
8 HYS	Hyst. Enable Field Select one out of next values for pad: FEC_TX_CLK. 0: Hysteresis Disabled 1: Hysteresis Enabled
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: FEC_TX_CLK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: FEC_TX_CLK. 0: Keeper 1: Pull

Table 498: Register IOMUXC_SW_PAD_CTL_PAD_FEC_TX_CLK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: FEC_TX_CLK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3-1	Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: FEC_TX_CLK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 499: Register: IOMUXC_SW_PAD_CTL_PAD_RTCK

Offset 0x03e4 (IOMUXC_SW_PAD_CTL_PAD_RTCK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	DSE		SRE
W																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0

Table 500: Register IOMUXC_SW_PAD_CTL_PAD_RTCK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: RTCK. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: RTCK. 0: Keeper 1: Pull

Table 500: Register IOMUXC_SW_PAD_CTL_PAD_RTCK Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: RTCK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: RTCK. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: RTCK. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0 SRE	Slew Rate Field Select one out of next values for pad: RTCK. 0: Slow Slew Rate 1: Fast Slew Rate

Table 501: Register: IOMUXC_SW_PAD_CTL_PAD_TDO

Offset 0x03e8 (IOMUXC_SW_PAD_CTL_PAD_TDO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 502: Register IOMUXC_SW_PAD_CTL_PAD_TDO Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for pad: TDO. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 503: Register: IOMUXC_SW_PAD_CTL_PAD_DE_B

Offset 0x03ec (IOMUXC_SW_PAD_CTL_PAD_DE_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 504: Register IOMUXC_SW_PAD_CTL_PAD_DE_B Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for pad: DE_B. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 505: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_A

Offset 0x03f0 (IOMUXC_SW_PAD_CTL_PAD_GPIO_A) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 506: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_A Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_A. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_A. 0: Keeper 1: Pull

Table 506: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_A Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_A. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_A. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_A. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 507: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_B

Offset 0x03f4 (IOMUXC_SW_PAD_CTL_PAD_GPIO_B) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 508: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_B Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B. 0: Keeper 1: Pull

Table 508: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_B Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 509: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_C

Offset 0x03f8 (IOMUXC_SW_PAD_CTL_PAD_GPIO_C) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	PUE	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Table 510: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_C Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_C. 0: Pull/Kepper Disabled 1: Enabled
6 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_C. 0: Keeper 1: Pull

Table 510: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_C Bits Description

Field	Description
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_C. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_C. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_C. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 511: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_D

Offset 0x03fc (IOMUXC_SW_PAD_CTL_PAD_GPIO_D) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 512: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_D Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_D. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_D. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 512: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_D Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_D. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_D. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 513: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_E

Offset 0x0400 (IOMUXC_SW_PAD_CTL_PAD_GPIO_E) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		ODE	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0

Table 514: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_E Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_E. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_E. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up

Table 514: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_E Bits Description

Field	Description
3 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_E. 0: Open Drain Disabled 1: Open Drain Enabled
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_E. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 515: Register: IOMUXC_SW_PAD_CTL_PAD_GPIO_F

Offset 0x0404 (IOMUXC_SW_PAD_CTL_PAD_GPIO_F) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 516: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_F Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_F. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_F. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3	Reserved

Table 516: Register IOMUXC_SW_PAD_CTL_PAD_GPIO_F Bits Description

Field	Description
2-1 DSE	Drive Strength Field Select one out of next values for pad: GPIO_F. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 517: Register: IOMUXC_SW_PAD_CTL_PAD_VSTBY_REQ

Offset 0x0408 (IOMUXC_SW_PAD_CTL_PAD_VSTBY_REQ) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 518: Register IOMUXC_SW_PAD_CTL_PAD_VSTBY_REQ Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: VSTBY_REQ. 0: Pull/Kepper Disabled 1: Enabled
6-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for pad: VSTBY_REQ. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 519: Register: IOMUXC_SW_PAD_CTL_PAD_VSTBY_ACK

Offset 0x040c (IOMUXC_SW_PAD_CTL_PAD_VSTBY_ACK) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 520: Register IOMUXC_SW_PAD_CTL_PAD_VSTBY_ACK Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: VSTBY_ACK. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: VSTBY_ACK. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3	Reserved

Table 520: Register IOMUXC_SW_PAD_CTL_PAD_VSTBY_ACK Bits Description

Field	Description
2-1 DSE	Drive Strength Field Select one out of next values for pad: VSTBY_ACK. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 521: Register: IOMUXC_SW_PAD_CTL_PAD_POWER_FAIL

Offset 0x0410 (IOMUXC_SW_PAD_CTL_PAD_POWER_FAIL) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	PKE	0	PUS		0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 522: Register IOMUXC_SW_PAD_CTL_PAD_POWER_FAIL Bits Description

Field	Description
31-8	Reserved
7 PKE	Pull / Keep Enable Field Select one out of next values for pad: POWER_FAIL. 0: Pull/Kepper Disabled 1: Enabled
6	Reserved
5-4 PUS	Pull Up / Down Config. Field Select one out of next values for pad: POWER_FAIL. 00: 100KOhm Pull Down 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up
3	Reserved

Table 522: Register IOMUXC_SW_PAD_CTL_PAD_POWER_FAIL Bits Description

Field	Description
2-1 DSE	Drive Strength Field Select one out of next values for pad: POWER_FAIL. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 523: Register: IOMUXC_SW_PAD_CTL_PAD_CLKO

Offset 0x0414 (IOMUXC_SW_PAD_CTL_PAD_CLKO) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 524: Register IOMUXC_SW_PAD_CTL_PAD_CLKO Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for pad: CLKO. 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 525: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_MISC

Offset 0x0418 (IOMUXC_SW_PAD_CTL_GRP_DVS_MISC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 526: Register IOMUXC_SW_PAD_CTL_GRP_DVS_MISC Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_MISC (Pads: CSPI1_MISO CSPI1_MOSI CSPI1_RDY CSPI1_SCLK CSPI1_SS0 CSPI1_SS1 FEC_MDC FEC_MDIO FEC_RDATA0 FEC_RDATA1 FEC_RX_DV FEC_TDATA0 FEC_TDATA1 FEC_TX_CLK FEC_TX_EN KPP_COL0 KPP_COL1 KPP_COL2 KPP_COL3 KPP_ROW0 KPP_ROW1 KPP_ROW2 KPP_ROW3 UART1_CTS UART1_RTS UART1_RXD UART1_TXD UART2_CTS UART2_RTS UART2_RXD UART2_TXD). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 527: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_FEC

Offset 0x041c (IOMUXC_SW_PAD_CTL_GRP_DSE_FEC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 528: Register IOMUXC_SW_PAD_CTL_GRP_DSE_FEC Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_FEC (Pads: FEC_MDC FEC_MDIO FEC_RDATA0 FEC_RDATA1 FEC_RX_DV FEC_TDATA0 FEC_TDATA1 FEC_TX_CLK FEC_TX_EN). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 529: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_JTAG

Offset 0x0420 (IOMUXC_SW_PAD_CTL_GRP_DVS_JTAG) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 530: Register IOMUXC_SW_PAD_CTL_GRP_DVS_JTAG Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_JTAG (Pads: DE_B RTCK SJC_MOD TCK TDI TDO TMS TRSTB). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 531: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_NFC

Offset 0x0424 (IOMUXC_SW_PAD_CTL_GRP_DSE_NFC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 532: Register IOMUXC_SW_PAD_CTL_GRP_DSE_NFC Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_NFC (Pads: D0 D1 D10 D11 D12 D13 D14 D15 D2 D3 D4 D5 D6 D7 D8 D9 NFALE NFCLE NFRB NFRE_B NFWE_B NFWP_B NF_CE0). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 533: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_CSI

Offset 0x0428 (IOMUXC_SW_PAD_CTL_GRP_DSE_CSI) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 534: Register IOMUXC_SW_PAD_CTL_GRP_DSE_CSI Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_CSI (Pads: CSI_D2 CSI_D3 CSI_D4 CSI_D5 CSI_D6 CSI_D7 CSI_D8 CSI_D9 CSI_HSYNC CSI_MCLK CSI_PIXCLK CSI_VSYNC). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 535: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_WEIM

Offset 0x042c (IOMUXC_SW_PAD_CTL_GRP_DSE_WEIM) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 536: Register IOMUXC_SW_PAD_CTL_GRP_DSE_WEIM Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_WEIM (Pads: A0 A1 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A2 A20 A21 A22 A23 A24 A25 A3 A4 A5 A6 A7 A8 A9 BCLK CS0 CS1 CS2 CS3 CS4 CS5 EB0 EB1 ECB LBA MA10 OE RW). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 537: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_DDR

Offset 0x0430 (IOMUXC_SW_PAD_CTL_GRP_DSE_DDR) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 538: Register IOMUXC_SW_PAD_CTL_GRP_DSE_DDR Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_DDR (Pads: CAS DQM0 DQM1 RAS SD0 SD1 SD10 SD11 SD12 SD13 SD14 SD15 SD2 SD3 SD4 SD5 SD6 SD7 SD8 SD9 SDBA0 SDBA1 SDCKE0 SDCKE1 SDCLK SDQS0 SDQS1 SDWE). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 539: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_CRM

Offset 0x0434 (IOMUXC_SW_PAD_CTL_GRP_DVS_CRM) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 540: Register IOMUXC_SW_PAD_CTL_GRP_DVS_CRM Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_CRM (Pads: BOOT_MODE0 BOOT_MODE1 CLKO CLK_SEL EXT_ARMCLK GPIO_A GPIO_B GPIO_C GPIO_D GPIO_E GPIO_F POR_B POWER_FAIL RESET_B TEST_MODE UPLL_BYPCLK VSTBY_ACK VSTBY_REQ). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 541: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_KPP

