Freescale Semiconductor
Application Note

AN2693
Rev. 0, 01/2005

New Features of the MPC56*x* READI Nexus Interface Module

by: Randy Dees & Bryan Weston
TECD Applications & Design

The MPC565 was the first publicly available microcontroller to feature a Nexus Interface (IEEE-ISTO 5001™-1999[1] Nexus Class 3). It allows advanced debug capabilities by providing high-speed access to the microcontroller core. These advanced debug capabilities include trace, without requiring extensive external circuitry to monitor an external address bus. The Nexus standard, as implemented on the MPC56*x*, is referred to as the Real-time Embedded Application Development Interface (READI). Revision D of the MPC565, revision D of the MPC561, and revision C of the MPC563 have an updated version of the READI module that includes new features that are described in this application note. These features are not available in earlier versions of these devices.

These changes address issues found with the Nexus standard and the READI implementation of the Nexus standard. The changes include an additional Nexus register accessible by the Nexus tool to enable new options, additional error codes, updated manufacturer

Table of Contents

1. Nexus 5001-1999 (Version 1.0) was the initial release of the standard, however the Nexus 5001-2003 was released in 2003. The MPC56*x* does include some features from the 2003 release.

ID, and enhancements for program trace. This document outlines the new features and the impact on tools and users.

The new features include the following:

- Updated manufacturer ID in the READI DID message/register
- Additional READI register (READI_MC) to select new options
- Nexus message queue doubled in size (from 16 to 32 messages)
- New Public message for Program Trace Resource Full
- Data Trace message change for 16-bit data
- Additional error codes in Error messages
- Selectable enhancements for additional trace support:
  - Instruction count field in Program Synchronization messages
  - Queue flush or queue entry halted on trace overflows
  - Program trace reconstruction without using sync messages and using trace of indirect show cycles only

NOTE

In this application note, the bit numbering in the register definitions of tool mapped registers follows the Nexus IEEE-ISTO 5001-1999 bit numbering convention of MSB = bit 31 and LSB = bit 0, unlike the MPC500 standard of MSB = bit 0 and LSB = bit 31. The bit description tables list the RCPU bit numbering and Nexus bit numbering.

1 Identification of the Updated READI Module

Revision 3.0 of the READI module introduces a number of new features that were not available on earlier revisions of the READI module. Due to these changes, it is necessary for tools and customers to determine which revision of the READI module is used on the MPC56*x* device being debugged. When the READI module is enabled, it transmits the Nexus Device ID (DID) message. Embedded in the DID is the manufacturer of the device, a device number, and a silicon revision level of the device. These can be used to determine the revision of the READI module used on the device. Table 1 shows the different revisions of the READI module, the manufacturer ID, and the silicon version that is transmitted in the DID message for all revisions of each of the MPC56*x* devices. This same information is available in the READI_DID register, which can be read by a Nexus tool.

In versions of the READI module prior to revision 3.0, an incorrect manufacturer identifier is transmitted for Freescale. The 3.0 revision of the READI module corrects this by changing the manufacturer ID from 0x1C to 0xE, a shift of 1 bit.

The device revision is also contained in the READI device ID message/register. The device revision field (REV) holds the major revision level of the device; this is the same as the top nibble of the MASKNUM field of the IMMR.

Table 1. MPC56*x* Mask Sets

Device	Mask Set	Nickname	Part Num	Mask Num	READI Version[1]	READI Manufacturer ID	READI Device REV	Part Number Marking[2]
MPC561	0K27S	0	0x35	0x00	1.0	0x1C	0	—
MPC561	0L40B	A	0x35	0x10	2.0	0x1C	1	—
MPC561	0L17K	C	0x35	0x20	2.1	0x1C	2	MPC561MZP56
MPC561	**L05S**	**D**	**0x35**	**0x30**	**3.0**	**0xE**	**3**	**REV D or MPC561MZP56D**
MPC563	1K73X	0a	0x36	0x00	2.0	0x1C	0	—
MPC563	0L74H	A	0x36	0x10	2.1	0x1C	1	—
MPC563	0L08N	B	0x36	0x20	2.1	0x1C	2	REV B or MPC563MZP56B
MPC563	**0L74H**	**C**	**0X36**	**0x30**	**3.0**	**0xE**	**3**	**REV C**
MPC565	1K85H	0a	0x33	0x00	1.0	0x1C	0	—
MPC565	K13Y	A	0x33	0x10	2.0	0x1C	1	—
MPC565	1K13Y	B	0x33	0x11	2.0	0x1C	1	—
MPC565	2K13Y	C	0x33	0x11	2.0	0x1C	1	MPC565MZP56
MPC565	**L99N**	**D**	**0x33**	**0x20**	**3.0**	**0xE**	**2**	**REV D or MPC565MZP56D**

NOTES:
1 This information is only available from the Design Information and Errata list for a particular mask set.
2 On devices that do not have the mask set marked on the device, the part number marking differentiates between revisions.

2 READI Tool Registers

The READI module has a number of registers that are accessible from the Nexus tool. An additional register, READI_MC, has been added to allow the Nexus tool to control some of the new enhanced features. The READI_MC register is located at address 11 (0xB). The MPC56*x* part powers up in a mode that is compatible with existing silicon. The new features must be enabled in the READI_MC register for the new features to be utilized.

Table 2. Tool Mapped Register Space

Access Opcode	Register	Access Type
8 (0x08)	Device ID Register (DID)[1]	Read Only
10 (0x0A)	Development Control Register (DC)	Read/Write
11 (0x0B)	Mode Control Register (MC)[2]	Read/Write
13 (0x0D)	User Base Address Register (UBA)	Read Only
15 (0x0F)	Read/Write Access Register (RWA)	Read/Write
16 (0x10)	Upload/Download Information Register (UDI)	Read/Write
20 (0x14)	Data Trace Attributes Register 1 (DTA1)	Read/Write
21 (0x15)	Data Trace Attributes Register 2 (DTA2)	Read/Write

NOTES:
1 The hard-coded fields in the READI_DID register have changed.
2 The READI_MC register is only available on revision D of the MPC565 (mask set L99N and later), revision C of the MPC563 (mask set L74H or later), and revision D of the MPC561 (mask set L05S and later).

2.1 Device ID (DID) Register

Accessing the DID register provides key MCU attributes to the development tool. This information is also transmitted via the auxiliary output port upon exit of READI reset ($\overline{\text{RSTI}}$), if $\overline{\text{EVTI}}$ is asserted at $\overline{\text{RSTI}}$ negation. Table 3 shows the definition of bit fields for this register.

