Freescale Semiconductor
Application Note

AN2754
Rev. 1, 01/2005

CPM Architecture and Downloading RAM Microcodes on the PowerQUICC II™ Family

by *NCSD Applications*
Freescale Semiconductor, Inc.
East Kilbride, Scotland

All devices in the PowerQUICC II™ family of processors (MPC825x, MPC826x, MPC827x, and MPC828x) implement an integrated communication processor module (CPM). This CPM has a RISC microcontroller that supports downloadable RAM-based sets of instructions used by this microcontroller. These RAM-based microcodes are required to patch or modify existing code in ROM to support the following:

- CPM errata (bug) fixes
- New protocols
- Protocol enhancements to existing functionality in microcode ROM, for example ATM AAL2

This application note describes the CPM microcode architecture and the microcode download procedure. It also includes example C code for downloading RAM-based microcode onto different silicon revisions of the PowerQUICC II family.

Contents

freescale™ semiconductor

1 References

Users are encouraged to consult the reference documentation in Table 1 for additional information.

Table 1. References

Document Category	Document Title	Document ID
Reference Manual	*MPC8260 PowerQUICC II Family Reference Manual*	MPC8260UM
	MPC8280 PowerQUICC II Family Reference Manual	MPC8280RM
	MPC8272 PowerQUICC II Family Reference Manual	MPC8272RM

2 Silicon Revisions

The internal memory register IMMR contains a read-only field called Mask_Num, which is programmed with a code corresponding to the mask number of the part on which the SIU is located. It is intended to help factory test and user code determine the silicon revision of the PQ2 device soldered on a PCB board which is sensitive to part changes.

In addition associated with each version of CPM microcode, is number (REV_NUM) that uniquely identifies that specific microcode. This number is hard-coded into the microcode which is stored in the CPM's internal ROM. At power-up, the Communication Processor (CP) reads this number, and stores it into a miscellaneous portion of the CPM's internal dual-port RAM (DPRAM). The user can then access this location in DPRAM to determine which version of CPM microcode is contained in that device.

Table 2 gives the IMMR[Mask_Num] and CPM microcode revision number for all PQ2 silicon revisions available

Table 2. Silicon Revision Information

Silicon Revision	Mask	Process	IMMR[16–31] [1]	Rev_Num [2]
A.1	1K22A	0.29 µm HIP3	0x0011	0x0001
B.3	3K23A	0.29 µm HIP3	0x0023	0x003B
C.2	6K23A	0.29 µm HIP3	0x0024	0x007B
MPC826xA (HIP4) Derivatives including 8260A, 8264A, 8265A and 8266A				
A.0	2K25A	0.25 µm HIP4	0x0060	0x000D
B.1	4K25A	0.25 µm HIP4	0x0062	0x002D
MPC8275, MPC8270 and MPC8280 (HIP7) Derivatives				
0.1	1K49A	0.13 µm HIP7	0x0A00	0x0070