Offset 0x0438 (IOMUXC_SW_PAD_CTL_GRP_DSE_KPP) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 542: Register IOMUXC_SW_PAD_CTL_GRP_DSE_KPP Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_KPP (Pads: KPP_COL0 KPP_COL1 KPP_COL2 KPP_COL3 KPP_ROW0 KPP_ROW1 KPP_ROW2 KPP_ROW3). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 543: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_SDHC1

Offset 0x043c (IOMUXC_SW_PAD_CTL_GRP_DSE_SDHC1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 544: Register IOMUXC_SW_PAD_CTL_GRP_DSE_SDHC1 Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_SDHC1 (Pads: SD1_CLK SD1_CMD SD1_DATA0 SD1_DATA1 SD1_DATA2 SD1_DATA3). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 545: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_LCD

Offset 0x0440 (IOMUXC_SW_PAD_CTL_GRP_DSE_LCD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 546: Register IOMUXC_SW_PAD_CTL_GRP_DSE_LCD Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_LCD (Pads: CONTRAST HSYNC LD0 LD1 LD10 LD11 LD12 LD13 LD14 LD15 LD2 LD3 LD4 LD5 LD6 LD7 LD8 LD9 LSCLK OE_ACD VSYNC). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 547: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_UART

Offset 0x0444 (IOMUXC_SW_PAD_CTL_GRP_DSE_UART) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 548: Register IOMUXC_SW_PAD_CTL_GRP_DSE_UART Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_UART (Pads: UART1_CTS UART1_RTS UART1_RXD UART1_TXD UART2_CTS UART2_RTS UART2_RXD UART2_TXD). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 549: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_NFC

Offset 0x0448 (IOMUXC_SW_PAD_CTL_GRP_DVS_NFC) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 550: Register IOMUXC_SW_PAD_CTL_GRP_DVS_NFC Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_NFC (Pads: D0 D1 D10 D11 D12 D13 D14 D15 D2 D3 D4 D5 D6 D7 D8 D9 NFALE NFCLE NFRB NFRE_B NFWE_B NFWP_B NF_CE0). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 551: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_CSI

Offset 0x044c (IOMUXC_SW_PAD_CTL_GRP_DVS_CSI) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 552: Register IOMUXC_SW_PAD_CTL_GRP_DVS_CSI Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_CSI (Pads: CSI_D2 CSI_D3 CSI_D4 CSI_D5 CSI_D6 CSI_D7 CSI_D8 CSI_D9 CSI_HSYNC CSI_MCLK CSI_PIXCLK CSI_VSYNC I2C1_CLK I2C1_DAT). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 553: Register: IOMUXC_SW_PAD_CTL_GRP_DSE_CSPI1

Offset 0x0450 (IOMUXC_SW_PAD_CTL_GRP_DSE_CSPI1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	DSE		0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 554: Register IOMUXC_SW_PAD_CTL_GRP_DSE_CSPI1 Bits Description

Field	Description
31-3	Reserved
2-1 DSE	Drive Strength Field Select one out of next values for group: DSE_CSPI1 (Pads: CSPI1_MISO CSPI1_MOSI CSPI1_RDY CSPI1_SCLK CSPI1_SS0 CSPI1_SS1). 00: Nominal Drive Strength 01: High Drive Strength 10: Max Drive Strength 11:
0	Reserved

Table 555: Register: IOMUXC_SW_PAD_CTL_GRP_DDRTYPE

Offset 0x0454 (IOMUXC_SW_PAD_CTL_GRP_DDRTYPE) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	DDR_TYPE		0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 556: Register IOMUXC_SW_PAD_CTL_GRP_DDRTYPE Bits Description

Field	Description
31-13	Reserved
12-11 DDR_TYPE	DDR Type Field Select one out of next values for group: DDRTYPE (Pads: A0 A1 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A2 A20 A21 A22 A23 A24 A25 A3 A4 A5 A6 A7 A8 A9 BCLK CAS CS0 CS1 CS2 CS3 DQM0 DQM1 EB0 EB1 LBA MA10 OE RAS RW SD0 SD1 SD10 SD11 SD12 SD13 SD14 SD15 SD2 SD3 SD4 SD5 SD6 SD7 SD8 SD9 SDBA0 SDBA1 SDCKE0 SDCKE1 SDCLK SDQS0 SDQS1 SDWE). 00: Mobile DDR type 01: 3.3v SDRAM type 10: DDR2 type 11:
10-0	Reserved

Table 557: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_SDHC1

Offset 0x0458 (IOMUXC_SW_PAD_CTL_GRP_DVS_SDHC1) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 558: Register IOMUXC_SW_PAD_CTL_GRP_DVS_SDHC1 Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_SDHC1 (Pads: SD1_CLK SD1_CMD SD1_DATA0 SD1_DATA1 SD1_DATA2 SD1_DATA3). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 559: Register: IOMUXC_SW_PAD_CTL_GRP_DVS_LCD

Offset 0x045c (IOMUXC_SW_PAD_CTL_GRP_DVS_LCD) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	DVS	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 560: Register IOMUXC_SW_PAD_CTL_GRP_DVS_LCD Bits Description

Field	Description
31-14	Reserved
13 DVS	Drive Voltage Select Field Select one out of next values for group: DVS_LCD (Pads: CONTRAST HSYNC LD0 LD1 LD10 LD11 LD12 LD13 LD14 LD15 LD2 LD3 LD4 LD5 LD6 LD7 LD8 LD9 LSCLK OE_ACD PWM VSYNC). 0: 3.3v Drive 1: 1.8v Drive
12-0	Reserved

Table 561: Register: IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT

Offset: 0x0460 (IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 562: Register IOMUXC_AUDMUX_P4_INPUT_DA_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_da_amx 0: Selecting Pad: EB1 for Mode: ALT4. 1: Selecting Pad: FEC_MDIO for Mode: ALT2.

Table 563: Register: IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT

Offset: 0x0464 (IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 564: Register IOMUXC_AUDMUX_P4_INPUT_DB_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_db_amx 0: Selecting Pad: EB0 for Mode: ALT4. 1: Selecting Pad: FEC_MDC for Mode: ALT2.

Table 565: Register: IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT

Offset: 0x0468 (IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 566: Register IOMUXC_AUDMUX_P4_INPUT_RXCLK_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_rxclk_amx 0: Selecting Pad: CS4 for Mode: ALT4. 1: Selecting Pad: FEC_TX_EN for Mode: ALT2.

Table 567: Register: IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT

Offset: 0x046c (IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 568: Register IOMUXC_AUDMUX_P4_INPUT_RXFS_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_rxfs_amx 0: Selecting Pad: CS5 for Mode: ALT4. 1: Selecting Pad: FEC_RDATA0 for Mode: ALT2.

Table 569: Register: IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT

Offset: 0x0470 (IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT)

Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 570: Register IOMUXC_AUDMUX_P4_INPUT_TXCLK_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_txclk_amx 0: Selecting Pad: OE for Mode: ALT4. 1: Selecting Pad: FEC_TDATA0 for Mode: ALT2.

Table 571: Register: IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT

Offset: 0x0474 (IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 572: Register IOMUXC_AUDMUX_P4_INPUT_TXFS_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p4_input_txfs_amx 0: Selecting Pad: RW for Mode: ALT4. 1: Selecting Pad: FEC_TDATA1 for Mode: ALT2.

Table 573: Register IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT

Offset: 0x0478 (IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 574: Register IOMUXC_AUDMUX_P7_INPUT_DA_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p7_input_da_amx 0: Selecting Pad: SD1_DATA1 for Mode: ALT3. 1: Selecting Pad: POWER_FAIL for Mode: ALT4.

Table 575: Register: IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT

Offset: 0x047c (IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 576: Register IOMUXC_AUDMUX_P7_INPUT_TXFS_AMX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: audmux, In Pin: p7_input_txfs_amx 0: Selecting Pad: SD1_DATA0 for Mode: ALT3. 1: Selecting Pad: VSTBY_REQ for Mode: ALT4.

Table 577: Register: IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT

Offset: 0x0480 (IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 578: Register IOMUXC_CAN1_IPP_IND_CANRX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: can1, In Pin: ipp_ind_canrx 0: Selecting Pad: FEC_RX_DV for Mode: ALT4. 1: Selecting Pad: GPIO_B for Mode: ALT6.

Table 579: Register: IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT

Offset: 0x0484 (IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 580: Register IOMUXC_CAN2_IPP_IND_CANRX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: can2, In Pin: ipp_ind_canrx 0: Selecting Pad: FEC_RDATA0 for Mode: ALT4. 1: Selecting Pad: GPIO_D for Mode: ALT6.

Table 581: Register: IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT

Offset 0x0488 (IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 582: Register IOMUXC_CSI_IPP_CSI_D_0_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: ipp_csi_d[0] 00: Selecting Pad: LD0 for Mode: ALT2. 01: Selecting Pad: UART1_RTS for Mode: ALT1. 10: Selecting Pad: KPP_ROW2 for Mode: ALT3.

Table 583: Register: IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT

Offset 0x048c (IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 584: Register IOMUXC_CSI_IPP_CSI_D_1_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: ipp_csi_d[1] 00: Selecting Pad: LD1 for Mode: ALT2. 01: Selecting Pad: UART1_CTS for Mode: ALT1. 10: Selecting Pad: KPP_ROW3 for Mode: ALT3.