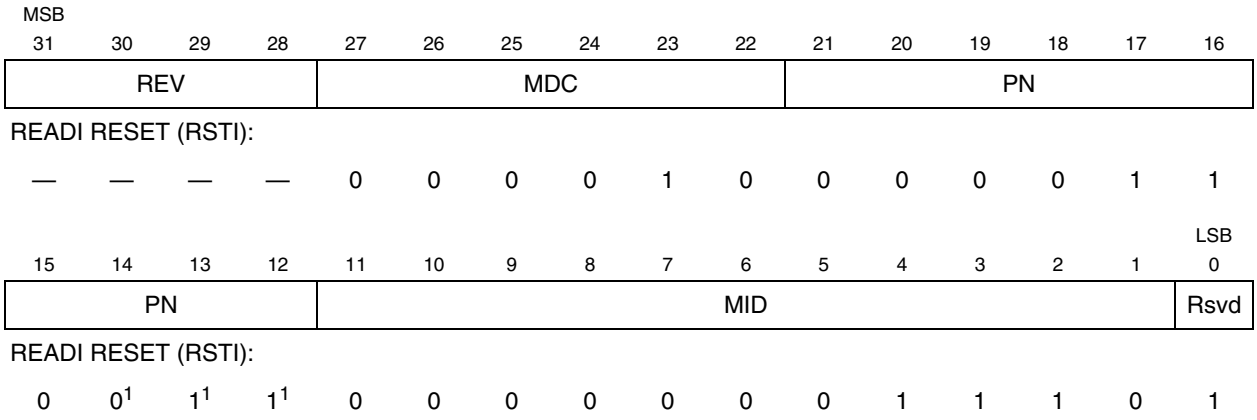

Figure 1. READI Device ID Register (READI_DID)—0x08

NOTES:
1 See “Part Num” column in Table 1. The part number for the MPC561 is 0x35, MPC563 is 0x36, and MPC565 is 0x33.

Table 3. READI_DID Bit Descriptions

RCPU Bits	Nexus Bits	Name	Description
0–3	31–28	REV	READI version number. Contains the revision level of the device. See Table 1.
4–9	27–22	MDC[1]	READI manufacturer design center. Identifies the manufacturer's design center. All of the MPC56*x* parts have a value of 0x2.
10–19	21–12	PN[1]	READI part number. Part number identification field. See Table 1.
20–30	11–1	MID	READI manufacturer ID. Identifies the manufacturer of the device. Freescale's ID is 0x0E.[2]
31	0	—	Reserved. **Note:** For JTAG compliancy, the LSB is reserved and must be set to 1 for IEEE 1149.1 compliancy.

NOTES:
1 The IEEE-ISTO 5001-1999 defines these two fields as a single combined field.
2 In early versions of the READI module, the MID was 0x1C. See Table 1.

2.2 Mode Control (READI_MC) Register

The READI_MC register is used for extended development control of the READI module. This new register allows the control of three new trace options. Figure 2 shows the location of register bits. The READI_MC register allows selection of the new program trace features that are now available. This register should be written by the Nexus tool before writing the READI_DC register. For consistent results, this READI_MC register and the READI_DC register should be written prior to the microcontroller exiting from HRESET.

MSB							LSB
7	6	5	4	3	2	1	0
QTST	Reserved				PTM	QFM	PTSM
READI RESET (RSTI):							
0	0	0	0	0	0	0	0

Figure 2. READI Mode Control (DC) Register (READI_MC)—0x0B

Table 4. READI_MC Bit Descriptions

RCPU Bits	Nexus Bits	Name	Description
0	7	QTST	Enables a factory test mode for structural testing of the queue. This bit can only be written in factory test mode. When set, no trace messages are queued. Users should always write this bit as a zero (0).
5	2	PTM	Program trace mode. Enables an enhanced method of program trace. This mode allows program trace to work with the ISCTL bits of the ICTRL register set to any value except 3. The value of 0x2 in the ICTRL[ISCTL] is recommended for optimal processor performance. The drawback of the enhanced mode is that direct branch messages are never synchronizing, so sync requests must be held until the next indirect branch. 0 Legacy program trace mode 1 Enhanced program trace mode
6	1	QFM	Queue flush mode. Selects whether information in the queue is discarded or transmitted at the time of an overrun. Discarding information allows trace to resume more quickly after an overrun, but makes it difficult to find the cause of the overrun. 0 Information in the queue is removed 1 Trace is stopped until the queue empties.
7	0	PTSM	Program trace sync mode. Indicates if the program trace messages contain the I-CNT packet. 0 Program trace with sync messages do not contain the I-CNT packet. 1 Program trace with sync message contains the I-CNT packet.

2.2.1 Program Trace Sync Mode (READI_MC[PTSM])

Setting the READI_MC[PTSM] bit changes program trace with sync messages to include the number of instructions that have been executed since the last trace message. The instruction count (I-CNT) field indicates the number of instructions. This message format is in line with the new IEEE-ISTO 5001-2003 standard.

In legacy mode (PTSM = 0), it is possible that the Nexus tool will not reconstruct all instructions executed by the processor. The initial version of the Nexus Standard (IEEE-ISTO 5001-1999) requires that every 255^{th} message must be a program sync message. Since the sync message does not indicate the number of instructions that have been executed since that last trace message, it is possible that trace tools cannot determine all of the instructions that were executed by the RCPU. In some cases, this can cause a loss of trace information.

2.2.2 Queue Flush Mode (READI_MC[QFM])

Queue flush mode (QFM) changes the operation of the queue when an overflow condition occurs.

With the QFM bit cleared, operation is the same as existing MPC56*x* silicon. When on overflow of the message queue occurs, all of the contents of the queue are flushed and a queue overflow message is put into the Nexus message queue. This mode allows trace to resume immediately after an overrun occurs, but it prevents the user or tool from determining the cause of the overflow condition.

Setting the READI_MC[QFM] bit prevents the message queue from being cleared, and holds new messages off until there is room for them in the queue. If an incoming message overflow occurs, the type

of overflow is signalled in the error message (for example, Ownership Trace overrun, Program Trace overrun, or Data Trace overrun). If multiple types of messages are lost, however, the generic OT/PT/DT overrun error code (0b00111) will be sent in the error message. (For more information, see Table 11.)

2.2.3 Program Trace Mode (READI_MC[PTM])

The program trace mode (PTM) bit enables an enhanced method of program trace. This mode allows program trace to work with the ISCTL bits of the ICTRL register set to any value except 0x3 (0b11, no instruction show cycle mode). The value of 0x2 (0b10, indirect branches only show cycle mode) is recommended for optimal processor performance. The drawback of this mode is direct branch messages are never synchronized; therefore, sync requests must be held until the next indirect branch.

The legacy mode (READI_MC[PTM] = 0) requires that the ISCTL bits of the ICTRL must be set to 1 (0b01, all changes of flow show cycles mode) to allow sufficient information for complete program trace.

3 Nexus Public and Vendor Defined Messages

The Nexus tool uses public messages to communicate between the Nexus tool and the Nexus target system. Table 5 shows the all of the public messages and Table 6 shows all of the vendor-defined messages that are recognized by the MPC56*x* devices. The modified or new messages are outlined in bold.

