Functional Differences Between MSC8101 (Mask 2K42A) and MSC8103 (Mask 2K87M)

1 Introduction

This document describes the differences between MSC8101 mask set 2K42A and MSC8103 mask set 2K87M. These differences include:

- System Interface Unit (SIU) changes:
  — Internal Memory Map Register (IMMR) MASKNUM field value
  — Addition of the Internal Memory Map Mirror Register (IMMMR)
- Reset changes:
  — Modes
  — Hard Reset Configuration Word (HRCW) layout
- Boot source changes
- Clock changes
  — Clock scheme
  — System Clock Mode Register (SCMR)
  — Addition of the CLKODIS field to the System Clock Control Register (SCCR)
  — Clock modes
  — CLKIN to CLKOUT delay change
  — Maximum clock frequencies change
- Memory map addition of IMMMR
- ORx in UPM mode, bit 27 functionality difference
- Host interface (HDI16) changes
- Direct memory access (DMA) controller transfer code (TC) definitions

CONTENTS

freescale™ semiconductor

- Interrupt system changes:
  — CPM Low Interrupt Priority Registers (SCPRR_L and SCPRR_L_EXT) definitions
  — Assignment of interrupt vector 44 to transmission convergence (TC) layer request
- Debugging system changes:
  — JTAG ID
  — EOnCE Status Register (ESR) values
- Removal of Enhanced Filter Coprocessor (EFCOP) support
- Communications processor module (CPM) changes:
  — RISC Controller Configuration Register (RCCR)
  — Dual-port RAM
  — Addition of ROM-based inverse multiplexing for ATM (IMA) microcode
  — Addition of TC layer functionality to the time-slot assigner (TSA) with support through FCC2
  — Addition of new MCC host commands
- Errata
- New bootloader program

2 Summary of Differences

Table 1 lists a summary of the differences between MSC8101 mask set 2K42A and MSC8103 mask set 2K87M.

Table 1. Difference Summary for MSC8101 Mask Set 2K42A and MSC8103 Mask Set 2K87M

Module	Function	MSC8101 Mask Set 2K42A	MSC8103 Mask Set 2K87M
System Interface Unit (SIU)	Internal Memory Map Register (IMMR)	MASKNUM bit field = 0x02	MASKNUM bit field = 0x12
	Internal Memory Map Mirror Register (IMMMR)	Not supported	Added
Reset	Supported modes	Host reset and hardware reset	Host reset, hardware reset, and reduced reset
	Hardware reset configuration word	Seventeen fields defined	Eighteen fields defined--software watchdog disable (SWDIS) bit added
Boot	Boot sources supported	From host (HDI16) or external memory (system bus)	From host (HDI16), external memory (system bus), or serial EPROM (I^2C interface)
Clock	Clock Scheme	Clock scheme configures SPLL PDF, SPLL MF, and Bus DF	Clock scheme configures SPLL PDF, SPLL MF, Bus DF, CPM DF, Core DF, CPLL PDF, and CPLL MF
	System Clock Mode Register (SCMR)	Defines seven fields	Defines eight fields
	System Clock Control Register (SCCR)	Defines one field (DFBRG)	Defines two fields (CLKODIS and DFBRG)
	Clock modes	Two valid modes	Twenty-seven valid modes
	CLKIN-to-CLKOUT delay	A function of frequency	Not a function of frequency
	Enabling the DLL	DLL-enabled mode is not supported and designs must use a zero-delay clock buffer.	To enable the DLL, the zero-delay clock buffer recommended for the 2K42A mask set must be placed in PLL-bypass mode or replaced.
	Disabling the DLL	—	For clock modes 5, 6, 46, and 57, apply offsets to DLL-disabled timing.
	Maximum clock frequencies	BCLK/CLKOUT/SCLK = 68.75 MHz CPMCLK = 137.5 MHz DSPCLK = 275 MHz	BCLK/CLKOUT/SCLK = 100 MHz CPMCLK = 200 MHz DSPCLK = 300 MHz

Table 1. Difference Summary for MSC8101 Mask Set 2K42A and MSC8103 Mask Set 2K87M (Continued)

Module	Function	MSC8101 Mask Set 2K42A	MSC8103 Mask Set 2K87M
Memory Map	IMMMR	Not supported	Added
	SCCR	Not supported	Added
Memory Controller	ORx in UPM mode	Bit 27 is a Wait State bit. When bit 27 is set, it add one wait state to the memory cycle.	Bit 27 is reserved.
Host Port (HDI16)	ICR and ISR	ICR and ISR do not support HDI6 bursts of different sizes	ICR and ISR redefined to support HDI16 burst of several sizes.
Direct Memory Access (DMA) controller	Transfer Code (TC) bit definitions	As defined in **Table 31** of this document.	The Transfer Code (TC) bit definitions are modified to conform to the original specification as listed in **Table 31** of this document.
Interrupts	SCPRR_L and SCPRR_L_EXT YC1P–YC8P field definitions	The value 100 is reserved.	The value 100 = “TC layer asserts its request in the YCCn position.”
	Interrupt vector 44	Reserved	Assigned to TC layer interrupt.
JTAG/ EOnCE system	JTAG ID	0x0188101D	0x1188101D
	EOnCE Status Register (ESR) values	ESR[REVNO] = 1 ESR[CORETP] = 0	ESR[REVNO] = 2 ESR[CORETP] = 2
EFCOP	Enhanced Filter Coprocessor	Supported as defined in the *MSC8101 Reference Manual*	Not supported
CPM	RISC Controller Configuration Register (RCCR)	Enable RAM Microcode (ERAM) field 3 bits wide	Enable RAM Microcode (ERAM) field changed to 4 bits wide
	Dual-port RAM	24 KB	32 KB
	IMA functionality	Not available	Added
	Serial Interface (SI) and time-slot assigner (TSA)	No TSA layer functionality.	TC layer functionality added to the TSA for ATM with a new interrupt (vector 44)
All	Errata	Documented in the current errata list	See **Table 48**, **Table 49**, and **Table 50** for details.
Bootloader Program	Code list	Available upon request.	Listed in **Appendix A**

3 SIU

3.1 Internal Memory Map Register (IMMR) Change

The IMMR is updated in 2K87M to reflect the mask change. The MASKNUM bit field has changed to 0x12 to reflect the correct revision level.

3.1.1 2K42A Mask Set IMMR Values

IMMR Internal Memory Map Register **0x101A8**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ISB															—
Type	R/W															
Reset	Depends on reset configuration sequence.															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	PARTNUM								MASKNUM							
Type	R															
Reset	—															

IMMR identifies a specific device as well as the base address for the internal memory map. Software can deduce availability and location of any on-chip system resources from the values in IMMR. PARTNUM and MASKNUM are mask programmed and cannot be changed for any particular device.

Table 2. IMMR Bit Descriptions

Bits	Description	Settings
ISB 0–14	**Internal Space Base** Defines the base address of the internal memory space. The value of ISB is configured at reset to one of seven addresses; the software can then change it to any value. The default address is based on the ISB bits in the Hard Reset Configuration Word. The default is zero, which maps to address 0xF0000000. ISB defines the 15 msbs of the memory map register base address. IMMR itself is mapped into the internal memory space region. As soon as the ISB is written with a new base address, the IMMR base address is relocated according to the ISB. ISB enables the configuration of multiple-MSC8101 systems.	Implementation-dependent
— 15	Reserved. Write to zero for future compatibility.	
PARTNUM 16–23	**Part Number** This field is mask-programmed with a code corresponding to the part number of the part on which the SIU is located. It helps factory test and user code that is sensitive to part changes. This field changes when the part number changes. For example, it would change if any new module is added or if the size of any memory module changes. It does not change if the part is changed to fix a bug in an existing module.	The MSC8101 has an ID of 0x50.
MASKNUM 24–31	**Mask Number** This field is mask-programmed with a code corresponding to the mask number of the part on which the SIU is located. It helps factory test and user code that is sensitive to part changes. It is programmed in a commonly changed layer and should be changed for all mask set changes.	The MSC8101 mask set 2K42A has an ID of 0x02.

3.1.2 2K87M Mask Set IMMR Values

IMMR Internal Memory Map Register **0x101A8**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ISB															—
Type	R/W															
Reset	Depends on reset configuration sequence.															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	PARTNUM								MASKNUM							
Type	R															
Reset	—															

IMMR identifies a specific device as well as the base address for the internal memory map. Software can deduce availability and location of any on-chip system resources from the values in IMMR. PARTNUM and MASKNUM are mask programmed and cannot be changed for any particular device.

Table 3. IMMR Bit Descriptions

Bits	Description	Settings
ISB 0–14	**Internal Space Base** Defines the base address of the internal memory space. The value of ISB is configured at reset to one of seven addresses; the software can then change it to any value. The default address is based on the ISB bits in the Hard Reset Configuration Word. The default is zero, which maps to address 0xF0000000. ISB defines the 15 msbs of the memory map register base address. IMMR itself is mapped into the internal memory space region. As soon as the ISB is written with a new base address, the IMMR base address is relocated according to the ISB. ISB enables the configuration of multiple-MSC8103 systems.	Implementation-dependent
— 15	Reserved. Write to zero for future compatibility.	
PARTNUM 16–23	**Part Number** This field is mask-programmed with a code corresponding to the part number of the part on which the SIU is located. It helps factory test and user code that is sensitive to part changes. This field changes when the part number changes. For example, it would change if any new module is added or if the size of any memory module changes. It does not change if the part is changed to fix a bug in an existing module.	The MSC8103 has an ID of 0x50.
MASKNUM 24–31	**Mask Number** This field is mask-programmed with a code corresponding to the mask number of the part on which the SIU is located. It helps factory test and user code that is sensitive to part changes. It is programmed in a commonly changed layer and should be changed for all mask set changes.	The MSC8103 mask set 2K87M has an ID of 0x12.

3.2 Internal Memory Map Mirror Register (IMMMR)

The 2K87M mask set adds a new Internal Memory Map Mirror Register (IMMMR) which has a fixed address controlled by the QBus Bank 1 (0x00F8FFC0) and the same register fields as the IMMR. It reflects the contents of the IMMR. If the ISB in the IMMR is modified, the base address of all SIU and CPM registers, including the IMMR, changes to the new value selected by the ISB. In such a case, the device or external masters may not be able to access the registers. Since the IMMMR does not reside in the same base memory area, it is always available at its fixed address. You can read the current ISB value from the IMMMR and determine the current internal base address from that value.

4 Reset

The 2K42A mask set offers two options for programming the Hard Reset Configuration Word (HRCW). The 2K87M mask set offers three options. In addition, the structure of the HRCW itself is different between the two mask sets.

4.1 Reset Configuration Options

Table 4 shows the options for programming the HRCW for each of the mask sets.

Table 4. HRCW Programming Options by Mask Set

2K42A	2K87M
Host reset. Through the HDI16 from a host *Hardware reset*. Through the system bus from external memory	*Host reset*. Through the HDI16 from a host. *Hardware reset*. Through the system bus from external memory. *Reduced reset*. From a serial EPROM using the I^2C protocol. Limited fields of the hard reset configuration word are configured via the appropriate data bus bits on the system bus.

4.1.1 Configuration Modes

The 2K42A mask set defines the configuration modes as shown in **Table 5**.

Table 5. MSC8101 2K42A Mask Set Reset Configuration Modes

RSTCONF	HPE/EE1	BTM[0-1]/ EE[4-5]	Mode
1	1	01	host reset configuration
0	0	00	master hardware reset configuration
1	0	00	slave hardware reset configuration

The 2K87M mask set adds a third “reduced reset configuration” method of programming the reset configuration word, in addition to the host reset and hardware reset configuration methods.

Table 6. MSC8103 2K87M Mask Set Reset Configuration Modes

RSTCONF	HPE/EE1	BTM[0-1]/ EE[4-5]	Mode
1	1	01	host reset configuration
0	0	00	master hardware reset configuration
1	0	00	slave hardware reset configuration
0	0	10	master reduced reset configuration
1	0	10	slave reduced reset configuration

4.1.2 Reduced Reset Configuration Information for Mask Set 2K87M

Reduced reset configuration is executed for serial boot only. Only the NMI_OUT (bit 12), ISB (bits 13–15), SWDIS (bit 26), and DLLDIS (bit 27) fields in the Hard Reset Configuration Word (HRCW) can be programmed using this method. The rest of the bits are programmed to their default values, but they can be reprogrammed later after reset. **Table 7** describes the 2K87M mask set hard reset configuration word. .

The major features of the reduce reset configuration include:

- The MSC8103 samples the signals described in **Table 6** at the rising edge of $\overline{\text{PORESET}}$. If the configured boot mode is serial, the values for the NMI_OUT, ISB, SWDIS, and DLLDIS fields are programmed from the system data bus. The mapping of the bits on the data bus is the same as for hardware reset configuration mode.
- MODCK_H cannot be programmed using this method and the value is set by default to 000. Therefore, only clock modes 0–1 and 4–7 can be used with this boot mode.
- In reduced configuration mode, the internal reset is extended for 1024 CLKIN cycles.
- Although there is no default HRCW in this mode, the bits in the HRCW that are not programmed assume the default values.
- During the first 8 CLKIN cycles D[12–15] and D[26–27] are sampled to configure the NMI_OUT, ISB, SWDIS, and DLLDIS fields in the HRCW. All other data bus bits are ignored. The simplest configuration scenario, where all data bus pins are ignored is not available for this mode.
- $\overline{\text{HRESET}}$ in simple slave mode does not change the reset configuration. In master/slave mode, $\overline{\text{HRESET}}$ does cause a new reset configuration.
- Configuration from EPROM for single or multiple chip is available only for the mentioned above 6 bits. All other bits on data bus are ignored.
- Multiple chip configuration in a system with no EPROM is also available only for the mentioned above 6 bits. All other bits on data bus are ignored.

4.2 Hard Reset Configuration Word Changes

The 2K87M mask set of the MSC8103 adds the SWDIS field (bit 26) and changes the definition of the ISB field (bits 13–15).

MSC8103 2K87M Mask Set Hard Reset Configuration Word

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EARB	EXMC	IRQ7 INT	EBM	BPS		SCDIS	ISPS	IRPC		DPPC		NMI OUT	ISB		
Type								R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	—	BBD	MMR		—		TCPC		BC1PC		SWDIS	DLLDIS	MODCK_H			—
Type								R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7. MSC8103 2K87M Mask Set Hard Reset Configuration Word Bit Descriptions

Name	Reset	Description	Settings
EARB 0	0	**External Arbitration** Defines the initial value for PPC_ACR[EARB].	0 = No external arbitration is assumed 1 = External arbitration is assumed
EXMC 1	0	**External MEMC** Defines the initial value of BR0[EMEMC].	0 = No external memory controller is assumed 1 = External memory controller is assumed
IRQ7 INT 2	0	**IRQ7 or INT_OUT Selection**	0 = $\overline{\text{IRQ7}}$/$\overline{\text{INT\_OUT}}$ pin is $\overline{\text{IRQ7}}$ 1 = $\overline{\text{IRQ7}}$/$\overline{\text{INT\_OUT}}$ pin is $\overline{\text{INT\_OUT}}$
EBM 3	0	**External 60x-compatible Bus Mode** Defines the initial value of BCR[EBM].	
BPS 4–5	0	**Boot Port Size** Defines the initial value of BR0[PS], the port size for memory controller bank 0.	00 = 64-bit port size 01 = 8-bit port size 10 = 16-bit port size 11 = 32-bit port size

Table 7. MSC8103 2K87M Mask Set Hard Reset Configuration Word Bit Descriptions (Continued)

Name	Reset	Description	Settings
SCDIS 6	0	**SC140 Disabled** Enables/disables the SC140. This bit cannot be changed after reset.	0 = SC140 enabled 1 = SC140 disabled
ISPS 7	0	**Internal Space Port Size** Defines the initial value of BCR[ISPS]. This bit must be set in order to use the host interface. Setting ISPS configures the MSC8103 to respond to accesses from a 32-bit external master to its internal space. This bit cannot be changed after reset.	0 = 60x-compatible data bus is 64 bits wide 1 = 60x-compatible data bus is 32 bits wide (lower 32 bits; upper 32 bits reserved for HDI16
IRPC 8–9	0	**Interrupt Pin Configuration** Defines the initial value of SIUMCR[IRPC].	
DPPC 10–11	0	**Data Parity Pin Configuration** Defines the initial value of SIUMCR[DPPC].	
NMI OUT 12	0	**NMI OUT** Defines the host core to handle a non-maskable Interrupt (NMI) event.	0 = NMI is serviced by SC140 core 1 = NMI is routed to external pin and serviced by the external host
ISB 13–15	0	**Initial Internal Space Base Select** Defines the initial value of IMMR(ISB[0–14]) and determines the base address of the internal memory space and the DSPRAM base address on the local bus. Note that the SC140 core internal address space spans from 0x00000000–0x00FFFFFF (16 MB). Therefore it is not advisable to map the Internal Memory Map Register (IMMR) in this space, since the SC140 core cannot access the registers of the SIU and CPM. QBus banks are mapped to address 0x00F00000, so using ISB value 101 causes the Dual-Port RAM (DPRAM) address space and the QBus address space to overlap. Modifying the QBus Base Registers to another address allows the user to access the DPRAM address space.	**ISB (10)** / **Internal Memory** / **DSPRAM (Bank 10)** 000 / 0xF0000000 / 0x02000000 001 / 0xF0F00000 / 0x03000000 010 / 0xFF000000 / 0x04000000 011 / 0xFFF00000 / 0x05000000 100 / Reserved / Reserved 101 / 0x00F00000 / 0x07000000 110 / 0x0F000000 / 0x08000000 111 / 0x0FF00000 / 0x09000000 **Note:** The 2K87M mask adds the DSPRAM location to the areas controlled by these bits.
— 16	0	Reserved. Write to zero for future compatibility.	
BBD 17	0	**Bus Busy Disable** Defines the initial value of SIUMCR[BBD].	
MMR 18–19	0	**Mask Masters Requests** Defines the initial value of SIUMCR[MMR].	
— 20–21	0	Reserved. Write to zero for future compatibility.	
TCPC 22–23	0	**Transfer Code Pin Configuration** Defines the initial value of SIUMCR[TCPC].	
BC1PC 24–25	0	**BC1PC Value** Defines the initial value of SIUMCR[BC1PC].	
SWDIS 26	0	**Software Watchdog Disable** Defines the initial state of the Software Watchdog Timer	0 = Software Watchdog Timer enabled 1 = Software Watchdog Timer disabled **Note:** This field is undefined in the 2K42A mask set.
DLLDIS 27	0	**DLL Disable** Defines whether the DLL mechanism is disabled.	0 = No DLL bypass 1 = DLL bypass
MODCK_H 28–30	0	**MODCK High Order Bits** High-order bits of the MODCK bus, which determine the clock reset configuration.	
— 31	0	Reserved. Write to zero for future compatibility.	

5 Boot

The 2K42A mask set does not support a serial boot mode. The 2K87M mask set supports a serial boot mode with a reduced reset configuration. The following sections describe the serial boot mode operations.

5.1 Boot Mode Definition Changes

The boot modes for mask set 2K42A are defined by the values of BTM[0–1] when sampled on the rising edge of $\overline{\text{PORESET}}$, as follows:

BTM0	BTM1	Boot Source
0	0	External memory
0	1	HDI16
1	0	Reserved
1	1	Reserved

The boot modes for mask set 2K87M are defined by the values of BTM[0–1] when sampled on the rising edge of $\overline{\text{PORESET}}$, as follows:

BTM0	BTM1	Boot Source
0	0	External memory
0	1	HDI16
1	0	Serial interface
1	1	Reserved

5.2 Serial EPROM Boot Procedure

The MSC8103 core programs the I^2C registers and parameter RAM and prepares buffers and buffer descriptors (BDs) for the boot loading process. The MSC8103 I^2C interface is programmed as the master, and it accesses the slave address 0b1010111 to read the boot source code. Booting the MSC8103 through the serial EPROM is useful for systems in which the MSC8103s are connected only through serial interfaces, such as TDM, Ethernet, ATM, and so forth.

Note: The bootloader supports access to serial EPROMS with a user-defined device address and 2-byte address specifications.

5.3 Software Watchdog Handling

The software watchdog timer can be enabled by clearing the SWDIS bit as part of the reduced reset configuration process. In serial boot mode, if the SWDIS bit is cleared, the bootloader program periodically executes a special service sequence to prevent the time-out of its counter and the assertion of a hardware reset during the bootloading process. For details on the software watchdog, refer to **Section 4.2.5**, *Software Watchdog Timer*, and **Section 4.3**, *SIU Programming Model* in the *MSC8103 Reference Manual*.

5.4 Source Program Data Stream Structure

The source program can be organized into several blocks. Each block can be either a data block or an instruction block, and it can be loaded to a different specified destination. Each block contains the block size, the location where the block is loaded, source program words, checksum, checksum enable and the location from which the next block is loaded.

When each block word is loaded, the routine calculates a checksum by XORing the current word bit by bit with the result of XORing previous words. The value of bit *i* of the current result is equal to XORing bit *i* of the current word with bit *i* of the previous result. After the entire block is loaded, the calculated checksum is compared to the loaded checksum. The checksum comparison and calculation can be skipped by clearing checksum enable bit. The last block is a special block that indicates end of code. It contains the boot execution start address. **Figure 1** shows the stream structure.

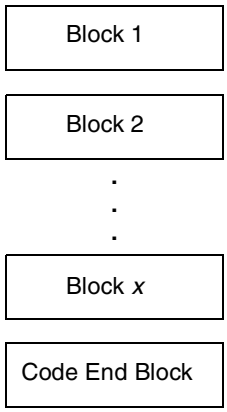


Figure 1. Boot Code Stream Structure

5.5 Source Program Block Structure

The data stream source programs must be structured in the format shown in **Table 8**.

Table 8. Block Structure

Word	Description
1	Block size + checksum enable bit (see **Figure 2** for layout description)
2	Next block address
3	Address where the first block of the source program is to be loaded, most significant part
4	Address where the first block of the source program is to be loaded, least significant part
5	First word of source program
n	Last word of source program
n + 1	Checksum—xor for first block
n + 2	Checksum—$\overline{\text{xor}}$ for first block
2nd offset + 1	Block size + checksum enable bit
2nd offset + 2	Next block address
2nd offset + 3	Address where the second block of the source program is to be loaded, most significant part
2nd offset + 4	Address where the second block of the source program is to be loaded, least significant part
2nd offset + 5	First word of source program
2nd offset + n	Last word of source program
2nd offset +n + 1	Checksum—xor for second block
2nd offset + n + 2	$\overline{\text{Checksum}}$—$\overline{\text{xor}}$ for second block

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	CSE[1]	Block Size														

Notes: **1.** CSE = checksum enable. Set this bit when a checksum comparison is needed.
2. Since the EPROMS that support the I^2C protocol are small, 15 bits are sufficient to define the block size.

Figure 2. Checksum Enable Bit + Block Size

In addition, the following rules apply:

- The source data must be in big-endian format, with the most significant part at the lower-order address.
- To enable checksum, set the CSE bit. To disable checksum, clear the CSE bit.
- Block size includes the checksum and $\overline{\text{checksum}}$.
- Each address must be aligned on a 16-byte boundary.
- Maximum size for a block is 64 KB.
- Checksum is performed on all block words, including addresses and sizes.
- The block size should be a multiple of 32 bits.
- If the next block address is 0x0, the bootloader treats the next block as sequential.
- All addresses should be located in SRAM. The address should be given as a DSP internal address.

The end of the boot code stream is indicated by a special boot end block with the structure shown in **Table 9**.

Table 9. Structure of the Boot End Block

Word	Description
1	0x0000
2	0x0000
3	Boot start address, most significant part
4	Boot start address, least significant part
5	0x0000
6	0x0000
7	Checksum—xor
8	$\overline{\text{Checksum}}$—$\overline{\text{xor}}$

The first two words indicate the end of the source blocks. At least one block of source code must be loaded when the bootloader is invoked. The boot start address indicates the address at which the boot program execution starts. This address must be aligned on a 16-byte boundary. The bootloader routine expects at least one code block in addition to the boot end block. The sequence is repeated for subsequent blocks, until the final block in the data stream is reached.

5.6 Load Procedure

The bootloader program uses the I^2C serial interface to access data in the EPROM and to program the I^2C parameter RAM and registers. The I^2C uses BDs and buffers to read and write data. The bootloader prepares and keeps track of the BDs and buffers for program loading.

To enable multi-master support, the first 14 bytes of the EPROM are reserved for an address table, which is also accessed in single-master mode. The table entries contain the location of the boot code.The loading process starts by calculating the entry address and reading the boot code location address. Then the code loading starts by reading the first 4 block words from the EPROM. The size of block, where to load it, the location of the next block in EPROM, and checksum enable are extracted.

Note: The bootloader supports access to serial EPROMs up to 512 KB with an address specification that is accessed by 2 bytes.