Table 585: Register: IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT

Offset 0x0490 (IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 586: Register IOMUXC_CSPI1_IPP_IND_SS3_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi1, In Pin: ipp_ind_ss3_b 0: Selecting Pad: NF_CE0 for Mode: ALT1. 1: Selecting Pad: VSTBY_ACK for Mode: ALT2.

Table 587: Register: IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT

Offset 0x0494 (IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 588: Register IOMUXC_CSPI2_IPP_CSPI_CLK_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_cspi_clk_in 0: Selecting Pad: LD14 for Mode: ALT2. 1: Selecting Pad: SD1_DATA0 for Mode: ALT1.

Table 589: Register: IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT

Offset: 0x0498 (IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 590: Register IOMUXC_CSPI2_IPP_IND_DATAREADY_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_ind_dataready_b 0: Selecting Pad: LD15 for Mode: ALT2. 1: Selecting Pad: SD1_DATA1 for Mode: ALT1.

Table 591: Register: IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT

Offset 0x049c (IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 592: Register IOMUXC_CSPI2_IPP_IND_MISO_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_ind_miso 0: Selecting Pad: LD13 for Mode: ALT2. 1: Selecting Pad: SD1_CLK for Mode: ALT1.

Table 593: Register: IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT

Offset: 0x04a0 (IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 594: Register IOMUXC_CSPI2_IPP_IND_MOSI_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_ind_mosi 0: Selecting Pad: LD12 for Mode: ALT2. 1: Selecting Pad: SD1_CMD for Mode: ALT1.

Table 595: Register: IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT

Offset: 0x04a4 (IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 596: Register IOMUXC_CSPI2_IPP_IND_SS0_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_ind_ss0_b 0: Selecting Pad: OE_ACD for Mode: ALT2. 1: Selecting Pad: SD1_DATA2 for Mode: ALT1.

Table 597: Register: IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT

Offset: 0x04a8 (IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 598: Register IOMUXC_CSPI2_IPP_IND_SS1_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi2, In Pin: ipp_ind_ss1_b 0: Selecting Pad: CONTRAST for Mode: ALT2. 1: Selecting Pad: SD1_DATA3 for Mode: ALT1.

Table 599: Register: IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT

Offset: 0x04ac (IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 600: Register IOMUXC_CSPI3_IPP_CSPI_CLK_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_cspi_clk_in 0: Selecting Pad: ECB for Mode: ALT6. 1: Selecting Pad: CSI_D4 for Mode: ALT7.

Table 601: Register: IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT

Offset: 0x04b0 (IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 602: Register IOMUXC_CSPI3_IPP_IND_DATAREADY_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_dataready_b 0: Selecting Pad: LBA for Mode: ALT6. 1: Selecting Pad: CSI_D5 for Mode: ALT7.

Table 603: Register: IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT

Offset 0x04b4 (IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 604: Register IOMUXC_CSPI3_IPP_IND_MISO_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_miso 0: Selecting Pad: CS5 for Mode: ALT6. 1: Selecting Pad: CSI_D3 for Mode: ALT7.

Table 605: Register: IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT

Offset 0x04b8 (IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 606: Register IOMUXC_CSPI3_IPP_IND_MOSI_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_mosi 0: Selecting Pad: CS4 for Mode: ALT6. 1: Selecting Pad: CSI_D2 for Mode: ALT7.

Table 607: Register: IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT

Offset: 0x04bc (IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 608: Register IOMUXC_CSPI3_IPP_IND_SS0_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_ss0_b 0: Selecting Pad: EB0 for Mode: ALT6. 1: Selecting Pad: CSI_D6 for Mode: ALT7.

Table 609: Register: IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT

Offset: 0x04c0 (IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 610: Register IOMUXC_CSPI3_IPP_IND_SS1_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_ss1_b 0: Selecting Pad: EB1 for Mode: ALT6. 1: Selecting Pad: CSI_D7 for Mode: ALT7.

Table 611: Register: IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT

Offset 0x04c4 (IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 612: Register IOMUXC_CSPI3_IPP_IND_SS2_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_ss2_b 0: Selecting Pad: CSI_D8 for Mode: ALT7. 1: Selecting Pad: GPIO_D for Mode: ALT7.

Table 613: Register: IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT

Offset: 0x04c8 (IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 614: Register IOMUXC_CSPI3_IPP_IND_SS3_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: cspi3, In Pin: ipp_ind_ss3_b 0: Selecting Pad: CSI_D9 for Mode: ALT7. 1: Selecting Pad: UART2_CTS for Mode: ALT6.

Table 615: Register: IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT

Offset: 0x04cc (IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 616: Register IOMUXC_ESDHC1_IPP_DAT4_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc1, In Pin: ipp_dat4_in 0: Selecting Pad: D12 for Mode: ALT6. 1: Selecting Pad: UART2_CTS for Mode: ALT1.

Table 617: Register: IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT

Offset: 0x04d0 (IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 618: Register IOMUXC_ESDHC1_IPP_DAT5_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc1, In Pin: ipp_dat5_in 0: Selecting Pad: D13 for Mode: ALT6. 1: Selecting Pad: UART2_RTS for Mode: ALT1.

Table 619: Register: IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT

Offset: 0x04d4 (IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 620: Register IOMUXC_ESDHC1_IPP_DAT6_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc1, In Pin: ipp_dat6_in 0: Selecting Pad: D14 for Mode: ALT6. 1: Selecting Pad: UART2_TXD for Mode: ALT1.

Table 621: Register: IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT

Offset: 0x04d8 (IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 622: Register IOMUXC_ESDHC1_IPP_DAT7_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc1, In Pin: ipp_dat7_in 0: Selecting Pad: D15 for Mode: ALT6. 1: Selecting Pad: UART2_RXD for Mode: ALT1.

Table 623: Register: IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT

Offset: 0x04dc (IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 624: Register IOMUXC_ESDHC2_IPP_CARD_CLK_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_card_clk_in 00: Selecting Pad: LD9 for Mode: ALT6. 01: Selecting Pad: CSI_D7 for Mode: ALT2. 10: Selecting Pad: FEC_MDIO for Mode: ALT1.

Table 625: Register: IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT

Offset: 0x04e0 (IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 626: Register IOMUXC_ESDHC2_IPP_CMD_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_cmd_in 00: Selecting Pad: LD8 for Mode: ALT6. 01: Selecting Pad: CSI_D6 for Mode: ALT2. 10: Selecting Pad: FEC_MDC for Mode: ALT1.

Table 627: Register: IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT

Offset 0x04e4 (IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 628: Register IOMUXC_ESDHC2_IPP_DAT0_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat0_in 00: Selecting Pad: LD10 for Mode: ALT6. 01: Selecting Pad: CSI_MCLK for Mode: ALT2. 10: Selecting Pad: FEC_TDATA0 for Mode: ALT1.

Table 629: Register: IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT

Offset: 0x04e8 (IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 630: Register IOMUXC_ESDHC2_IPP_DAT1_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat1_in 00: Selecting Pad: LD11 for Mode: ALT6. 01: Selecting Pad: CSI_VSYNC for Mode: ALT2. 10: Selecting Pad: FEC_TDATA1 for Mode: ALT1.

Table 631: Register: IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT

Offset: 0x04ec (IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 632: Register IOMUXC_ESDHC2_IPP_DAT2_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat2_in 00: Selecting Pad: LD12 for Mode: ALT6. 01: Selecting Pad: CSI_HSYNC for Mode: ALT2. 10: Selecting Pad: FEC_TX_EN for Mode: ALT1.

Table 633: Register: IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT

Offset 0x04f0 (IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 634: Register IOMUXC_ESDHC2_IPP_DAT3_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat3_in 00: Selecting Pad: LD13 for Mode: ALT6. 01: Selecting Pad: CSI_PIXCLK for Mode: ALT2. 10: Selecting Pad: FEC_RDATA0 for Mode: ALT1.

Table 635: Register: IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT

Offset 0x04f4 (IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 636: Register IOMUXC_ESDHC2_IPP_DAT4_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat4_in 0: Selecting Pad: CSI_D2 for Mode: ALT2. 1: Selecting Pad: FEC_RDATA1 for Mode: ALT2.

Table 637: Register: IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT

Offset 0x04f8 (IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 638: Register IOMUXC_ESDHC2_IPP_DAT5_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat5_in 0: Selecting Pad: CSI_D3 for Mode: ALT2. 1: Selecting Pad: FEC_RX_DV for Mode: ALT2.

Table 639: Register: IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT

Offset: 0x04fc (IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 640: Register IOMUXC_ESDHC2_IPP_DAT6_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat6_in 0: Selecting Pad: CSI_D4 for Mode: ALT2. 1: Selecting Pad: FEC_TX_CLK for Mode: ALT2.

Table 641: Register: IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT

Offset: 0x0500 (IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 642: Register IOMUXC_ESDHC2_IPP_DAT7_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: esdhc2, In Pin: ipp_dat7_in 0: Selecting Pad: CSI_D5 for Mode: ALT2. 1: Selecting Pad: RTCK for Mode: ALT2.

Table 643: Register: IOMUXC_FEC_FEC_COL_SELECT_INPUT

Offset 0x0504 (IOMUXC_FEC_FEC_COL_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 644: Register IOMUXC_FEC_FEC_COL_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_col 00: Selecting Pad: A18 for Mode: ALT7. 01: Selecting Pad: LD9 for Mode: ALT5. 10: Selecting Pad: UART2_RTS for Mode: ALT2.