Table 5. Public Messages Supported

Message Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Device ID	6	6	Fixed	TCODE number = 1	From Device
	32	32	Fixed	Device ID information	
Ownership Trace Message	6	6	Fixed	TCODE number = 2	From Device
	32	32	Fixed	Task/Process ID tag	
Program Trace—Direct Branch Message	6	6	Fixed	TCODE number = 3	From Device
	1	8	Variable	number of sequential instructions executed since last taken branch	
Program Trace—Indirect Branch Message	6	6	Fixed	TCODE number = 4	From Device
	1	8	Variable	number of sequential instructions executed since last taken branch	
	1	23	Variable	unique portion of the target address for taken branches and exceptions	
Data Trace—Data Write Message	6	6	Fixed	TCODE number = 5	From Device
	1	25	Variable	unique portion of the data write address	
	8	32	Variable	data write value (8, 16, 32 bits)[1]	

Table 5. Public Messages Supported (continued)

Message Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Data Trace—Data Read Message	6	6	Fixed	TCODE number = 6	From Device
	1	25	Variable	unique portion of the data read address	
	8	32	Variable	data read value (8, 16, 32 bits)[1]	
Error Message[2]	6	6	Fixed	TCODE number = 8	From Device
	5	5	Fixed	error code	
Program Trace Correction Message	6	6	Fixed	TCODE number = 10 (0xA)	From Device
	1	8	Variable	correcting the number of instructions in the trace	
Program Trace—Direct Branch Synchronization Message (PTSM=0)	6	6	Fixed	TCODE number = 11 (0xB)	From Device
	1	1	Variable	number of program trace messages cancelled	
	1	23	Variable	full target address	
Program Trace—Direct Branch Synchronization Message (PTSM==1)	6	6	Fixed	TCODE number = 11 (0xB)	From Device
	1	8	Variable	number of sequential instructions (I-CNT) executed since last taken branch	
	1	23	Variable	full target address	
Program Trace—Indirect Branch Synchronization Message (PTSM=0)	6	6	Fixed	TCODE number = 12 (0xC)	From Device
	1	1	Variable	number of program trace messages cancelled	
	1	23	Variable	full target address	
Program Trace—Indirect Branch Synchronization Message (PTSM=1)	6	6	Fixed	TCODE number = 12 (0xC)	From Device
	1	8	Variable	number of sequential instructions (I-CNT) executed since last taken branch	
	1	23	Variable	full target address	
Data Trace—Data Write Synchronization Message	6	6	Fixed	TCODE number = 13 (0xD)	From Device
	1	1	Variable	number of messages canceled	
	1	25	Variable	full target address	
	8	32	Variable	data write value (8, 16, 32 bits)[1]	
Data Trace—Data Read Synchronization Message	6	6	Fixed	TCODE number = 14 (0xE)	From Device
	1	1	Variable	number of messages canceled	
	1	25	Variable	full target address	
	8	32	Variable	data read value (8, 16, 32 bits)[1]	

Table 5. Public Messages Supported (continued)

Message Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Watchpoint Message	6	6	Fixed	TCODE number = 15 (0xF)	From Device
	6	6	Fixed	number indicating watchpoint source	
Auxiliary Access—Device Ready for Upload/Download Message	6	6	Fixed	TCODE number = 16 (0x10)	From Device
Auxiliary Access—Upload Request Message	6	6	Fixed	TCODE number = 17 (0x11)	From Tool
	8	8	Fixed	opcode to enable selected configuration, status or data upload from MCU	
Auxiliary Access—Download Request Message	6	6	Fixed	TCODE number = 18 (0x12)	From Tool
	8	8	Fixed	opcode to enable selected configuration or data download to MCU	
	8	80	Variable	Depending upon opcode selected for download, information to be downloaded to device will vary.	
Auxiliary Access—Upload/Download Information Message	6	6	Fixed	TCODE number = 19 (0x13)	From Device / Tool
	8	80	Variable	1). For an access, depending on word size selected (SZ field in RWA register), variable-length packets of information (10, 18, or 34 bits) will be uploaded/downloaded from/to device. 2). Depending upon opcode selected for upload from internal READI registers, information to be uploaded to the device will vary.	
Resource Full Message	6	6	Fixed	TCODE number = 27 (0x1B)	From Device
	1	4	Variable	resource code	

NOTES:
1 On newer revisions of silicon, 16 bit data fields are padded with an extra 8 bits in Full Port Mode.
2 Refer to Table 11 for the error code encoding for error messages.

Table 6. Vendor-Defined Messages Supported

TCODE Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Dev Port Access—DSDI Data Message	6	6	Fixed	TCODE number = 56 (0x38)	From Tool
	10	35	Variable	BDM Development Serial Data In (DSDI)	
Dev Port Access—DSDO Data Message	6	6	Fixed	TCODE number = 57 (0x39)	From Device
	10	35	Variable	BDM Development Serial Data Out (DSDO)	
Dev Port Access—BDM Status Message	6	6	Fixed	TCODE number = 58 (0x3A)	From Device
	1	1	Fixed	BDM status	
Program Trace—Indirect Branch Message With Compressed Code	6	6	Fixed	TCODE number = 59 (0x3B)	From Device
	1	8	Variable	Number of sequential instructions executed since last taken branch	
	6	6	Fixed	Bit address	
	1	23	Variable	Unique portion of the target address for taken branches and exceptions (compressed code)	
Program Trace—Direct Branch Synchronization Message With Compressed Code (PTSM=0)	6	6	Fixed	TCODE number = 60 (0x3C)	From Device
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	
Program Trace—Direct Branch Synchronization Message With Compressed Code (PTSM=1)	6	6	Fixed	TCODE number = 60 (0x3C)	From Device
	1	8	Variable	Number of sequential instructions (I-CNT) executed since last taken branch	
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	
Program Trace—Indirect Branch Synchronization Message With Compressed Code (PTSM=0)	6	6	Fixed	TCODE number = 61 (0x3D)	From Device
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	
Program Trace—Indirect Branch Synchronization Message With Compressed Code (PTSM=1)	6	6	Fixed	TCODE number = 61 (0x3D)	From Device
	1	8	Variable	Number of sequential instructions (I-CNT) executed since last taken branch	
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	

3.1 Updated Program Trace Messages

By setting the PTSM field in the READI_MC register, the messages cancelled field of Program Trace with Synchronization messages are changed to include the number of instructions that have been executed (I-CNT) instead.

In addition, tool vendors will see a change in the number of instructions reported when an exception occurs. The number of instructions will always be one instruction less when using READI version 3.0 than was previously reported in READI module versions prior to 3.0.

The following table shows the comparison of the Program Trace Synchronization messages that are different based on the setting of the READI_MC[PTSM] bit.