The bootloader allocates a buffer for the first code block according to the address and size given in the opening 4 words and creates a BD that describes it. After the BD is ready, the bootloader starts reading the code block from the EPROM. If checksum is enabled, the bootload calculates a checksum on the block word and compares the calculated checksum to the loaded checksum. If the checksum is correct, the bootloader continues to the next block. If the checksum fails, the bootloader tries to read the block again. After a second unsuccessful try, the bootloader aborts.

To skip the checksum comparison, clear the checksum enable bit. If checksum is disabled, the bootloader program continues loading the next code block after finishing the current one. When all code blocks are loaded, boot execution starts from the start address given in the end block.

The bootloader program polls the BDs to check when a code block has been received. It does not use interrupts.

The load procedure is described in **Figure 3**.

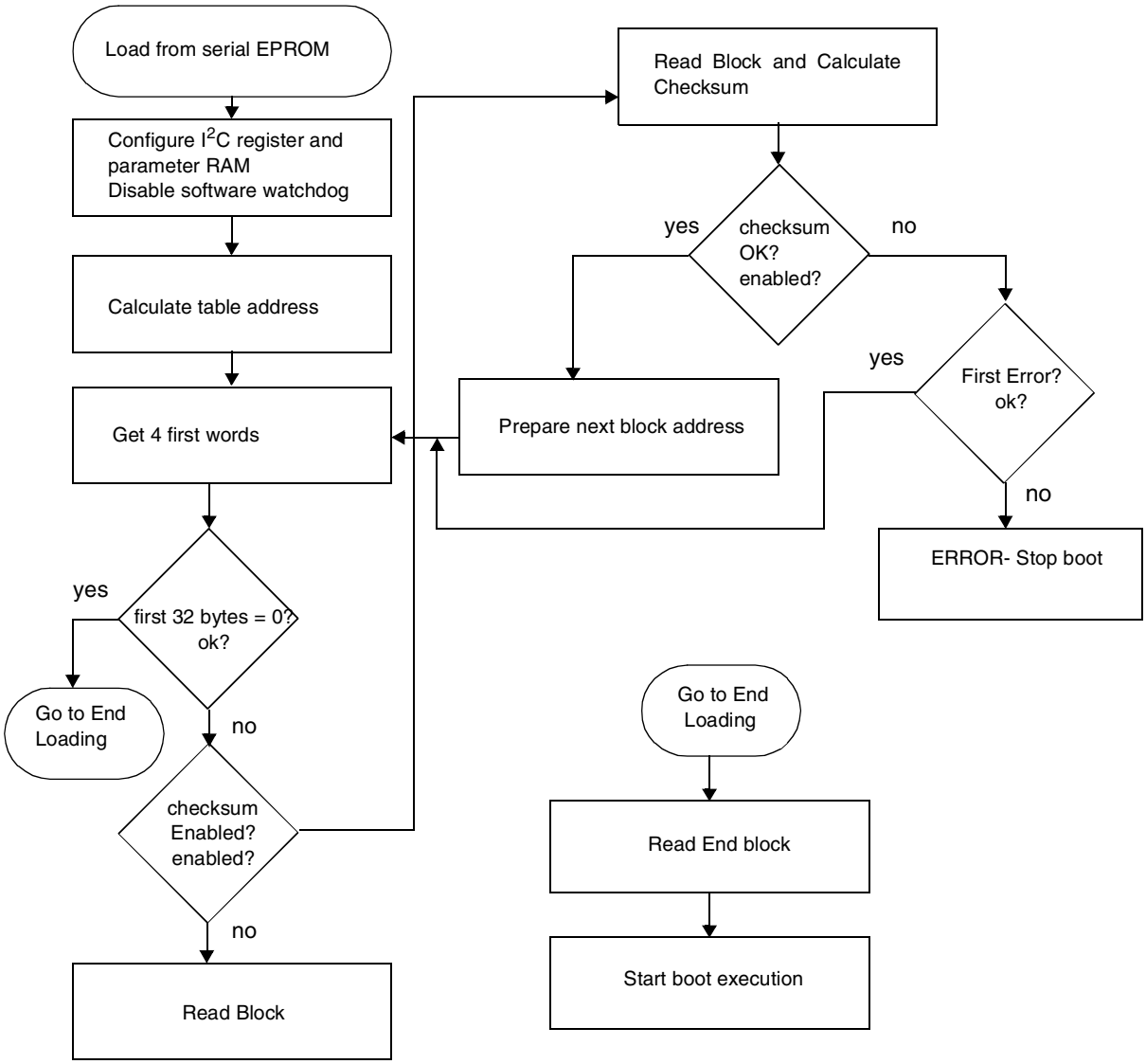


Figure 3. Serial Boot Load Procedure

5.7 I²C Clocking

The MSC8103 I²C interface accesses the serial EPROM using a clock frequency derived by dividing the core frequency by 4000. This division sets the I²C clock to 75 KHz or less, a clock frequency that most EPROMs support. If the MSC8103 I²C gets an NAK during transmission, that is, the EPROM does not respond to the address asserted by MSC8103, the boot loader stops execution. If the software watchdog timer is enabled, the counter will eventually expire causing a system reset. After reset, the bootloading process restarts.

5.8 Reset Configuration Word

The only reset configuration mode that can be used while serial boot mode is enabled is reduced configuration word. In the reduced configuration word, only the NMI_OUT, ISB, SWDIS, and DLLDIS fields can be set. For details, see **Section 4.1.2**.

5.9 Default Programming During Boot

The bootloader routine provides default programming for parts of the MSC8103. The routine programs the UPMC and the GPCM as required to support the internal SRAM and peripherals. The UPMC is programmed to support the bus-to-CPM clock ratio. The routine checks the value of the SCMR[BUSDF] bits and selects the bus-to-CPM clock ratio. The routine also programs the correct values for the ELIR[A–F] registers in the programmable interrupt controller (PIC) to determine the interrupt mode (edge-triggered or level-triggered).

5.10 Multi-Processor Booting from a Serial EPROM

The I²C protocol supports multi-master environments. All MSC8103 booting devices act as I²C masters and try to access the serial EPROM. The first 14 bytes of the serial EPROM are reserved for an address table. Each master accesses a different entry in the table according to its ISB[0–2] bits in the IMMR, allowing seven I²C masters in the system. Each entry in the table contains the actual address where the boot program resides in the 2-byte increments. The master accesses the table, reads the boot location address, and starts loading boot from there.If two or more masters try to put information onto the bus, the first to produce a 1 when another produces a 0 loses the arbitration (collision). If a collision occurs, the master that loses the arbitration retries.

Note: If there are less than seven masters and not all 14 bytes of the table are needed, the boot code can start inside the first 14 bytes of the table.

A MSC8103 I²C master should access a table entry equal to its ISB[0–2] value multiplied by two. The bootloading process always starts by accessing the address table. In a single master environment, the boot code may start immediately after the address entry.

Note: During the booting process, multiple masters should not address each other. There is no support for multi-master errors.

5.11 Boot Code Changes

The boot code for the 2K87M mask set supports the I²C protocol. In addition, the code reflects changes in the device design that fix a host checksum errata, make the SRAM base address ISB dependent, and allow the software watchdog timer to be enabled or disabled via software. For a detailed listing of the updated boot code, see **Appendix A**.

6 Clock System

6.1 Clock Scheme Modifications

The MSC8101 mask set 2K42A and the MSC8103 mask set 2K87M use different clock schemes. The following paragraphs describe the clock schemes.

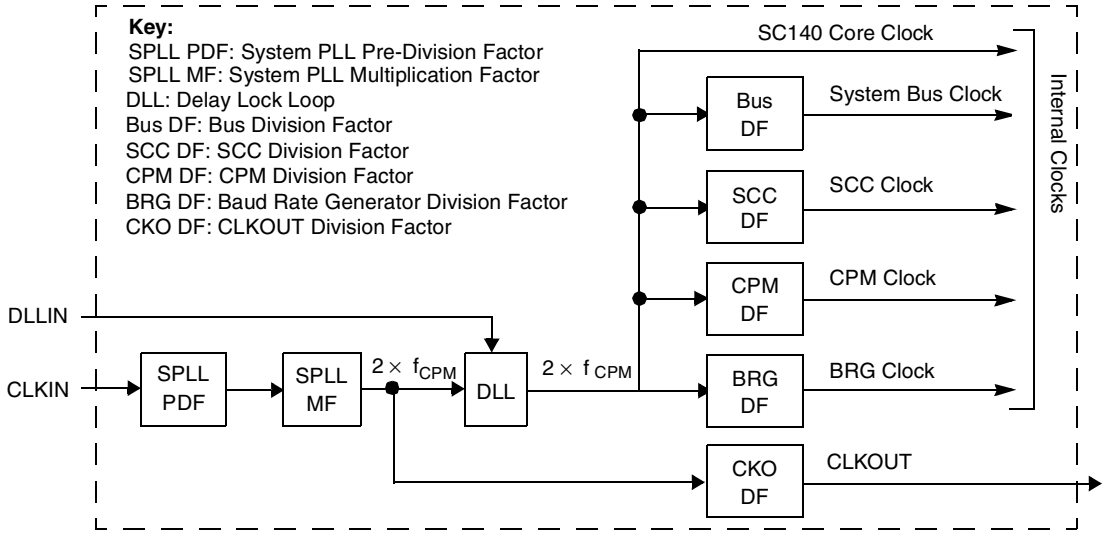
6.1.1 Mask Set 2K42A Clock Scheme

Six bit values map the MSC8101 clocks to one of the valid configuration mode options. Each option determines the CLKIN, SC140 core, system bus, SCC clock, CPM, and CLKOUT frequencies. The six bit values are derived from three dedicated input pins (MODCK[1–3]) and three bits from the reset configuration word (MODCK_H). To configure the SPLL pre-division factor, SPLL multiplication factor, and the frequencies for the SC140 core, SCC clocks, CPM parallel I/O ports, and system buses, the MODCK[1–3] pins are sampled and combined with the MODCK_H values when the internal power-on reset (internal $\overline{\text{PORESET}}$) is deasserted. Clock configuration changes only when the internal $\overline{\text{PORESET}}$ signal is deasserted.

The following factors are configured:

- SPLL pre-division factor (SPLL PDF)
- SPLL multiplication factor (SPLL MF)
- Bus post-division factor (Bus DF)

Figure 4 shows the functional block diagram for the 2K42A mask set. The 2K42A mask set contains an internal system phase-lock loop (SPLL)



Note: SPLL PDF determined by the clock configuration mode; SPLL MF determined by the clock configuration mode; Bus DF = CLKOUT DF and is 4 or 5, determined by the clock configuration mode; SCC DF is always 4; CPM DF is always 2; BRG DF is set by the System Clock Control Register (SCCR) and is 4, 16 (default), 64, or 256.

Figure 4. MSC8101 Clock Functional Block Diagram for 2K42A Mask Set

The SCC division factor (SCC DF) is fixed at 4 and the CPM division factor (CPM DF) is fixed at 2. The BRG division factor (BRG DF) is configured through the System Clock Control Register (SCCR) and can be 4, 16 (default after reset), 64, or 256.

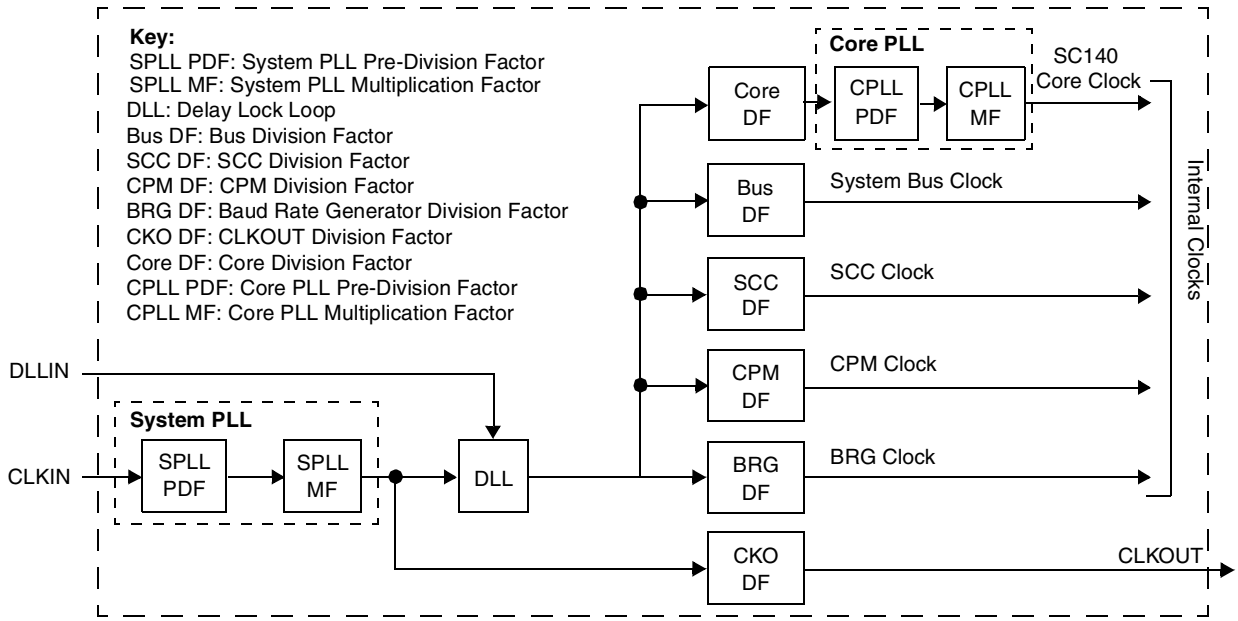
6.1.2 Mask Set 2K87M Clock Scheme

Six bit values map the MSC8101 clocks to one of the valid configuration mode options. Each option determines the CLKIN, SC140 core, system bus, SCC clock, CPM, and CLKOUT frequencies. The six bit values are derived from three dedicated input pins (MODCK[1–3]) and three bits from the hard reset configuration word (MODCK_H). To configure the SPLL pre-division factor, SPLL multiplication factor, and the frequencies for the SC140 core, SCC clocks, CPM parallel I/O ports, and system buses, the MODCK[1–3] pins are sampled and combined with the MODCK_H values when the internal power-on reset (internal $\overline{\text{PORESET}}$) is deasserted. Clock configuration changes only when the internal $\overline{\text{PORESET}}$ signal is deasserted.

The following factors are configured:

- SPLL pre-division factor (SPLL PDF)
- SPLL multiplication factor (SPLL MF)
- Bus post-division factor (Bus DF)
- CPM division factor (CPM DF)
- Core division factor (Core DF)
- CPLL pre-division factor (CPLL PDF)
- CPLL multiplication factor (CPLL MF)

Figure 5 shows the functional block diagram for the 2K87M mask set. The 2K87M mask set contains an internal system phase-lock loop (SPLL) and a core phase-lock loop (CPLL).



Note: SPLL PDF, SPLL MF, Core DF, CPLL PDF, CPLLMF, and CPM DF are determined by the clock configuration mode; Bus DF = CKO DF and is determined by the clock configuration mode; SCC DF is always 4; BRG DF is set by the System Clock Control Register (SCCR) and is 4, 16 (default), 64, or 256.

Figure 5. MSC8103 Clock Functional Block Diagram for 2K87M Mask Set

6.2 System Clock Mode Register (SCMR) Changes

The 2K42A and 2K87M mask sets use different SCMR definitions. The following sections describe the SCMR definitions for each mask set.

6.2.1 2K42A Mask Set SCMR Field Definitions

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—		COREPDF		COREMF				BUSDF				CPMDF			
Type									R							
Reset									—							
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	SPLLPDF				SPLLMF				—	DLLDIS	—					
Type									R							
Reset									—							

Figure 6. 2K42A Mask Set System Clock Mode Register (SCMR)—**0x10C88**

SCMR is a read-only register that is updated during power-on reset (PORESET) and provides the mode control signals to the PLLs, DLL, and clock logic. This register reflects the currently defined configuration settings.

Table 10. 2K42A Mask Set SCMR Bit Descriptions

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
— 0–1	—	—	Reserved	
COREPDF 2–3	Configuration Pins	Unaffected	**Core PLL Pre-Division Factor**	Not used.
COREMF 4–7	Configuration Pins	Unaffected	**Core Multiplication Factor**	Not used
BUSDF 8–11	Configuration Pins	Unaffected	**60x Bus Division Factor**	0010 Bus DF = 3 0011 Bus DF = 4 0100 Bus DF = 5 All other combinations not used.
CPMDF 12–15	Configuration Pins	Unaffected	**CPM Division Factor**	0001 CPM DF = 2 All other combinations are not used.
SPLLPDF 16–19	Configuration Pins	Unaffected	**SPLL Pre-Division Factor**	0000 SPLL PDF = 1 0001 SPLL PDF = 2 0010 SPLL PDF = 3 0011 SPLL PDF = 4 All other combinations not used
SPLLMF 20–23	Configuration Pins	Unaffected	**SPLL Multiplication Factor**	0110 SPLL MF = 12 0111 SPLL MF = 14 1000 SPLL MF = 16 1001 SPLL MF = 18 1010 SPLL MF = 20 1011 SPLL MF = 22 1100 SPLL MF = 24 1101 SPLL MF = 26 1110 SPLL MF = 28 1111 SPLL MF = 30 All other combinations not used
— 24	—	—	Reserved	
DLLDIS 25	Configuration Pins	Unaffected	**DLL Disable**	0 DLL operation is enabled 1 DLL is disabled

Table 10. 2K42A Mask Set SCMR Bit Descriptions (Continued)

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
— 26–31	—	—	Reserved	

6.2.2 2K87M Mask Set SCMR Field Definitions

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	COREPDF				COREMF				BUSDF				CPMDF			
Type									R							
Reset									—							
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	SPLLPDF				SPLLMF				—	DLLDIS	—		COREDF			
Type									R							
Reset									—							

Figure 7. 2K87M Mask Set System Clock Mode Register (SCMR)—**0x10C88**

SCMR is a read-only register that is updated during power-on reset (PORESET) and provides the mode control signals to the PLLs, DLL, and clock logic. This register reflects the currently defined configuration settings.

Table 11. 2K87M Mask Set SCMR Field Descriptions

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
COREPDF 0–3	Configuration Pins	Unaffected	**Core PLL Pre-Division Factor**	0000 CPLL PDF = 1 0001 CPLL PDF = 2 0010 CPLL PDF = 3 0011 CPLL PDF = 4 All other combinations not used.
COREMF 4–7	Configuration Pins	Unaffected	**Core Multiplication Factor**	0101 CPLL MF = 10 0110 CPLL MF = 12 0111 CPLL MF = 14 All other combinations not used.
BUSDF 8–11	Configuration Pins	Unaffected	**60x-compatible Bus Division Factor**	0001 Bus DF = 2 0010 Bus DF = 3 0011 Bus DF = 4 0100 Bus DF = 5 0101 Bus DF = 6 All other combinations not used.
CPMDF 12–15	Configuration Pins	Unaffected	**CPM Division Factor**	0000 CPM DF = 1 0001 CPM DF = 2 0010 CPM DF = 3 All other combinations not used.
SPLLPDF 16–19	Configuration Pins	Unaffected	**SPLL Pre-Division Factor**	0000 SPLL PDF = 1 0001 SPLL PDF = 2 0010 SPLL PDF = 3 0011 SPLL PDF = 4 0100 SPLL PDF = 5 0101 SPLL PDF = 6 All other combinations not used

Table 11. 2K87M Mask Set SCMR Field Descriptions (Continued)

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
SPLLMF 20–23	Configuration Pins	Unaffected	**SPLL Multiplication Factor**	0101 SPLL MF = 10 0110 SPLL MF = 12 0111 SPLL MF = 14 1000 SPLL MF = 16 1001 SPLL MF = 18 1010 SPLL MF = 20 1011 SPLL MF = 22 1100 SPLL MF = 24 1101 SPLL MF = 26 1110 SPLL MF = 28 1111 SPLL MF = 30 All other combinations not used
— 24	—	—	Reserved	
DLLDIS 25	Configuration Pins	Unaffected	**DLL Disable**	0 DLL operation is enabled 1 DLL is disabled
— 26–27	—	—	Reserved	
COREDF 28–31	Configuration Pins	Unaffected	**Core Division Factor**	0000 CORE DF = 1 0001 CORE DF = 2 0010 CORE DF = 3 0011 CORE DF = 4 0100 CORE DF = 5 0101 CORE DF = 6 All other combinations not used.

6.3 System Clock Control Register (SCCR)

The 2K87M mask set adds a new field (CLKODIS) to the SCCR. The functionality of the common field (DFBRG) is the same. **Figure 8** shows the register layout and **Table 12** lists the bit descriptions.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	Reserved															
Reset	—															
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	—											CLKODIS	—		DFBRG	
Type	Reserved											R/W	Reserved		R/W	
Reset	—											0	—		0	1

Figure 8. System Clock Control Register (SCCR)—**0x10C80**

Table 12. SCCR Bit Descriptions

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
— 0–26	—	—	Reserved. Write to 0 fro future compatibility.	
CLKODIS 27	0	Unaffected	**CLKOUT Disable** Disables the CLKOUT signal. The value of CLKOUT when disabled is indeterminate (can be 1 or 0).	0 CLKOUT enabled (default) 1 CLKOUT disabled **Note**: This bit is reserved in mask set 2K42A.

Table 12. SCCR Bit Descriptions (Continued)

Name Bit No.	Defaults		Description	Settings
	PORESET	Hard Reset		
— 28–29	—	—	Reserved. Write to 0 fro future compatibility.	
DFBRG 30–31	01	Unaffected	**Division Factor for the BRG Clock** Defines the BRGCLK frequency. Changing this value does not result in a loss of lock condition.	00 Divide by 4 01 Divide by 16 (default value) 10 Divide by 64 11 Divide by 256

6.4 Clock Mode Changes

6.4.1 2K42A Mask Set Clock Configuration Modes

The MSC8101 2K42A mask set supports the following valid clock modes:

Table 13. 2K42A Mask Set Clock Configuration Modes

Mode #	MODCK_H	MODCK[1–3]	SPLL PDF	SPLL MF	Bus DF	Ratio Bus:CPM:Core	Bus/ CLKIN
46	101	110	2	16	4	1 : 2 : 4	4
57	111	001	2	10	5	1 : 2.5 : 5	1

6.4.2 2K87M Mask Set Clock Modes

The MSC8103 2K87M mask set supports the clock modes listed in **Table 14**.

Table 14. 2K87M Mask Set Clock Configuration Modes

Mode #	MODCK_H[1]	MODCK[1–3][2]	SPLL PDF	SPLL MF	Bus DF	Core DF	CPM DF	CPLL PDF	CPLL MF	Ratio Bus:CPM:Core	Bus/ CLKIN
0	000	000	1	12	4	4	2	4	12	1 : 2 : 3	3
1	000	001	2	16	4	4	2	4	12	1 : 2 : 3	2
2	000	010	Reserved								
3	000	011	Reserved								
4	000	100	2	16	4	4	2	4	14	1 : 2 : 3.5	2
5	000	101	1	12	4	2	2	5	10	1 : 2 : 4	3
6	000	110	3	12	4	2	2	5	10	1 : 2 : 4	1
7	000	111	1	10	5	5	2	2	10	1 : 2.5 : 5	2
8	001	000	1	16	4	4	2	4	12	1 : 2 : 3	4
9	001	001	1	20	4	4	2	4	12	1 : 2 : 3	5
10	001	010	3	12	4	4	2	4	12	1 : 2 : 3	1
11	001	011	3	12	4	4	2	4	14	1 : 2 : 3.5	1
12	001	100	1	12	4	4	2	4	14	1 : 2 : 3.5	3
13	001	101	1	16	4	4	2	4	14	1 : 2 : 3.5	4
14	001	110	Reserved								
15	001	111	Reserved								
16	010	000	1	12	6	2	2	5	10	1 : 3 : 6	2
17	010	001	2	12	6	2	2	5	10	1 : 3 : 6	1
18	010	010	1	16	8	8	2	2	12	1 : 4 : 6	2
19	010	011	2	16	8	8	2	2	12	1 : 4 : 6	1
20	010	100	Reserved								
21	010	101	Reserved								

Table 14. 2K87M Mask Set Clock Configuration Modes (Continued)

Mode #	MODCK_H[1]	MODCK[1–3][2]	SPLL PDF	SPLL MF	Bus DF	Core DF	CPM DF	CPLL PDF	CPLL MF	Ratio Bus:CPM:Core	Bus/ CLKIN
22	010	110	Reserved								
23	010	111	Reserved								
24	011	000	2	12	6	6	2	3	12	1 : 3 : 4	1
25	011	001	1	12	6	3	2	5	10	1 : 3 : 4	2
26	011	010	1	18	6	3	2	5	10	1 : 3 : 4	3
27	011	011	2	10	5	3	2	5	12	1 : 2.5 : 4	1
28	011	100	Reserved								
29	011	101	Reserved								
30	011	110	Reserved								
31	011	111	2	12	6	8	2	3	12	1 : 3 : 3	1
32	100	000	Reserved								
33	100	001	2	16	8	8	4	2	10	1 : 2 : 5	1
34	100	010	1	12	4	4	2	2	10	1 : 2 : 5	3
35	100	011	2	16	8	8	4	2	12	1 : 2 : 6	1
36	100	100	1	16	8	8	4	2	12	1 : 2 : 6	2
37	100	101	Reserved								
38	100	110	Reserved								
39	100	111	Reserved								
40	101	000	Reserved								
41	101	001	Reserved								
42	101	010	Reserved								
43	101	011	Reserved								
44	101	100	Reserved								
45	101	101	Reserved								
46[3]	101	110	2	16	4	2	2	5	10	1 : 2 : 4	2
47	101	111	Reserved								
48	110	000	Reserved								
49	110	001	Reserved								
50	110	010	Reserved								
51	110	011	Reserved								
52	110	100	Reserved								
53	110	101	Reserved								
54	110	110	Reserved								
55	110	111	Reserved								
56	111	000	Reserved								
57[3]	111	001	2	10	5	5	2	2	10	1 : 2.5 : 5	1
58	111	010	Reserved								
59	111	011	Reserved								
60	111	100	Reserved								
61	111	101	Reserved								
62	111	110	Reserved								
63	111	111	Reserved								

Notes:
1. MODCK_H is a 3-bit field that occupies bits 28–30 of the Hard Reset Configuration Word. The bits are listed in the table in the following order: bit 28, bit 29, bit 30. For example, the value 110 indicates that bit 28 = 1, bit 29 = 1, and bit 30 = 0.
2. MODCK[1–3] are external signal inputs that are either pulled up (1) or pulled down (0) to configure the system clock mode. The values are listed in the table in the following order: MODCK1, MODCK2, MODCK3. For example, the value 110 indicates that MODCK1 is pulled up, MODCK2 is pulled up, and MODCK3 is pulled down.
3. This mode is compatible with the same mode for mask set 2K42A.