Table 645: Register: IOMUXC_FEC_FEC_CRS_SELECT_INPUT

Offset 0x0508 (IOMUXC_FEC_FEC_CRS_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 646: Register IOMUXC_FEC_FEC_CRS_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_crs 00: Selecting Pad: A25 for Mode: ALT7. 01: Selecting Pad: CONTRAST for Mode: ALT5. 10: Selecting Pad: SD1_DATA3 for Mode: ALT2.

Table 647: Register: IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT

Offset 0x050c (IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 648: Register IOMUXC_FEC_FEC_RDATA_2_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rdata[2] 00: Selecting Pad: A20 for Mode: ALT7. 01: Selecting Pad: LD11 for Mode: ALT5. 10: Selecting Pad: SD1_CMD for Mode: ALT2.

Table 649: Register: IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT

Offset: 0x0510 (IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 650: Register IOMUXC_FEC_FEC_RDATA_3_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rdata[3] 00: Selecting Pad: A21 for Mode: ALT7. 01: Selecting Pad: LD12 for Mode: ALT5. 10: Selecting Pad: SD1_CLK for Mode: ALT2.

Table 651: Register: IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT

Offset 0x0514 (IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 652: Register IOMUXC_FEC_FEC_RX_CLK_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rx_clk 00: Selecting Pad: A24 for Mode: ALT7. 01: Selecting Pad: LD15 for Mode: ALT5. 10: Selecting Pad: SD1_DATA2 for Mode: ALT2.

Table 653: Register: IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT

Offset 0x0518 (IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 654: Register IOMUXC_FEC_FEC_RX_ER_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: fec, In Pin: fec_rx_er 00: Selecting Pad: A19 for Mode: ALT7. 01: Selecting Pad: LD10 for Mode: ALT5. 10: Selecting Pad: UART2_CTS for Mode: ALT2.

Table 655: Register: IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT

Offset 0x051c (IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 656: Register IOMUXC_I2C2_IPP_SCL_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c2, In Pin: ipp_scl_in 0: Selecting Pad: FEC_RDATA1 for Mode: ALT1. 1: Selecting Pad: GPIO_C for Mode: ALT2.

Table 657: Register: IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT

Offset 0x0520 (IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 658: Register IOMUXC_I2C2_IPP_SDA_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c2, In Pin: ipp_sda_in 0: Selecting Pad: FEC_RX_DV for Mode: ALT1. 1: Selecting Pad: GPIO_D for Mode: ALT2.

Table 659: Register: IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT

Offset 0x0524 (IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 660: Register IOMUXC_I2C3_IPP_SCL_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c3, In Pin: ipp_scl_in 00: Selecting Pad: HSYNC for Mode: ALT2. 01: Selecting Pad: GPIO_A for Mode: ALT4. 10: Selecting Pad: GPIO_E for Mode: ALT1.

Table 661: Register: IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT

Offset 0x0528 (IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 662: Register IOMUXC_I2C3_IPP_SDA_IN_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: i2c3, In Pin: ipp_sda_in 00: Selecting Pad: VSYNC for Mode: ALT2. 01: Selecting Pad: CSPI1_SS1 for Mode: ALT1. 10: Selecting Pad: GPIO_B for Mode: ALT4.

Table 663: Register: IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT

Offset: 0x052c (IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 664: Register IOMUXC_KPP_IPP_IND_COL_4_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[4] 0: Selecting Pad: FEC_RDATA1 for Mode: ALT6. 1: Selecting Pad: GPIO_C for Mode: ALT3.

Table 665: Register: IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT

Offset: 0x0530 (IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 666: Register IOMUXC_KPP_IPP_IND_COL_5_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[5] 0: Selecting Pad: FEC_RX_DV for Mode: ALT6. 1: Selecting Pad: GPIO_D for Mode: ALT3.

Table 667: Register: IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT

Offset 0x0534 (IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 668: Register IOMUXC_KPP_IPP_IND_COL_6_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[6] 0: Selecting Pad: LD14 for Mode: ALT4. 1: Selecting Pad: CSI_D8 for Mode: ALT1.

Table 669: Register: IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT

Offset: 0x0538 (IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 670: Register IOMUXC_KPP_IPP_IND_COL_7_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_col[7] 0: Selecting Pad: LD15 for Mode: ALT4. 1: Selecting Pad: CSI_D9 for Mode: ALT1.

Table 671: Register: IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT

Offset: 0x053c (IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 672: Register IOMUXC_KPP_IPP_IND_ROW_4_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[4] 0: Selecting Pad: FEC_TX_EN for Mode: ALT6. 1: Selecting Pad: GPIO_A for Mode: ALT3.

Table 673: Register: IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT

Offset 0x0540 (IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 674: Register IOMUXC_KPP_IPP_IND_ROW_5_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[5] 0: Selecting Pad: FEC_RDATA0 for Mode: ALT6. 1: Selecting Pad: GPIO_B for Mode: ALT3.

Table 675: Register: IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT

Offset 0x0544 (IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 676: Register IOMUXC_KPP_IPP_IND_ROW_6_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[6] 0: Selecting Pad: LD12 for Mode: ALT4. 1: Selecting Pad: CSI_D6 for Mode: ALT1.

Table 677: Register: IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT

Offset 0x0548 (IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 678: Register IOMUXC_KPP_IPP_IND_ROW_7_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: kpp, In Pin: ipp_ind_row[7] 0: Selecting Pad: LD13 for Mode: ALT4. 1: Selecting Pad: CSI_D7 for Mode: ALT1.

Table 679: Register: IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT

Offset: 0x054c (IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 680: Register IOMUXC_SIM1_PIN_SIM_RCVD1_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim1, In Pin: pin_sim_rcvd1_in 0: Selecting Pad: A19 for Mode: ALT6. 1: Selecting Pad: LD5 for Mode: ALT4.

Table 681: Register: IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT

Offset 0x0550 (IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 682: Register IOMUXC_SIM1_PIN_SIM_SIMPD1_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim1, In Pin: pin_sim_simpd1 0: Selecting Pad: A18 for Mode: ALT6. 1: Selecting Pad: LD4 for Mode: ALT4.

Table 683: Register: IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT

Offset 0x0554 (IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 684: Register IOMUXC_SIM1_SIM_RCVD1_IO_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim1, In Pin: sim_rcvd1_io 0: Selecting Pad: A17 for Mode: ALT6. 1: Selecting Pad: LD3 for Mode: ALT4.

Table 685: Register: IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT

Offset: 0x0558 (IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT)

Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 686: Register IOMUXC_SIM2_PIN_SIM_RCVD1_IN_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim2, In Pin: pin_sim_rcvd1_in 0: Selecting Pad: A25 for Mode: ALT6. 1: Selecting Pad: OE_ACD for Mode: ALT4.

Table 687: Register: IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT

Offset 0x055c (IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 688: Register IOMUXC_SIM2_PIN_SIM_SIMPD1_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim2, In Pin: pin_sim_simpd1 0: Selecting Pad: A24 for Mode: ALT6. 1: Selecting Pad: LSCLK for Mode: ALT4.

Table 689: Register: IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT

Offset 0x0560 (IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT) Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 690: Register IOMUXC_SIM2_SIM_RCVD1_IO_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sim2, In Pin: sim_rcvd1_io 0: Selecting Pad: A23 for Mode: ALT6. 1: Selecting Pad: VSYNC for Mode: ALT4.

Table 691: Register: IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT

Offset: 0x0564 (IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 692: Register IOMUXC_UART3_IPP_UART_RTS_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: ipp_uart_rts_b 0: Selecting Pad: CSPI1_SS1 for Mode: ALT2. 1: Selecting Pad: KPP_ROW2 for Mode: ALT1.

Table 693: Register: IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT

Offset: 0x0568 (IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 694: Register IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart3, In Pin: ipp_uart_rxd_mux 0: Selecting Pad: CSPI1_MOSI for Mode: ALT2. 1: Selecting Pad: KPP_ROW0 for Mode: ALT1.

Table 695: Register: IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT

Offset: 0x056c (IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 696: Register IOMUXC_UART4_IPP_UART_RTS_B_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: ipp_uart_rts_b 00: Selecting Pad: LD10 for Mode: ALT2. 01: Selecting Pad: KPP_COL2 for Mode: ALT1. 10: Selecting Pad: VSTBY_REQ for Mode: ALT6.

Table 697: Register: IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT

Offset: 0x0570 (IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 698: Register IOMUXC_UART4_IPP_UART_RXD_MUX_SELECT_INPUT Bits Description

Field	Description
31-2	Reserved
1-0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart4, In Pin: ipp_uart_rxd_mux 00: Selecting Pad: LD8 for Mode: ALT2. 01: Selecting Pad: KPP_COL0 for Mode: ALT1. 10: Selecting Pad: GPIO_E for Mode: ALT6.

Table 699: Register: IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT

Offset: 0x0574 (IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 700: Register IOMUXC_UART5_IPP_UART_RTS_B_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: ipp_uart_rts_b 0: Selecting Pad: CS5 for Mode: ALT3. 1: Selecting Pad: CSI_D4 for Mode: ALT1.

Table 701: Register: IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT

Offset: 0x0578 (IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 702: Register IOMUXC_UART5_IPP_UART_RXD_MUX_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: uart5, In Pin: ipp_uart_rxd_mux 0: Selecting Pad: LBA for Mode: ALT3. 1: Selecting Pad: CSI_D2 for Mode: ALT1.

Table 703: Register: IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT

Offset: 0x057c (IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 704: Register IOMUXC_USB_TOP_IPP_IND_OTG_USB_OC_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb_top, In Pin: ipp_ind_otg_usb_oc 0: Selecting Pad: D10 for Mode: ALT6. 1: Selecting Pad: GPIO_B for Mode: ALT2.