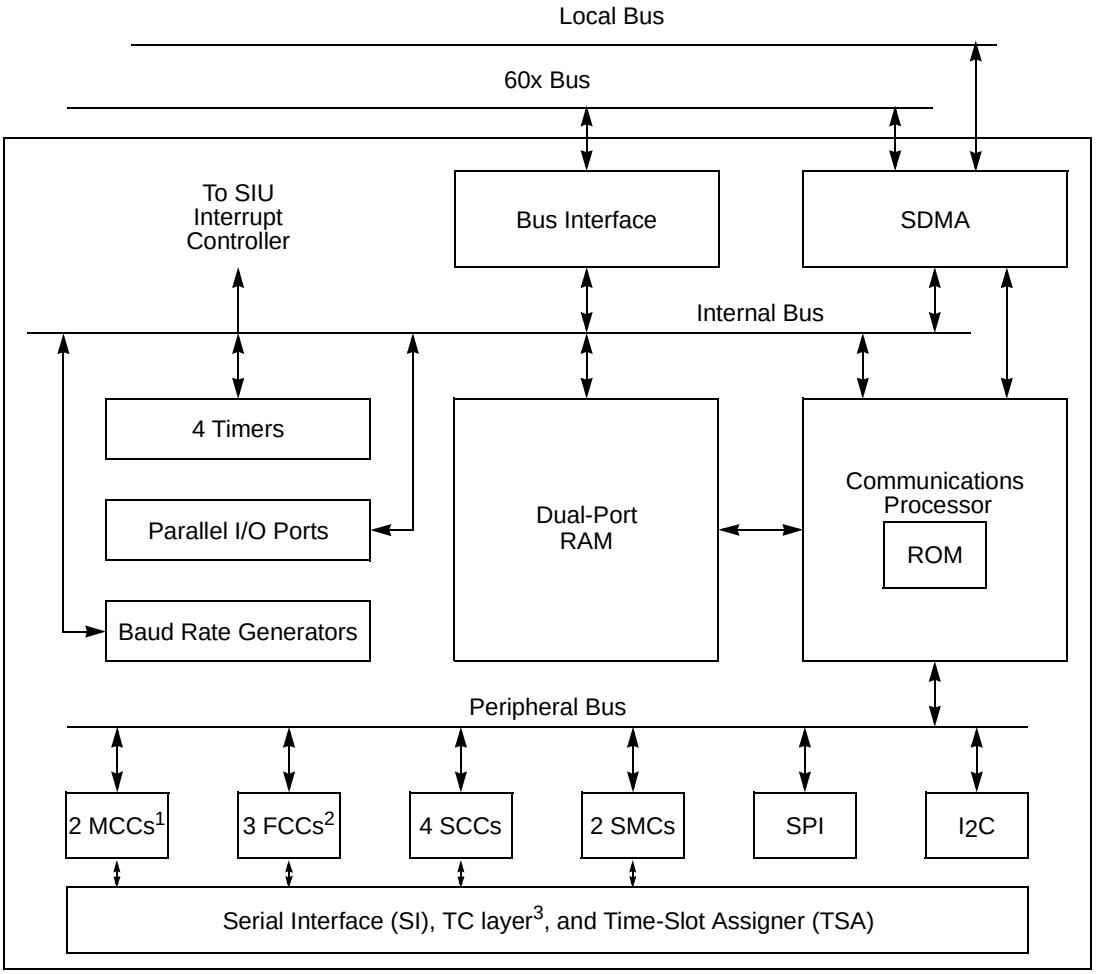
1 The IMMR[16–31] indicates the mask number, which changes only when metal layers are modified. Located at offset 0x101A8.

2 The Rev_Num located at offset 0x8AF0 in DPRAM indicates the CPM microcode revision number.

3 CPM Architecture

A block diagram of the communication processor module (CPM) is given in Figure 1. The CPM consists of the following major components:

- CP RISC microcontroller
- Dual-port RAM (DPRAM)
- ROM
- DMA unit
- Serial peripherals



Note
[1] One MCC on the MPC8250, MPC8255, MPC8270, and MPC8275
[2] Two FCCs on the MPC8255
[3] MPC8264, MPC8266, and MPC8280 only

Figure 1. PowerQUICC II CPM Block Diagram

3.1 RISC Microcontroller

The communications processor (CP), also called the RISC microcontroller, is a 32-bit controller for the CPM that resides on a separate bus from the G2 core, it can therefore perform tasks independently of the G2 core. The CP handles lower-layer communications tasks and DMA control, freeing the G2 core to handle higher-layer activities. The CP works with the peripheral controllers and parallel ports to implement user-programmable protocols and manage the serial DMA (SDMA) channels that transfer data between the I/O channels and memory. It also manages the IDMA (independent DMA) channels and contains an internal timer used to implement up to 16 additional software timers.

The CP RISC is a request driven engine. Requests can be generated by peripherals, timers, or external commands. The CP RISC has a hardware block called the scheduler, responsible for storing requests and dispatching the appropriate task. When several requests are present at the same time, the one with the highest priority will be serviced. Since the CP RISC is request-driven, tasks are atomic and cannot preempt each other.