6.5 CLKIN-to-CLKOUT Delay Change

For the 2K42A mask set, the CLKIN-to-CLKOUT delay is a function of frequency. For the 2K87M mask set, the CLKIN-to-CLKOUT delay is a not a function of frequency. Therefore, the two versions cannot be used together in a DLL-disabled system.

6.6 Enabling the DLL

For the 2K42A, system erratum QSIU14 requires that the MSC8101 use only DLL-disabled mode and a zero-delay buffer. To use the MSC8103 mask set 2K87M in such a design with the DLL-enabled, the zero-delay buffer must either be placed in PLL-bypass mode or replaced.

6.7 Disabling the DLL

The 2K87M AC timing specification listed in the *MSC8103 Technical Data* sheet is for DLL-enabled mode only and is therefore referenced to DLLIN. For 2K87M clock modes 5, 6, 46, and 57 only, apply the following offsets to accommodate for the fact that the signal is referenced to CLKOUT instead of DLLIN:

- For SIU outputs, add 400 ps.
- For DMA outputs, add 400 ps to specification 76.
- For SIU inputs, subtract 400 ps.
- For DMA inputs, subtract 400 ps from specifications 72–75.

6.8 Clock Frequency Changes

lists the maximum clock frequencies for each mask set.

Table 15. Maximum Clock Frequencies for Each Mask Set

Clock	2K42A Maximum Frequency	2K87M Maximum Frequency
Input Clock (CLKIN)	68.75 MHz	75 MHz
Internal Bus Clock (BCLK)	68.75 MHz	100 MHz
Output Clock (CLKOUT)	68.75 MHz	100 MHz
SCC Clock (SCLK)	68.75 MHz	100 MHz
CPM Clock (CPMCLK)	137.5 MHz	200 MHz
SC140 Core Clock (DSPCLK)	275 MHz	300 MHz

Note: Refer to *MSC8103 Technical Data* for details on all frequency ranges and limitations.

7 Memory Map

The 2K87M mask adds the IMMMR to the memory map. The 2K87M memory map for Bank1 of the QBus is shown in **Table 16**.

Table 16. Mask Set 2K87M QBus Memory Map—Bank1

SC140 Core Internal Address	Mnemonic	Name	Size
Boot ROM			
0000–07FF	BOOTROM	MSC8103 boot ROM	2 KB
0800–FFBF	Reserved	Leave unchanged for future compatibility	63 KB – 64 bytes
FFC0	IMMMR	Internal Memory Map Mirror Register	32 bits
FFC4–FFFF	Reserved	Leave unchanged for future compatibility	60 bytes

8 ORx in UPM Mode

In UPM mode, the 2K42A mask uses the previously undefined bit 27 in the ORx to add a wait state to the memory cycle when set. In the 2K87M mask set, this bit is undefined.

9 HDI16

The 2K87M mask set defines additional bits in the Interface Control Register (ICR) and Interface Status Register (ISR) to support HDI16 bursts of various sizes. This affects the conditions used to assert or deassert the HREQ signal. The following sections compare the register layouts for the two mask sets and indicate the changes to the logic used to assert the HREQ signal.

9.1 Interface Control Register (ICR) Changes

The following sections list the ICR layout and definitions for the 2K42A and 2K87M mask sets.

9.1.1 2K42A Mask Set ICR Definitions

ICR Mask Set 2K42A Interface Control Register (DMA = 0, HICR = 1) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—					HF0	HF1	—	INIT	HM		HF2	HF3	HDRQ	TREQ	RREQ
Type	R/W															
Reset	See **Table 18** on page 25															

ICR Mask Set 2K42A Interface Control Register (DMA = 1, HICR = 1) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—					HF0	HF1	—	INIT	HM		HF2	HF3	—	TREQ	RREQ
Type	R/W															
Reset	See **Table 18** on page 25															

ICR Mask Set 2K42A Interface Control Register (DMA = 0, HICR = 0) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—					HF0	HF1	—	INIT	HDM		HF2	HF3	HDRQ	TREQ	RREQ
Type	R/W									R		R/W				
Reset	See **Table 18** on page 25															

ICR Mask Set 2K42A Interface Control Register (DMA = 1, HICR = 0) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—					HF0	HF1	—	INIT	HDM		HF2	HF3	—	Note 1	Note 2
Type	R/W									R		R/W			R/W	
Reset	See **Table 18** on page 25															

Notes: **1.** $\overline{\text{HDM0}}$ when read, TREQ when written
2. HDM0 when read, RREQ when written

ICR is a read/write control register that allows the use of bit manipulation instructions on control register bits. The host processor uses the ICR to control the HDI16 interrupts and flags. The SC140 core cannot access the ICR. The HPCR[DMA] bit and the HCR[HICR] bit control the function of the ICR bits.

Table 17. Mask Set 2K42A ICR Bit Descriptions

Name	Description
— 0–4	Reserved. Write to zero for future compatibility.
HF0 5	**Host Flag 0** A general-purpose flag for host-to-core communication. The host processor can set or clear HF0. HF0 is reflected in the HSR on the core side of the HDI16.
HF1 6	**Host Flag 1** A general-purpose flag for host-to-core communication. The host processor can set or clear HF1. HF1 is reflected in the HSR on the core side of the HDI16.
— 7	Reserved. Write to zero for future compatibility.
INIT 8	**Force Initialization** Used by the host processor to force initialization of the HDI16 hardware (which may or may not be necessary, depending on the software design of the interface). During initialization, the HDI16 transmit and receive control bits are configured. The type of initialization performed when the INIT bit is set depends on the state of TREQ and RREQ in the HDI16. The INIT command, which is local to the HDI16, conveniently configures the HDI16 into the desired data transfer mode. The effect of the INIT command is described in **Table 19**. When the host sets the INIT bit, the HDI16 hardware executes the INIT command. The interface hardware clears the INIT bit after the command executes.

Table 17. Mask Set 2K42A ICR Bit Descriptions (Continued)

Name	Description
HM/HDM 9–10	**Host Mode/Host DMA Mode** When host DMA mode is enabled, if HCR[HICR] is set, the HREQ pin requests DMA transfers, the TREQ and RREQ bits select the direction of DMA transfers, and the HACK input pin is used as a DMA transfer acknowledge input, if OAE in HPCR is cleared. If the DMA direction is from core to host, the contents of the selected register are written to the host data bus when HACK is asserted. If the DMA direction is from host to core, the selected register is read from the host data bus when HACK is asserted. If HPCR[OAE] is set, a host read or write to host address 0x4 is used as a DMA transfer acknowledge. If the DMA direction is from core to host, the contents of the selected register are written to the host data bus when the host reads from host address 0x4. If the DMA direction is from host to core, the selected register is read from the host data bus when the host writes to host address 0x4. HM also controls the size of the DMA word to be transferred. The HDI16 data register selected during a host DMA transfer is determined by a 2-bit address counter, which is preloaded with the value in HM. The address counter replaces the HA[0–1] bits of the HDI16 during a host DMA transfer. The address counter can be initialized with the INIT bit feature. After each DMA transfer on the host data bus, the address counter is decremented. When the address counter reaches the last register, the address counter is loaded with the value in HM. Thus, 16-bit, 32-bit, 48-bit, or 64-bit data can be transferred in a circular fashion, and the need is eliminated for the DMA controller to supply the HA0–2] pins (HPCR bit OAE=0) or to read/write at host address 0x4 (HPCR bit OAE=1). For 32-, 48- or 64-bit data transfers, the core CPU interrupt rate is reduced by a factor of 2, 3, or 4, respectively, from the host request rate. That is, for every two or three host processor data transfers of one byte each, there is only one 64-bit core CPU interrupt. This bit is available only in ICR mode (HCR[HICR] = 1). When the HDI16 is in ICR priority non-DMA mode (the HPCR[DMA] bit is cleared and HCR[HICR] is set), data transfer size is defined by HM, as described in **Table 22**. The transfer size causes the RX*x*/TX*x* register read/write at the last (trigger) address to clear the RXDF/TXDE bits, respectively.
HF2 11	**Host Flag 2** A general-purpose flag for host-to-core communication. The host processor can set or clear HF2. HF2 is reflected in the HSR on the core side of the HDI16.
HF3 12	**Host Flag 3** A general-purpose flag for host-to-core communication. The host processor can set or clear HF3. HF3 is reflected in the HSR on the core side of the HDI16.
HDRQ 13	**HREQ/HTRQ and HACK/HRRQ Pin Control** Controls the HREQ/HTRQ and HACK/HRRQ pins. If HDRQ is cleared, the HREQ/HTRQ pin functions as a single HREQ. If HDRQ is set, the HREQ/HTRQ and HACK/HRRQ pins function as HTRQ and HRRQ, respectively. This bit is available only in non-DMA (interrupt) mode (HPCR[DMA] = 0).
TREQ/$\overline{\text{HDM0}}$ 14	**HREQ/HTREQ Pin Control** Controls the HREQ/HTREQ pin for host transmit data transfers. In non-DMA (interrupt) mode (DMA = 0 in the HPCR), TREQ enables host requests via the host request (HREQ or HTRQ) pin when the Transmit Data Register Empty (TXDE) status bit in the ISR is set. If TREQ is cleared, TXDE interrupts are disabled. If TREQ and TXDE are set, the host request pin is asserted. In DMA modes (HPCR[DMA] = 1 and HCR[HICR] = 1), software must set or clear TREQ to select the direction of DMA transfers. Setting TREQ sets the direction of the DMA transfers as host-to-core and enables the HREQ pin to request data transfers. When HCR[HICR] is cleared and HPCR[DMA] is set, a TREQ read reflects the status of NOT HDM0 in HCR. When written, TREQ affects the INIT mode. See **Table 20**.
RREQ/HDM0 15	**HREQ and HRREQ Pin Control** Controls the HREQ and HRREQ pins for host receive data transfers. In non-DMA (interrupt) mode (HPCR[DMA] = 0), RREQ enables host requests via the host request (HREQ or HRRQ) pin when the Receive Data Register Full (RXDF) status bit in the ISR is set. If RREQ is cleared, ISR[RXDF] interrupts are disabled. If RREQ is set, the host request pin (HREQ or HRRQ) is asserted if ISR[RXDF] is set. In host DMA mode (DMA = 1 in the HPCR and HICR = 1 in the HCR), RREQ must be set or cleared by software to select the direction of DMA transfers. Setting RREQ sets the direction of the host DMA transfers to core-to-host and enables the HREQ pin to request data transfers. When HICR in HCR is cleared and DMA in HPCR is set, RREQ, when read, reflects the status of HDM0 in HCR. When HCR[HICR] is cleared and HPCR[DMA] is cleared, an RREQ read reflects the status of HCR[HDM0]. When written, RREQ affects the INIT mode. See **Table 20**.

Table 18 shows the result of the four kinds of reset on bits in each of the HDI16 registers accessible to the host processor. The hardware reset is caused by asserting the $\overline{\text{RESET}}$ signal. The individual reset is caused by clearing HPCR[HEN].

Table 18. Host-Side Registers After Reset

Register Name	Register Data	Reset Type	
		Hardware (HW) Reset	Individual (IR) Reset
ICR	All Bits	0	—
CVR	NMI	0	0
	HC	0	0
	HV[0–6]	0	—
ISR	HREQ	0	1 if TREQ is set; 0 otherwise
	HF[4–7]	0	—
	TRDY	1	1
	TXDE	1	1
	RXDF	0	0
RX	RX[0–3]	empty	empty
TX	TX[0–3]	empty	empty

Table 19 shows the effects of the INIT command in relation to the values of TREQ and RREQ.

Table 19. Effects of the INIT Command

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
0	0	INIT = 0	None
0	1	INIT = 0; ISR[RXDF] = 0; HSR[HTFE/HTFNF] = 1	Core to host
1	0	INIT = 0; ISR[TXDE] = 1; HSR[HRFF/HRFNE] = 0	Host to core
1	1	INIT = 0; ISR[RXDF] = 0; HSR[HTFE/HTFNF] = 1; ISR[TXDE] = 1; HSR[HRFF/HRFNE] = 0	Host to/from core

Table 20 and **Table 21** summarize the effect of RREQ and TREQ on the HREQ and HRRQ signals.

Table 20. TREQ and RREQ Modes (HDRQ = 0) in Non-DMA Mode (HICR=1)

TREQ	RREQ	HREQ Signal
0	0	No interrupts (polling)
0	1	ISR[RXDF] request (interrupt)
1	0	ISR[TXDE] request (interrupt)
1	1	ISR[RXDF] and ISR[TXDE] request (interrupts)

Table 21. TREQ and RREQ Modes (HDRQ = 1) in Non-DMA Mode (HICR=1)

TREQ	RREQ	HTRQ Signal	HRRQ Signal
0	0	No interrupts (polling)	No interrupts (polling)
0	1	No interrupts (polling)	ISR[RXDF] request (interrupt)
1	0	ISR[TXDE] request (interrupt)	No interrupts (polling)
1	1	ISR[TXDE] request (interrupt)	ISR[RXDF] request (interrupt)

Table 22 shows the valid HM values and their meaning.

Table 22. HM Host DMA Mode Values (HICR=1)

HM		Mode	
0	1	DMA Mode	Non-DMA Mode
0	0	16-bit words	64-bit words. Last read/write (trigger) address: 0x4
0	1	32-bit words	48-bit words. Last read/write (trigger) address: 0x5
1	0	48-bit words	32-bit words. Last read/write (trigger) address: 0x6
1	1	64-bit words	16-bit words. Last read/write (trigger) address: 0x7

9.1.2 2K87M Mask Set ICR Definitions

ICR 2K87M Mask Set Interface Control Register (DMA = 0, HICR = 1) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	HRRA		HTRA		—	HF0	HF1	—	INIT	HM		HF2	HF3	HDRQ	TREQ	RREQ
Type	R/W															
Reset	See **Table 24** on page 29															

ICR 2K87M Mask Set Interface Control Register (DMA = 1, HICR = 1) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	HRRA		HTRA		—	HF0	HF1	—	INIT	HM		HF2	HF3	—	TREQ	RREQ
Type	R/W															
Reset	See **Table 24** on page 29															

ICR 2K87M Mask Set Interface Control Register (DMA = 0, HICR = 0) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	HRRA		HTRA		—	HF0	HF1	—	INIT	HDM		HF2	HF3	HDRQ	TREQ	RREQ
Type	R/W									R		R/W				
Reset	See **Table 24** on page 29															

ICR 2K87M Mask Set Interface Control Register (DMA = 1, HICR = 0) **0x0**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	HRRA		HTRA		—	HF0	HF1	—	INIT	HDM		HF2	HF3	—	Note 1	Note 2
Type	R/W									R		R/W			R/W	
Reset	See **Table 24** on page 29															

Notes: **1.** $\overline{\text{HDM0}}$ when read, TREQ when written
2. HDM0 when read, RREQ when written

ICR is a read/write control register that allows the use of bit manipulation instructions on control register bits. The host processor uses the ICR to control the HDI16 interrupts and flags. The SC140 core cannot access the ICR. The HPCR[DMA] bit and the HCR[HICR] bit control the function of the ICR bits.

Table 23. 2K87M Mask Set ICR Bit Descriptions

Name	Description	Settings
HRRA 0–1	**HDI Receive Request Assertion** The receive request is asserted a specified time based on the number of data bytes to receive.	00 HRRQ asserted for 8 bytes (RX full) 01 HRRQ asserted for 16 bytes (RX full + 1 HOTX entry full) 10 HRRQ asserted for 32 bytes (RX full + 3 HOTX entries full) 11 Reserved
HTRA 2–3	**HDI Transmit Request Assertion** The transmit request is asserted a specified time based on the number of data bytes to transmit.	00 HTRQ/HREQ asserted for 8 bytes (TX empty) 01 HTRQ/HREQ asserted for 16 bytes (TX empty + 1 HOTX entry empty) 10 HTRQ/HREQ asserted for 32 bytes (TX empty + 3 HOTX entries empty) 11 Reserved
— 4	Reserved. Write to zero for future compatibility.	
HF0 5	**Host Flag 0** A general-purpose flag for host-to-core communication. The host processor can set or clear HF0. HF0 is reflected in the HSR on the core side of the HDI16.	
HF1 6	**Host Flag 1** A general-purpose flag for host-to-core communication. The host processor can set or clear HF1. HF1 is reflected in the HSR on the core side of the HDI16.	
— 7	Reserved. Write to zero for future compatibility.	
INIT 8	**Force Initialization** Used by the host processor to force initialization of the HDI16 hardware (which may or may not be necessary, depending on the software design of the interface). During initialization, the HDI16 transmit and receive control bits are configured. The type of initialization performed when the INIT bit is set depends on the state of TREQ and RREQ in the HDI16. The INIT command, which is local to the HDI16, conveniently configures the HDI16 into the desired data transfer mode. The effect of the INIT command is described in **Table 25** on page 30. When the host sets the INIT bit, the HDI16 hardware executes the INIT command. The interface hardware clears the INIT bit after the command executes.	

Table 23. 2K87M Mask Set ICR Bit Descriptions (Continued)

Name	Description	Settings
HM/HDM 9–10	**Host Mode/Host DMA Mode** When host DMA mode is enabled, if HCR[HICR] is set, the HREQ pin requests DMA transfers, the TREQ and RREQ bits select the direction of DMA transfers, and the HACK input pin is used as a DMA transfer acknowledge input, if OAE in HPCR is cleared. If the DMA direction is from core to host, the contents of the selected register are written to the host data bus when HACK is asserted. If the DMA direction is from host to core, the selected register is read from the host data bus when HACK is asserted. If HPCR[OAE] is set, a host read or write to host address 0x4 is used as a DMA transfer acknowledge. If the DMA direction is from core to host, the contents of the selected register are written to the host data bus when the host reads from host address 0x4. If the DMA direction is from host to core, the selected register is read from the host data bus when the host writes to host address 0x4. HM also controls the size of the DMA word to be transferred. The HDI16 data register selected during a host DMA transfer is determined by a 2-bit address counter, which is preloaded with the value in HM. The address counter replaces the HA[0–1] bits of the HDI16 during a host DMA transfer. The address counter can be initialized with the INIT bit feature. After each DMA transfer on the host data bus, the address counter is decremented. When the address counter reaches the last register, the address counter is loaded with the value in HM. Thus, 16-bit, 32-bit, 48-bit, or 64-bit data can be transferred in a circular fashion, and the need is eliminated for the DMA controller to supply the HA0–2] pins (HPCR bit OAE=0) or to read/write at host address 0x4 (HPCR bit OAE=1). For 32-, 48- or 64-bit data transfers, the core CPU interrupt rate is reduced by a factor of 2, 3, or 4, respectively, from the host request rate. That is, for every two or three host processor data transfers of one byte each, there is only one 64-bit core CPU interrupt. This bit is available only in ICR mode (HCR[HICR] = 1). When the HDI16 is in ICR priority non-DMA mode (the HPCR[DMA] bit is cleared and HCR[HICR] is set), data transfer size is defined by HM, as described in **Table 28**. The transfer size causes the RX*x*/TX*x* register read/write at the last (trigger) address to clear the RXDF/TXDE bits, respectively.	
HF2 11	**Host Flag 2** A general-purpose flag for host-to-core communication. The host processor can set or clear HF2. HF2 is reflected in the HSR on the core side of the HDI16.	
HF3 12	**Host Flag 3** A general-purpose flag for host-to-core communication. The host processor can set or clear HF3. HF3 is reflected in the HSR on the core side of the HDI16.	
HDRQ 13	**HREQ/HTRQ and HACK/HRRQ Pin Control** Controls the HREQ/HTRQ and HACK/HRRQ pins. If HDRQ is cleared, the HREQ/HTRQ pin functions as a single HREQ. If HDRQ is set, the HREQ/HTRQ and HACK/HRRQ pins function as HTRQ and HRRQ, respectively. This bit is available only in non-DMA (interrupt) mode (HPCR[DMA] = 0).	

Table 23. 2K87M Mask Set ICR Bit Descriptions (Continued)

Name	Description	Settings
TREQ/$\overline{\text{HDM0}}$ 14	**HREQ/HTREQ Pin Control** Controls the HREQ/HTREQ pin for host transmit data transfers. In non-DMA (interrupt) mode (DMA = 0 in the HPCR), TREQ enables host requests via the host request (HREQ or HTRQ) pin when the Transmit Data Register Empty (TXDE) status bit in the ISR is set. If TREQ is cleared, TXDE interrupts are disabled. If TREQ and TXDE are set, the host request pin is asserted. In DMA modes (HPCR[DMA] = 1 and HCR[HICR] = 1), software must set or clear TREQ to select the direction of DMA transfers. Setting TREQ sets the direction of the DMA transfers as host-to-core and enables the HREQ pin to request data transfers. When HCR[HICR] is cleared and HPCR[DMA] is set, a TREQ read reflects the status of NOT HDM0 in HCR. When written, TREQ affects the INIT mode. See **Table 25**.	
RREQ/HDM0 15	**HREQ and HRREQ Pin Control** Controls the HREQ and HRREQ pins for host receive data transfers. In non-DMA (interrupt) mode (HPCR[DMA] = 0), RREQ enables host requests via the host request (HREQ or HRRQ) pin when the Receive Data Register Full (RXDF) status bit in the ISR is set. If RREQ is cleared, ISR[RXDF] interrupts are disabled. If RREQ is set, the host request pin (HREQ or HRRQ) is asserted if ISR[RXDF] is set. In host DMA mode (DMA = 1 in the HPCR and HICR = 1 in the HCR), RREQ must be set or cleared by software to select the direction of DMA transfers. Setting RREQ sets the direction of the host DMA transfers to core-to-host and enables the HREQ pin to request data transfers. When HICR in HCR is cleared and DMA in HPCR is set, RREQ, when read, reflects the status of HDM0 in HCR. When HCR[HICR] is cleared and HPCR[DMA] is cleared, an RREQ read reflects the status of HCR[HDM0]. When written, RREQ affects the INIT mode. See **Table 25**.	

Table 24. Host-Side Registers After Reset

Register Name	Register Data	Reset Type	
		Hardware (HW) Reset	Individual (IR) Reset
ICR	All Bits	0	—
CVR	NMI	0	0
	HC	0	0
	HV[0–6]	0	—
ISR	TXDE32	1	1
	TXDE16	1	1
	RXDF32	0	0
	RXDF16	0	0
	HREQ	0	1 if TREQ is set; 0 otherwise
	HF[4–7]	0	—
	TRDY	1	1
	TXDE	1	1
	RXDF	0	0
RX	RX[0–3]	empty	empty
TX	TX[0–3]	empty	empty

Table 25 shows the effects of the INIT command in relation to the values of TREQ and RREQ.

Table 25. Effects of the INIT Command

TREQ	RREQ	After INIT Execution	Transfer Direction Initialized
0	0	INIT = 0	None
0	1	INIT = 0; ISR[RXDF] = 0; HSR[HTFE/HTFNF] = 1	Core to host
1	0	INIT = 0; ISR[TXDE] = 1; HSR[HRFF/HRFNE] = 0	Host to core
1	1	INIT = 0; ISR[RXDF] = 0; HSR[HTFE/HTFNF] = 1; ISR[TXDE] = 1; HSR[HRFF/HRFNE] = 0	Host to/from core

Table 26 and **Table 27** summarize the effect of RREQ and TREQ on the HREQ and HRRQ signals.