Table 7. Message Field Size Comparison[1]

Message	TCODE	Field # 1	Field # 2	Field # 3	Max Size[2]	Min. Size[3]
Program Trace — Direct Branch Synchronization Message (PTSM==0)	11 (0xB)	Variable Max = 1 Min = 1	Variable Max = 23 Min = 1	NA	30 bits	8 bits
Program Trace — Direct Branch Synchronization Message (PTSM==1)	11 (0xB)	Variable Max =8 Min = 1	Variable Max = 23 Min = 1	NA	37 bits	8 bits
Program Trace — Indirect Branch Synchronization Message (PTSM==0)	12 (0xC)	Variable Max = 1 Min = 1	Variable Max = 23 Min = 1	NA	30 bits	8 bits
Program Trace — Indirect Branch Synchronization Message (PTSM==1)	12 (0xC)	Variable Max = 8 Min = 1	Variable Max = 23 Min = 1	NA	37 bits	8 bits
Resource Full Message	27 (0x1B)	Variable Max = 4 Min = 1	NA	NA	10 bits	7 bits
Program Trace — Direct Branch Synchronization Message With Compressed Code (PTSM==0)	60 (0x3C)	Fixed = 6	Variable Max = 23 Min = 1	NA	35 bits	13 bits
Program Trace — Direct Branch Synchronization Message With Compressed Code (PTSM==1)	60 (0x3C)	Variable Max = 8 Min = 1	Fixed = 6	Variable Min = 1 Max = 23	43 bits	14 bits

Table 7. Message Field Size Comparison[1] (continued)

Message	TCODE	Field # 1	Field # 2	Field # 3	Max Size[2]	Min. Size[3]
Program Trace—Indirect Branch Synchronization Message With Compressed Code (PTSM==0)	61 (0x3D)	Fixed = 6	Variable Max = 23 Min = 1	NA	35 bits	13 bits
Program Trace—Indirect Branch Synchronization Message With Compressed Code (PTSM==1)	61 (0x3D)	Variable Max = 8 Min = 1	Fixed = 6	Variable Min = 1 Max = 23	43 bits	14 bits

NOTES:
1 Only modified messages are shown.
2 Maximum information size. The actual number of bits transmitted is dependant on the number of MDO pins.
3 Minimum information size. The actual number of bits transmitted is dependent on the number of MDO pins.

3.1.1 Direct Branch Synchronization Message

The program trace direct branch synchronization message has the following formats, depending on how the PTSM bit in the READI_MC register is configured:

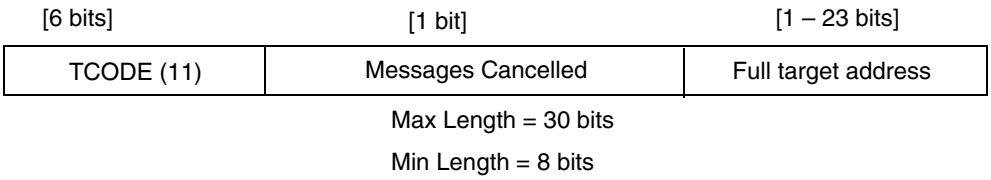

Figure 3. Direct Branch Synchronization Message Format (PTSM==0)

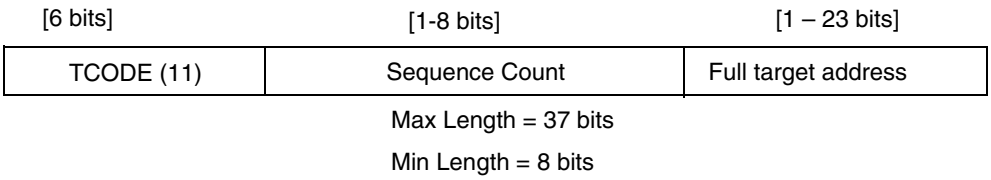

Figure 4. Direct Branch Synchronization Message Format (PTSM==1)

3.1.2 Indirect Branch Synchronization Message

The program trace indirect branch synchronization message has the following formats depending on how the PTSM bit in the READI_MC register is configured:

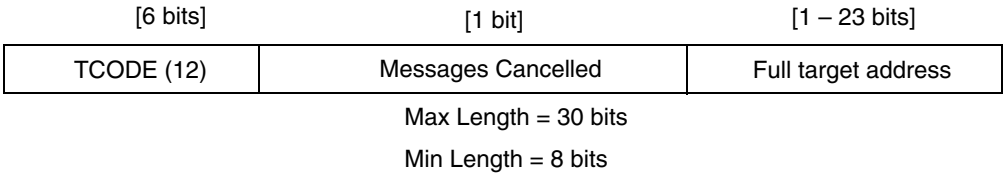

Figure 5. Indirect Branch Synchronization Message Format (PTSM==0)

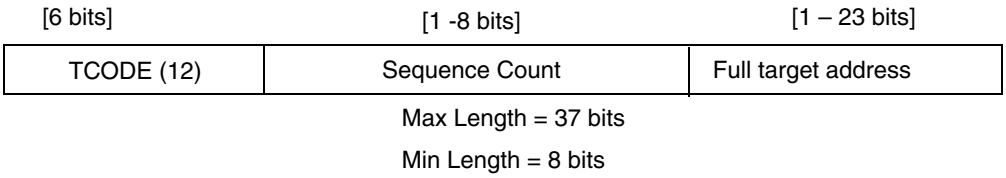

Figure 6. Indirect Branch Synchronization Message Format (PTSM==1)

3.1.3 Direct Branch Synchronization Message With Compressed Code

For compressed code support, an additional 6 bits indicate the starting bit address within the word of the compressed instruction. The program trace direct branch synchronization with compressed code message has the following formats, depending on the setting of the PTSM bit in the READI_MC register:

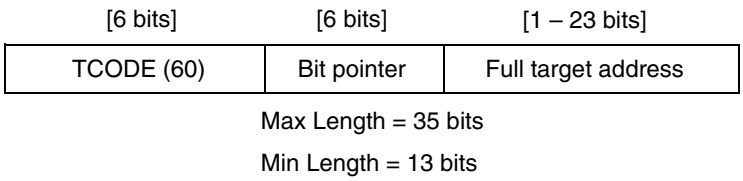

Figure 7. Direct Branch Synchronization Message Format With Compressed Code (PTSM==0)

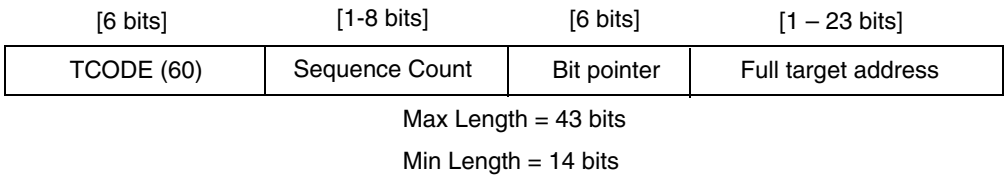

Figure 8. Direct Branch Synchronization Message Format With Compressed Code (PTSM==1)

3.1.4 Indirect Branch Synchronization Message with Compressed Code

For compressed code support, an additional 6 bits indicate the starting bit address within the word of the compressed instruction. The program trace indirect branch synchronization with compressed code message has the following formats depending on the setting of the PTSM bit in the READI_MC register:

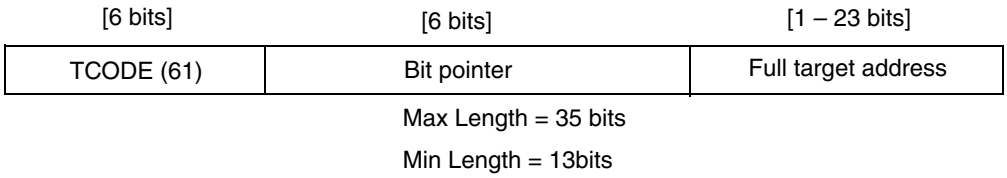

Figure 9. Indirect Branch Synchronization Message Format With Compressed Code (PTSM==0)