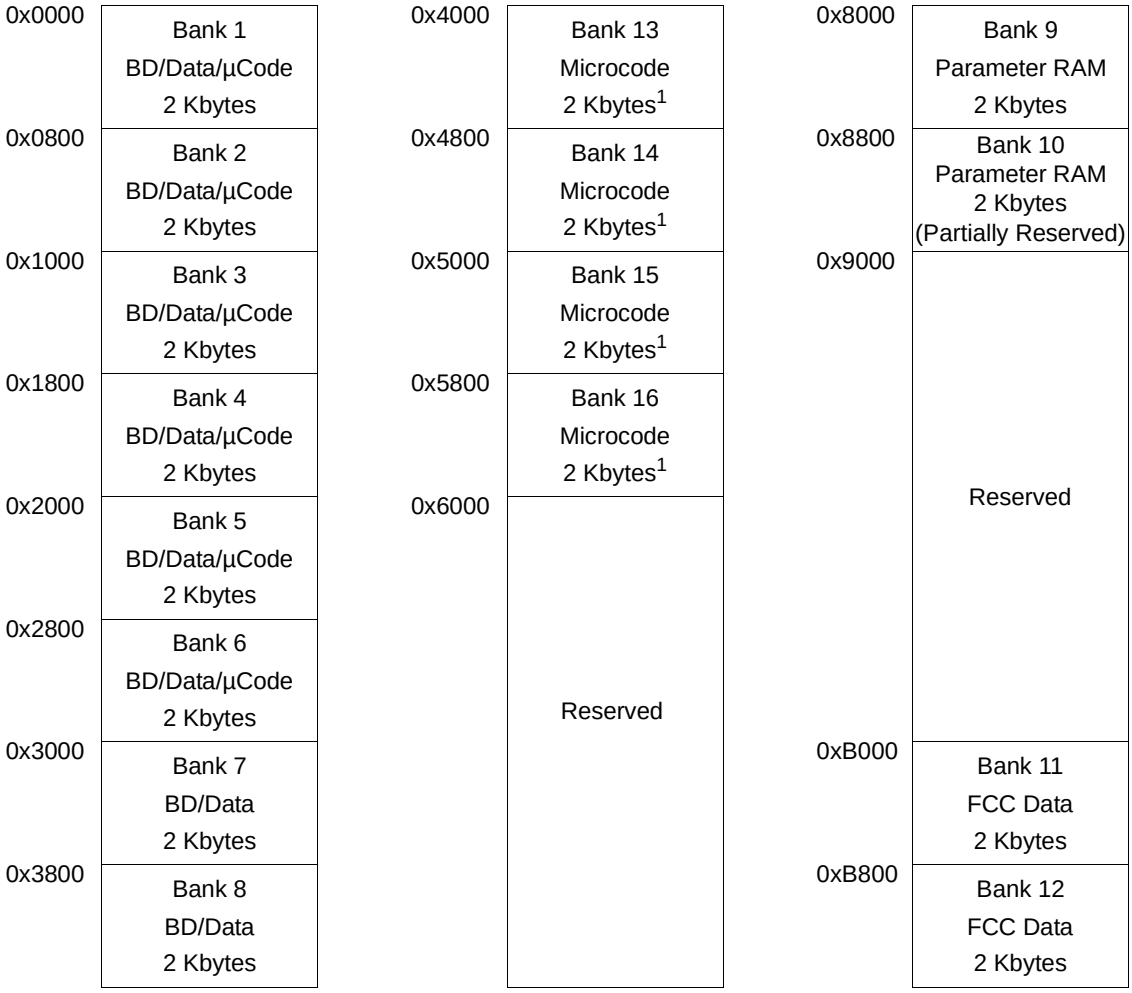
The CP's architecture and instruction set are optimized for data communications and data processing required by many wire-line and wireless communications standards. An example of this is instructions that trigger CRC calculations of transmitted or received data, this provides a performance increase as these tasks are offloaded from the RISC's instruction set.

3.2 DPRAM

DPRAM can be accessed by either the CP RISC or the G2 core and is primarily used for storing any or all of the following:

- A custom, downloadable RAM microcode for the CP RISC
- Parameters used by peripherals
- Buffer descriptors for slow serials (SMC, SPI, I^2C, and SCC)
- Temporary data

The CPM has 24 Kbytes of DPRAM on 0.29-µm (HiP3) devices and 32-Kbytes of DPRAM on 0.25-µm (HiP4) devices. Figure 2 shows the memory map of dual-port RAM on 0.29-µm (HiP3) devices and 0.25-µm (HiP4) devices.



[1] Reserved on .29 µm (HiP3) devices.

Figure 2. Dual-Port RAM (DPRAM) Memory Map

Only parameter RAM and RAM microcodes require fixed addresses to be used. Buffer descriptors, data buffers, and scratchpad RAM can be located in DPRAM or in any unused parameter RAM (that is, any area made available when a peripheral controller or sub-block is not being used). Note that buffer descriptors for the FCC and MCC protocols must reside in external memory on the 60x or local bus; the CPM accesses these BDs through an SDMA transaction.