Table 26. TREQ and RREQ Modes (HDRQ = 0) in Non-DMA Mode (HICR=1)

TREQ	RREQ	HREQ Signal
0	0	No interrupts (polling)
0	1	ISR[RXDF] request (interrupt)
1	0	ISR[TXDE] request (interrupt)
1	1	ISR[RXDF] and ISR[TXDE] request (interrupts)

Table 27. TREQ and RREQ Modes (HDRQ = 1) in Non-DMA Mode (HICR=1)

TREQ	RREQ	HTRQ Signal	HRRQ Signal
0	0	No interrupts (polling)	No interrupts (polling)
0	1	No interrupts (polling)	ISR[RXDF] request (interrupt)
1	0	ISR[TXDE] request (interrupt)	No interrupts (polling)
1	1	ISR[TXDE] request (interrupt)	ISR[RXDF] request (interrupt)

Table 28 shows the valid HM values and their meaning.

Table 28. HM Host DMA Mode Values (HICR=1)

HM		Mode	
0	1	DMA Mode	Non-DMA Mode
0	0	16-bit words	64-bit words. Last read/write (trigger) address: 0x4
0	1	32-bit words	48-bit words. Last read/write (trigger) address: 0x5
1	0	48-bit words	32-bit words. Last read/write (trigger) address: 0x6
1	1	64-bit words	16-bit words. Last read/write (trigger) address: 0x7

9.2 Interface Status Register (ISR) Changes

The following sections lists the ICR layout and definitions for the 2K42A and 2K87M mask sets.

9.2.1 2K42A Mask Set ISR Definitions

ISR 2K42A Mask Set Interface Status Register **0x2**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								HREQ	HF4	HF5	HF6	HF7	TRDY	TXDE	RXDF
Type	R															
Reset	See Table 24, “Host-Side Registers After Reset,” on page 29															

ISR is a status register by which the host processor interrogates the status and flags of the HDI16. The host processor can write to this address without affecting the internal state of the HDI16. The SC140 core cannot access ISR.

Table 29. 2K42A Mask Set ISR Bit Descriptions

Name	Description	Settings
— 0–7	Reserved. Write to zero for future compatibility.	
HREQ 8	**HREQ Status** HREQ indicates the status of the external transmit and receive request output signals (HTRQ and HRRQ) if HDRQ is set. If HDRQ is cleared, it indicates the status of the external host request output signal (HREQ). The HREQ bit can be set under either or both of two conditions: the Receive Word Registers (RX[0–3]) are full or the Transmit Word Registers (TX[0–3]) are empty. These conditions are indicated by the ISR RXDF and TXDE status bits, respectively. If the interrupt source has been enabled by the associated request enable bit in the ICR, HREQ is set if one or more of the two enabled interrupt sources is set.	If HDRQ is cleared: 0 No host processor interrupts requested. 1 Interrupt requested. If HDRQ is set: 0 No host processor interrupts requested (HTRQ and HRRQ cleared). 1 Interrupt requested (HTRQ or HRRQ set).
HF[4–7] 9–12	**Host Flags 4–7** Indicates the state of host flags 4–7 in the HCR on the core side. Only the SC140 core can change HF[4–7].	
TRDY 13	**TRDY Status** Indicates whether the Transmit Word Registers TX[0–3] and the HORX FIFO are empty. TRDY is set if TXDE is set and HRFNE is cleared. If TRDY is set, the data that the host processor writes to TX[0–3] is immediately transferred to the core side of the HDI16 and can be read by the SC140 core. This feature has many applications. For example, if the host processor issues a host command that causes the SC140 core to read HORX, the host processor can be certain that the data it just transferred to the HDI16 is the same data received by the SC140 core.	0 TX[0–3] and the HORX FIFO are not empty. 1 TX[0–3] and the HORX FIFO are empty.
TXDE 14	**Transmit Data Empty** Indicates whether the Transmit Word Registers (TX[0–3]) are empty and can be written by the host processor. TXDE is set when the contents of the TX[0–3] registers are transferred to the HORX register. TXDE is cleared when the host processor writes to the transmit data registers (TX) and the HORX FIFO is full. The host processor sets TXDE using the initialize function. TXDE can assert the external HREQ/HTRQ signal if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE indicates whether the TX registers are full and data can be latched in so that the host processor can use polling techniques.	0 The TX registers are not empty. 1 The TX registers are empty and can be written by the host processor.

Table 29. 2K42A Mask Set ISR Bit Descriptions (Continued)

Name	Description	Settings
RXDF 15	**Receive Data Full** Indicates whether the Receive Word Registers (RX[0–3]) contain data from the SC140 core and can be read by the host processor. RXDF is set when the HOTX is transferred to the RX[0–3] registers. RXDF is cleared when the host processor reads the receive data registers (RX) at the last address and the HOTX FIFO is empty. The host processor can clear RXDF using the initialize function. RXDF can be used to assert the external HREQ/HRRQ signal if RREQ is set. Regardless of whether the RXDF interrupt is enabled, RXDF indicates whether the RX registers are full and data can be latched out, so that the host processor can use polling techniques.	0 The RX registers are not full. 1 The RX registers are full and can be read by the host processor.

9.2.2 2K87M Mask Set ISR Definitions

ISR 2K87M Mask Set Interface Status Register **0x2**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—				TXDE32	TXDE16	RXDF32	RXDF16	HREQ	HF4	HF5	HF6	HF7	TRDY	TXDE	RXDF
Type	R															
Reset	See **Table 30**															

ISR is a status register by which the host processor interrogates the status and flags of the HDI16. The host processor can write to this address without affecting the internal state of the HDI16. The SC140 core cannot access ISR.

Table 30. 2K87M Mask Set ISR Bit Descriptions

Name	Description	Settings
— 0–3	Reserved. Write to zero for future compatibility.	
TXDE32 4	**Transmit Queue 32 Bytes Empty**	0 < 32 bytes are empty 1 32 bytes are empty (24 bytes in HORX and 8 bytes in TX)
TXDE16 5	**Transmit Queue 16 Bytes Empty**	0 < 16 bytes are empty 1 16 bytes are empty (8 bytes in HORX and 8 bytes in TX)
RXDF32 6	**Receive Queue 32 Bytes Full**	0 < 32 bytes are full 1 32 bytes are full (24 bytes in HOTX and 8 bytes in RX)
RXDF16 7	**Receive Queue 16 Bytes Full**	0 < 16 bytes are full 1 16 bytes are full (8 bytes in HOTX and 8 bytes in RX)

Table 30. 2K87M Mask Set ISR Bit Descriptions (Continued)

Name	Description	Settings
HREQ 8	**HREQ Status** HREQ indicates the status of the external transmit and receive request output signals (HTRQ and HRRQ) if HDRQ is set. If HDRQ is cleared, it indicates the status of the external host request output signal (HREQ). The HREQ bit can be set under either or both of two conditions: the Receive Word Registers (RX[0–3]) are full or the Transmit Word Registers (TX[0–3]) are empty. These conditions are indicated by the ISR RXDF and TXDE status bits, respectively. If the interrupt source has been enabled by the associated request enable bit in the ICR, HREQ is set if one or more of the two enabled interrupt sources is set.	If HDRQ is cleared: 0 No host processor interrupts requested. 1 Interrupt requested. If HDRQ is set: 0 No host processor interrupts requested (HTRQ and HRRQ cleared). 1 Interrupt requested (HTRQ or HRRQ set).
HF[4–7] 9–12	**Host Flags 4–7** Indicates the state of host flags 4–7 in the HCR on the core side. Only the SC140 core can change HF[4–7].	
TRDY 13	**TRDY Status** Indicates whether the Transmit Word Registers TX[0–3] and the HORX FIFO are empty. TRDY is set if TXDE is set and HRFNE is cleared. If TRDY is set, the data that the host processor writes to TX[0–3] is immediately transferred to the core side of the HDI16 and can be read by the SC140 core. This feature has many applications. For example, if the host processor issues a host command that causes the SC140 core to read HORX, the host processor can be certain that the data it just transferred to the HDI16 is the same data received by the SC140 core.	0 TX[0–3] and the HORX FIFO are not empty. 1 TX[0–3] and the HORX FIFO are empty.
TXDE 14	**Transmit Data Empty** Indicates whether the Transmit Word Registers (TX[0–3]) are empty and can be written by the host processor. TXDE is set when the contents of the TX[0–3] registers are transferred to the HORX register. TXDE is cleared when the host processor writes to the transmit data registers (TX) and the HORX FIFO is full. The host processor sets TXDE using the initialize function. TXDE can assert the external HREQ/HTRQ signal if the TREQ bit is set. Regardless of whether the TXDE interrupt is enabled, TXDE indicates whether the TX registers are full and data can be latched in so that the host processor can use polling techniques.	0 The TX registers are not empty. 1 The TX registers are empty and can be written by the host processor.
RXDF 15	**Receive Data Full** Indicates whether the Receive Word Registers (RX[0–3]) contain data from the SC140 core and can be read by the host processor. RXDF is set when the HOTX is transferred to the RX[0–3] registers. RXDF is cleared when the host processor reads the receive data registers (RX) at the last address and the HOTX FIFO is empty. The host processor can clear RXDF using the initialize function. RXDF can be used to assert the external HREQ/HRRQ signal if RREQ is set. Regardless of whether the RXDF interrupt is enabled, RXDF indicates whether the RX registers are full and data can be latched out, so that the host processor can use polling techniques.	0 The RX registers are not full. 1 The RX registers are full and can be read by the host processor.
Note: A status bit is set when there are a number of empty/full bytes more than the status bit indicates. For example, if there are 32 empty bytes, all three relevant status bits (TXDE32, TXDE16, and TXDE) are set.		

9.3 HREQ Logic Differences

The following sections describe the implementation of HREQ logic for the 2K42A and 2K87M mask sets.

9.3.1 2K42A Mask Set HREQ Logic

When HREQ is connected to the host processor interrupt input, the HDI16 asserts HREQ to request service from the host processor. HREQ is asserted when TXDE is set and/or ISR[RXDF] is set, and the corresponding enable bit (TREQ or RREQ, respectively) is set, as **Figure 9** shows.

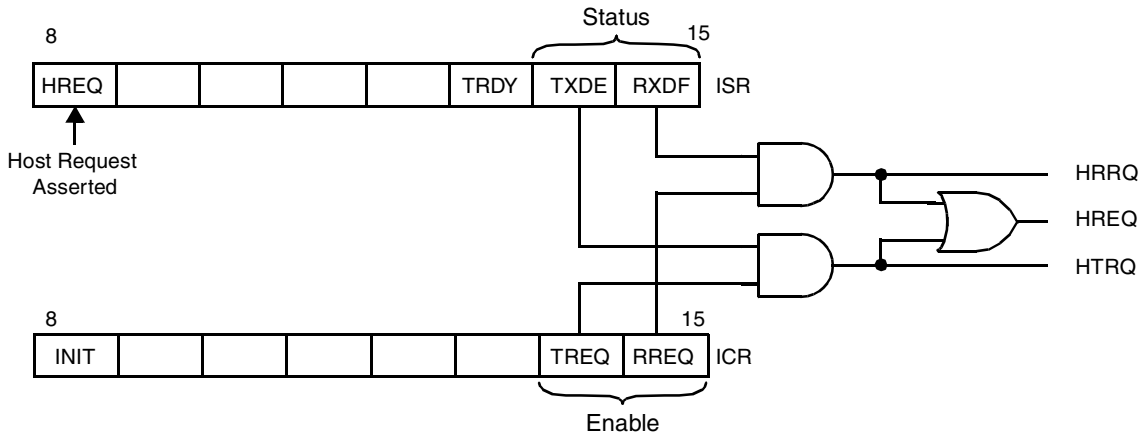


Figure 9. HDI16 Host Request Structure

The host processor acknowledges host interrupts by executing an interrupt service routine. The host processor tests ISR[RXDF] and ISR[TXDE] to determine the interrupt source. The host processor interrupt service routine must read or write the appropriate HDI16 data register to clear the interrupt. HREQ is deasserted under the following conditions:

- The enabled request is cleared or masked.
- The SC140 core is reset.

9.3.2 2K87M Mask Set HREQ Logic

When HREQ is connected to the host processor interrupt input, the HDI16 asserts HREQ to request service from the host processor. HREQ is asserted according to the programming of ICR[HTRA] and the status bits TXDE, TXDE16, and TXDE2 when ICR[TREQ] is set, or according to the programming of ICR[HRRA] and the status bits RXDF, RXDF16, and RXDF32 when ICR[RREQ] is set, as **Figure 10** shows. The programming of HRRA or HTRA chooses the status bit, and depending on its value, HREQ is asserted or deasserted.

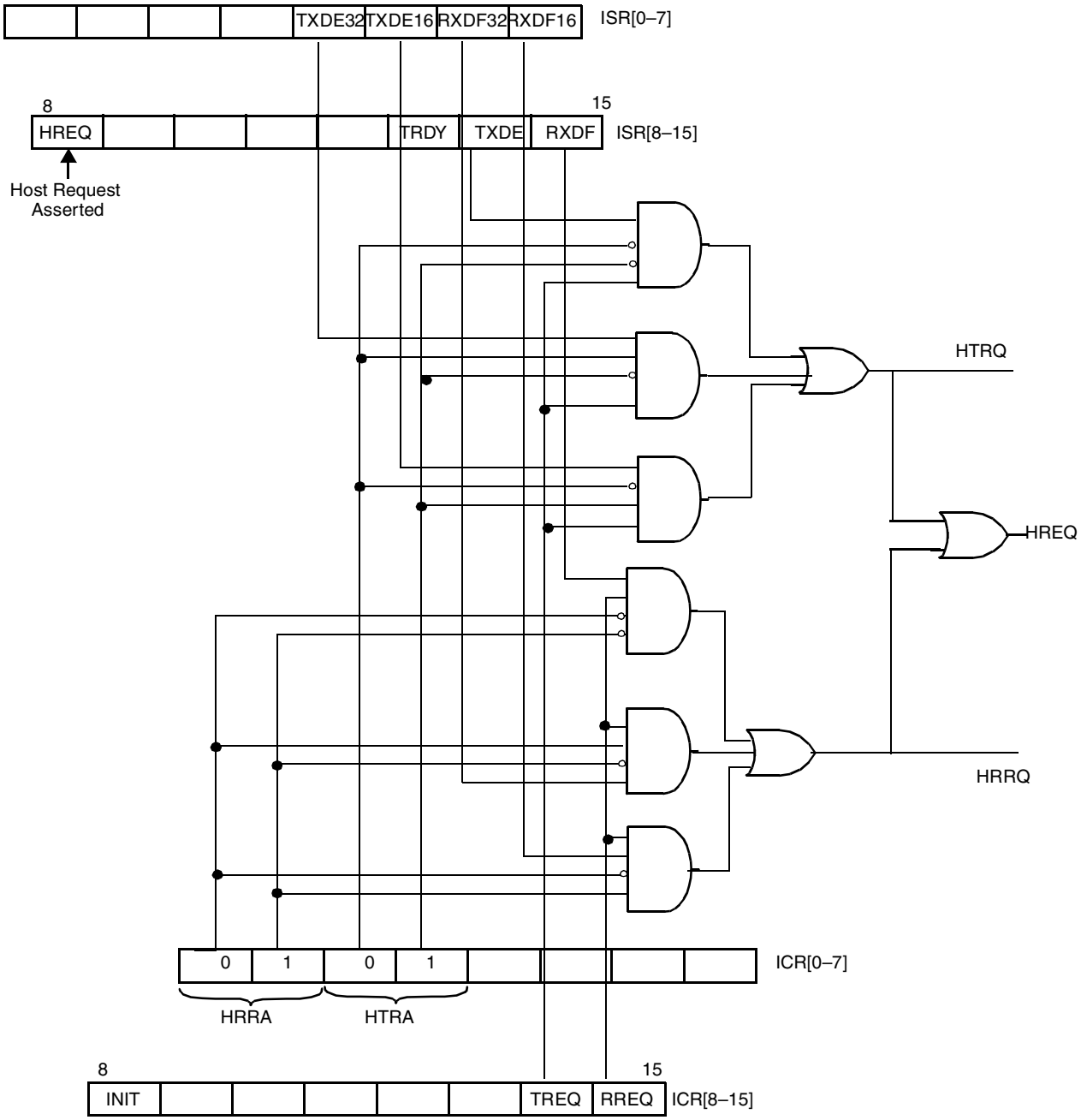


Figure 10. HDI16 Host Request Structure

The host processor acknowledges host interrupts by executing an interrupt service routine. The host processor tests the appropriate status bit (TXDE, TXDE16, TXDE32, RXDF, RXDF16, or RXDF32, depending on the application) to determine the interrupt source. The host processor interrupt service routine must read or write the appropriate HDI16 data register to clear the interrupt. HREQ is deasserted under the following conditions:

- The enabled request is cleared or masked.
- The SC140 core is reset.

10 DMA Transfer Code Definitions

The TC code definitions for mask sets 2K42A and 2K87M are as follows:

Table 31. Transfer Code Encoding

TC[0–2]	2K42A	2K87M
000	Reserved	Reserved
001	Reserved	Reserved
010	Reserved	Reserved
011	Reserved	Reserved
100	Reserved	DMA function code 0
101	SC140 core	SC140 core/DMA function code 1
110	SDMA/DMA function code 0	SDMA
111	SDMA/DMA function code 1	SDMA

11 Interrupt System

11.1 SCPRR_L and SCPRR_L_EXT Changes

For mask set 2K87M, the TC layer interrupts are configured with the SCCs using the YCC entries defined by the CPM Low Interrupt Priority Registers (SCPRR_L and SCPRR_L_EXT). For the fields YC1P–YC8P in each register, the value 100 (which was reserved for mask set 2K42A) is reassigned to “TC layer asserts its request in the YCCn position.”

11.2 TC Layer Interrupts

The TC layer accesses use interrupt vector 44 (this was reserved for mask set 2K42A).

12 Debugging

12.1 JTAG ID Changes

The JTAG ID value for the 2K42A mask set is 0x0188201D. The JTAG ID value for the 2K87M mask set is 0x1188201D.

12.2 EOnCE Status Register (ESR) Values

For the 2K42A mask set, ESR[REVNO] = 1 and ESR[CORETP] =1. For the 2K87M mask, ESR[REVNO] = 2 and ESR[CORETP] = 2.

12.3 EOnCE Counter Register Values

For the 2K42A mask set, the Event Counter Value Register (ECNT_VAL) and the Extension Counter Value Register (ECNT_EXT) have a maximum value of 0xFFFFFFFF. For the 2K87M mask set, the most significant bit (MSB) of these registers is hard-wired to 0, limiting the maximum register value to 0x7FFFFFFF. When reading either register, the MSB is always zero. For future compatibility, always write a 0 to the MSB of these registers.

Note: When the counter value is written to the trace buffer, it is shifted one position to the left so the debugger can use the least significant bit (LSB) to identify a new entry in the trace buffer.

13 EFCOP

The MSC8103 mask set 2K87M does not support an EFCOP module.

14 CPM

The 2K87M CPM adds functionality. The contents of the RISC Controller Configuration Register (RCCR) changed to accommodate the increased size of the Dual-Port RAM. Support for a ROM-based Inverse Multiplexing for ATM (IMA) microcode was added. Transmission convergence (TC) layer functionality required to support IMA was added to the Time-Slot Assigner (TSA). Also, new MCC host commands are added. The following sections discuss these changes.

14.1 RISC Controller Configuration Register (RCCR) Changes

The 2K87M mask set RISC Controller Configuration (RCCR) expands the Enable RAM Microcode (ERAM) field from 3 to 4 bits wide (adding RCCR to 19 bit which was reserved in the 2K42A mask set) to reflect the availability of dual-port RAM space starting at address 0x4000.

RCCR 2K87M Mask Set RISC Controller Configuration Register **0x119C4**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	TIME	—	TIMEP						EXT[*1–2*]M[1]		EXT*1*P[1]		—	SCD	EXT*2*P[1]	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	ERAM				EDM*[1–4]*[1]				EXT[*3–4*]M[1]		EXT*3*P[1]		—		EXT*4*P[1]	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Notes: **1.** Reserved for future microcode applications

RCCR configures the CP to run microcode from ROM or RAM and controls the CP internal timer.

Table 32. 2K87M Mask Set RCCR Bit Descriptions

Name	Reset	Description	Settings
TIME 0	0	**Timer Enable** Enables the CP internal timer that generates a tick to the CP based on the value programmed into the TIMEP field. TIME can be modified at any time to start or stop the scanning of the RISC timer tables.	0 Timer disabled 1 Timer enabled
— 1	0	Reserved. Write to zero for future compatibility.	
TIMEP 2–7	0	**Timer Period** Controls the CP timer tick. The RISC timer tables are scanned on each timer tick, and the input to the timer tick generator is the general system clock (150 MHz) divided by 1,024. The formula is (TIMEP + 1) × 1,024 = (general system clock period). Thus, a value of 0 stored in these bits gives a timer tick of 1 × (1,024) = 1,024 general system clocks and a value of 63 (decimal) gives a timer tick of 64 × (1,024) = 65,536 general system clocks.	
EXT*[1–2]***M** 8–9	0	**External Request Mode** Controls the external request sensitivity **Note:** Reserved for future microcode applications.	0 EXT*x* is edge sensitive according to EDMx 1 EXT*x* is level sensitive according to EDMx
EXT1P 10–11	0	**External Request Priority** Controls the external request priority relative to communication controllers **Note:** Reserved for future microcode applications.	00 EXT*x* has higher priority than the communication controllers (default) 01 EXT*x* has lower priority than the communications controllers 10 EXT*x* has the lowest priority 11 Reserved
— 12	0	Reserved. Write to zero for future compatibility.	
SCD 13	0	**Scheduler Configuration** Configure as instructed in the download process of a Freescale-supplied RAM microcode package.	0 Normal operation 1 Alternate configuration of the scheduler
EXT2P 14–15	0	**External Request Priority** Controls the external request priority relative to communication controllers **Note:** Reserved for future microcode applications.	00 EXT*x* has higher priority than the communication controllers (default) 01 EXT*x* has lower priority than the communications controllers 10 EXT*x* has the lowest priority 11 Reserved

Table 32. 2K87M Mask Set RCCR Bit Descriptions (Continued)

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th><th>Settings</th></tr>
<tr><td>ERAM
16–19</td><td>0</td><td>Enable RAM Microcode
Configure as instructed in the download process of a Freescale-supplied RAM microcode package.</td><td>
<table>
<tr><th>ERAM</th><th>RAM Microcode</th></tr>
<tr><td>0000</td><td>Disable microcode program execution from the dual-port RAM.</td></tr>
<tr><td colspan="2">For the following, execution starts at 0x0000 after reset:</td></tr>
<tr><td>0010</td><td>Microcode uses the first 2 KB + 8 KB from 0x4000</td></tr>
<tr><td>0100</td><td>Microcode uses the first 4 KB + 8 KB from 0x4000</td></tr>
<tr><td>0110</td><td>Microcode uses the first 6 KB + 8 KB from 0x4000</td></tr>
<tr><td>1000</td><td>Microcode uses the first 8 KB + 8 KB from 0x4000</td></tr>
<tr><td>1010</td><td>Microcode uses the first 10 KB + 8 KB from 0x4000</td></tr>
<tr><td>1100</td><td>Microcode uses the first 12 KB + 8 KB from 0x4000</td></tr>
<tr><td colspan="2">For the following, execution starts at 0x4000 after reset:</td></tr>
<tr><td>0011</td><td>Microcode uses the first 2 KB starting from 0x4000</td></tr>
<tr><td>0101</td><td>Microcode uses the first 4 KB starting from 0x4000</td></tr>
<tr><td>0111</td><td>Microcode uses the first 6 KB starting from 0x4000</td></tr>
<tr><td>1001</td><td>Microcode uses the first 8 KB starting from 0x4000</td></tr>
<tr><td colspan="2">All other combinations are reserved.</td></tr>
</table>
Note: this area was expanded from the 3-bit field used by the 2K42A mask set.
</td></tr>
<tr><td>EDM[1–4]
20–23</td><td>0</td><td>Edge Detect Mode
EXTx asserts requests according to settings

Note: Reserved for future microcode applications.</td><td>0 Low to high change for EXT x = 0, active high for EXT x = 1
1 High to low change for EXT x = 0, active low for EXT x = 1</td></tr>
<tr><td>EXT[3–4]M
24–25</td><td>0</td><td>External Request Mode
Controls the external request sensitivity

Note: Reserved for future microcode applications.</td><td>0 EXTx is edge-sensitive according to EDMx
1 EXTx is level-sensitive according to EDMx</td></tr>
<tr><td>EXT3P
26–27</td><td>0</td><td>External Request Priority
Controls the external request priority relative to communication controllers

Note: Reserved for future microcode applications.</td><td>00 EXTx has higher priority than the communication ‘controllers (default)
01 EXTx has lower priority than the communications controllers
10 EXTx has the lowest priority
11 Reserved</td></tr>
<tr><td>—
28–29</td><td>0</td><td colspan="2">Reserved for future applications</td></tr>
</table>

Table 32. 2K87M Mask Set RCCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
EXT4P 30–31	0	**External Request Priority** Controls the external request priority relative to communication controllers **Note:** Reserved for future microcode applications.	00 EXT*x* has higher priority than the communication controllers (default) 01 EXT*x* has lower priority than the communications controllers 10 EXT*x* has the lowest priority 11 Reserved

14.2 Dual-Port RAM Change

The dual-port RAM has been expanded to 32 KB of static RAM. An extra 8 KB starting at address 0x4000 is available for microcode execution only and cannot be used for data buffers or BDs. However, when not used for microcode, the extra 8 KB can be accessed from the system bus for general purpose internal storage.