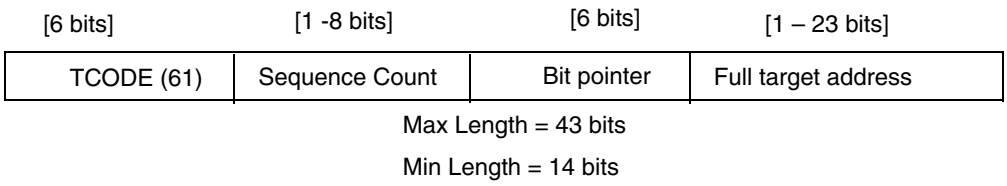

Figure 10. Indirect Branch Synchronization Message Format With Compressed Code (PTSM==1)

3.2 New Program Trace Resource Full Message

With the addition of the sequence count (I-CNT) field to the Branch with Synchronization messages, a new type of message was required to handle the case when the I-CNT is greater than 256 instructions in a row without a branch (since the I-CNT field has a maximum width of 8 bits). The Program Trace Resource Full Message indicates that the maximum I-CNT has been reached.The total instruction count can be found by adding 256 for each Program Trace Full message received to the sequence count of the following trace message. The Program Trace Full message has the following format:

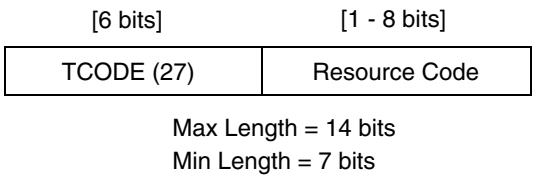

Figure 11. Program Trace Full Message Format

At this time, the Program Trace sequential count full code is the only defined option for this message.

Table 8. Resource Codes

Error Code	Description
0000	Program Trace Sequential Count Full
0001–1111	Vendor Defined or Reserved

3.3 16 Bit Data Trace Change

When using an 8-bit port (Full Port mode), tools cannot differentiate between an 8-bit data message and a 16-bit data message that has zeros in the upper 8 bits. 16-bit data trace messages have been changed to always send 24 bits of data. This includes 8 bits of padding. This allows 8-bit data to be differentiated from 16-bit data.

The data read message has the following format:

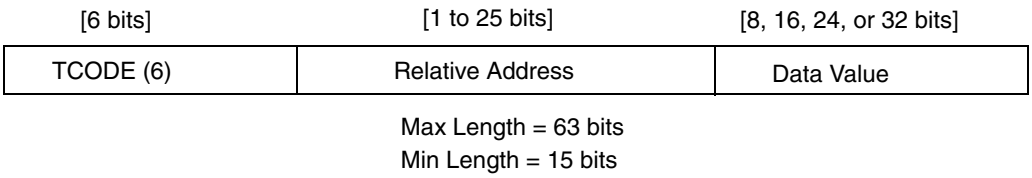

Figure 12. Data Read Message Format

This same change also affects Data Trace Write messages, Data Read with Synchronization messages, and Data Write with Synchronization messages.

Even though, with a full-width auxiliary port, the address field can be from 1 to 25 bits of address, because of the Nexus variable length fields' padding and the requirement that variable length fields cannot end on the same clock in which they are started, the relative address field will have anywhere from 10 to 26 bits. Table 9 shows an example of a 16-bit read data trace message that includes the maximum number of bits in the relative address.

Table 9. Data Trace Message—16 Bit Data with the Maximum Relative Address

Clock	MDO[7]	MDO[6]	MDO[5]	MDO[4]	MDO[3]	MDO[2]	MDO[1]	MDO[0]	MDO[7:0]	MSEO	
0	—	—	—	—	—	—	—	—	—	1	Idle or end of message
1	0	0 LSB	0 MSB	0	0	1	1	0 LSB	0x06	0	Start of message
2	0	0	0	0	0	0	0	0	0x00	0	
3	0	0	0	0	1	0	0	0	0x00	0	
4	0	1	0	0	0	0	0	0	0x00	1	End of Field #1 - Relative Address
5	0	0	0	0	0	0	0	1	0x01	0	Start of Field #2 - 16 Bit Data
6	0	0	0	0	0	0	0	0	0x00	0	
7	0	0	0	0	0	0	0	0	0x0	1	End of Field #2 and End of Message

Padding
Field #1 (relative address)
Field #2 (16-bit data)
TCODE

Table 10 shows an example of an 8 bit data trace message that has a minimum size address field.

Table 10. Data Trace Message—8 Bit with the Minimum Relative Address

Clock	MDO[7]	MDO[6]	MDO[5]	MDO[4]	MDO[3]	MDO[2]	MDO[1]	MDO[0]	MDO[7:0]	MSEO	
0	—	—	—	—	—	—	—	—	—	1	Idle or end of message
1	0	1 LSB	0 MSB	0	0	1	1	0 LSB	0x46	0	Start of message
2	0	0	0	0	0	0	0	0	0x00	1	End of Field #1 - Relative Address
3	0	0	0	0	0	0	0	1	0x01	0	Start of Field #2 - 8 Bit Data
4	0	0	0	0	0	0	0	0	0x00	1	End of Field #2 and End of Message

Padding
Field #1 (relative address)
Field #2 (16-bit data)
TCODE

3.4 Updated Error Messages

Previously, the only program trace error sent by the MPC56*x* devices was a program/data/ownership trace overrun (error code number 0b00111). On the new silicon with READI_MC[QFM] = 1, the READI module can generate the three previously reserved error messages: ownership trace overrun (0b00000), program trace overrun (0b00001), and data trace overrun (0b00010). These will be generated if only one type of trace message is discarded. If more than one type of trace message is discarded, then the program/data/ownership trace overrun (0b00111) will be sent.

Table 11. Error Codes

Error Code	Description
00000	Ownership trace overrun[1]
00001	Program trace overrun[1]
00010	Data trace overrun[1]
00011	Read/write access error
00100	Invalid message
00101	Invalid access opcode
00110	Watchpoint overrun
00111	Program/data/ownership trace overrun
01000–10111	Reserved
11000–11111	Vendor Defined

NOTES:
1 Early revisions of silicon (with READI modules versions prior to 3.0) will not generate this error code.

4 Nexus Message Queue Size

On revision 3.0 (and later) of the READI module, the Nexus message queue size has been increased from 16 to 32 messages. This will allow more messages to be queued and handle more Data Trace messages along with Program Trace messages without having a queue overflow.

This enhancement is not visible to the user, only the affect of having a larger buffer can be seen: the ability to store and transmit more trace messages (data or program trace).

5 READI Priorities

Versions of the READI module prior to 3.0 used the same priority mechanism for messages input into the Nexus output queue as the priority of messages output via the MDO[0:7] (or MDO[0:1]).