Table 705: Register: IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT

Offset: 0x0580 (IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT) — Access: User read / write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DAISY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 706: Register IOMUXC_USB_TOP_IPP_IND_UH2_USB_OC_SELECT_INPUT Bits Description

Field	Description
31-1	Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb_top, In Pin: ipp_ind_uh2_usb_oc 0: Selecting Pad: D8 for Mode: ALT6. 1: Selecting Pad: PWM for Mode: ALT6.

Appendix B
Revision History

This appendix provides a list of the major differences between the *MCIMX25 Multimedia Applications Processor Reference Manual*, Revision 1 through Revision 2.

Section, Page	Changes
5.1, 5-1	In Section 5.1, "External Clock Sources." the first paragraph has been removed.
7.3.1.1, 7-3	In Table 7-2, "Fuse Description," the BT_BUS_WIDTH row has been updated for WEIM (NOR) as following:

BT_BUS_WIDTH[1:0]	Selects boot device bus mode.	BT_MEM_CTL[1:0] = WEIM (NOR) 00 Reserved 01 16-bit address/data unmultiplexed interface 10 Reserved 11 Reserved

7.3.1.2, 7-5	In Section 7.3.1.2, "Boot Fuses and Associated Contacts," the second sentence of the first paragraph has been replaced with the following sentences: "The input contacts listed are latched during POR; however, the value is stored once POR is released. It overrides fuse values when the GPIO_BT_SEL fuse is unblown."
7.3.3.3, 7-10	In Section 7.3.3.3, "External Device Selection," the last bulleted item has been replaced with the following text: "EEPROM boot using SPI and I^2C."
7.3.3.3, 7-10	In Section 7.3.3.3, "External Device Selection," the following bulleted item has been included at the end of the bulleted list: "Serial Flash using SPI."
7.3.3.9, 7-20	In Table 7-11, "Flash Header Structure Fields," the fourth row has been updated as follows:

dcd_ptr_ptr double pointer	This pointer must be set to point to the dcd_ptr also contained in the Flash header structure.

7.3.3.10, 7-25	In Section 7.3.3.10, "Device Configuration Data (DCD)," the last two sentences of the last paragraph have been replaced with the following sentences: "This DCD table is an array of structures containing three elements (access type, address and value) preceded by a barker code and length field. The number of DCD structure entries is limited to 60."

7.3.3.10, 7-25 Under the first bulleted list of Section 7.3.3.10, “Device Configuration Data (DCD):”

– The third bulleted item has been replaced with the following text:

“Universal Synchronous Bus (USB)”

– The fifth bulleted item has been replaced with the following text:

“Watchdog (WDOG)”

– The following bulleted item has been added:

“Real Time Integrity Checker (RTIC)”

7.3.3.10, 7-25 In Section 7.3.3.10, “Device Configuration Data (DCD),” the first two sentences of the paragraph after the first bulleted list have been replaced with the following sentences:

“ROM bootstrap has a look-up table of lower and upper addresses for each module allowed to be programmed through the DCD. This table is utilized for verifying any attempt to configure a register.”

7.3.3.10, 7-25 In Section 7.3.3.10, “Device Configuration Data (DCD),” the first line of code in the third typedef structure has been replaced with the following line of code:

```
“UINT32 type; /* Type of pointer (byte=0x1, halfword=0x2, word=0x4) */.”
```

7.3.3.10, 7-26 In Table 7-12, “DCD Block Description,” the first row under “Followed by an array of structures with following fields” has been updated as follows:

DCD access type	Type of pointer (byte=0x1 halfword=0x2, word=0x4) in the following field	4	—

7.3.3.10, 7-28 In Table 7-13, “Valid DCD Address Range,” the eleventh row has been updated as follows:

USB memory	0x53FF4000	0x00004000

7.3.3.10, 7-28 In Table 7-13, “Valid DCD Address Range,” a row has been added at the end of the table as follows:

RTIC register	0x53FEC000	0x00004000

7.3.5, 7-29 In Figure 7-10, “USB/UART Boot Flow,” “90 timer over” has been replaced with “90 sec timer over.”

7.3.5.1, 7-31 In Section 7.3.5.1, “USB,” the first sentence of the first paragraph has been replaced with the following sentence:

“The i.MX25 USB support is provided by the USB module (high performance USB On-The-Go (OTG) functionality, compatible with the USB 2.0 specification, the OTG supplement and the ULPI specification).”

7.3.5.1.1, 7-33 In Table 7-16, “VID/PID and Strings for USB Device Driver,” the third row has been updated as follows:

String Descriptor1 (iManufacturer)	Freescale Semiconductor Inc.	

7.3.5.2, 7-34 In Table 7-17, “UART Configuration,” the last two rows have been updated as follows:

Receive contact	RxD
Transmit contact	TxD

7.7.1, 7-38 In the first paragraph of Section 7.7.1, “High-Assurance Boot (HAB) Security Types,” the two instances of the word “Flash” have been replaced with the term “the boot device”.

7.7.1, 7-38 In Section 7.7.1, “High-Assurance Boot (HAB) Security Types,” the first sentence of the second paragraph has been replaced with the following sentence:

“The CSF provides the signature and certificate information of the piece of software to authenticate.”

7.7.3, 7-43 In Table 7-22, “HAB API Jump Table Addresses,” the last row has been updated as follows:

pu_irom_boot_decision	0x00406968

7.7.4.1, 7-43 In Section 7.7.4.1, “HAB Support for DryIce in Valid/Non-Valid State,” the text has been updated completely.

7.8.5, 7-46 In Section 7.8.5, “Write File,” the following bulleted item has been included at the end of the bulleted list:

“In case of application file type, the Flash header should still be provided. It must be located with a 0x0 offset, meaning right before the application code that will be authenticated (if secure boot), and executed.”

7.8.6, 7-47 In Section 7.8.6, “Completed,” the first sentence of the first paragraph has been replaced with the following sentence:

“The Completed command is required after the Write File command with application file type (0xAA) in order to terminate the protocol.”

8.2, 8-1 In Section 8.2, “Power Saving Methodology,” the first bulleted item under first paragraph has been replaced with the following text:

“Active power savings, including clock gating, software controlled dynamic voltage and frequency scaling (DVFS), dynamic process and temperature compensation (DPTC).”

8.3, 8-3 Section 8.3, “Power-Up and Power-Down Sequence” has been replaced completely.

9.1, 9-1 In Figure 9-1, “1-Wire Module Block Diagram,” the term “OWDAT” has been replaced with term “OWIRE_LINE.”

9.2, 9-2 In Table 9-1, “1-Wire Module Signal,” the first row has been updated as follows:

OWIRE_LINE	I/O	1-Wire bus Requires an external pull-up resistor. The recommended resistor value is specified by the generic 1-Wire device used in a given system.

9.4.1.3, 9-10 In Figure 9-9, “Byte Transfers,” the term “OWDAT” has been replaced with the term “OWIRE_LINE.”

17.3.1, 17-3 Section 17.3.1, “Data Transfer With The Embedded DMA Controllers” has been replaced completely.

17.4.2, 17-9 Section 17.4.2, “End Of Frame Interrupt (EOF_INT)” has been replaced completely.

17.6.3, 17-18 In Table 17-8, “CSI Control Register 1 Field Descriptions,” the row for the bit 8 FCC has been updated as follows:

8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected. **Note:** FCC should only be used when CSI DMA burst size and FIFO water-mark fill level match. If they do not match and FCC bit is set this may cause corrupted images.

17.6.4, 17-21 In Table 17-9, “CSI Control Register 2 Description,” the first and second rows have been updated as follows:

31–30 DMA_BURST_TY PE_RFF	Burst Type of DMA Transfer from RxFIFO. Selects the burst type of DMA transfer from RxFIFO. 00 INCR8 01 INCR4 10 INCR8 11 INCR16 **Note:** The optimal setting is INCR8 so that the CSI burst size matches the ESDRAMC burst size.
29–28 DMA_BURST_TY PE_SFF	Burst Type of DMA Transfer from STATFIFO. Selects the burst type of DMA transfer from STATFIFO. 00 INCR8 01 INCR4 10 INCR8 11 INCR16 **Note:** The optimal setting is INCR8 so that the CSI burst size matches the ESDRAMC burst size.

17.6.8, 17-25 In Section 17.6.8, “CSI RX Count Register (CSIRXCNT),” the second paragraph has been replaced with the following text:

“There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or the embedded DMA controller, the counter value is updated and compared with this register. If

the values match, then an EOF interrupt is triggered. For example, one FIFO word is 4 bytes, so for a picture of 640 × 480 using 2 bytes/pixel, RXCNT should be set to 640 × 480 × 2/4."

18.3.3.2, 18-7 — In Table 18-7, "TXDATA Register Field Description," the two occurrences of BIT_COUNT have been changed to BURST_LENGTH.

20.1.1, 20-3 — In Section 20.1.1, "Features," the last two sub-bulleted items of the second bulleted item have been replaced with the following:

- Support x16 SDR SDRAM (up to 1-Gbit at 133 MHz)
- Support x16 LPDDR SDRAM (up to 1-Gbit at 266 MHz)

20.4.5.1, 20-20 — In Table 20-5, "EMI Outputs to IC Pins," the last 16 rows under the title "SDRAM/LPDDR Dedicated Data Pins" have been removed.

20.4.5.1, 20-20 — In Table 20-5, "EMI Outputs to IC Pins," the last two rows under the title "SDRAM/LPDDR Mask (Byte Enable)" have been removed.