14.2.1 2K42A Mask Set 24 KB Dual-Port RAM Memory Map

Figure 11 shows the MSC8101 2K42A mask set memory map of the dual-port RAM.

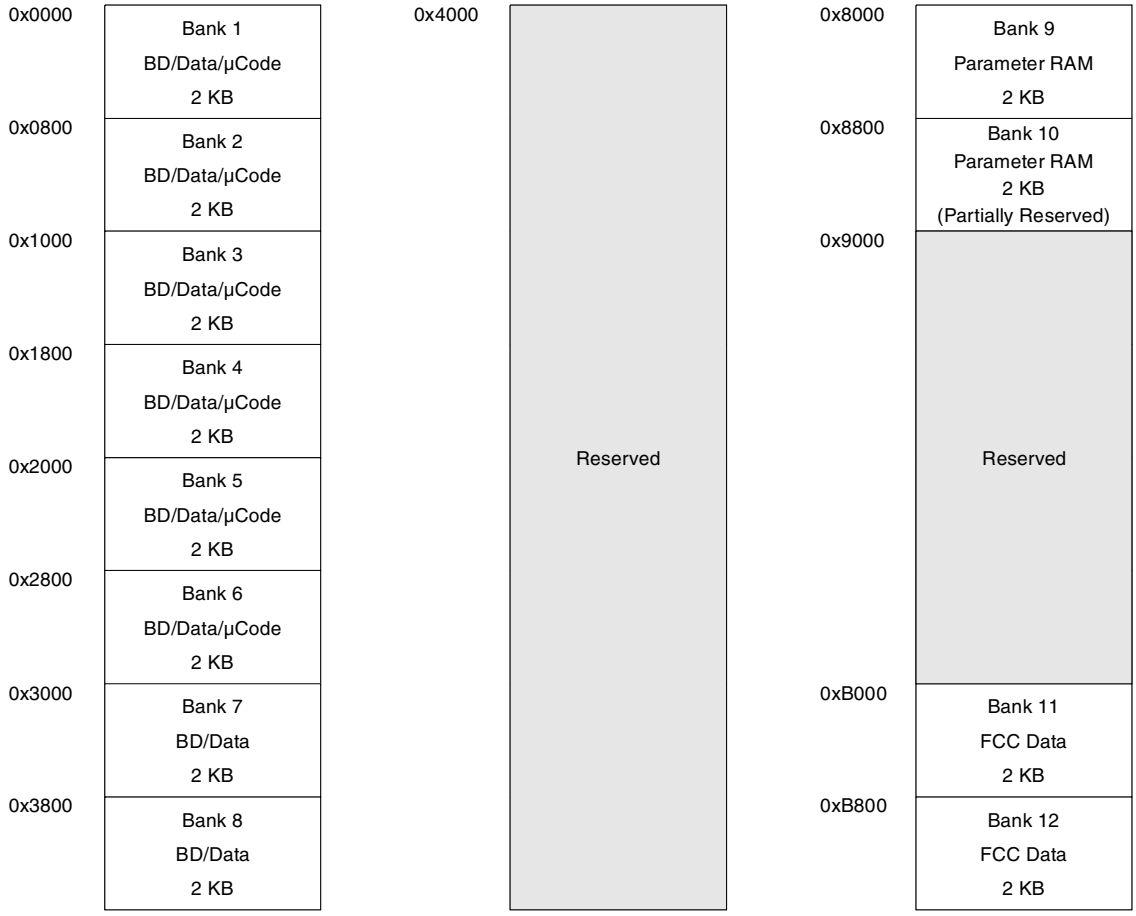


Figure 11. 2K42A Mask Set 24 KB Dual-Port RAM Memory Map

14.2.2 2K87M Mask Set Dual-Port RAM Memory Map

Figure 12 shows the MSC8103 2K87M mask set memory map of the dual-port RAM.

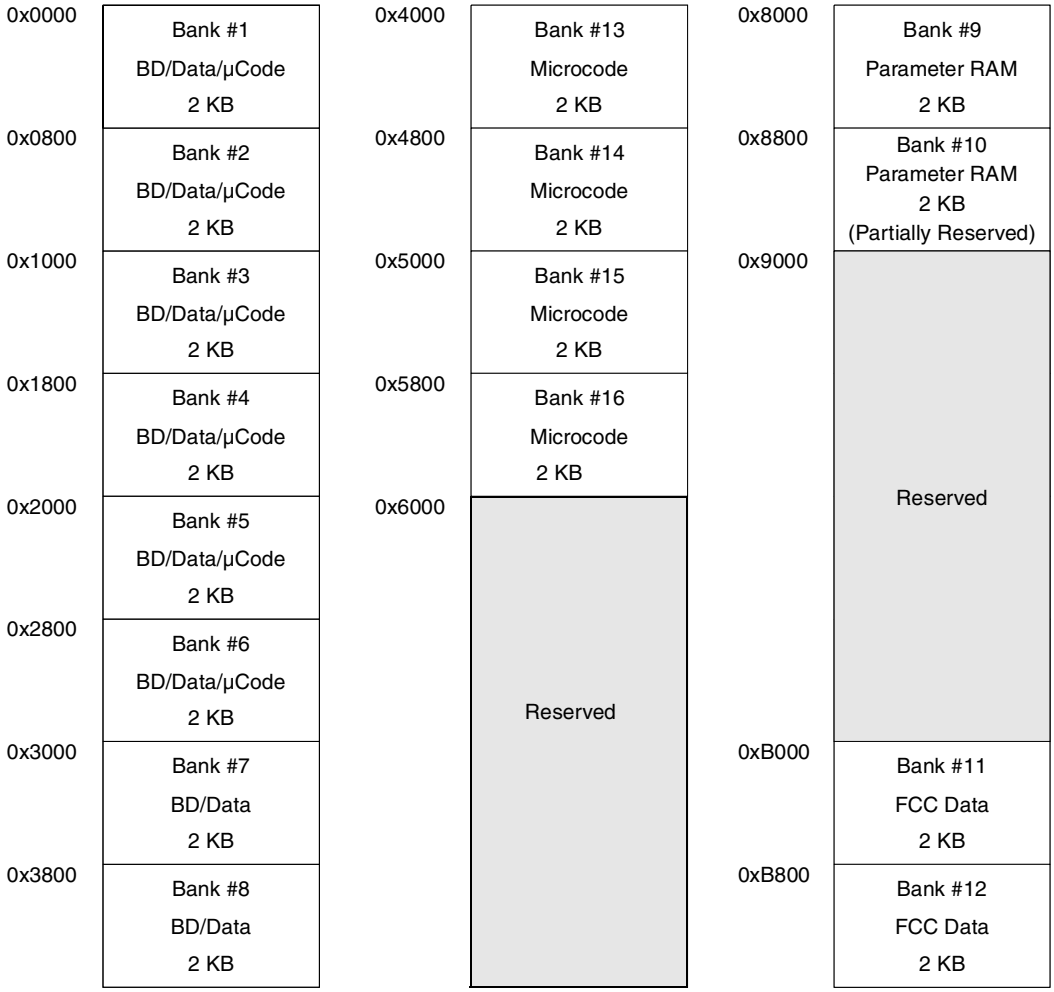


Figure 12. 2K87M Mask Set 32 KB Dual-Port RAM Memory Map

14.3 IMA Functionality

IMA functionality is similar to the updated functionality described in *MPC8266AUMAD/D* for updates to the MPC826x product family. This document can be accessed from the Freescale website listed on the back page of this document.

14.4 TSA Changes

The 2K87M mask set adds functionality required to support the new ATM transmission convergence (TC) layer added to the communications processor module (CPM) in the time-slot assigner (TSA). This functionality can only be configured and used by the fast communications controller 2 (FCC2) for ATM interfaces using the IMA functionality.

14.4.1 TC Layer Functionary

Figure 13 shows a block diagram of the SI blocks and TSA.The TSA also contains transmission convergence (TC) layer hardware.

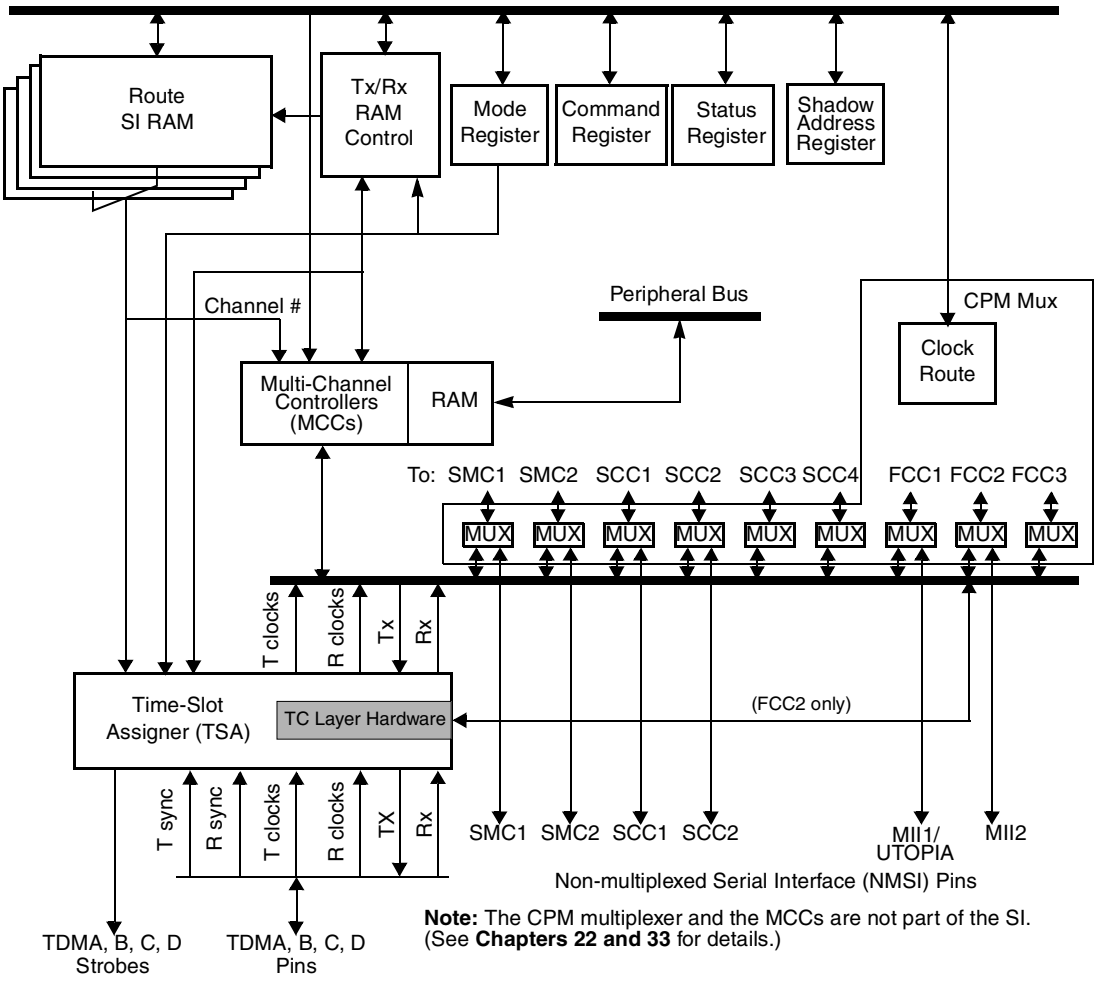


Figure 13. SI Block Diagram

Note: TC layer support for ATM (FCC2 only) with TC layer 1 supported through TDMA1 and TC layers 6–8 supported through TDMB2, TDMC2, and TDMD2, respectively.

For the FCC2 ATM UTOPIA 8, the SI supports the following:

- Four TDM channels routed in hardware to a TC layer block
- Protocol-specific overhead bits may be discarded or routed to other controllers by the SI
- Performing ATM TC layer functions (according to ITU-T I.432)
- Transmit (Tx) updates include:
  — Cell HEC generation
  — Payload scrambling using self synchronizing scrambler (programmable by the user)
  — Coset generation (programmable by the user)
  — Cell rate by inserting idle/unassigned cells
- Receive (Rx) updates include:
  — Cell delineation using bit by bit HEC checking and programmable ALPHA and DELTA parameters for the

delineation state machine
— Payload descrambling using self synchronizing scrambler (programmable by the user)
— Coset removing (programmable by the user)
— Filtering idle/unassigned cells (programmable by the user)
— Performing HEC error detection and single bit error correction (programmable by the user)
— Generating loss of cell delineation status/interrupt (LOC / LCD)

- Serial loop back mode
- Cell echo mode
- Supports both FCC transmit modes:
  — External rate mode—Idle cells are generated by the FCC (microcode) to control data rate
  — Internal rate mode (sub-rate)—FCC transfers only the data cells using the required data rate. The TC layer generates idle/unassigned cells to maintain the line bit rate
- Supports the TC layer and PMD (physical medium dependant) WIRE interface (according to the ATM-Forum af-phy-0063.000)

14.4.2 ATM TC Layer Support

The MSC8103 supports applications that receive ATM traffic over the standard serial protocols like E1, T1, and xDSL via its serial interface (SIx TDMx and NMSI) ports because the ATM TC-layer functionality is implemented internally. This allows the use of standard low-cost PHY devices in system applications instead of PHYs that support UTOPIA bus devices. A typical TC layer application requires the use of one SI TDM channel per TC block.

As shown in **Figure 14**, all TC blocks are internally connected to FCC2. In addition, **Figure 14** shows FCC1 connected to a UTOPIA 8-bit MPHY bus that can be routed outside and operated independently from the TC block

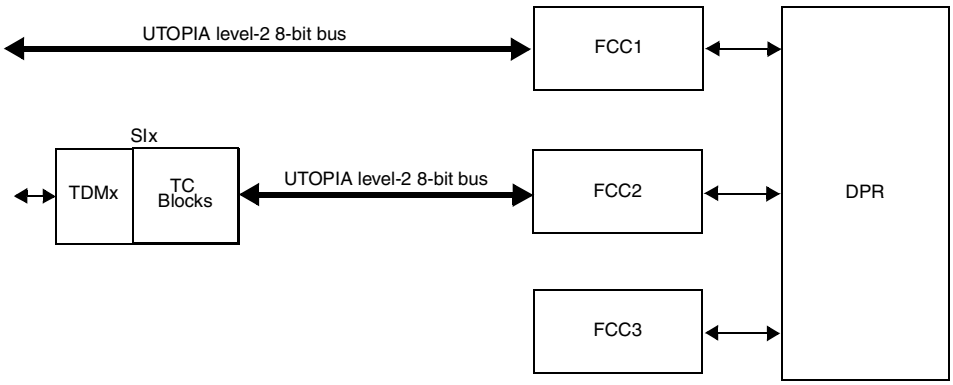


Figure 14. Serial ATM Interface Using FCC2 and TC blocks (single channel)

14.4.3 TC Layer Support

The TC layer block is shown in **Figure 15**. The transmit and the receive parts are independent; the only case in which they are synchronized is in cell echo mode.

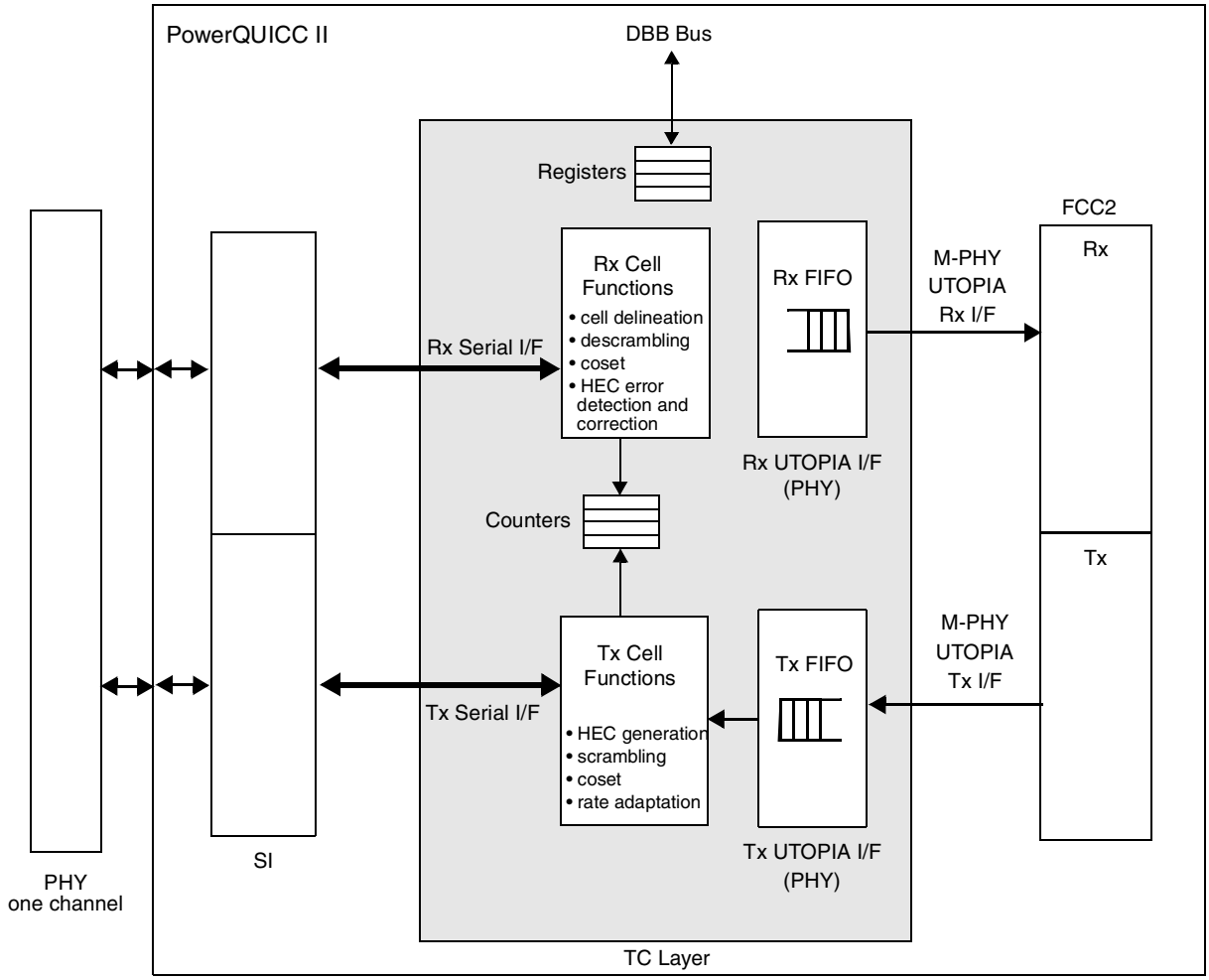


Figure 15. TC Layer Block Diagram

14.4.4 Signals

The TC layer operates via an SI TDM port or NMSI port using a serial protocol. Synchronization signals are required for some applications and must be supported. **Table 33** describes the signals for operating the TC layer.

Table 33. TC Layer Signals

Signal	Direction	Description
TXC	Input	Transmit Clock. Clocks Tx data out of the TC to external device.
TXD	Output	Transmit Data from TC to external device.
Tx Sync	Input	Transmit Synch. Synchronizes the transmit data to the beginning of a frame.
RXC	Input	Receive Clock. Clocks the data into the TC.
RXD	Input	Receive Data. From external device to the TC.
Rx Sync	Input	Receive Synch. Synchronizes the received data. Not required in NMSI mode.

14.4.5 Receive ATM Cell Functions

The ATM receive cell functions block (RCF) performs the receive functions of the TC block. It performs cell delineation, cell payload descrambling, HEC verification and correction, and idle/unassigned cell filtering.

Cell delineation is the process of framing data to ATM cell boundaries using the header error check (HEC) received in the ATM cell header. The HEC is a CRC-8 calculation over the first four octets of the ATM cell header. The cell delineation algorithm assumes that repetitive correct HEC calculations over consecutive cells indicate valid ATM cell boundaries.

The RCF performs a sequential bit-by-bit hunt for a correct HEC sequence, and the cell delineation state machine is in HUNT state. When a correct HEC is found, the RCF locks on the particular cell boundary and enters the PRESYNCH state, which indicates that the previously detected HEC pattern is not a false indication. If a correct HEC pattern is false, an incorrect HEC is received within the next DELTA cells. If an incorrect cell is detected, there is a transition to the HUNT state. If an incorrect HEC is not detected in the PRESYNCH state, there is a transition to the SYNCH state. In the SYNCH state, the TC is assumed to be synchronized so that other functions can be applied to the received cell. A transition back to the HUNT state is made only after ALPHA consecutive incorrect HEC patterns are detected.

The cell delineation state machine is shown in **Figure 16**. The ALPHA and DELTA parameters determine the robustness of the delineation method. ALPHA determines the robustness against false misalignment due to bit errors. DELTA determines the robustness against false delineation in the synchronization. Both parameters are programmable for each TC block and are provided to help tune the system according to the line error characteristics of a specific application.

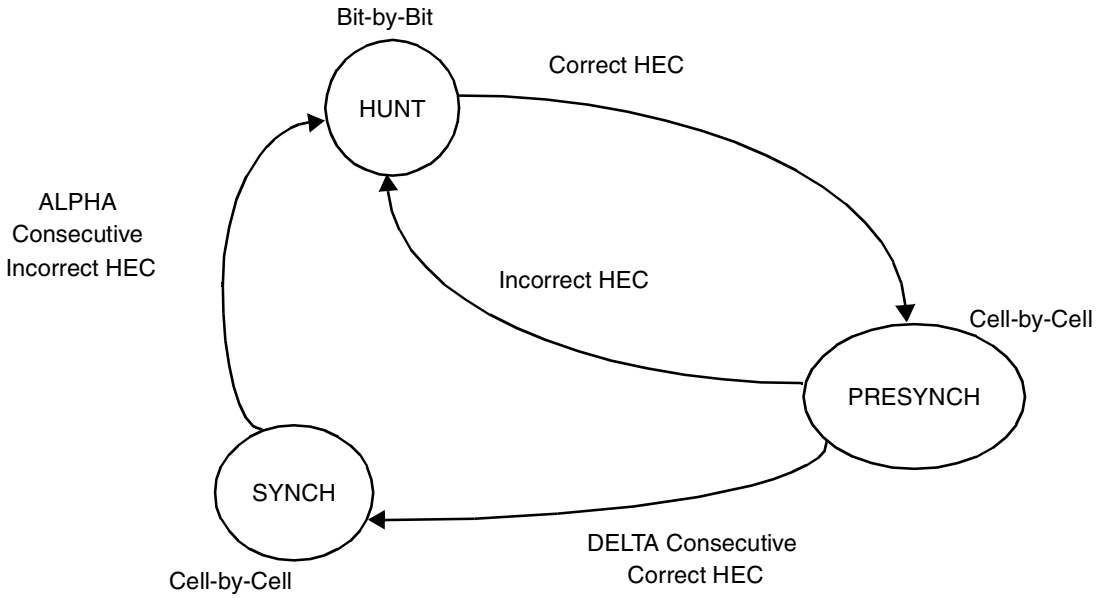


Figure 16. TC Cell Delineation State Machine

The RCF descrambles (programmable) the cell payload using the self-synchronizing descrambler with a polynomial of $x^{43} + 1$.

The HEC calculation is a CRC-8 calculation over the first four octets of the ATM cell header. The RCF verifies the received HEC using the accumulation polynomial, $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) to the received HEC octet before comparison with the calculated result (programmable).

The RCF can perform single bit error correction on the header. If multiple bit errors are found in the HEC, the cell is discarded. If the single bit error correction mode is not enabled (TCMODE[SBC] = 1), the cell is also discarded when a single bit error is found in the header.

When the cell delineation state machine is in the SYNCH state, the HEC verification state machine (see **Figure 17**) implements the correction algorithm. This state machine ensures that a single cell header is corrected at a time. If consecutive cells are detected with single bit errors in their headers, only the first cell error is corrected and the rest are discarded. This state machine reduces the delivery of cells with headers containing errors under bursty error conditions.