On 0.29 µm (HiP3) devices, DPRAM is divided into twelve memory banks. The first 12 Kbytes (Banks 1 through 6) can be used for custom RAM microcode execution. RAM microcode has to start at the beginning of Bank 1, and it has to use subsequent banks. This means a RAM microcode could overlay an area of DPRAM required for MCC1 and MCC2 channel-specific parameters. For example, if a RAM microcode occupies the first 2 Kbytes of DPRAM, MCC1 channels 0–31 are no longer available.

On 0.25-µm (HiP4) devices, DPRAM is divided into sixteen memory banks. The first 12 Kbytes (Banks 1 through 6) can be used for custom RAM microcode execution. In addition, an extra 8 Kbytes of DPRAM (Banks 13 through 16) are available for microcode execution only and cannot be used for data buffers or BDs.

Banks 7, 8, 11, and 12 of the DPRAM can be used for storing buffer descriptors, parameters, temporary data, or any data specific to a particular task.

Therefore on HiP3 devices, 12 Kbytes of DPRAM starting at address IMMR + 0x0000 is available for RAM microcode downloads. On HiP4 devices, 20 Kbytes of DPRAM starting at address IMMR + 0x4000 (for the first 8 Kbytes) and address IMMR + 0x0000 (for the remaining 12 Kbytes) is available for RAM microcode downloads.

On 0.13-µm (HiP7) devices, the CPM has 64 Kbytes of static RAM. This RAM is divided into two 32-Kbyte blocks as follows:

- 32-Kbyte CP-RISC instruction RAM. This RAM is used to store a microcode package of up to 32-Kbyte instructions, and cannot be used for data buffers, BDs, or protocol parameters
- 32-Kbyte CP-RISC data RAM. This RAM is used to store CP-RISC parameter RAM and data structures as defined in the *MPC8260 PowerQUICC II Family Reference Manual.*

Figure 3 shows a memory map of the internal instruction RAM 0.13-µm (HiP7) devices.