20.4.5.1, 20-20 — In Table 20-5, "EMI Outputs to IC Pins," the seventh, eighth and ninth rows under the title "SDRAM/LPDDR Command Dedicated Pins" have been removed.

Chapter 24, 24-1 — In Chapter 24, "Enhanced SDRAM Controller (ESDRAMC)," the sentence "It supports 64-, 128-, 256-, or 512-Mbit and 1- or 2-Gbit 4-bank SDRAM by two independent chip selects and with up to 64 Mbytes addressable memory per chip select" has been replaced with "It supports 64-,128-, 256-, or 512-Mbit and 1-Gbit 4-bank SDRAM per chip select."

24.1.9, 24-5 — In Section 24.1.9, "Features," the following changes have been done:

- The bullet point "LPDDR/DDR burst length configuration of 8" has been replaced with "LPDDR/DDR2 burst length configuration of 8."
- The bullet point "Supports 64-, 128-, 256-, or 512-Mbit and 1- or 2-Gbit sizes of 4-bank, single data rate, synchronous SDRAM, LPDDR and non-mobile DDR1 devices" has been replaced with "Supports 64-, 128-, 256-, or 512-Mbit and 1-Gbit sizes of 4-bank, single data rate, synchronous SDRAM, LPDDR and non-mobile DDR1 devices."
- The bullet point "Up to 256 Mbytes per chip select" has been replaced with "Up to 128 Mbytes per chip select."
- The bullet point "Support for 16- and 32-bit mobile/low power DDR266 devices" has been replaced with "Support for 16-bit LPDDR/DDR2 devices."
- The bullet point "16- or 32-bit memory data bus width (equal in both chip selects)" has been replaced with "16-bit memory data bus width (equal in both chip selects)."

24.2.1, 24-7 — In Table 24-1, "ESDRAMC Signal Properties," the following changes have been done:

- The first row has been updated as follows:

SDCLK	—	Clock to SDRAM (up to 133MHz)	1	Output

- The second row has been removed.

– The third and fourth rows have been updated as follows:

SDCKE0	—	Clock to SDRAM to SDRAM 0	0	Output
SDCKE1	—	Clock to SDRAM to SDRAM 1	0	Output

– The seventh and eighth rows have been replaced with the following row:

SDATA[15:0]	—	Data bus	0	I/O

– The tenth row has been replaced with the following row:

SDBA[1:0]	—	Bank address	0	Output

– The following rows have been removed:

DQM3	—	Data qualifier mask byte 3 (D[31:24])	0	Output
DQM2	—	Data qualifier mask byte 2 (D[23:16])	0	Output

– The text in the first column of the fifteenth row has been modified to read as $\overline{\text{SWE}}$.

– The first row under the title “Mobile LPDDR signals” has been modified as follows:

SDCLK_B	—	Inverted clock to LPDDR SDRAM	0	Output

– The second to seventh rows under the title “Mobile LPDDR signals” have been removed.

– The last two rows under the title “Mobile LPDDR signals” have been updated as follows:

SDDQS1	—	Data strobe byte 1 (D[15:8])	0	Input
SDQS0	—	Data strobe byte 0 (D[7:0])	0	Input

24.2.2, 24-8 In Table 24-2, “ESDRAMC Detailed Signal Description,” the following changes have been done:

– The first row has been updated as follows:

SDCLK	O	SDRAM clock. The SDCLK output provides the timing reference for the memory devices. All other SDRAM interface signals are referenced to this clock. SDCLK is synchronous to the system clock, but is gated off during low power operating modes when both SDCKE0 and SDCKE1 are negated.

– The second row has been removed.

– The third row has been replaced with the following row:

SDCKE0 SDCKE1	O	SDRAM/MMDR clock enables. Clock enable outputs to the SDRAM memory devices. SDCKE0 corresponds to SDRAM/LPDDR array 0 and SDCKE1 to SDRAM/LPDDR array 1. Activates the memory’s clock input when high, indicating a stable clock is being supplied. Deactivates the memory’s clock input when low. SDCKEx low initiates power-down and self-refresh modes to the SDRAM.

– The fifth row has been updated as follows:

SDATA[15:0]	I/O	Read/write data bus. The 16 data pins are used to transfer data between the enhanced SDRAM controller and memory. Data bit 15 is the most significant bit. Bit 0 is the least significant. 16-bit memory alignment is selectable according to the setting of the SDRAM memory data width (DSIZ field in the ESDCTLn register) (See Section 24.3.3, “Register Descriptions”).

– The seventh row has been updated as follows:

SDBA[1:0]	O	Bank address bus. The bank address pins specify to which bank the current command is targeted. Table 24-17 document which address pins are used as the bank address bus for given device configuration.

– The eighth row has been updated as follows:

DQM1, DQM0	O	Data qualifier mask. During read cycles, DQMx controls the SDRAM data output buffers. DQMx asserted high disables the output buffers leaving them in a high-impedance state. DQMx low allows the data buffers to drive normally. During write cycles, DQMx controls which bytes are written in the SDRAM. DQMx driven low enables a write to the corresponding byte, while DQMx asserted high leaves the byte unchanged. DQM0 corresponds to D[7:0] and DQM1 to D[15:8]. The ESDRAMC takes care of the endian operating mode, and drives the respective DQM strobe required. **Note:** When the controller is used to interface mobile/low power DDR, DQM changes twice each cycle (like the data) and is aligned to DQS edges.

– The ninth row has been updated as follows:

$\overline{\text{SDWE}}$	O	Write enable. Write enable is part of the three bit command field (RAS and CAS make up the other two bits) used by the SDRAM. Generally, $\overline{\text{SDWE}}$ is asserted low if a command transfers data to the memory. A detailed summary of the supported SDRAM commands is provided in Table 24-22.

– The twelfth row has been updated as follows:

SDCLK_B	O	LPDDR SDRAM inverted clock. SDCLK and SDCLK_B are mobile/low power DDR differential clocks. All LPDDRs address and control input signals are sampled on the crossing of the positive edge of SDCLK and negative edge of SDCLK_B. Internal clock signals are derived from SDCLK/SDCLK_B.

– The thirteenth row has been removed.

– The fourteenth row has been updated as follows:

SDQS1, SDQS0	I	Data strobes inputs outputs. During read cycles, SDQSx are generated by the memory devices and are edge aligned with read data. During write cycles, they are generated by the ESDRAMC and are centered with write data. The SDQS delay module is used to delay the SDQSx signal and center it in the data valid window.

– The fifteenth row has been removed.

24.3.3, 24-15 In Table 24-7, “Enhanced SDRAM Control Register (ESDCTLn) Field Descriptions,” the following changes have been done:

– The eighth row has been updated as follows:

17–16 DSIZ	SDRAM memory data width. This field defines the width of the SDRAM memory external data bus. Memories aligned to D[31:16] use DQM2 and DQM3. Memories aligned to D[15:0] use DQM0 and DQM1. 00 Reserved 01 16-bit memory width aligned to D[15:0] (reset value for CSD0) 10 Reserved (reset value for CSD1) 11 Reserved

– A note has been added to the following row:

<table>
<tr><td>11–10
PWDT</td><td>Power-down timer. This field determines whether the SDRAM is placed in a power-down condition after a selectable delay from the last access. The power-down timeout can be triggered on either the absence of an active bank (PWDT=01) or a clock (HCLK) count from the last access (PWDT=10 or 11). Count-based timeouts do not force the SDRAM into an idle condition (for example, any active banks remain open). The power-down timers feature is disabled by hardware reset. See sections Section 24.4.6.3, “Power-Down Modes” and Section 24.4.6.3.2, “Active Power-Down Mode” for a comprehensive description of this operating mode.
<table>
<tr><th>PWDT[1:0]</th><th>PowerDown Timeout</th><th>Memory Device Operating Mode</th></tr>
<tr><td>00</td><td>Disabled (bit field reset value)</td><td>Run mode</td></tr>
<tr><td>01</td><td>Any time no banks are active</td><td>Precharge power-down</td></tr>
<tr><td>10</td><td>64 clocks (HCLK) after completion of last access[1]</td><td>Active power-down</td></tr>
<tr><td>11</td><td>128 clocks (HCLK) after completion of last access[1]</td><td>Active power-down</td></tr>
</table>
[1] This setting cannot be used if the PRCT (precharge timer) is enabled.

Note: When PWDT is enabled and lpmd or dvfs requests are required by system, PWDT must be disabled before the requests are granted. PWDT can only be re-enabled after these modes have been exited.</td></tr>
</table>

24.4, 24-27 In Section 24.4, “Functional Description” the first two sentences of the third paragraph have been replaced with “The enhanced SDRAM controller is designed to support a broad range of JEDEC standard SDRAM/LPDDR configurations including devices of 64-, 128-, 256-, 512-Mbit and 1-Gbit densities. The design support memory devices with data widths of 16 bits.”

24.4, 24-27 In Table 24-13, “JEDEC Standard Single/Double–Data Rate SDRAMs,” 2-GBit column has been removed.

24.4.3.1, 24-43 In Table 24-16, “CPU to SDRAM/LPDDR Translation,” the following changes have been done:

– 32-bit SDRAM column has been removed as 32-bit is not supported in i.MX25.

– The third footnote has been replaced to read as “CPU A0 defines how the data masks are driven.”

24.4.3.1, 24-43 In Section 24.4.3.1, “Multiplexed Address Bus” the following changes have been done:

– The sentence “The enhanced SDRAM controller multiplexed address bus is aligned to the column addresses so that address line A1 always appears on pin MA0 for 16-bit memory devices and A2 always appears on MA0 for 32-bit memory devices” has been replaced with “The enhanced SDRAM controller multiplexed address bus is aligned to the column addresses so that address line A1 always appears on pin MA0 for 16-bit memory devices.”