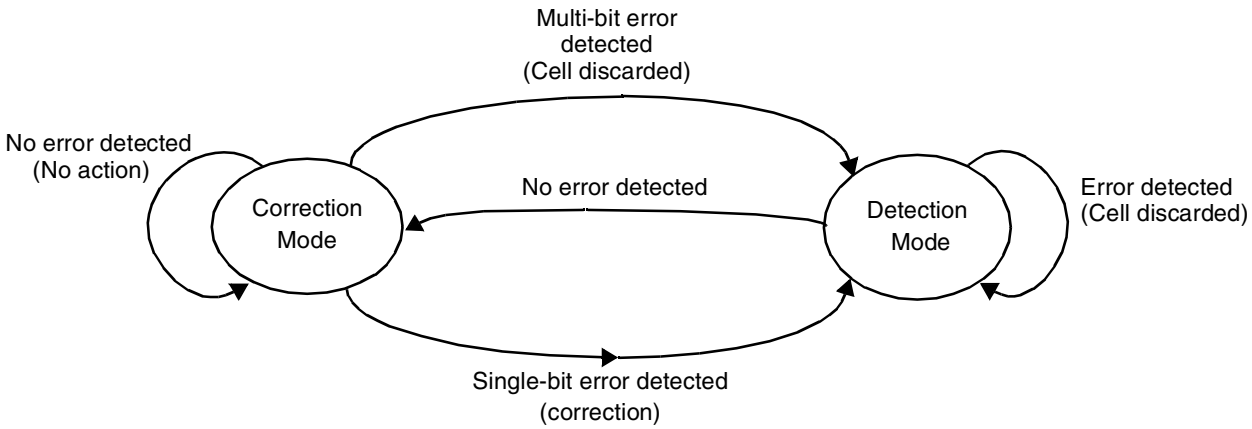


Figure 17. HEC: Receiver Modes of Operation

The RCF can also perform idle/unassigned cell filtering. Both features are programmable (TCMODE[CF]). Cells that are detected to be idle/unassigned are discarded, that is, not forwarded to the UTOPIA interface Rx FIFO.

14.4.6 Receive ATM 2-Cell FIFO

The receive FIFO provides FIFO management and an interface to the UTOPIA receive cell interface. The receive FIFO can hold 2 ATM cells, thereby providing the cell rate decoupling function between the transmission system physical layer and the ATM layer. FIFO management includes filling the FIFO, indicating to the UTOPIA interface that it contains cells, maintaining the FIFO read and write pointers, and detecting FIFO overrun (TCER[OR]) conditions.

14.4.7 Transmit ATM Cell Functions

The transmit ATM cell functions block (TCF) performs the ATM cell payload scrambling and is responsible for the HEC generation and the idle/unassigned cell generation. The TCF scrambles (programmable by the user) the cell payload using the self-synchronizing scrambler with polynomial $x^{43} + 1$. The HEC is generated using the polynomial $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) (programmable by the user) to the calculated HEC octet. The result overwrites the HEC octet on the transmitted cell. When the transmit FIFO is empty, the TCF inserts idle/unassigned cells (counted in ICC). The TCF accumulates the number of transmitted assigned cells in a counter (TCC).

14.4.8 Transmit ATM 2-Cell FIFO

The transmit FIFO provides FIFO management and an interface to the UTOPIA transmit interface. The FIFO provides the cell rate decoupling between the transmission system physical layer and the ATM layer.

The FIFO management includes emptying cells from the transmit FIFO, indicating to the UTOPIA interface that it is full, maintaining the FIFO read and write pointers, and detecting FIFO underrun (TCER[UR]) conditions.

14.4.9 Rx UTOPIA Interface

T he receive interface with the FCC via the UTOPIA bus implements the UTOPIA level-2 (multi-PHY) 8-bit PMD side (slave) interface.

14.4.10 Tx UTOPIA Interface

T he transmit interface with the FCC via the UTOPIA bus implements the UTOPIA level-2 (multi-PHY) 8-bit PMD side (slave) interface.

14.4.11 TC Layer Registers

Each TC layer block is controlled by registers in the block and accessed from the 60x-compatible bus.

14.4.12 TC Layer Mode Register (TCMODE)

Each TC layer block is configured using a TC layer mode register TCMODEx, as shown in **Figure 18**.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXEN	TXEN	RPS	TPS	RC	TC	SBC	CF		URE	LB		TBA	IMA	SM	CM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 18. TCMODEx

Table 34 describes TCMODE fields.

Table 34. TCMODE Field Descriptions

Name	Description	Settings	
RXEN 0	**TC Layer Receive Enable** Enables the TC Layer Rx block operation	0 1	TC Layer Rx operation disabled TC Layer Rx operation enabled
TXEN 1	**TC Layer Transmit Enable** Enables the TC Layer Tx block operation	0 1	TC Layer Tx operation disabled TC Layer Tx operation enabled
RPS 2	**Receive Payload Descrambling Disable** Disables payload descrambling on received payload data	0 1	Received payload descrambling No received payload descrambling
TPS 3	**Transmit Payload Scrambling Disable** Disables payload scrambling on transmitted payload data	0 1	Transmit payload scrambling No transmit payload scrambling
RC 4	**Receive Coset Disable** Disables XOR with 0xAA on received HEC	0 1	XOR with 0xAA on received HEC No XOR with 0xAA on received HEC
TC 5	**Transmit Coset Disable** Disables XOR with 0xAA on transmitted HEC	0 1	XOR with 0xAA on transmitted HEC No XOR with 0xAA i on transmitted HEC
SBC 6	**Header Single Bit error Correction Disable** Disables single bit correction on the header according to HEC in Sync mode.	0 1	Single bit error correction No single bit error correction

Table 34. TCMODE Field Descriptions (Continued)

Name	Description	Settings	
CF 7–8	**Receive Idle/unassigned Cells Filtering** Selects the type of filtering to perform on received cells. The header of an idle cell (ITU-T I.361) is 0b00000000_00000000_00000000_00000001. The header of an unassigned cell (ITU-T I.361) is 0b00000000_00000000_00000000_0000xxx0 Physical layer cells bypass the TC layer; they are not filtered. The filter works on the header only and ignores the HEC.	00 01 10 11	No cell filtering is done on Rx cells. Idle cell filtering is done and idle cells are discarded. Unassigned cell filtering is done and unassigned cells are discarded. Idle and unassigned cell filtering is done and both idle and unassigned cells are discarded.
URE 9	**Underrun interrupt (TCER[UR]) enable** An underrun interrupt may be set when an idle cell is generated by the TC.	0 1	Underrun interrupt disabled Underrun interrupt enabled
LB 10–11	**Loopback/echo modes** For echo mode operation, clear TCMODE[SM], independent of the FCC multi-PHY mode configuration.	00 01 10 11	Normal operation. Cell echo mode operation. Received cells are transmitted and do not go out to the UTOPIA bus. Data loopback mode operation. Transmit data stream is connected to the receive data stream. Not used.
TBA 12	**Transmit Byte Align** Enables alignment of transferred bytes to the Tx Sync signal.	0 1	Tx data is transferred as soon as it is enabled Tx data is transferred byte-aligned to the Tx Sync signal
IMA 13	**IMA mode** Enables IMA mode.	0 1	Rx is not in IMA Rx is in IMA mode
SM 14	**Single Mode** Indicates whether the TC is the only PHY on the UTOPIA interface.	0 1	TC is not the only PHY on UTOPIA TC is the only PHY on UTOPIA
CM 15	**Cell Counters Mode** Disables the counter read and then clear function.	0 1	Reading a cell counter clears the counter Reading a cell counter does not change the counter value

14.4.13 Cell Delineation State Machine Register (CDSMRx)

The cell delineation state machine register (CDSMR), shown in **Figure 19**, holds the ALPHA and DELTA parameters of the cell delineation state machine.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ALPHA					DELTA					—					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 19. CDSMRx

Table 35 describes the CDSMR fields.

Table 35. CDSMR Field Descriptions

Name	Description
ALPHA 0-4	**ALPHA** Consecutive received cells with incorrect HEC are counted by the cell delineation state machine to pass from state SYNCH to state HUNT.
DELTA 5-9	**DELTA** Consecutive received cells with correct HEC are counted by the cell delineation state machine to pass from state PRESYNCH to state SYNCH.
— 10–15	Reserved. Write to 0 for future compatibility.

14.4.14 TC Layer Event Register (TCERx)

The TC layer event registers (TCERx), as shown in **Figure 20**, records error events for each TC block. TCER event bits are cleared by writing ones to them.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OR	UR	CDT	MS	PARE	—					ROF	TOF	EOF	COF	IOF	FOF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 20. TCERx

The TCER bits are described in **Table 36**.

Table 36. TCER Field Descriptions

Name	Description		Description
OR 0	**Overrun** Indicates whether the Rx FIFO is full when a complete cell is received.	0 1	The Rx FIFO is not full or another complete cell is not received. The Rx FIFO is full and another complete cell is received. The cell is discarded.
UR 1	**Underrun** Indicates whether there is no ATM cell to transmit. The interrupt is enabled only if TCMODE[URE] is set.	0 1	The Tx FIFO is not empty or the cell transmission is not completed. The Tx FIFO is empty and the cell transmission is completed. An idle cell is sent. The idle cell header is 0x00000001 (I.432), whose HEC is 0x52. The idle cell payload is 0x6A (I.432).
CDT 2	**Cell delineation toggled** Indicates whether the cell delineation bit (TCGSR[CD]) bit changed.	0 1	TCGSR[CD] did not change. TCGSR[CD] changed.
MS 3	**Misplaced Tx Sync signal** Indicates whether the Tx Sync is out of place. The first Tx Sync is by definition always in place.	0 1	Tx Sync is not out of place. Tx Sync is out of place.
PARE 4	**Parity event** Indicates whether the parity is incorrect.	0 1	The transmit parity from UTOPIA is correct. The transmit parity from UTOPIA is wrong.
— 5–9	Reserved. Write to 0 for future compatibility.		
ROF 10	**Received cell counter overflow** Indicates that the received cells counter passed its maximum value.	0 1	The received cells counter did not pass its maximum value. The received cells counter passed its maximum value.
TOF 11	**Transmitted cell counter overflow** Indicates that the transmitted cells counter passed its maximum value.	0 1	The transmitted cells counter did not pass its maximum value. The transmitted cells counter passed its maximum value.
EOF 12	**Errorred cells counter overflow** Indicates that the errorred cells counter passed its maximum value.	0 1	The errored cells counter did not pass its maximum value. The errored cells counter passed its maximum value.
COF 13	**Corrected cells counter overflow** Indicates that the corrected cells counter passed its maximum value.	0 1	The corrected cells counter did not pass its maximum value. The corrected cells counter passes its maximum value.
IOF 14	**Tx Idle cells counter overflow** Indicates that the Tx idle cells counter passed it maximum value.	0 1	The Tx idle cells counter did not pass its maximum value. The Tx idle cells counter passes its maximum value.
FOF 15	**Filtered cells counter overflow** Indicates that the filtered cells counter passed its maximum value.	0 1	The filtered cells counter did not pass its maximum value. The filtered cells counter passed its maximum value.

14.4.15 TC Layer Mask Register (TCMRx)

The TCMRx field description is identical to that of TCER (refer to **Section 14.4.14**). Each bit that is set in TCMR enables an interrupt when the corresponding bit in TCER is set.

14.4.16 TC Layer General Registers

The TC layer general registers are distributed to all of the TC blocks. Each TC block is represented by specific bits. When accessing a general register, each TC block is responsible only for its bits.

14.4.17 TC Layer General Event Register (TCGER)

The TC layer general event register (TCGER), shown in **Figure 21**, summarizes the events for all the TC blocks. Each bit stands for an ORed event register of a TC block. When a bit is set, it indicates that one or more event bits are set in the corresponding TC block event register.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TC1	—				TC6	TC7	TC8	—							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 21. TCGER

Table 37 describes TCGER fields.

Table 37. TCGER Field Descriptions

Name	Description	Settings	
TC1 0	**TC1 Event** Indicates whether a TC layer 1 event occurred.	0 1	No event. One or more bits are set in TC1 event register.
— 1–4	Reserved. Write to 0 for future compatibility.		
TC6 0	**TC6 Event** Indicates whether a TC layer 6 event occurred.	0 1	No event. One or more bits are set in TC6 event register.
TC7 0	**TC7 Event** Indicates whether a TC layer 7 event occurred.	0 1	No event. One or more bits are set in TC7 event register.
TC8 0	**TC8 Event** Indicates whether a TC layer 8 event occurred.	0 1	No event. One or more bits are set in TC8 event register.
— 8–15	Reserved. Write to 0 for future compatibility.		

14.4.18 TC Layer General Status Register (TCGSR)

Figure 22 shows the TC layer general status register (TCGSR), which records the cell delineation and transmit FIFO status for all TC blocks.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	CD1	—				CD6	CD7	CD8	—							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R															

Figure 22. TCGSR

Table 38 describes TCGSR fields.

Table 38. TCGSR Field Descriptions

Name	Description	Settings
CD1 0	**Cell Delineation** The cell delineation state machine status of TC1.	0 Cell delineation state machine is in Hunt or Pre-Synch mode. 1 Cell delineation machine is in Synch mode.
— 1–4	Reserved. Write to 0 for future compatibility.	
CD6 0	**Cell Delineation** The cell delineation state machine status of TC6.	0 Cell delineation state machine is in Hunt or Pre-Synch mode. 1 Cell delineation machine is in Synch mode.
CD7 0	**Cell Delineation** The cell delineation state machine status of TC7.	0 Cell delineation state machine is in Hunt or Pre-Synch mode. 1 Cell delineation machine is in Synch mode.
CD8 0	**Cell Delineation** The cell delineation state machine status of TC8.	0 Cell delineation state machine is in Hunt or Pre-Synch mode. 1 Cell delineation machine is in Synch mode.
8–15	Reserved. Write to 0 for future compatibility.	

14.4.19 TC Layer Cell Counters

Each TC block maintains six memory-mapped 16-bit performance cell counters listed in **Table 39** that are updated during operation and can be read by the host. If a counter overflows, it wraps back to zero and generates a maskable interrupt. These counters are automatically cleared when read if TCMODE[CM] = 0; see **Section 14.4.12**.

Table 39. TC Performance Cell Counters

Counter Name	Description
Received Cell Counter (RCC)	This cell counter is updated whenever a received cell without HEC errors is passed to the Rx UTOPIA FIFO.
Transmitted Cell Counter (TCC)	This cell counter is updated whenever the transmission of a cell is completed.
Errored Cell Counter (ECC)	This cell counter is updated whenever a received errored cell (cell with header error) is discarded.
Corrected Cell Counter (CCC)	This cell counter is updated whenever a received cell with a HEC single bit error is corrected. If header single bit error correction is not enabled (TCMODE[SBC] is set), this counter is not updated. (All errored cells are counted by the errored cell counter (ECC).)
Tx IDLE Cell Counter (ICC)	This cell counter is updated whenever an idle cell is transmitted.
Filtered Cell Counter (FCC)	This cell counter is updated whenever an idle/unassigned cell is filtered (discarded). If cell filters are not enabled (TCMODE[CF] is cleared), this counter is not updated.

14.4.20 Programming FCC2

FCC2 is designed to work with the TC blocks. The TC blocks are located on fixed addresses on the UTOPIA bus internally. FCC2 should be programmed to work with the TC blocks as if the TC blocks are external PHYs located on the lowest eight (or fewer) addresses.

14.4.21 Programming and Operating the TC Layer

The host should first program the mode registers of each TC block to be active, according to the number of TC channels required. Then, FCC2 is programmed to work on the UTOPIA interface with the active TC blocks. Finally, FCC2 and the PHYs of the TC channels are enabled. The transmit channels for each TC block are enabled by setting TCMODEx[TXEN]. The receive channels for each TC block are enabled by setting TCMODEx[RXEN]. The host polls the CD bits of each enabled TC block to verify that its receive cell delineation state machines are synchronized. For every TC block that is synchronized, the host clears the CDT bit in its event register. When all the enabled TC blocks are synchronized, the host terminates its initialization routine, and the system starts normal operation.

When a TC block gets out of synchronization, the corresponding TCGSR[CD] is cleared. This change causes a TCER[CDT] interrupt to the host (if enabled).

On the receive path, the TC performs the following functions:

1. Receives the bit stream via the SI.
2. Attempts to gain synchronization on the ATM cell boundaries by checking each byte (HEC candidate) against the HEC calculated on the preceding 32 bits (ATM cell header candidate).
3. Once synchronized, performs the descrambling function on the cell payload (if enabled), performs the coset function on the HEC (if enabled), checks for HEC errors and corrects single HEC errors when found (again, if enabled). Cells containing multi-bit header errors (at least 2 errors) are discarded. Idle and unassigned cells are filtered (discarded) when detected (if the filters are enabled). Once a cell is processed, it passes to the TC receive FIFO, and the internal TC cell counters are updated. The cell is passed from the TC receive FIFO via the UTOPIA interface to the FCC2 receive FIFO.

An overrun condition occurs when the TC receive FIFO is full, the CP is busy, and the FCC cannot read a cell from it via the UTOPIA interface before another valid cell is received. The incoming cell is discarded and TCER[OR] interrupt is sent to the host (if enabled).

On the transmit path, once enabled, the TC performs the following functions:

1. Starts requesting for cells to send via the UTOPIA interface.
2. When a cell is passed via the UTOPIA interface to the TC transmit block, it is stored in the TC transmit FIFO.
3. When a cell is to be sent, it is read from the TC transmit FIFO and is processed. The scrambling function is performed on its payload (if enabled), its header HEC value is calculated and the coset function is performed on the HEC (if enabled).
4. The cell is then sent to the PHY via the SI. Once the cell transmission is complete, the relevant TC cell counters are updated.

An underrun condition occurs when a cell is to be sent to maintain the bit rate, but the TC transmit FIFO is empty. An idle cell is sent instead. This condition generates a TCER[UR] interrupt (if not masked) if TCMODE[URE] is set. When a TC cell counter overflows, an interrupt is set (if enabled).

A TC channel provides the data rate dictated by the PMD device by operating the FCC2 in one of two modes:

- *External Rate*. In this mode, the external device determines the data rate. The CP keeps the FCC FIFO full by inserting ATM cells or idle cells (if ATM cells are not available) into the FCC FIFO whenever it is not full. This operation ensures that the cell stream is the data rate required by the PMD. In general, the TC transmit FIFO is never empty, and thus would not need to generate idle cells. However, if the CP is busy, and the TC is forced to generate an idle cell because its transmit FIFO is empty, an underrun condition occurs. The UR interrupt is sent to the host (if not masked) if TCMODE[URE] is set. See **Figure 23**. This mode generates a greater CP load than the internal rate mode.

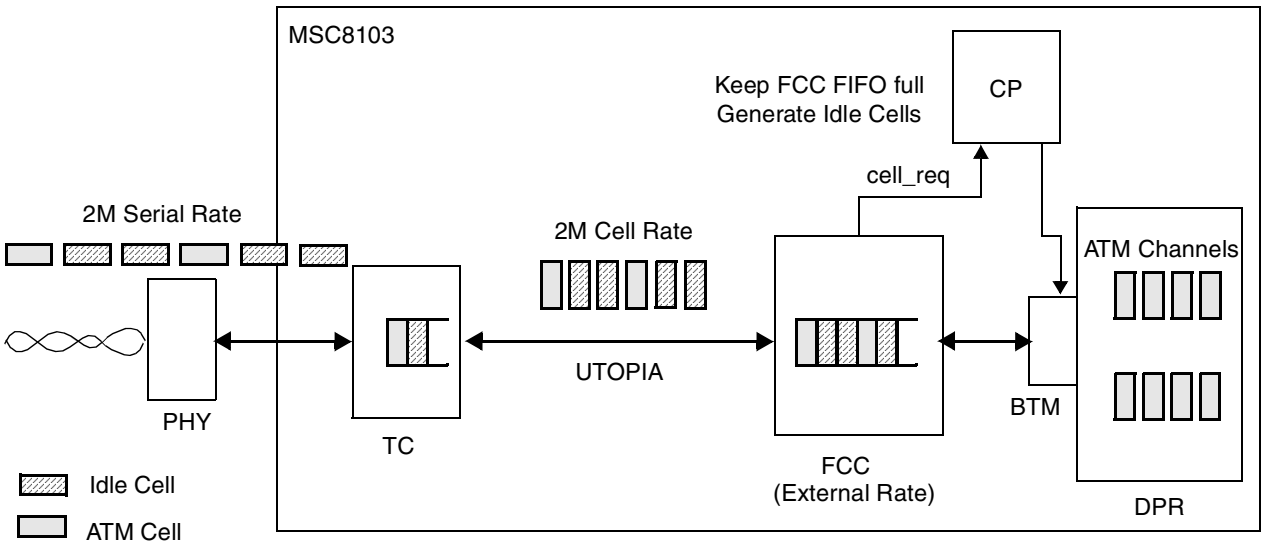


Figure 23. TC Operation in FCC External Rate Mode

- *Internal Rate (Sub Rate)*. In this mode, the FCC supplies only four PHY devices that have four distinct addresses. Each channel is controlled by a dedicated hardware timer that is programmed and tuned to the data rate needed. When a timer expires, a valid cell is sent to the corresponding PHY. The PHY sends idle cells to keep its synchronization and transmission rate. The same is true for the TC. Once activated with FCC2 in this mode, the TC requests cells and sends idle cells (UR interrupts can be disabled by clearing TCMODE[URE]) until a valid cell is transferred via the UTOPIA bus from the FCC. See **Figure 24**.

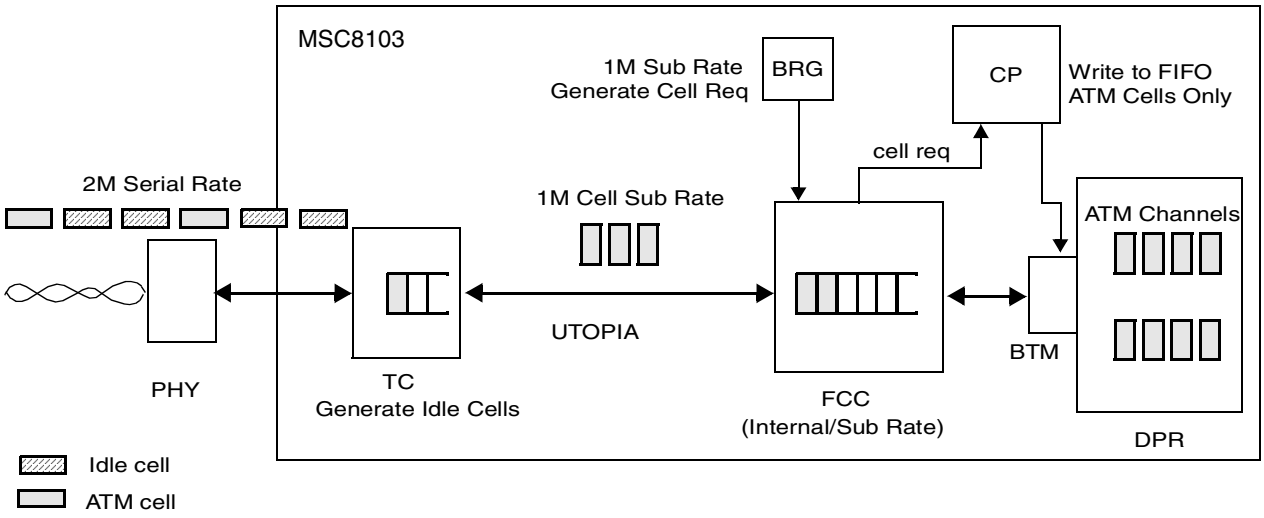


Figure 24. TC Operation in FCC Internal Rate Mode (Sub Rate Mode)

Operation in byte aligned mode (TCMODE[xTBA] = 1) is required for T1/E1 mainly. In this mode, once the TC is enabled, it waits for the first Txsyn pulse to start transmit the first byte of the first cell. This ensures that subsequent Txsyn pulses are byte-aligned to the cell boundaries.

14.4.22 Operating the TC Layer at Higher Frequencies

The operation of the TC layer requires a minimum frequency ratio of 1:2.5 between the serial clock and the UTOPIA clock (in Rx and Tx separately). Using the TC for serial frequencies greater than 10 MHz requires using higher UTOPIA frequencies to preserve that ratio.

14.4.23 Programming a T1 Application

This section describes how to implement a T1 application using a single TC layer block. Using two or more TCs requires FCC2 to work in MPHY mode. Assuming that the required ATM parameters and data structures have been set up and initialized, implementing a T1 application requires the following steps:

1. *Program FCC2*. To set up and initialize FCC2, program the FPSMR and GFMR as shown in **Table 40**. This is for working with one TC block operating in a single PHY environment. The transmitter and receiver should not be enabled at this time. In this example, FCC2 does not discard idle cells.