1. Invalid message
2. READI register access handshakes (device ready/download information)[1]
3. Watchpoint messages
4. Read/write access message
5. RCPU development access message
6. Queued messages (program trace, data trace, and ownership trace)

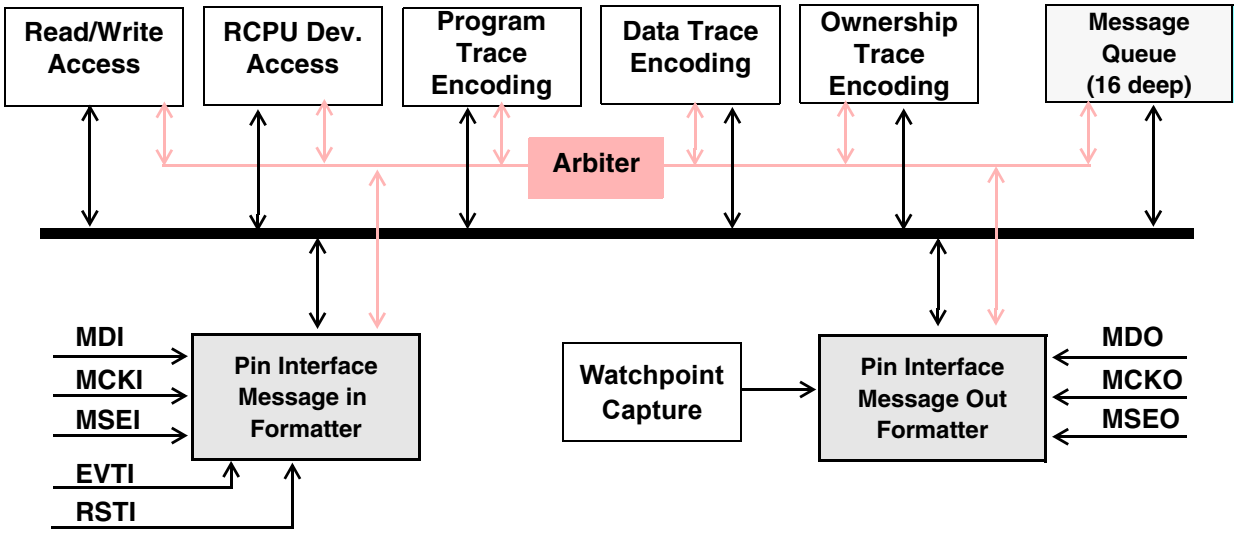

Figure 13. READI Version 1.0 through 2.1 Message Queue Block Diagram

In the version 3.0 of the READI module, the priority of messages on the input side of the output queue have a separate priority than the message output queue.

1. High Priority Data Trace message[2]
2. Program Trace messages
3. Low Priority Trace messages
4. Ownership Trace messages

The output side of the READI message queue still has the same priority as earlier revisions of the READI module.

1. READI register access handshake messages are actually the same priority as Invalid messages since both cannot happen at the same time.
2. Data Trace messages are made high priority if they are in danger of being lost due to the message queue not being available while Program Trace messages are being added to the queue.

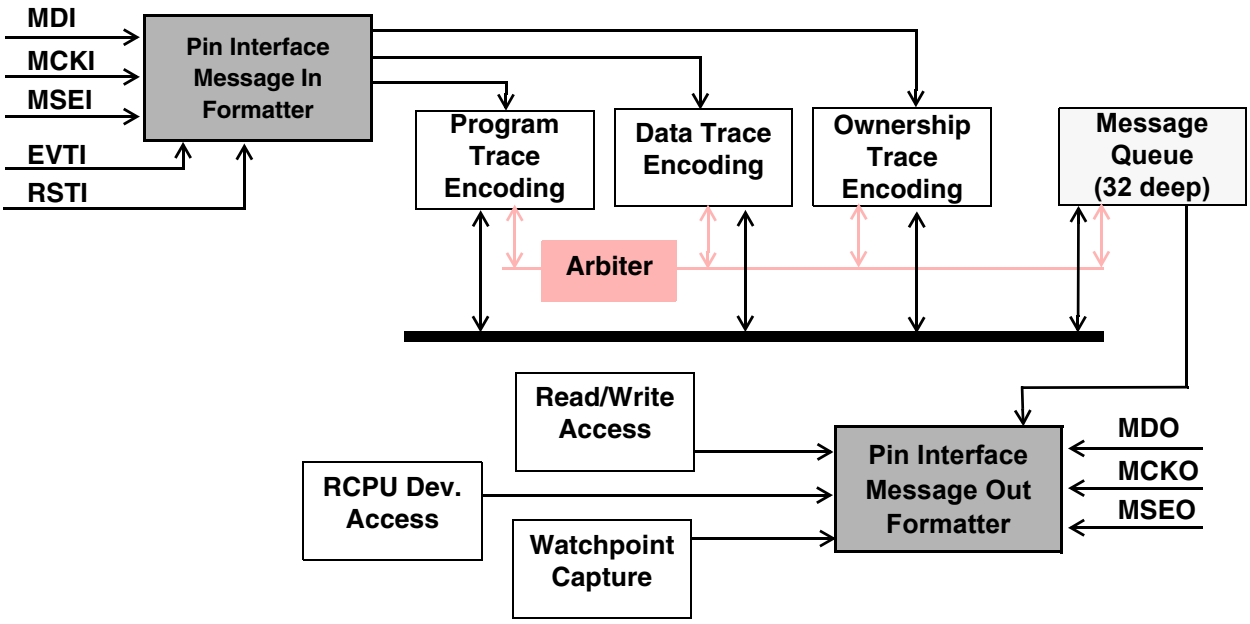

Figure 14. READI Version 3.0 Message Queue Block Diagram

6 Number of Clocks Between MDI Messages

Silicon that does not have version 3.0 (or later) of the READI module requires that Nexus input messages be separated by at least 4 MCKI clocks. In revision 3.0 (and later) of the READI module, this requirement has been removed and messages can be input into the Nexus input port with only 1 clock between messages. In most cases, however, the user or tool should wait until an acknowledgement message is received from the device via the Nexus output port (MDO[0:7] or MDO[0:1]).

7 Hints for Debugging Using Nexus

The following subsections outline commonly overlooked information that users and tool vendors should keep in mind when working with the MPC56*x* devices and Nexus debugging.

7.1 Recommended Register Settings for Trace

The many debug options offered for MPC56*x* devices have features that are controlled in several different registers. The READI module snoops both the internal U-bus and L-bus. When using the Nexus interface, there are optimum settings of these registers for performance that still allow access to the debug information. These are summarized in Table 12.

Table 12. Summary of MCU Trace Settings for Nexus Trace

Register	Field	Setting	Description
ICTRL	ISCT_SER	0b101	Show all instruction change of flows
ICTRL	ISCT_SER	0b110[1]	Show all indirect changes of flows
SIUMCR	NOSHOW	0x1	Do not put show cycle information on the external bus
SIUMCR	DSHW	0x0	Do not show data show cycles on the internal U-bus
L2U_MCR	LSHOW	0b00	Do not show L-bus show cycles on the U-bus

NOTES:
1 Not all tools may be able to handle this setting. Check with your specific tool vendor.