– The sentence “Table 24-17 summarizes the multiplex options supported by the controller for 16 and 32-bit devices respectively” has been replaced with “Table 24-17 summarizes the multiplex options supported by the controller for 16-bit devices.”

24.4.3.1, 24-43 — Table 24-22, “Address Multiplexing by Column/Row Width for 32-bit Devices,” has been removed as 32-bit is not supported in i.MX25.

24.4.4, 24-45 — In Section 24.4.4, “Multiplexed Address Bus During Precharge or Load Mode Registers Modes” the sentence “For example, to drive the MA10 bit (PRECHARGE ALL command) the CPU A10 bit is required to be set (for both 16- or 32-bit external devices)” has been replaced with “For example, to drive the MA10 bit (PRECHARGE ALL command) the CPU A10 bit is required to be set.”

24.4.5, 24-46 — In Figure 24-31, “Hardware Refresh Timing Diagram,” the following changes have been done:

– Label DQ[31:0] has been replaced with DQ[15:0].

– The NOTE has been replaced with “Because refresh commands (requires all banks to be in idle state, achieved by precharge-all) are issued automatically by the enhanced SDRAM controller at each 32 KHz clock, address bit A10 cannot be shared with other peripherals’ address bus in the system.”

24.4.5, 24-47 — In Figure 24-32, “Hardware Refresh with Pending Bus Cycle Timing Diagram,” the label DQ[31:0] has been replaced with DQ[15:0].

24.4.6.1, 24-48 — In Section 24.4.6.1, “Self-Refresh Mode for SDRAM/LPDDR Devices,” two instances of the term “p_lpmd” have been replaced with “low power mode (p_lpmd).”

24.4.8, 24-62 — In Table 24-21, “SDRAM/LPDDR Burst Access Support,”

– 32-bit column of External Memory Device has been removed as it is not supported in i.MX25.

– The second sentence of the second table footnote has been replaced with the following:

For LPDDR devices only BL=8 is supported.

24.4.8, 24-62 — In Section 24.4.8, “Normal Read/Write Mode,” the first NOTE has been replaced with the following:

NOTE

- ESDRAMC handles the HUNALIGN accesses in the following way. The M3IF module converts the original access address to a word align address to

	the ESDRAMC. The HBSTRB are used by the ESDRAMC to drive the correct value on the DQM signals to the external memory. • 32-bit examples are not relevant for i.MX25 as only the 16-bit mode is supported.
24.4.8.1, 24-84	In Section 24.4.8.1, "SDR Cycle Accurate Enhanced SDRAM Controller Accesses," the first paragraph has been replaced with the following: "This section provides cycle accurate timing diagrams for several ESDRAMC (AMBA AHB-Lite) supported read and write accesses to bit data width SDR memory devices from only one master. The diagrams are provided to emphasize ESDRAMC performance for single master hit (ACTIVE row) requests. The CAS latency for all diagrams is 2 cycles and burst length is set to 4 words for 16-bit memory."
24.4.8, 24-62	The Section 24.4.8.2, "Single Write Word Access to 32-bit Memory," has been removed.
24.4.8.2, 24-89	In Table 24-22, "SDRAM Command Sequence for Burst Accesses," the 32 bit rows of WRAP8, INCR8, WRAP4 and INCR4 have been removed.
24.5.4.3, 24-100	In Section 24.5.4.3, "SDRAM Memory Configuration Examples," the first and second paragraphs have been replaced with the following: "15 different SDRAM (SDR and LPDDR) configurations are demonstrated. These examples are 64 Mbyte, 128 Mbyte, and 256 Mbyte SDRAM memories in single x16 configurations. All single-configuration 16-bit examples are shown connected to the lower half of the data bus."
24.5.4.3, 24-100	Section 24.5.4.3.6, "Dual 64-Mbit (4 Mbits x16) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.7, "Dual 128-Mbit (8 Mbits x16) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.8, "Dual 256-Mbit (16 Mbits x16) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.9, "Single 64-Mbit (2 Mbits x 32) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.10, "Single 128-Mbit (4 Mbits x32) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.11, "Single 256-Mbit (8 Mbits x 32) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.12, "Single 512-Mbit (16 Mbits x 32) SDRAM Configuration," has been removed.
24.5.4.3, 24-100	Section 24.5.4.3.13, "Single 1-Gbit (32 Mbits x 32) SDRAM Configuration," has been removed.

24.5.4.3, 24-100 — Section 24.5.4.3.14, “Single 2-Gbit (64 Mbits x 32) SDRAM Configuration,” has been removed.

24.5.4.3, 24-100 — Section 24.5.4.3.15, “Single 512-Mbit (16 Mbits x 32) Mobile DDR SDRAM Configuration,” has been removed.

Chapter 28, 28-1 — In Figure 28-1, “General Purpose Timer (GPT) Block Diagram,” the label “lpp_ind_clkin” has been removed.

Chapter 28, 28-2 — In Figure 28-2, “GPT Counter Clocks Diagram,” the label “External clock (lpp_ind_clkin)” has been replaced with “Clock off” and the gray square box with the ground symbol.

28.2, 28-2 — In Section 28.2, “Features,” the first bulleted item has been replaced with the following:

– One 32-bit upcounter with clock source selection.

28.4, 28-3 — In Table 28-1, “External Signal Description,” the following Note has been added: “Not all of the capin or cmpout signals are implemented in all GPT instances. Users must check which signals are present in the IOMUX table.”

28.4, 28-3 — Section 28.4.1, “External Clock Input (ipp_ind_clkin),” has been removed.

28.5.2.1, 28-6 — In Table 28-5, “Register Field Descriptions,” the row for 8-6 CLKSRC has been updated as follows:

8-6 CLKSRC	Clock source select. These bits selects which clock goes to the prescaler and subsequently be used to run the GPT counter. This bit field value should only be changed after disabling the GPT by clearing the EN bit in this register. For other programming requirements while changing clock source, see Section 28.7.1, “Change of Clock Source”. 000 No clock 001 ipg_clk 010 ipg_clk_highfreq 011 Reserved 1xx ipg_clk_32k

28.6.1.1, 28-15 — In Section 28.6.1.1, “Clocks,” the last bulleted item has been removed along with the text following it.

28.6.1.5, 28-18 — In Section 28.6.1.5, “Low-Power Mode Behavior,” the first sentence has been replaced with the following text:

“In low-power modes (stop mode and wait mode) the counter continues to run if the clock from the selected clock source is available and if the control bit for that mode is set.”

33.2.9, 33-10 — Section 33.2.9.1, “Pin Configuration for LCDC Signals,” has been removed.

33.2.9.2, 33-16 — In Table 33-3, “TFT Color Channel Assignments,” the following Note has been added: “Need to add 24bpp paralled mode.”

33.3.26, 33-45 In Table 33-29, “LCDC Graphic Window Control Register Description,” the first row has been updated as follows:

31–24 GWAV	Graphic window alpha value. Defines the graphic window alpha value used for alpha blending between graphic window and background plane 0b00000000 Graphic window totally transparent—not displayed on LCD screen ... 0b11111111 Graphic window totally opaque—completely visible on LCD screen

40.4.1, 40-16 In Table 40-6, “JTAG Register Summary,” the following changes have been done:

– CPCR row has been removed.

– PLLBR row has been removed.

– The rows starting from TESTREG row to RWVALCR row have been removed.

45.2, 45-1 In Table 45-1, “Acronyms and Abbreviations,” the first row has been updated as follows:

USB XCVR <=> transceiver	Universal Serial Bus

45.4, 45-3 Section 45.4, “Interface Signal Descriptions,” has been renamed as Section 45.4, “External Signal Descriptions.”

45.4, 45-3 In Section 45.4, “External Signal Descriptions,” the first sentence has been replaced with “Refer Table 45-2 for detail signal list for UTMI-USB-PHY.”

45.4, 45-3 The title of Table 45-3, “Detail port List for UTMI-USB-PHY,” has been replaced with “Detail signal List for UTMI-USB-PHY.”

45.4, 45-3 In Table 45-2, “Detail signal List for UTMI-USB-PHY,” the following changes have been done:

– The first two rows under “Analog Interface Signals” have been removed.

– The third row has been updated as following:

RREF	I	1	Used by Analog block for generating accurate bias current. A 10 Kbyte ± 1% precision resistor is connected on board to ground.

– The fourth row has been removed.

– The fifth and sixth rows have been updated as following:

DP	I/O	NA	USB 2.0 D+ line from cable
DM	I/O	NA	USB 2.0 D- line from cable

– The ninth row has been removed.

– All rows under title “UTMI Signals” have been removed.

– All rows under titles “Testability Interface Signals” and “UTMI Full Speed/Low Speed Serial Interface Signals” have been removed.

Chapter 45, 45-1 Section 45.5, “Detailed Pin Description,” has been removed.

45.5.1.1.3, 45-4 — In Section 45.5.1.1.3, “PLL,” the first paragraph has been replaced with the following:

“PLL is a clock generating module for USB application. It has an fixed frequency Analog-PLL to produce 480 MHz ± 500 ppm for different input frequencies. The clock out represents the phases of the PLL used by the transceiver to transmit and receive data. It requires an input reference of 24 MHz which comes from the 24 MHz oscillator. It is required that the external clock / crystal source jitter be less than 2 ps-rms (in the frequency range100 KHz to10 MHz).”