Table 40. Programming GFMR and FPSMR to Set Up FCC2

Init Values	Description
FPSMR2 = 0x0080_0000	UTOPIA Rx and Tx in master mode, idle cells are not discarded
GFMR2 = 0x0000_000A	ATM protocol mode, receiver and transmitter are disabled

2. *Set up I/O ports and clocks*. Because the FCC2 UTOPIA bus is internally connected to the TC UTOPIA bus, program the parallel ports and BRGs for the active TDM(s).
3. *Enable Tx/Rx on FCC2*. To enable receiving and transmitting, program FCC2 as shown in **Table 41**.

Table 41. Enable FCC2

Init Values	Description
GFMR2 = 0x0000_003A	Enable Rx and Tx

4. *Program CPM multiplexing*. To define the connection of FCC2, program the CPM multiplexing as shown in **Table 42**.

Table 42. Programming the CPM Multiplexing for a TI Application

Init Values	Description
CMXFCR = 0x0080_0000	FCC2 is connected to the TC Layer
CMXUAR = 0x0000	FCC2 as UTOPIA master

5. *Program the TC block*. The TCx layer block is configured using the TCMODEx and CDSMR1 registers as shown in **Table 43**. Note that the TC layer must be enabled after both FCC2 and the CPM multiplexing are programmed.

Table 43. Programming the TC Layer Block

Init Values	Description
TCMODE1 = 0xC202	Enable TC Layer Rx and Tx, no error correction on header, the TC is the only PHY on UTOPIA
CDSMR1 = 0x3980	ALPHA = 7, DELTA = 6 (default values)

6. *Program the serial interface (SI)*. Program the SI to retrieve the data bits (192 bits) out of the T1 frame (193 bits). The SI frame pattern is programmed in the SI RAM (Rx or Tx), as shown in **Table 44**.

Table 44. Programming the SI RAM (Rx or Tx) for a T1 Application

Init Values	Description
SI_RAM[00] = 0x0000	1 bit is ignored.
SI_RAM[02] = 0x015E	Route 8 bytes to FCC2.
SI_RAM[04] = 0x015E	Route 8 bytes to FCC2.
SI_RAM[06] = 0x015F	Route 8 bytes to FCC2 and go back to the first entry in table.

7. *Enable TDM*. The Initialize the serial interface registers and enable TDM*x*—in this case TDMA on SI1, as shown in **Table 45**.

Table 45. Programming SI Registers to enable TDM

Init Values	Description
SI1AMR = 0x0040	Common Receive and Transmit Pins for TDMa
SI1GMR = 01	Enable TDMA

14.5 New MCC Host Commands

Four new MCC commands are added by the 2K87M mask set as part of the CP command opcodes. **Table 46** lists the new CP command opcodes changed in the 2K87M mask set. The fields are identical to those defined in the 2K42A mask set, except that definitions have been added for the MCC.

Table 46. 2K87M Mask Set CP Command Opcodes

Opcode	Channel									
	FCC	SCC	SMC (UART/ Transparent)	SMC (GCI)	SPI	I^2C	IDMA	MCC	Timer	Special
0011	ENTER HUNT MODE	ENTER HUNT MODE	ENTER HUNT MODE	—	—	—	—	INIT RX AND TX PARAMS (one channel)	—	—
0101	GRACEFUL STOP TX	GRACEFUL STOP TX	—	—	—	—	—	INIT TX PARAMS (one channel)	—	—
0110	RESTART TX	RESTART TX	RESTART TX	—	—	—	—	INIT RX PARAMS (one channel)	—	—
0111	CLOSE RX BD	CLOSE RX BD	CLOSE RX BD	—	CLOSE RX BD	CLOSE RX BD	—	MCC RESET	—	—

Table 47 describes the additional commands listed in **Table 46**; all other commands are unchanged and are as described in the *MSC8101 Reference Manual*.

Table 47. 2K87M Mask Set New Command Descriptions

Command	Description
INIT MCC RX AND TX PARAMS — ONE CHANNEL	Initializes the receive and transmit parameters of the peripheral controller. Differs from INIT RX AND TX PARAMS in that, for the MCCs, issuing INIT RX AND TX PARAMS initializes 32 consecutive channels beginning with the channel number specified in CPCR[MCN]. However, issuing INIT MCC RX AND TX—ONE CHANNEL initializes only the channel in the command.
INIT MCC RX PARAMS— ONE CHANNEL	Initializes MCC receive parameters for only a single channel according to MCC channel number field.
INIT TX PARAMS— ONE CHANNEL	Initializes MCC transmit parameters for only a single channel according to MCC channel number field.
MCC RESET	Provides a hard reset to the MCC FIFOs. To use this command, software should execute the following sequence: **1.** Disable the TDM by clearing the appropriate enable bit in SIxGMR[4-7]. **2.** Issue the MCC RESET command. **3.** Issue the INIT RX AND TX command. **4.** Reprogram the specific MCC channel, global parameters, and any BDs that need to be updated. **5.** Set the appropriate enable bit in SIxGMR[4–7].

15 Errata

Table 48 lists the functional errata that the 2K87M mask removes and the current 2K87M mask set errata. Always refer to the Freescale website listed on the back page of this document for a current list of device errata.

Table 48. Errata Resolved in Mask Set 2K87M

Errata Number	Errata Description	Applies to Mask
SIU1	**Wrong Timer Advancement on RCCR** **Date Added:** 5/30/2000: **Description:** PQ2 treats the RCCR[TIMEP] value (UC timer) differently then QUICC. In QUICC the timer advanced (N+1)*1024 cycles and in MSC8101 and MPC826X the timer advances N*1024 cycles. **Workaround:** None **System Number:** 1399	0K40A 1K42A 2K42A
SIU4	**Incorrect Masking of MCP** **Date Added:** 5/30/2000: **Description:** MCP (Machine Check Interrupt) due to data errors (parity / ECC) is masked by the SWRI bit in SYPCR. **Workaround:** Clear the SWRI bit in SYPCR to get data errors indication. **System Number:** 3695	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
SIU8	**Parity Checking Error** **Date Added:** 6/13/2000: **Description:** During a read from a device with port size less than 64 bits, from an address not aligned to 64 bits, the parity bits for parity check are not taken from the correct locations. For example, for a read of 4 bytes from a 32-bit port size from address 4, the parity is checked against DP[4–7] while it should be checked against DP[0–3]. The bug exists for both normal and rmw parity, and for both System and Local buses. **Workaround:** None **System Number:** 5839	0K40A 1K42A 2K42A
SIU9	**Bus Monitor Asserts Spurious $\overline{\text{TEA}}$ After Address Retry.** **Date Added:** 1/28/2001 **Description:** The bus monitor will not recognize the competition of an Address Retry transaction and will assert TEA if there is no bus activity for a time equal to the expiration time. **Workaround:** Disable the bus monitor in systems where Address Retry cycles are used (e.g. systems which include PowerSpan).	0K40A 1K42A 2K42A
SIU10	**Strict Enforcement of Requirement to Assert $\overline{\text{DBG}}$ and $\overline{\text{TS}}$ in Same Cycle When Core Enabled** **Date Added:** 1/28/2001 **Description:** This is a compatibility note. An external arbiter must assert $\overline{\text{DBG}}$ in the same clock in which $\overline{\text{TS}}$ is asserted (there may be a one clock delay if the PPC_ACR[DBGD] bit is set, however, after reset this bit is not set by default). Some external arbiters, including the one implemented in PowerSpan, violate this requirement. As a result, the system hangs following the first bus access after reset. **Workaround:** Use only a compliant external arbiter or the internal MSC8101 arbiter.	0K40A 1K42A 2K42A
QSIU3	**TEA May Hang 60x-Compatible Bus** **Date Added:** 5/30/2000 **Description:** TEA may hang the 60x-compatible bus if it is asserted between specific address and data phases during a split transaction. **Workaround:** Enable Bus Monitor. **Fix Plan:** Rev. A	0K40A 1K42A 2K42A
QSIU5	**Incorrect Data on 60x-Compatible Bus** **Date Added:** 6/13/2000 **Description:** The following sequence on the 60x-compatible bus can result in incorrect data: 1. Read transaction with $\overline{\text{DACK}}$ before $\overline{\text{AACK}}$. 2. Failed atomic write transaction. 3. Write transaction. **Workaround:** None. **System Number:** 5823	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

<table>
<tr><th>Errata Number</th><th>Errata Description</th><th>Applies to Mask</th></tr>
<tr><td>QSIU6</td><td>EE[4–5] Pins are Sampled on $\overline{\text{HRESET}}$,$\overline{\text{SRESET}}$ and $\overline{\text{PORESET}}$
Date Added: 8/31/2000
Description: The EE[4–5] pins should be sampled only on $\overline{\text{PORESET}}$, but they are sampled on $\overline{\text{SRESET}}$ and $\overline{\text{HRESET}}$ as well. Changing the value of the EE[4–5] pins after $\overline{\text{PORESET}}$ deassertion might prevent the chip from booting, because their value might choose a different or undefined boot configuration.
Workaround: The EE[4–5] values should be kept constant and equal to the values on $\overline{\text{PORESET}}$.</td><td>0K40A
1K42A
2K42A</td></tr>
<tr><td>QSIU7</td><td>Pin $\overline{\text{IRQ7}}$_$\overline{\text{INTOUT}}$ Not Open Drain
Date Added: 9/6/2000
Description: INOUT pin $\overline{\text{IRQ7\_INTOUT}}$ should be open-drain but not implemented as one.
Workaround: Buffer $\overline{\text{INTOUT}}$ on the board when it is wire ORed.</td><td>0K40A
1K42A
2K42A</td></tr>
<tr><td>QSIU12</td><td>Limited Clock Modes
Date Added: 4/2/2001, Modified 2/19/2002
Description: Single-master and Multi-master systems are limited to clock mode 57 for 1K42A and 46 or 57 for 2K42A (see also QSIU14). Note that in mode 46 a 33MHz CLKIN can be used which will result in a 16.5MHz input to the SPLL. Although this is slower than the specified 18MHz (refer to the MSC8101 Data Sheet), this is not an issue in mode 46.
Available 1K42A Clock Modes
<table><tr><th>MODCK</th><th>BUS:CPM:CORE</th><th>BUS:CLKIN</th><th>MAX BUS</th><th>MAX CPM</th><th>MAX CORE</th></tr><tr><td>57</td><td>1:2.5:5</td><td>1.0</td><td>55</td><td>138</td><td>275</td></tr></table>Available 2K42A Clock Modes
<table><tr><th>MODCK</th><th>BUS:CPM:CORE</th><th>BUS:CLKIN</th><th>MAX BUS</th><th>MAX CPM</th><th>MAX CORE</th></tr><tr><td>46</td><td>1:2:4</td><td>2.0</td><td>69</td><td>138</td><td>275</td></tr><tr><td>57</td><td>1:2.5:5</td><td>1.0</td><td>55</td><td>138</td><td>275</td></tr></table></td><td>1K42A
2K42A</td></tr>
<tr><td>QSIU13</td><td>Software Watchdog Cannot be Enabled after Boot from Host
Date Added: 8/5/2001
Description: The software watchdog is disabled when booting from host. It cannot be subsequently enabled because the SYPCR can only be written once.
Workaround: None</td><td>0K40A
1K42A
2K42A</td></tr>
</table>

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
QSIU14	**Non-Functional DLL** **Date Added:** 11/25/2001 **Description:** DLL may fail to lock. **Workaround:** Use DLL disabled mode by setting the DLLDIS bit in the Reset Configuration Word. To maximize bus performance, use a zero-delay buffer for CLKOUT for both single-master and multi-master systems. **System Number:** 7427	1K42A 2K42A
QSIU15	**60x Compatible Global Transaction Fail on RETRY** **Date Added:** 5/30/2000 **Description:** Data may be lost on RETRY when global transactions are performed in 60x compatible mode. **Workaround:** When global transactions are used, 60x compatible mode cannot be used. **System Number:** 5678	0K40A 1K42A 2K42A
QSIU16	**ALE Output During Reset** **Date Added:** 2/20/2003 **Description:** ALE behavior is not guaranteed during reset. This affects only multi-master systems which perform reset configuration from external memory and which use ALE for all memory accesses. ALE recovers with the first access after reset. **Workaround:** None **System Number:** **Fix Plan:** RevA	0K40A 1K42A 2K42A
DMA1	**DMA Data Corruption on either PPC Bus or Local Bus** **Date Added:** 2/19/2002 **Description:** Data transferred by the DMA on either the PPC Bus or Local Bus may be corrupted. **Workaround:** For DMA accesses on the PPC Bus - disable PPC bus pipeline by setting BCR[PLDP]=1. For DMA accesses on the Local Bus the UPMC programming patch is available. **System Number:** 7462	0K40A 1K42A 2K42A
BOOT1	**Incorrect Checksum for Host Bootload** **Date Added:** 8/15/2000: **Description:** The host bootload calculates erroneous checksum. **Workaround:** Clear ICR[HF3] so that the host bootload ignores the checksum comparison result. **System Number:** 6178	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask

BOOT2	**Boot Interference in Multi-Master System with Shared Memory**
	Date Added: 8/5/2001
	Description: The DSPRAM address in the memory map as programmed by the boot loader code in the ROM is the same for all processors. During simultaneous boot, this will cause interference of one processor with another.
	Workaround: For up to 4 MSC8101's including the configuration master.
	Boot the processors one after the other, and not at the same time. Reprogram unique DSPRAM address for each processor. The configuration master is set also to be the arbitration master and the memory controller for the system. The configuration word for the master should set the MMR field of the reset word (bits [18:19]) to 2'b11. This will mask all the external bus requests of the configuration slaves. After the master completes its boot, the user should:
	1. Clear all external requestors from the Arbitration Level Register (ALR) of the arbitration master.
	2. Reprogram the UPM of the DSPRAM bank to unique address (see programming example below)
	3. Set priority for the next configuration slave in the Arbitration Level Register (ALR) of the arbitration master.
	4. Enable external bus requests by clearing SIUMCR MMR field (bits [16:17]).
	5. Repeat stages 2,3 for consecutive slaves.
	Configuration master programming code:
	`move.l PPC_ALRH,D7 ; Step #1`
	`move.l PPC_ALRL,D8`
	`bmclr.w #$f,D7.L`
	`bmclr.w #$ff00,D8.H`
	`move.l D7,PPC_ALRH`
	`move.l D8,PPC_ALRL`
	`upmc init $04000000 ; Step #2 (DSPRAM base address $0400_0000)`
	`move.l PPC_ALRH,D7 ; Step #3`
	`nop`
	`bmset.w #$7,D7.L`
	`nop`
	`move.l D7,PPC_ALRH`
	`move.l SIUMCR,D7; Step #4`
	`nop`
	`bmclr.w #$c000,D7.L`
	`nop`
	`move.l D7,SIUMCR`
		0K40A
		1K42A
		2K42A

GEN1	**Device Withstands MM ESD of 75V Instead of 100V** **Date Added:** 5/21/2002 **Description:** Device meets the ESD specifications for Human Body Model (HBM) of 1000V and Charged Device Model (CDM) of 500V but does not withstand the Machine Model (MM) requirement of 100V. All pins guaranteed to withstand 75V MM. **Workaround:** None.	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
GEN2	**Unexpected Outputs During Boundary Scan** **Date Added:** 10/18/2002 **Description:** D[32:60] outputs may behave indeterminately during boundary scan. **Workaround:** Reset the device by asserting PORESET while HPE/EE1=0. Alternatively, assert PORESET throughout boundary scan. **Fix Plan:** RevA	0K40A 1K42A 2K42A
SC1	**PC Cannot Be Updated in Debug Mode Entered From Asynchronous Interrupt** **Date Added:** 10/5/2000 **Description:** When a DEBUG instruction is executed in a static or dynamic delay slot created by an asynchronous interrupt, the core enters Debug mode, but the PC cannot be updated. **Workaround:** In the dynamic case, the debugger can use a status bit in the ESR, which indicates whether the core entered a debug in a delay slot. Software workarounds are available for all the static cases. A detailed description was sent by StarCore and can be resent upon request.	0K40A 1K42A 2K42A
SC3	**Erroneous Trace Buffer Value** **Date Added:** 10/5/2000 **Description:** An erroneous 62-bit value may be written to the trace buffer when the EOnCE is programmed to write both event counters (ECNT_VAL and ECNT_EXT). **Workaround:** None.	0K40A 1K42A 2K42A
CPM2	**CAM Access Not Atomic** **Date Added:** 5/30/2000: **Description:** The bus atomicity mechanism for CAM access may not function correctly when the CPM's DMA accesses the CAM. This only affects systems in which multiple CPMs will access the CAM. **Workaround:** None **System Number:** 1410	0K40A 1K42A 2K42A
CPM4	**No CTS Lost Indication with HDLC** **Date Added:** 5/30/2000 **Description:** When CTS is deasserted at the end of hdlc frame, (last flag or one bit before) transmission will be aborted. However there is no CTS-LOST indication. There is only abort indication. **Workaround:** None **System Number:** 1670	0K40A 1K42A 2K42A
CPM5	**Data Corruption on SDMA Flyby** **Date Added:** 5/30/2000 **Description:** The data of a SDMA write, which follows a SDMA flyby read in the local bus may be corrupted. **Workaround:** None **System Number:** 1720	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM6	**Erroneous Report of Overrun on FCC** **Date Added:** 5/30/2000 **Description:** Spurious overrun indications on the FCC may occur in the following cases: **1.** After stop transmit command is issued. **2.** Following CTS lost condition. **3.** Late collision under ethernet. **Workaround:** None **System Number:** 1746	0K40A 1K42A 2K42A
CPM7	**Erroneous Report of Overrun With Fast Ethernet** **Date Added:** 5/30/2000 **Description:** If the CRS (carrier sense) signal is deasserted while fast ethernet frame is transmitted, an overrun error may occur and the FCC may have to be reset. **Workaround:** None **System Number:** 1752	0K40A 1K42A 2K42A
CPM8	**Error on Transmit On Demand Register** **Date Added:** 5/30/2000 **Description:** The TODR mechanism may freeze serial channels. **Workaround:** Do not use TODR. **System Number:** 2484	0K40A 1K42A 2K42A
CPM9	**Erroneous Reception of ATM Cell** **Date Added:** 5/30/2000 **Description:** Under certain conditions, an ATM receiver may receive cells of PHYs which were not addressed for it. Details of the condition: • ATM receiver in UTOPIA slave mode. • FIFO full condition occurred (this happens only when the transmitter violates the UTOPIA standard requirements: transmits data without CLAV). • Transmitter changed selected PHY number. • FIFO full condition ended (CPM read some data from FIFO). **Workaround:** Use different VPI/VCI for different PHYs or expect the cells to be discarded by higher-level protocol software. **System Number:** 2493	0K40A 1K42A 2K42A
CPM10	**Error in ATM Underrun Report** **Date Added:** 5/30/2000 **Description:** In ATM, a Transmit internal rate underrun error is not reported correctly in the TIRU field of the FCCE register. In most cases, TIRU is not set in the FCCE when an internal rate underrun error occurs. In some rare cases that depend on internal sequences within the communications controller, the TIRU bit may be set as expected when the error should be reported. **Workaround:** None **System Number:** 2611	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM11	**False Indication of Shared Flag** **Date Added:** 5/30/2000 **Description:** FCC-TX HDLC - FCT_TXD (data out) changes from 1-->0 for 1 ser_clock period, few clocks after the reset command from MAIN is given. A false shared flag can be detected at the receiver if the last bit before reset was 0, and the receiver considers it as a closing flag of the frame. In most of cases, a CRC error is generated and the frame is discarded. **Workaround:** None **System Number:** 3024	0K40A 1K42A 2K42A
CPM13	**Error in Random Number Generation** **Date Added:** 5/30/2000 **Description:** In Fast Ethernet transmit, when more then one (up to four) frames reside in the FCC FIFO, random number generation (for collision wait) may produce the same number for all four frames. **Workaround:** None **System Number:** 3421	0K40A 1K42A 2K42A
CPM14	**Corruption of ATM Cells** **Date Added:** 5/30/2000 **Description:** Corruption of ATM cells may occur when the following combination is used: AAL1 with UDC in which the user-defined header size is 9 to 12 octets and PM is not used. **Workaround:** Since this problem appears in a very specific condition as described above, avoiding any of the elements (e.g., using cell header of 8 octets) eliminates it.	0K40A 1K42A 2K42A
CPM15	**Corruption of Port D Registers** **Date Added:** 5/30/2000 **Description:** The PDATA, PDATB, PDATC, and PDATD registers can only be written with a 32-bit write instruction. (i.e., stw). When 8- or 16-bit write instructions (i.e., sth, stb) are used, the bits not being written may be corrupted. **Workaround:** Use a 32-bit write instruction only to write to the PDATA, PDATB, PDATC, and PDATD registers. **System Number:** 3679	0K40A 1K42A 2K42A
CPM17	**Error in Reporting UTOPIA Error Condition** **Date Added:** 5/30/2000 **Description:** An FCC receiver which is configured as single PHY master does not detect a UTOPIA error condition when SOC and CLAV are not asserted simultaneously. **System Number:** 3728	0K40A 1K42A 2K42A
CPM21	**False Indication of Collision in Fast Ethernet** **Date Added:** 5/30/2000 **Description:** In the Fast Ethernet a false COL is reported whenever a collision occurs on the preamble of the previous frame. **Workaround:** S/W should ignore COL indications when the CRC of the frame is correct. **System Number:** 3927	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM22	**False Defer Indication in Fast Ethernet** **Date Added:** 5/30/2000 **Description:** In the fast ethernet, if a frame is transmitted due to defer and this frame also gets late collision, a false defer indication is indicated for the next frame. **Workaround:** None **System Number:** 3981	0K40A 1K42A 2K42A
CPM24	**Error in Indicating IDLE Between Frame** **Date Added:** 5/30/2000 **Description:** In the FCC HDLC transmitter, if slow serial clock (cpm_freq/serial_clock > 16) is used, RTS does not transition to IDLE between frames. This means that all the frames are transmitted back-to-back in case there is valid data in the transmitter's FIFO. **Workaround:** None **System Number:** 3998	0K40A 1K42A 2K42A
CPM27	**Error in Heartbeat Checking in FCC** **Date Added:** 5/30/2000 **Description:** The heartbeat checking in FCC transmit ethernet 10Mbps does not work properly. The standard requires that the collision pulse from the PHY should be checked within a window of 4 µsec from the falling edge of the carrier sense. The PQ2 samples the collision signal only once at exactly 4 µsec (10 serial clocks) after the falling edge of the carrier sense signal. **Workaround:** None **System Number:** 4155	0K40A 1K42A 2K42A
CPM28	**Error in Receive Frame Threshold** **Date Added:** 5/30/2000 **Description:** In the SCC Rx in HDLC mode, RFTHR does not work. There is no way to get interrupts on the receive side after a programmable number of frames. **Workaround:** RFTHR should be programed to 1. **System Number:** 4163	0K40A 1K42A 2K42A
CPM29	**MAXD1 and MAXD2 May Not Be Less Than MFLR** **Date Added:** 5/30/2000 **Description:** In SCC Rx ethernet, the option of transferring only part of a frame into memory (MAXD1 and MAXD2 < MFLR) does not work. **Workaround:** None **System Number:** 4166	0K40A 1K42A 2K42A
CPM30	**Graceful Stop Command Does Not Work** **Date Added:** 5/30/2000 **Description:** The graceful stop command does not work in SCC Tx in the following protocols: Ethernet, HDLC, Transparent. **Workaround:** None **System Number:** 4167	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM35	**Data Corruption in SCC Transparent Mode** **Date Added:** 5/30/2000 **Description:** When SCC transparent, envelope mode is used and the received frame size is (4*n) + 1, the last byte is corrupted. When GSMR_H(RFW) - rx FIFO width is used, the received data is completely corrupted, not just the last byte. **Workaround:** The bug can be worked around with a microcode patch. **System Number:** System number; 4350	0K40A 1K42A 2K42A

CPM36	**SI SYNC Timing Restriction**
	Date Added: 5/30/2000
	Description: SI's sync signal may not change exactly on clock edge in the following cases. The bug affects operation only when the SI is in one of two modes:
	`fsd = 00, ce = 0, fe = 0, dsc=1`
	(Sync sampled with falling edge -> Sync should not change on rising edge)
	`fsd = 00, ce = 1, fe = 1, dsc=1`
	(Sync sampled with rising edge -> Sync should not change on falling edge).
	Workaround: When working in these modes, the sync signal to the SI should be manipulated such that it does not change on the exact edge of the serial clock. Toggle the sync at least 5ns after the edge. One way to implement such a workaround is to add a noninverting buffer between the device that generates the sync signal and the MSC8101 that uses it.
	System Number: 3258
		0K40A
		1K42A
		2K42A