7.1.1 Enable Internal Instruction Show Cycles

The ICTRL[ISCT_SER] controls whether show cycle information of executed instructions is shown on the internal U-bus. It also controls whether the RCPU code runs in serialized or non-serialized mode. The non-serialized mode allows some overlapping of instructions to increase performance by pipelining instructions and by allowing the different subprocessors to operate in parallel when possible. These subprocessors include the Branch Processor (BPU), the Arithmetic/Bit Field Unit (ALU/BFU), the Load/Store Unit (LSU), the Floating Point Unit (FPU), and the Integer Multiply/Divide Unit (IMWL/IDIV).

For the READI module to see the instructions that are being executed in the RCPU core, instruction show cycles must be enabled. For maximum instruction trace information, the ICTRL[ISCT_SER] should be set to 0b101 (0x5). This enables non-serialized mode and shows all changes of flow. This allows the READI module to generate Nexus trace packets. READI version 3.0 (and later) also supports Nexus trace messaging with the ICTRL[ISCT_SER] = 0b110 (0x6). See Section 2.2.1, “Program Trace Sync Mode (READI_MC[PTSM]),” for additional information. Note that all tool vendors may not support trace reconstruction with this setting. The user should check with their specific tool vendor.

7.1.2 Don’t Show Trace Information On the External Bus

Normally when instruction or data show cycles are enabled, this information needs to be shown on the external bus to allow logic analyzers to see the internal information for code trace or code reconstruction. With Nexus, this information no longer needs to be shown on the external data and address buses. Setting the SIUMCR[NOSHOW] (= 0b1) disables the external show cycles.

7.1.3 Don’t Show Data Show Cycles on U-Bus

Normally data show cycles must be shown on the internal U-bus such that the data show cycles can be shown on the external data and address buses. Setting the SIUMCR[DSHW] = 0b1 enables data show cycles on the U-bus. For Nexus trace, this does not need to be enabled. Clearing the SIUMCR[DSHW] = 0b0 bit disables the data show cycles on the U-bus.

7.1.4 Don’t show L-bus Transactions on the U-bus

To show the internal CALRAM (SRAM) bus cycles onto the external bus, the L2U_MCR[LSHOW] bit must be enabled for the data show cycles to be shown from the L-bus to the U-bus. The READI module snoops the L-bus and can see all data transactions that go to or from the CALRAM, any IMB module, USIU or BBC register, or external bus. Therefore, LSHOW does not need to be enabled. For Nexus data trace, the L2UMCR[LSHOW] should be set to 0b00 (disable L-bus show cycles).

7.2 READI Limitations

This section highlights common issues that users should keep in mind when using Nexus debuggers with the RCPU-based microcontrollers. These are covered in the following sections.

7.2.1 Limitations of Using Watchpoints and Breakpoints with Exception Relocation and the Enhanced Interrupt Controller

Watchpoints or breakpoints at relocated addresses will not match in the RCPU core. This is not a problem with the device. Relocation is performed outside of the RCPU core (in the Burst Buffer Controller—BBC2); therefore, watchpoints and breakpoints will never match the relocated addresses in the core. Users should place watchpoints/breakpoints at the non-relocated addresses (0xFFF0_XX00).

CAUTION

Placing breakpoints at the first location of an exception handler is not recommended. Breakpoints should be set after the machine state is saved and the exception handler has made the exception recoverable (MSR[RI] = 1).

On version 3.0 and later of the READI module, users should not jump directly to an exception vector address (in the range of 0xFFF0_0000 to 0xFFF0_1F00) with exception relocation enabled. This will cause a corruption of the trace information. It is permissible to jump directly to the actual physical address where the exception handler is located.

7.2.2 Nexus Read/Write Access Limitations

The Nexus RWA (Read/Write Access) register has a limit of 25 bits of address for the destination read or write address. This means that a Nexus tool can only access memory location in the lower 32 Mbytes of the MPC56*x* address space (0x0 to 0x01FF_FFFF) with Nexus Read/Write access messages. Access to memory above this address space (starting at 0x0200_0000) can only be accomplished via the vendor-defined Development Port Access messages (TCODES 56 [0x38] and 57 [0x39]). This allows access from the Nexus port through the Background Debug port (BDM) to the RCPU core processor. The use of the Development Port Access messages require that the RCPU core be halted for access through BDM, since this is an intrusive memory access. Some tools may not support this type of operation.

7.2.3 READI Trace Limitations

The MPC56*x* will generate both program and data trace information (if enabled) for the complete memory address space. The READI module, however, masks off the upper 7 bits of the address for all trace messages and when setting the data trace region. All Instruction Trace messages do not include the upper 7 bits of address space. The data trace attributes registers (DTA1, DTA2) only allow for 25 bits of address when setting the address ranges to be traced. For example, if data trace messages are enabled for the memory range of 0x01F0_0000 through 0x01F0_2000, the READI module will show data accesses made to 0x03F0_0000 through 0x03F0_2000, 0x07F0_0000 through 0x07F0_2000, ..., and 0xFFF0_0000 through 0xFFF0_2000. The user should keep this in mind when setting the base address of any external memories to ensure that memory addresses are unique in the lower 25 bits of the address. See Table 13 for examples of mask values that can be programmed into the MPC56*x* memory controller to allow external memory to appear in multiple address spaces. This allows tools to be able to correlate program or data trace address to addresses readable via Nexus messages.

Table 13. Memory Controller Mask Programming for Mirrored Space

Address Space	Memory Size	Mask ORx[AM]	Mirrored Base Address	Mirrored End of Address	Example Memory
0xFFF0_0000–0xFFFF_FFFF	1 Mbyte	0x01F0_0000	0x01F0_0000	0x01FF_FFFF	AMD AM29BDD160G Flash
0xFFF0_0000–0xFFF7_FFFF	512 Kbytes	0x01F0_0000	0x01F0_0000	0x01F7_FFFF	Cypress CY7C1338 SRAM
0xFF00_0000–0xFFFF_FFFF	16 Mbytes	0x0100_0000	0x0100_0000	0x01FF_FFFF	2 × AMD AM29LV640MH/L Flash

7.2.4 MPC56*x* SPR Access

The RCPU supports special purpose registers (SPRs). These are located in a special internal memory space that is normally addressed using special instructions, Move To Special Purpose register (**mfspr**) and Move From Special Purpose Register (**mfspr**). SPRs that are NOT located on the RCPU core itself can be accessed via the Nexus Read/Write access register by setting the MAP bit (RWA[MAP] = 1). These are shown in Table 14.