45.5.1.1.4, 45-4 — In Section 45.5.1.1.4, “Bias,” the first paragraph has been replaced with the following:

“The Bias block generates bias currents to be used in different parts of the XCVR. A off-chip precision resistor of 10 Kbyte ± 1% is connected at RREF for this purpose.”

45.5.1.1.6, 45-4 — In Section 45.5.1.1.6, “Termination Calibration,” the first paragraph has been replaced with the following:

“The FS/HS 45 Ω terminations are calibrated by generating precision current from a off-chip precision (10 Kbyte ± 1%) resistor on the pin RREF. All DP/DM terminations are internal to the PHY.”

45.5.1, 45-3 — Section 45.6.1.2, “Digital Section of the PHY,” has been removed.

Chapter 45, 45-1 — Section 45.7, “Testability,” has been removed.

Chapter 45, 45-1 — Section 45.8, “Power Up Sequencing and Initialization,” has been removed.

Chapter 45, 45-1 — Section 45.10, “Electrical Specs,” has been removed.

47.5.2.6.1, 47-42 — In Section 47.5.2.6.1, “USB Command Register (USBCMD),” the Figure 47-25, “USB Command Register (USBCMD)” has been updated for bit 12 and bit 14.

47.5.2.6.1, 47-42 — In Table 47-22, “USB Command Register Field Descriptions,” the row for the bit 14 has been replaced with the following row:

12	Reserved, read as 0.

47.5.2.6.1, 47-42 — In Table 47-22, “USB Command Register Field Descriptions,” the row for the bit 12 has been replaced with the following row:

14 ATDTW	Add dTD Trip Wire (device mode only). This bit is used as a semaphore to ensure the to proper addition of a new dTD to an active (primed) endpoint’s linked list. This bit is set and cleared by software, and is also cleared by hardware when the state machine is in a state for which adding a dTD to a primed endpoint may go unrecognized. For more information on the use of this bit, see the Device Operational Model section of the USB-HS OTG High-Speed USB On-The-Go DEV reference manual.

47.5.2.6.21, 47-74 In Table 47-42, “Endpoint Control x Registers Field Descriptions,” the following row has been updated:

3–2 RXT	RX endpoint type – Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt Endpoint

49.1, 49-2 In Figure 49-1, “WEIM Block Diagram,” the labels GUARD, STROBE, WR_GUARD, IO_DIR [53:0], BOOT_CFG [5:0], and BIGEND have been removed along with their corresponding arrows.

49.2.1, 49-4 In Table 49-1, “Signal Properties,” the rows for BIGEND, BOOT_CFG, GUARD, IO_DIR, and WR_GUARD signals have been removed.

49.2.2, 49-5 In Table 49-2, “WEIM Detailed Signal Descriptions,” the fourth row has been updated as follows:

BCLK_FB	I	Burst Clock Feedback. This input is used to provide the input data sampling clock at high data speeds. It is an internal feedback from the I/O pin of the BCLK output signal that is used to align the clock used by the memory with the clock used to sample the read data.

49.2.2, 49-5 In Table 49-2, “WEIM Detailed Signal Descriptions,” the rows for BIGEND, BOOT_CFG, GUARD, IO_DIR, STROBE, and WR_GUARD signals have been removed.

49.2.2, 49-7 In Table 49-3, “WEIM Out/In Data in Case AHB Out/In Data is 0xB3B2B1B0,” the rows with Big endian entries have been removed since i.MX25 does not support Big endian mode.

49.3, 49-7 In Section 49.3, “Memory Map and Register Definition,” the statement “but for chip select 0 the registers’ reset states depend on the BOOT_CFG input” has been updated to read as following:

but for chip select 0 the registers’ reset states depend on the boot configuration inputs, for more details, refer Chapter 7, “System Boot."

49.3.1, 49-8 In Table 49-4, “WEIM Memory Map,” the table footnote has been updated as “Some bits are set according to boot configuration input.”

49.3.3, 49-10 In Table 49-7, “WEIM Register Summary,” the row for Base Address 0x0060 (WCR) has been updated as follows:

0x0060 (WCR)	R													ECP5	ECP 4	ECP3	EC P2
	W																
	R	ECP1	EC P0	AUS 5	AUS 4	AUS 3	AUS 2	AUS 1	AUS 0						BC M		MA S
	W																

49.3.3, 49-10 In Table 49-7, “WEIM Register Summary,” the row for Base Address “0xBASE_60 (WCR)” which was appearing twice has been removed.

49.3.3.1, 49-13 In Table 49-8, “Chip Select x Upper Control Register Field Descriptions,” the row for the bits 15-14 CNC has been updated as follows:

15–14 CNC	Chip Select Negation Clock Cycles. This bit field specifies the minimum number of clock cycles a chip select must remain negated after it is negated (but doesn't guarantee negation for back-to-back accesses, it requires EDC using) according to the settings shown below. See examples on the Figure 49-12, and Figure 49-13. CNC has no effect on write accesses when any CSA bit is set. CNC is cleared by a hardware reset. The number of clock cycles of this field can be increased using the CNC2 bit in the appropriate Chip Select Additional Control Register. The number of AHB clock cycles produced by both bit fields is shown in. 00 0 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 01 2 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 10 3 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated. 11 4 Minimum number of AHB clock cycles $\overline{CS}$ must remain negated.

49.3.3.1, 49-17 In Table 49-10, “WSC Bit Field Values,” the rows from seven to fourteen have been updated as follows:

110111	55	55	56	62	54	53
111000	56	56	57	63	55	54
111001	57	57	58	64	56	55
111010	58	58	59	65	57	56
111011	59	59	60	66	58	57
111100	60	60	61	67	59	58
111101	61	61	62	68	60	59
111110	62	62	63	69	61	60

49.3.3.2, 49-18 In Table 49-4, “Chip Select x Lower Control Register,” the text “BOOT_CFG” has been replaced with “boot configuration” in the third footnote.

49.3.3.2, 49-19 In Table 49-11, “Chip Select x Lower Control Register Field Descriptions,” the row corresponding to DSZ has been updated as follows:

10–8 DSZ	Data Port Size. This bit field defines the width of an external device's data port as shown in the Table 49-12. DSZ is set by a hardware reset for CSCR0L by the value of the boot configuration pins. DSZ and MUM (multiplexed mode) affected on data port location as it shown in the Table 49-12. DSZ is cleared by a hardware reset for CSCR1L-CSCR5L.

49.3.3.2, 49-21 In Table 49-12, “DSZ Bit Field Values,” the following changes have been done:

– For MUM=0, the DSZ bits 000, 001, 100, 110, and 111 are set to reserved.

– For MUM=0, the DSZ bits 100 and 111 are set to reserved.

49.3.3.3, 49-22 In Figure 49-5, “Chip Select x Additional Control Register,” the first footnote has been updated to read as follows:

“MUM (bit 15) reset value is determined by the settings of the boot configuration inputs (refer to Chapter 7, “System Boot,” for details on boot configuration signals).”

49.3.3.3, 49-22 In Table 49-13, “Chip Select x Additional Control Register Field Descriptions,” the text “MUM is configured at reset time with the BOOT_CFG[4] (see BOOT_CFG description in Table 49-3)” has been replaced with “MUM is configured at reset time with the boot configuration inputs (refer to Chapter 7, “System Boot” for more details).”

49.3.3.3, 49-22 In Table 49-13, “Chip Select x Additional Control Register Field Descriptions,” the row for the bits 14-13 LAH has been updated as follows:

14–13 LAH	$\overline{\text{LBA}}$ to Address Hold. This bit field determines address hold time after LBA negation for MUM = 1 only. See example on the Figure 49-43 and Figure 49-44. LAH is cleared by a hardware reset for CSCR1A - CSCR5A. For CSCR0A LAH is set to 10 by hardware reset. 00 2 AHB half clock cycles between $\overline{\text{LBA}}$ negation and address invalid. 01 3 AHB half clock cycle between $\overline{\text{LBA}}$ negation and address invalid. 10 4 AHB half clock cycles between $\overline{\text{LBA}}$ negation and address invalid. 11 5 AHB half clock cycles between $\overline{\text{LBA}}$ negation and address invalid.

49.3.3.4, 49-25 In Figure 49-6, “WCR Register,” the RESET values for bits 8 and 0 have been updated to 0. The corresponding footnotes have also been removed.

49.3.3.4, 49-25 In Table 49-16, “WEIM Control Register Field Descriptions,” the following changes have been done:

– In the description for bit 8, the text “This bit is configured at reset time with the BOOT_CFG[5] pin” has been replaced with “By default this bit comes up low (set to 0) after initial power up.”

– The row for bit 1 with reserved status has been removed.

– The row for bit 1 CRER has been updated as follows:

1 CRER	CRE Output Relocation. 0 CRE on ADDR[23] (ADDR[25:24] are used as EB[3:2] in the 32-bit multiplexed mode MUM=1). 1 Invalid setting.

– In the row for bit 0 MAS, the text “This bit is configured at reset time with the BOOT_CFG[32] pin” has been replaced with “By default this bit comes up low (set to 0) after initial power up.”

49.4.1, 49-27 In Section 49.4.1, “Configurable Bus Sizing,” the text “An 8-bit port can reside on DATA_IN/OUT bus bits [31:34], [23:16], [15:8] or [7:0]. A 16 bit port can reside on DATA_IN/OUT bus bits [13:16] or [15:0].” has been replaced with “An 8-bit port can reside on DATA_IN/OUT bus bits [15:8] or [7:0]. A 16 bit port can reside on DATA_IN/OUT bus bits [15:0].”