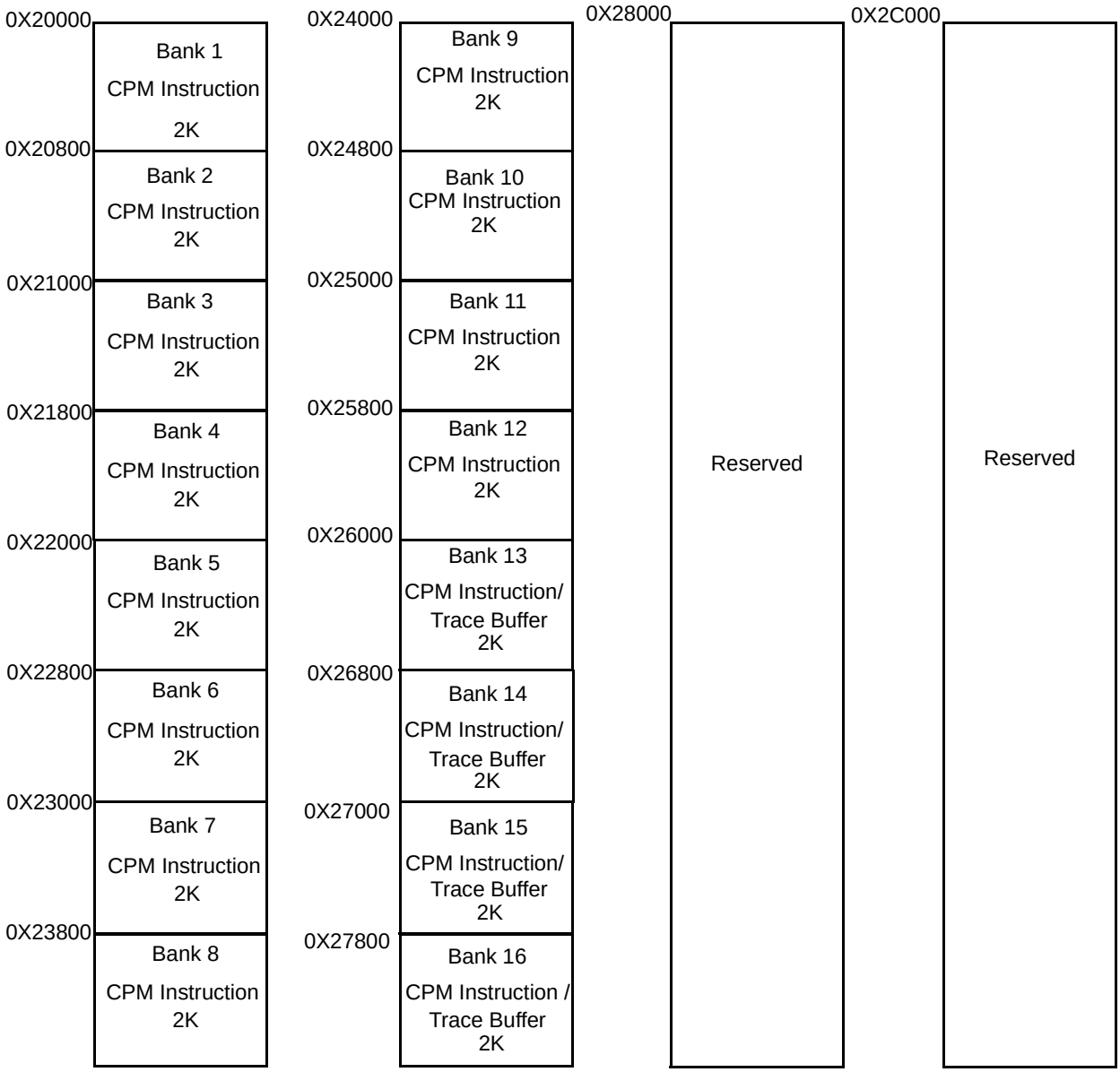


Figure 3. Instruction RAM Partitioning

Figure 4 shows a memory map of the internal data RAM on 0.13 µm (HiP7) devices. Note that the addresses refer to CPU address space.