Table 14. Special Purpose Registers accessible Via Nexus RWA

SPR	Symbol	Register	Internal Address
SPR 22	DEC	Decrementer	0x2C00
SPR 268	TBU	Time Base Lower — Read (TBL)	0x1880
SPR 269	TBL	Time Base Upper — Read (TBU)	0x1A80
SPR 284	TBU	Time Base Lower — Write (TBL)	0x3880
SPR 285	TBL	Time Base Upper — Write (TBU)	0x3A80
SPR 528	MI_GRA	MI Global Region Attribute Register	0x2100
SPR 529	EIBADR	External Interrupt Relocation Base Address	0x2300
SPR 536	L2U_GRA	L2U Global Region Attribute Register	0x3100

Table 14. Special Purpose Registers accessible Via Nexus RWA (continued)

SPR	Symbol	Register	Internal Address
SPR 560	BBCMCR	Burst Buffer Controller Module Configuration Register	0x2110
SPR 568	L2U_MCR	L-Bus to U-Bus Module Configuration Register	0x3110
SPR 630	DPDR	Development Port Data Register	0x2D30
SPR 638	IMMR	Internal Memory Mapping Register	0x3D30
SPR 784	MI_RBA0	MI Region 0 Base Address Register	0x2180
SPR 785	MI_RBA1	MI Region 1 Base Address Register	0x2380
SPR 786	MI_RBA2	MI Region 2 Base Address Register	0x2580
SPR 787	MI_RBA3	MI Region 3 Base Address Register	0x2780
SPR 792	L2U_RA0	L2U Region 0 Base Address Register	0x3180
SPR 793	L2U_RA1	L2U Region 1 Base Address Register	0x3380
SPR 794	L2U_RA2	L2U Region 2 Base Address Register	0x3580
SPR 795	L2U_RA3	L2U Region 3 Base Address Register	0x3780
SPR 816	MI_RA0	MI Region 0 Attribute Register	0x2190
SPR 817	MI_RA1	MI Region 1 Attribute Register	0x2390
SPR 818	MI_RA2	MI Region 2 Attribute Register	0x2590
SPR 819	MI_RA3	MI Region 3 Attribute Register	0x2790
SPR 824	L2U_RA0	L2U Region 0 Attribute Register	0x3190
SPR 825	L2U_RA1	L2U Region 1 Attribute Register	0x3390
SPR 826	L2U_RA2	L2U Region 2 Attribute Register	0x3590
SPR 827	L2U_RA3	L2U Region 3 Attribute Register	0x3790

The remaining SPR registers, all of the process general purpose registers (GPR0–GPR31) and floating point registers (FPR0–FPR31, FPSCR), the condition register (CR), and the machine state register (MSR) can only be accessed by using the Development Port Access messages (BDM TCODES). Use of the Development Support Access messages to access these registers requires that the RCPU be halted in debug mode. The special purpose registers are shown in Table 15.

Table 15. SPR Accessible only via BDM Accesses

SPR	Symbol	Register
SPR 1	XER	Integer Exception Register
SPR 8	LR	Link Register
SPR 9	CTR	Count Register
SPR 18	DSISR	DAE/Source Instruction Service Register
SPR 19	DAR	Data Address Register
SPR 26	SRR0	Machine Status Save/Restore Register 0

Table 15. SPR Accessible only via BDM Accesses (continued)

SPR	Symbol	Register
SPR 27	SRR1	Machine Status Save/Restore Register1
SPR 80	EIE	External Interrupt Enable
SPR 81	EID	External Interrupt Disable
SPR 82	NRI	Non-Recoverable Interrupt Register
SPR 144–SPR 147	CMPA–CMPD	Comparator A-D Value Register
SPR 148	ECR	Exception Cause Register
SPR 149	DER	Debug Enable Register
SPR 150	COUNTA	Breakpoint Counter A Value and Control Register
SPR 151	COUNTB	Breakpoint Counter B Value and Control Register
SPR 152–SPR 153	CMPE–CMPF	Comparator E-F Value Register
SPR 154–SPR 155	CMPG–CMPH	Comparator G-H Value Register
SPR 156	LCTRL1	L-bus Support Control Register 1
SPR 157	LCTRL2	L-bus Support Control Register 2
SPR 158	ICTRL	I-bus Support Control Register
SPR 159	BAR	Breakpoint Address Register
SPR 272–SPR 275	SPRG0–SPRG3	General Special-Purpose Registers 0–3
SPR 287	PVR	Processor Version Register
SPR 1022	FPECR	Floating-Point Exception Cause Register

8 READI Module Design Changes

To date, there have been three major and one minor versions of READI module.

8.1 READI Versions and Errata

Table 1 shows the versions of READI modules that are used on each revision of the MPC56*x* devices. A summary of all of the current READI module errata is shown in Table 16. This table also shows which of these errata are fixed on the different versions of the READI modules.

Table 16. Errata that Affect the READI Module

Errata Number	Errata Description	READI Modules Affected			
		1.0	2.0	2.1	3.0
AR_698	READI Input message requires 2 MCKI idle after READI Enabled.	X	X	X	X
AR_734	READI Module cannot be enabled without Nexus input clock (MCKI).	X			
AR_783[1]	READI input messages must be 4 MCKI apart.	X	X	X	
AR_846	READI: Synchronize the MCKI input clock to the MCKO output clock.	X	X	X	X

Table 16. Errata that Affect the READI Module (continued)

Errata Number	Errata Description	READI Modules Affected			
		1.0	2.0	2.1	3.0
AR_924	READI: Communication lost when clock freq is changed via Nexus with BDM enabled	X	X		
AR_1021[1]	READI: Manufacturer ID in Device ID register is incorrect	X	X	X	
AR_1041	READI: Trace may show incorrect Addresses When Exception Relocation is used	X	X	X	
AR_1050	READI: Unexpected READI Overflow Error Messages	X	X	X	
AR_1051	READI: Trace can't handle multiple change of flow without a show cycle	X	X	X	
AR_1059[1]	READI: 8-bit and some 16-bit data trace messages can't be differentiated	X	X	X	
AR_1060[1]	READI: Program Trace Sync Messages do not include sequential instruction count	X	X	X	
AR_1061[1]	READI: New Feature added to allow queue mode to empty instead of flush	X	X	X	
AR_1065[1]	READI: Queue entries to change from 16 to 32 on future revisions	X	X	X	
AR_1066[1]	READI: Program trace requires all change of flow show cycles	X	X	X	
AR_1073	CALRAM: Aborted overlay accesses could halt processor				
AR_1096[2]	READI: New features to enhance Nexus debug				X

NOTES:
1 This is a new feature that is described earlier in this application note.
2 This customer information (from the errata list) documents the new features (covered as missing in AR_1059, AR_1060, AR_1061, and AR_1065) that are now available.

Table 17. Non-READI Errata that Affect Only Specific Versions of Devices

Errata Number	Errata Description	Device					
		MPC561			MPC563	MPC565	
		0	A,C	D	0,0a, A, B	0, 0a, A, B, C	D
AR_829	USIU: Low power down mode does not work when READI is enabled	X					
AR_836	READI cannot be enabled prior to hreset_b negation.	X					
AR_1073	CALRAM: Aborted overlay accesses could halt processor	X	X		X	X	
AR_1076	RCPU: Treat VF queue flush information value of 6 as 2[1]	X	X		X	X	

NOTES:
1 Revision 3.0 of the READI module will handle this case if it were not fixed in the RCPU core.

9 Revision History

Table 18 is a revision history of this document.

Table 18. Revision History

Rev. Number	Substantive Changes	Date of Release
0	Initial release.	01/2005

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
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 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN2693
Rev. 0
01/2005