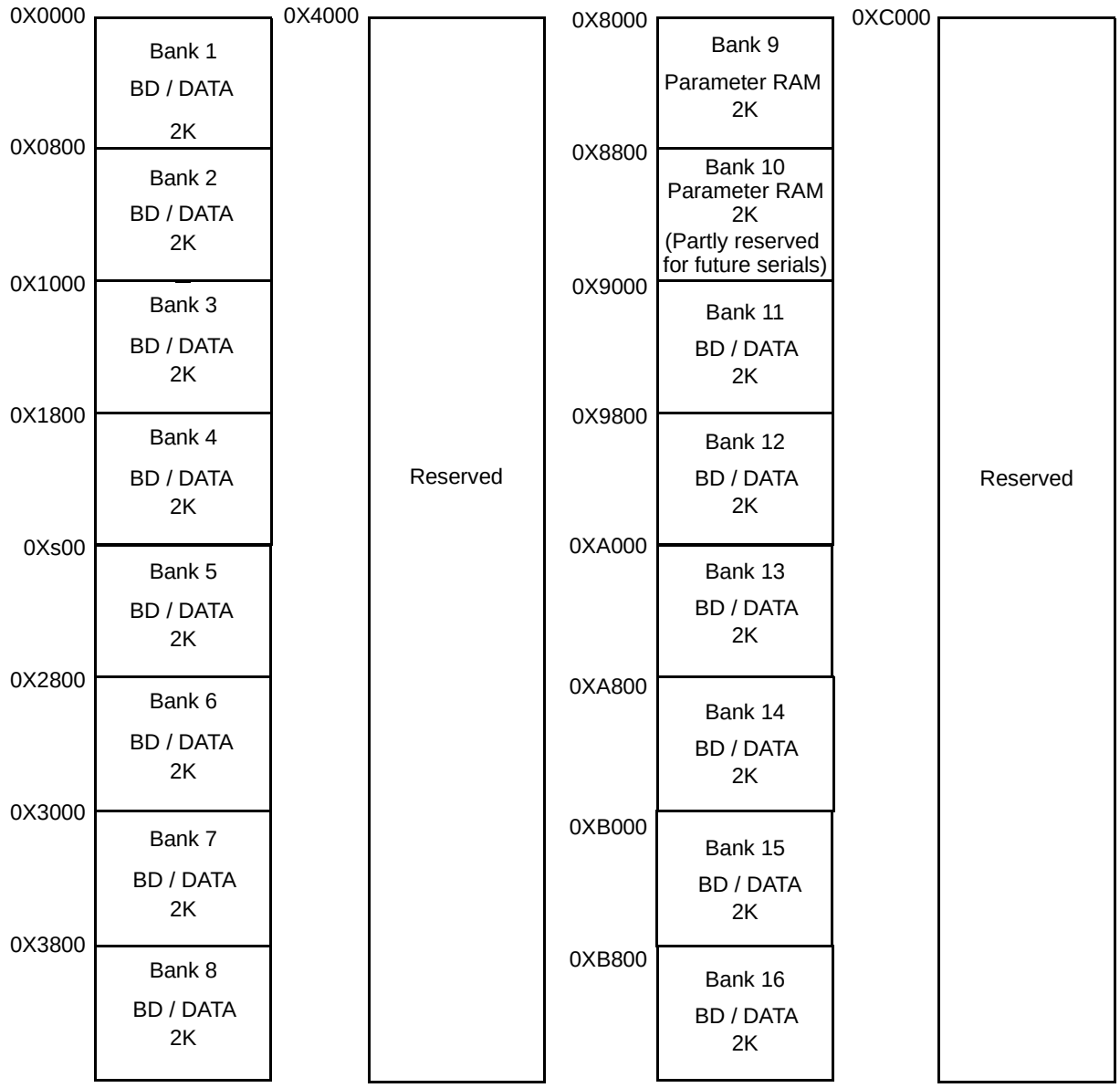


Figure 4. Internal Data RAM Memory Map

The internal data RAM data bus is 64 bits wide. The RAM is used for five possible tasks:

1. To store parameters associated with the FCCs, SCCs, SMCs, SPI, I^2C, and IDMAs in the 2,048-byte parameter RAM
2. To store the BDs that describe where data is to be received and transmitted on slow peripherals
3. To store parameters required to implement a given communications protocol, for example, APC tables when the FCC is running in ATM mode
4. Temporary storage between FCC FIFO and external memory for FCC data that is transferred by the block transfer module (BTM) which moves data between FCC FIFOs.

5. For additional RAM space for user software

Note that all banks of internal data RAM on 0.13-µm (HiP7) devices are interleaved supporting concurrent accesses for improved performance. It is recommended that FCC, BTM, and SDMA-based parameters and APC parameters are placed in the interleaved banks of DPRAM. Table 3 summarizes which banks should contain these FCC parameters.

Table 3. Interleaved DPRAM Banks vs. Silicon Revision

Revision	DPRAM Banks For FCC-, BTM-, and SDMA-Based Parameters
0.29 µm and 0.25 µm (Hip3 and Hip4)	7,8, 11, and 12
0.13 µm (Hip7)	1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, and 16

The following FCC parameters should therefore contain pointers to these banks.

- HDLC, Ethernet, and Transparent Mode: RIPTR and TIPTR
- ATM Mode: TCELL_TMP_BASE, RCELL_TMP_BASE, APC Tables, INT_RCT_BASE, INT_TCT_BASE, and INT_TCTE_BASE (if used)
- For AAL2: RxQD_Base_Int and CID_Mapping_Table_Base (assuming internal table)

3.3 ROM

ROM is reserved for the microcode developed by Freescale. This microcode implements several protocols such as HDLC, ATM, and Ethernet. The *MPC8260 PowerQUICC II Family Reference Manual* explains in detail the protocols implemented by the ROM microcode and provides information on how to use them.

3.4 DMA Unit

The DMA unit provides the CP RISC with access to additional memory located on either the PPC or local buses. When instructed, the DMA unit can transfer a block of data between dual-port RAM and external memory or vice versa, without intervention by the CP RISC.

3.5 Serial Peripherals

The CP RISC has access to the set of communication controllers through the peripheral bus. The peripheral bus is internal to the CPM and visible only to the CP RISC. The communication controllers have hardware specialized for handling various streams of data.

Each FCC and each SCC has separate receive and transmit FIFOs. The FCC FIFOs are 192 bytes. The SCC FIFOs are 32 bytes. The SMCs, SPI, and I^2C are all double-buffered, creating effective FIFO sizes of two characters.

The *MPC8260 PowerQUICC II Family Reference Manual* gives a general overview of each type of controller as well as the information to properly configure them.

4 Registers

4.1 RISC Configuration Control Register

The CP has an option to execute microcode from a portion of user RAM located in the DPRAM. In this mode, the CP fetches instructions from both the dual-port RAM and its own private ROM. This mode allows new protocols or enhancements to the PowerQUICC II to be added in the form of RAM microcode packages.

The amount of DPRAM (number of banks) used for RAM microcodes is configured by setting the ERAM field of the RCCR register. RAM microcode has to be placed in the DPRAM prior to setting RCCR[ERAM]. As long as RCCR[ERAM] is set, banks selected for microcode can be accessed only by the CP RISC instruction fetcher. Attempts to access this RAM area through the RISC load/store bus, the PowerPC™ 60x slave, or the SDMA unit have no effect on the content of RAM. Reads are returned with arbitrary values, and writes are ignored. It is illegal to place any parameter in a locked portion of DPRAM. The reason for this behavior is explained by the CPM's dual-bus architecture. Instructions fetched from the banks of DPRAM, as selected by the setting of RCCR[ERAM], are transferred using an internal RAM instruction bus. All other data is transferred using the RISC's load/store bus, so in essence, the CP RISC is based on the Harvard architecture. This ensures that a new instruction is fetched each clock cycle regardless of whether the RAM data bus is busy.

The RCCR[ERAM] settings for HiP3 and Hip4 devices to configure the CP to execute microcode from a portion of user RAM are shown in Table 4.

Table 4. RCCR [ERAM] Field Description for HiP3 and HiP4 Devices

Bits	Name	Description
16–18	ERAM	.29 µm (HiP3) devices: Enable RAM microcode. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. (For 25 µm (HiP4) devices RCCR[ERAM] = [16–19]. See the following for settings.) 000 Disable microcode program execution from the dual-port RAM. 001 Microcode uses the first 2 Kbytes of the dual-port RAM. 010 Microcode uses the first 4 Kbytes of the dual-port RAM. 011 Microcode uses the first 6 Kbytes of the dual-port RAM. 100 Microcode uses the first 8 Kbytes of the dual-port RAM. 101 Microcode uses the first 10 Kbytes of the dual-port RAM. 110 Microcode uses the first 12 Kbytes of the dual-port RAM. 111 Reserved

Table 4. RCCR [ERAM] Field Description for HiP3 and HiP4 Devices (continued)

Bits	Name	Description
19	—	.29-µm (HiP3) devices: Reserved
	ERAM	.25-µm (HiP4) devices: ERAM[16–19].Enable RAM microcode. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. 0000 Disable microcode program execution from the dual-port RAM. (That is, microcode execution starts at ROM address 0x0000 after reset.) In the following configurations, RAM microcode execution starts at address 0x0000 after reset: 0010 Microcode uses the first 2 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. 0100 Microcode uses the first 4 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. 0110 Microcode uses the first 6 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. 1000 Microcode uses the first 8 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. 1010 Microcode uses the first 10 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. 1100 Microcode uses the first 12 Kbytes of the dual-port RAM + 8 Kbytes starting from 0x4000. In the following configurations, microcode execution starts at RAM address 0x4000 after reset: 0011 Microcode uses 2 Kbytes starting from dual-port RAM address 0x4000. 0101 Microcode uses 4 Kbytes starting from dual-port RAM address 0x4000. 0111 Microcode uses 6 Kbytes starting from dual-port RAM address 0x4000. 1001 Microcode uses 8 Kbytes starting from dual-port RAM address 0x4000. All other configurations not listed are reserved.

The RCCR[ERAM] settings for HiP7 devices to configure the CP to run microcode from ROM or RAM are shown below in Table 5.

Table 5. RCCR [ERAM] Field Description for HiP7 Devices

Bits	Name	Description
16–19	ERAM	Enable RAM microcode. Configure this field as instructed during the downloading process of a Freescale-supplied RAM microcode package. Otherwise, it should not be used. 0000 Disable microcode program execution from the internal RAM. 0100 Microcode is executed from the Instruction RAM. Other combinations of these bits are not valid and must not be used.

4.2 Trap Registers

Trap registers allow the flow of microcode to be changed. Each trap register allows ROM code to be patched and has a dedicated RAM entry address. A match between the register contents and the program counter causes a trap to RAM.

There are four trap registers on 0.29-µm (HiP3) and 0.25-µm (HiP4) devices. On 0.13-µm (HiP7) devices there are eight trap registers.

Table 6 below shows the number of trap registers available and their location in the internal memory map.

Table 6. Trap Registers

Register [1]	Address Offset
RCCR[ERAM]	IMMR + 0x119C4
RCTR1	IMMR + 0x119CC
RCTR2	IMMR + 0x119CE
RCTR3	IMMR + 0x119D0
RCTR4	IMMR + 0x119D2
RCTR5	IMMR + 0x119B8
RCTR6	IMMR + 0x119BA
RCTR7	IMMR + 0x119BC
RCTR8	IMMR + 0x119BE

1 Trap registers 5–8 only available on 0.13-µm (HiP7) devices.

Trap registers should be programmed to the value stated in the user download instructions accompanying the s record or C language array. Note that there are many advantages in this ROM/RAM-based microcode architecture. This allows custom microcode or enhancements to existing microcodes without the need for a PCB re-spin or another revision of silicon.

4.3 CPCR

The communication processor configuration register (CPCR) allows the G2 core to instruct the RISC to perform a pre-defined microcoded host command or host commands. This register usage is essential for proper configuration and operation of the CPM peripherals. Both the MCC- and FCC-based protocols must be provided an INIT command; otherwise, erratic operation will occur.

When the G2 writes a command to this memory-mapped register, a request is placed in the RISC scheduler at a high priority (priority 4). This means that host commands take precedence over serial FIFO-based requests. These host commands are essential in order to give the application software full control over the communications tasks running on the CP RISC. For example, a host command could be issued to stop a particular protocol or channel and then another host command could be issued later in order to restart the protocol or even another protocol on the same serial channel. By using this register, the application can yield full control over the CP RISC.

5 Download Procedure

The RAM microcode package can be downloaded to PowerQUICC II silicon by completing Steps 1-5. There should not be any CPM activity until RAM microcode is downloaded (Steps 1-3) and enabled, via the trap registers (Step 4) and the RCCR (Step 5).

1. Clear RCCR[ERAM].
2. Download the RAM microcode package provided in three formats as follows:
   — ads script (Downloadable via internal low-level debugger)
   — c array (To be added to user source code during initialization)
   — s record (Binary file downloadable via debugger on the COP interface)
3. Users must ensure that software is downloaded to the correct address for IMMR.

NOTE

Silicon-specific starting address

- HiP3—IMMR + 0x0000
- HiP4—IMMR + 0x4000 and/or IMMR +0x0000
- HiP7—IMMR + 0x20000

4. Set the RCCR[ERAM] according to the silicon revision and RAM microcode documentation.
5. Set trap registers according to the silicon revision and RAM microcode documentation.

Users should never set trap registers without first locking the microcode by setting the RCCR[ERAM] field to the value instructed in the user download documentation.

6 Example C Code

This example C code for RAM microcode initialization was generated and compiled specifically to run on the MPC82xx FADS pilot board. However, it can be readily adapted to run on other hardware platforms. The source files implemented include two C files, two header files, and one assembly startup file as listed below. These files accompany this application note.

- **download.c—**Main RAM microcode initialization ‘C’ source file.
- **ucode.c—**C arrays for RAM microcode (In this example microcode patch for errata CPM115)
- **mpc82xx.h**—General data structures for the FCC parameter RAM and internal memory map (IMMR) for the MPC82xx registers.
- **netcomm.h**—Header file containing global data type definitions.
- **startup.s**—Initializes the stack, Instruction and Data Caches, Memory Management Unit (MMU), and local bus initialization.

The download.c driver uses certain functions to initialize and implement microcode initialization on the MPC82xx. The functions and their purpose are given in Table 7 below.

Table 7. download.c Functions

Function	Purpose
main()	Responsible for calling the respective initialization routines. Generates internal memory map pointer, initializes RCCR and microcode trap registers.
RevNum()	This function determines the silicon revision from the IMMR[Mask_Num] and CPM Revision Number.
LoadUCODE(DPR_BASE)	This function loads the correct RAM microcode patch using a C array for errata CPM115 based on the silicon revision.
SetRegs()	This function initializes the RCCR and microcode trap registers based on the silicon revision.
GP1led()	Turns ON LED GP1 on MPC82xx FADs board
FlashGP1led()	Flashes ON / OFF LED GP1 on MPC82xx FADs board

7 Document Revision History

Table 8 below provides a revision history on this document

Table 8. Document Revision History

Rev. No.	Date	Substantive Change(s)
1	01/06/2005	Updated Section 5, “Download Procedure.” Made minor edits.
0	09/25/2004	Initial release of document.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
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.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-0047 Japan
0120 191014
+81 3 3440 3569
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
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@
hibbertgroup.com

AN2754
Rev. 1
01/2005

freescale™
semiconductor