CPM38	**Heartbeat Error and Carrier Sense Lost Error On Two Frames** **Date Added:** 5/30/2000 **Description:** There are rare cases when a heartbeat error and carrier sense lost error are reported on two frames. The error is reported in the frame in which it occurred, but in those rare cases it is also reported on an adjacent frame. **Workaround:** None **System Number:** 1547,1550	0K40A 1K42A 2K42A
CPM39	**Corruption in AAL0 Cell Payload** **Date Added:** 5/30/2000 **Description:** There is a rare case in using the ATM AAL0 transmitter that the AAL0 cell payload may be corrupted. This can occur in certain internal sequence of events that the user cannot detect or control. **Workaround:** 1. Use the available microcode patch from the web site. 2. When working with AAL0 SAR, place the TCELL_TMP_BASE 64 byte align plus 4. For example use TCELL_TMB_BASE = 0x2d04 not 0x2d00. **System Number:** 4648a	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM40	**Corruption in AAL0 IDLE Cell** **Date Added:** 5/30/2000 **Description:** There is a rare case when transmitting an ATM idle cell that the idle cell may be corrupted. This can occur in certain internal sequences of events that cannot be controlled or detected by the user. **Workaround:** Place the Idle Base template at address 64 byte align minus 4. For example use Idle_BASE = 0x2cfc not 0x2d00. **System Number:** 4648b	0K40A 1K42A 2K42A
CPM41	**Limitation in ATM Controller** **Date Added:** 5/30/2000 **Description:** There are some limitations in the ATM controller. The first limitation is that only the first 8 PM tables can be used instead of 64. When using these 8 tables, the user must clear the 5 most significant bits of TBD_BASE (in case of Tx PM) or RBD_BASE (in case of Rx PM). The second limitation is that only the first 2048 ATM channel numbers can be used. **Workaround:** There is a microcode patch that can fix the PM limitation. The ATM channel number limitation has no workaround. **System Number:** 4744	0K40A 1K42A 2K42A
CPM42	**Data Corruption in MCC** **Date Added:** 5/30/2000 **Description:** Data corruption may occur in the receive buffers of MCC channels when more then one TDM slot uses 7 bits of contiguous data. **Workaround:** It is possible to avoid this problem by splitting the 7 bits slots between two SI ram entries such that one entry will represent 4 bits of the slot and the other SI entry will represent 3 bits of the slot. The errata occurs only when all the 7 bits are represented by one entry in the SI ram. **System Number:** 4743	0K40A 1K42A 2K42A
CPM43	**TxCLAV Ignored By UTOPIA in Single PHY Mode** **Date Added:** 5/30/2000 **Description:** When the FCC transmitter is configured to work in UTOPIA single PHY master mode, it ignores deassertion of the TxCLAV signal. Therefore, the UTOPIA slave cannot control the flow of cells by deasserting TxCLAV. Note that this bug affects Rev A.1 chips only. **Workaround:** Configure the FCC to work in multi-master mode but limit the number of PHYs to 1 by programming: FPSMR[LAST PHY] = 5'b0 **System Number:** 4882	0K40A 1K42A 2K42A
CPM44	**Zero Insertion Error on MCC** **Date Added:** 5/30/2000 **Description:** When the MCC transmitter is used in HDLC super channel mode, a zero insertion at the last bit before the flag fails to occur. **Workaround:** There is a microcode patch which fixes the problem. **System Number:** 4941	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM45	**Error in CLAV Sample Point** **Date Added:** 5/30/2000 **Description:** In FCC ATM transmit master mode (multiple PHY only), the CLAV signal is sampled 5 clocks before the end of the cell instead of 4 clocks. This is relevant only for back-to-back transmission sequence. **Workaround:** In multiple PHY fix priority polling mode, by adding a dummy PHY, it is possible to ensure that the dummy PHY is polled at payload 44 (5 clocks before the end of the cell). This is possible since the cell length is constant and the number of PHY to poll is also constant. **System Number:** 5031	0K40A 1K42A 2K42A
CPM46	**Error in Internal Prioritization of CPM Resource** **Date Added:** 5/30/2000 **Description:** Each of the communication controllers (FCC, MCC, SCC, ...) issues request for service to the CPM with different priorities in order to receive the necessary assistance in time. Because of an internal connection error, the FCC3 request for service is issued with a much lower priority than intended. Because of this, FCC3 may sporadically overrun when the CPM is heavily loaded. **Workaround:** None **System Number:** 5663	0K40A 1K42A 2K42A
CPM48	**Error in TDM** **Date Added:** 5/30/2000 **Description:** Disabling TDMx may interfere with the operation of TDMy if TDMy uses the SI-RAM blocks directly above those used by TDMx. For example: start address of TDMc = 0 start address of TDMb = 2 start address of TDMa = 4 start address of TDMd = 6 when disabling TDMa, TDMb is affected. when disabling TDMb, TDMc is affected. when disabling TDMd, TDMa is affected. when disabling TDMc, No TDM is affected. **Workaround:** Instead of disabling a TDM, the user can switch to a shadow RAM that contains only non supported slots in its entries. **System Number:** 5714	0K40A 1K42A 2K42A
CPM49	**Error in FEC CAM address recognition** **Date Added:** 3/14/2002 **Description:** External CAM address recognition in Fast Ethernet controller does not function. **Workaround:** Use microcode patch available from Freescale. **System Number:** 5404	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM50	**MCC Transparent Super Channel Loss of Alignment** **Date Added:** 5/30/2000 **Description:** When the MCC is configured to work in Transparent, super channel first sync slot synchronization, loss of alignment may occur when the first data (idles) on the Rx data line matches the value of the RCVSYNC parameter. **Workaround:** Write to RCVSYNC a pattern which cannot appear in the Rx data line. **System Number:** 5833	0K40A 1K42A 2K42A
CPM54	**Error in switching to and from shadow SI ram.** **Date Added:** 12/10/2000, modified 15/0/2002 **Description:** Dynamic switching in SIRAM may not be executed properly. **Workaround:** In SI RAM, when working with shadow RAM, the last entry (n) and the entry immediately before the last entry (n-1) MUST have at least one common bit in the CNT or BYT fields. For example: SIRAM Entry CNT BYT n-1 000 1 n 010 1 The above is okay n-1 101 0 n 001 0 The above is okay n-1 100 0 n 001 0 The above is not okay. **System Number:** 6282, 6283	0K40A 1K42A 2K42A
CPM55	**Error in ATM_Transmit command.** **Date Added:** 12/10/2000 **Description:** The ATM_Transmit command does not execute correctly when used on APC priority above 4. **Workaround:** None. **System Number:** 6162	0K40A 1K42A 2K42A
CPM57	**AAL5 Cell Corruption.** **Date Added:** 1/28/2001 **Description:** The second part of a second cell may overwrite the second part of the first cell in an AAL5 frame. **Workaround:** Use microcode patch.	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM64	**AAL5 RxBD[LNE] error generated if PDU length exceeds 65512 bytes** **Date Added:** 5/31/2001 **Description:** When the CPM receives an AAL5 PDU between 65512-65535 bytes (maximum length) the CPM sets the RxBD[LNE] indicating a receive length error, however the memory buffer contents for the PDU are correct. **Workaround:** Receive AAL5 PDU less than 65512 bytes or use microcode patch. **System Number:** 7025	0K40A 1K42A 2K42A
CPM65	**SS7 Microcode in ROM does not function correctly** **Date Added:** 8/5/2001 **Description:** The SS7 microcode in ROM does not function correctly and should not be used. **Workaround:** Use the Enhanced SS7 microcode package.	0K40A 1K42A 2K42A
CPM71	**CPM does not snoop MCC buffer descriptors.** **Date Added:** 8/5/2001 **Description:** When the MCC performs a DMA read or write of the buffer descriptor, GBL is not asserted and TC2 is always driven low.Therefore cache snooping will not be enabled for MCC BDs, therefore BDs in memory will not match the data cache. Also the bus used for the DMA is always the 60x, therefore if the BDs are on the local bus then the DMA consumes bandwidth on both the 60x and Local bus. **Workaround:** If GBL and/or TC2 are set in the MCC TSTATE/RSTATE parameters, use microcode patch available from Freescale which fixes the above problem. If GBL and TC2 are not set to improve performance move the MCC BDs to the 60x bus. The microcode patch will fix both the GBL/TC2 and the bus performance issue. **System Number:** 7018	0K40A 1K42A 2K42A
CPM72	**MCC Global Underruns** **Date Added:** 8/5/2001 **Description:** An MCC transmitter global underrun (GUN) error may result from intensive activity on FCC1 (e.g.burst of short back to back Ethernet frames).This is due to the prioritization of the MCC transmitter relative to FCC1 receiver and transmitter as well as the MCC receiver. **Workaround:** Each serial channel above can request a service at normal or emergency level. In case of emergency, the request is handled before normal (non-emergency) request of the channels at a higher priority level.The proposed change is to allow the MCC transmitter to assert an emergency request instead of normal request. The impact on FCC1 in this case is minimal.This feature will controlled by a new MCC mode bit in future MSC8101 revisions. This new MCC mode bit will allow users to continue to use the current CPM priority scheme in their applications if required.	0K40A 1K42A 2K42A
CPM73	**SI RAM Corruption** **Date Added:** 8/5/2001 **Description:** (7049)An access to the SI RAM bank from the 60x bus while the corresponding TDM is active may result in data corruption within the SI RAM. **Workaround:** Associate the SI RAM bank with an inactive TDM before attempting to access it. Once the accesses has been made, the SI RAM bank should be re-assigned to the active TDM. **System Number:** 7049	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM74	**FCC HDLC Controller Stops Transmitting When Using Nibble Mode With MFF=0** **Date Added:** 8/27/2001 **Description:** When running an FCC in HDLC nibble mode with the multi-frame per FIFO bit off (MFF=0) the CPM may lose synchronization with the FCC HDLC controller. As a result the HDLC controller will get stuck and stop transmission. **Workaround:** When running the FCC in HDLC nibble mode set the MFF=1 or alternatively run the FCC in HDLC bit mode.	0K40A 1K42A 2K42A
CPM76	**First transmitted bit zero in FCC Transparent Mode with GFMR[CTSS]=1** **Date Added:** 8/27/2001 **Description:** When using an FCC in transparent mode the first bit of a frame is transmitted as zero every time RTS is asserted before CTS is asserted when CTS is sampled synchronously with data (GFMR[CTSS]=1). If CTS is in pulse mode (GFMR[CTSP]=1) only the first frame is affected because CTS is ignored thereafter. If CTS is not in pulse mode (GFMR[CTSP]=0) then every frame is affected separately. **Workaround:** If the receiver synchronizes on a 8/16-bit sync pattern stored in the FDSR register (GFMR[SYNL]=1x) ensure that the synchronization pattern starts with a “0”. If no synchronization pattern is used (GRMR[SYNL]=0x) add a one-byte dummy buffer before sending the real data buffers.	0K40A 1K42A 2K42A
CPM79	**FCC Fast Ethernet Flow Control** **Date Added:** 3/14/2002 **Description:** When the FCC receives a flow control pause message with MAC parameter =0xffff, it sets a zero delay instead of maximum delay.	0K40A 1K42A 2K42A
CPM80	**MCC CES User Template** **Date Added:** 3/14/2002 **Description:** If the transparent MCC Tx CES channel requires the user template (CHAMR[UTM]=1) only the first 8 bytes of the user defined pattern are transmitted. Then the transmitter will continue to send bytes 4-7 of the pattern continuously until the counter reaches 0. Any bytes defined in the pattern after byte 7 are never transmitted. **Workaround:** Use a template size of 8 bytes.	0K40A 1K42A 2K42A
CPM85	**Only One BSY Interrupt Generated for AAL0** **Date Added:** 5/21/2002 **Description:** When using AAL0, only one BSY interrupt will be received regardless of the number of BSY events that are generated. **Workaround:** None.	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM86	**Random PHY Number For FCC RX in Single-PHY Master Mode** **Date Added:** 5/21/2002 **Description:** When the FCC Receive ATM controller is configured for Single PHY Master mode (FPSMR[RUMP]=0, FPSMR[RUMS]=0) and FPSMR [LAST PHY / PHY ID] is not equal to zero, a random PHY ID might be allocated to the incoming cells instead of the expected zero (for Single-PHY). This will result in a loss of cells. This configuration is typical when using the FCC Transmit ATM controller in Multi-PHY Master mode together with the FCC Receive ATM controller in Single PHY Master mode. **Workaround:** The Address Lookup Mechanism should be created in such a way that for any PHY Addr input, the Output will be as for PHY 0.	0K40A 1K42A 2K42A
CPM88	**MCC Transmit GUN when MCC STOP RX CPCR Command is used** **Date Added:** 10/15/2002 **Description:** An MCC may experience a highly intermittent transmit GUN event indication, related to MCC receive channels that have been stopped via the MCC STOP RX host CPCR command. This GUN can happen unrelated to internal CPM loading or other external factors. **Workaround:** Avoid using MCC STOP RX command using one of the following mechanisms: 1. Simply stop the TDM or 2. Use shadow RAM and dynamically remove the desired MCC RX channel entry from SIRAM programming (see 8260 User manual chapter 14). The following procedure should be utilized, using an extra redundant shadow RAM switch. This is done to provide a full TDM frame's amount of time to ensure receive activity is complete and will avoid the issue: a. Re-program shadow SIRAM to remove channel to be stopped. b. Switch to shadow SIRAM and wait for that TDM's bit in the SIxSTR register to change to indicate switch complete. c. Copy this new shadow RAM programming back to the main SIRAM bank. d. Switch to active RAM, again wait for switch to complete. e. Then software can re-initialize or modify the removed channel's RX parameters. **System Number:** 2905 **Fix Plan:** RevA	0K40A 1K42A 2K42A
CPM95	**ATM False Indication of Mis-Inserted Cells** **Date Added:** 2/25/2003 **Description:** There is a false indication of unassigned bits in the PHY:VPI:VCI which could cause ATM cells to be treated as mis-inserted cells and therefore be discarded. **Workaround:** Use microcode patch available from Freescale. **Fix Plan:** RevA	0K40A 1K42A 2K42A

Table 48. Errata Resolved in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM100	**ABR TCTE Address Miscalculation** **Description:** When using the AAL5 ABR ROM microcode with external ATM channels it is possible for the EXT_TCTE_BASE word value (written by the user to DPRAM) to be misread. In this case calculations performed by the microcode to access the users programmed external TCTE will be incorrect with a high chance of the access resulting in a CPM crash. **Workaround:** Use the micro code patch available from Freescale. **System Number:** 9131 **Fix Plan:** Rev. A	0K42A 1K42A 2K42A
CPM110	**FCC1 Prioritization** **Date Added:** 12/19/2003 **Description:** The FCC1 receiver in Ethernet, HDLC, or Transparent controller mode is not elevated to emergency status (priority 4 in Table 19-2 of the Reference Manual, "Peripheral Prioritization"), which may lead to a FIFO overrun if the system is heavily loaded (FCC1 receiver has the highest priority excluding emergency status of other peripherals). **Workaround:** When allocating FCCs, assign FCC2 and FCC3 for Ethernet, HDLC or Transparent before FCC1, or assign FCC1 to the lowest bit rate interface. If FCC1 is allocated for ATM and requires higher CPM usage than the other FCCs, disable its emergency status. **System Number:** 11062 **Fix Plan:**	0K40A 1K42A 2K42A

Table 49. Errata Resolved by Specification Change in Mask Set 2K87M

Errata Number	Errata Description	Applies to Mask
QSIU8	**Core TEA Not Supported as NMI** **Date Added:** 9/7/2000 **Description:** Q2PPC TEA is not supported on PIC NMI5. **Workaround:** Use PIC $\overline{\text{IRQ19}}$. **Resolution:** Specification is changed to reflect this state in mask set 2K87M.	0K40A 1K42A 2K42A 2K87M
SIU15	**Data Not Written to the SDRAM in RMW Parity Mode** **Date Added:** 5/21/2002 **Description:** When using a read-modify-write parity mode and pipelined addresses on the SDRAM interface, the write portion of the RMW might be performed as a read. As a result, the data is not written to the external memory. **Workaround:** Use BCR[PLDP]=1 for a pipeline depth of zero. **Fix Plan:** None, specification is changed so that this is defined as normal functionality and becomes a documentation errata.	2K87M

Table 49. Errata Resolved by Specification Change in Mask Set 2K87M (Continued)

Errata Number	Errata Description	Applies to Mask
CPM37	**Requirement for Software to Disable FCC After Error** **Date Added:** 5/30/2000 **Description:** There are four errors in the FCC transmitter that require software to disable and enable the transmitter before it can continue to operate correctly. The four errors are: **1.** CTS-LOST indication in the HDLC transmitter **2.** Late collision in the fast ethernet transmitter **3.** Underrun in any of the FCC transmitter protocols **4.** Expiration of RL in fast ethernet **Workaround:** None available. **System Number:** 4040, 3980, 2314 **Fix Plan:** Specification was changed to include details for error handling software.	0K40A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
SIU13	**SDAMUX not Valid in Single-MSC8103 Mode** **Date Added:** 3/14/2002 **Description:** SDAMUX signal is disabled (stuck at '0') when SDRAM machine handles the memory access and the chip is programmed to single-MSC8103 mode (BCR[EBM]=0). **Workaround:** None.	0K40A 1K42A 2K42A 2K87M
SIU16	**Bus Busy Disable Mode Can Hang 60X Bus in Multi-Master Systems** **Date Added:** 5/21/2002 **Description:** The bus busy disable mode (SIUMCR[BBD=1]) can not be used if the MSC8103 is not the only master on the 60x-compatible bus. Using this mode in such a system can cause the 60x-compatible bus to hang. **Workaround:** **1.** If the external master supports the $\overline{\text{ABB}}$ signal, do not use the bus busy disable mode and connect this signal to the MSC8103. The $\overline{\text{DBB}}$ signal can either be connected or can be pulled up. **2.** If the external master does not support the $\overline{\text{ABB}}$ signal do one of the following: a. Do not use the bus busy disable mode and generate the $\overline{\text{ABB}}$ signal externally. The $\overline{\text{DBB}}$ signal can either be connected or can be pulled up. The following external $\overline{\text{ABB}}$ implementation should be sufficient to work around the problem: Assert the $\overline{\text{ABB}}$ signal whenever a qualified bus grant for the external master is sampled (Bus grant asserted while $\overline{\text{ARTRY}}$ and $\overline{\text{ABB}}$ are deasserted). Deassert the $\overline{\text{ABB}}$ signal when there is no qualified bus grant. The deassertion of $\overline{\text{ABB}}$ should be as follows: Drive $\overline{\text{ABB}}$ to V_{DD} for half a clock cycle and then stop driving it (high impedance). b. If using the internal arbiter and up to two external masters, connect the external bus grants (through an AND gate if more than one) to an available external bus request and define the priority for that request to be the highest in the PPC_ALRH register. The $\overline{\text{DBB}}$ signal can either be connected or can be pulled up. **Fix Plan:** TBD	2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
SIU18	**ARTRY Assertion When Using Pipeline Depth of Zero** **Date Added:** 10/15/2002 **Description:** Internal (60x) slave maintains a pipeline depth of zero by asserting AACK only after TA. When ARTRY is asserted the 60x bus access will be terminated and TA will not be asserted. Therefore the Internal (60x) slave will not assert AACK since TA was not asserted. **Workaround:** Use a pipeline depth of one (BCR[PLDP]=0) for applications that require memory coherency. **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
SIU19	**Bus Monitor Timeout When Using External Slave** **Date Added:** 10/15/2002 **Description:** When using an external 60x bus slave with the bus monitor activated, PSDVAL is not asserted when the external slave is accessed, which could cause the bus monitor to time-out and TEA to be asserted. **Workaround:** The following workarounds 1. Use pipeline depth of zero (BCR[PLDP]=1) when using an external 60x bus slave. 2. Disable 60X bus monitor, SYPCR[PBME]=0. 3. If the external 60x bus slave is another 810x or 826x device, connect the PSDVAL signals together. **Fix Plan:** TDB	0K40A 1K42A 2K42A 2K87M
QSIU4	**Extended Mode on Local Bus** **Date Added:** 6/13/2000 **Description:** Using Extended mode on the local bus can generate incorrect transactions in certain combinations of consecutive reads and writes. **Workaround:** Do not use Extended mode on the local bus. **System Number:** 5959 **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
EFC1	**Inaccurate EFCOP IIR Outputs for Two or Fewer Coefficients** **Date Added:** 8/15/2000 **Description:** When using normal (dual) DMA or flyby DMA transfers which have a maximum transfer size greater than 32 bits with the EFCOP to perform IIR filtering with two or less IIR coefficients, the first output of the IIR filter is lost. The rest of the outputs are shifted and inaccurate. **Workaround:** Use only DMA 32-bit maximum transfer size for both input and output channels. **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
GEN3	**Device Withstands ESD CDM Stress of 400 V Instead of 500 V** **Date Added:** 10/31/2002 **Description:** Device meets the ESD specifications for Human Body Model (HBM) of 1000 V and Machine Model (MM) of 100 V but does not withstand the Charged Device Model (CDM) of 500 V. All pins are guaranteed to withstand CDM of 400 V. **Workaround:** NA **Fix Plan:** TBD	2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
SC4	**SC140 Core May Hang after Write to the PCTL0 Register** **Date Added:** 2/19/2002 **Description:** Write to the PCTL0 freezes the core immediately for 150–190 cycles. If the system is busy (for example, doing pre-fetch transactions), the core may not exit the freeze state. **Workaround:** Option A: 1. Ensure the EFCOP is not active. 2. Ensure the local bus to L1 memory is not active. 3. Ensure that the program that writes to PCTL0 is in internal memory. 4. Write to PCTL0 immediately after reset before any external accesses. Option B: Do not write to PCTL0. **System Number:** 7560 **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
SC5	**SC140 Core May Hang after Illegal Execution Set** **Date Added:** 2/19/2002 **Description:** Upon receipt of an illegal execution set, the SC140 core may enter a freeze state that can only be released by reset. **Workaround:** None available. **System Number:** 7541 **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
SC6	**Incorrect Data on Trace Buffer During Core Freeze** **Date Added:** 2/19/2002 **Description:** After writing data to the Trace Buffer (TB), the TB is disabled in order to read from it. There are two options to read the TB and the problem occurs in both: 1. Reading the TB by software. If a core freeze occurs while the software reads the TB into a core register, data can overwrite the previous data. 2. Reading the TB from JTAG. If a core freeze occurs while the JTAG is reading the TB, the data is not correctly sampled. **Workaround:** 1. Software: Read the TB by software when there is no core freeze: a. Ensure that the program is in internal memory. b. Ensure that the EFCOP is not active. c. Ensure that the local bus to L1 memory is not active. d. Ensure that the Write Buffer is empty. e. Ensure that there are no other MOVE commands except for the TB read. 2. JTAG: a. Read the TB from JTAG only when the core is in Debug mode. b. Before reading the TB, flush the Write Buffer. **System Number:** 7604 **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
SC7	**Change of Flow May Cause Incorrect Trace-Buffer Data** **Date Added:** 2/19/2002 **Description:** When the EOnCE module is programmed for tracing events of change of flow (TCHOF) and interrupts (TINT), the Trace Buffer is updated on every event by the source and destination addresses. In the event of a change of flow (CHOF) to another CHOF with an interrupt request in between, the Trace Buffer is updated with additional data. The additional data is incorrect and is not needed for the trace. **Workaround:** Perform post-processing after reading the Trace Buffer. Search in the data for a source address with its destination listed before it. Delete the source and the previous data. **System Number:** 7794 **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
SC8	**Debug Exception Request From JTAG is Not Accepted During Core Freeze** **Date Added:** 5/21/2002 **Description:** JTAG debug exception request is not accepted by the core during freeze. If the request is asserted and deasserted during a core freeze, the request is discarded. **Workaround:** Assert Debug request from JTAG. When entering the exception routine, use software to assert an external pin (one of the EE pins, for example) to signal that the Exception Service Routine was executed. After that, a new JTAG instruction can be written. **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
SC9	**EE Pins Do Not Enable Different EOnCE Modules During Core Freeze** **Date Added:** 5/21/2002 **Description:** If EE pins are asserted to enable events in the EOnCE modules during a core freeze and the request is deasserted during the same core freeze, the event is not enabled. **Workaround:** Poll core status from JTAG. After the core is not in a freeze state, assert the EE pin(s) for at least three cycles. **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M
CPM94	**FCC $\overline{\text{RTS}}$ Signal Not Asserted Correctly** **Date Added:** 2/25/2003 **Description:** At the beginning of an HDLC frame transmission which is preceded by more than one opening flag, $\overline{\text{RTS}}$ will not be asserted if $\overline{\text{CTS}}$ is negated. This may cause a deadlock if the modem waits for the assertion of $\overline{\text{RTS}}$ before asserting $\overline{\text{CTS}}$. **Workaround:** Implement one of the following: 1. Transmit no flags between or before frames. 2. Clear FPSMR[NOF] bit. Set GFMR[RTSM]=1 to ensure RTS/ is asserted when FCC is enabled. However no hand shaking activities with the modem will occur for all the proceeding frames. **Fix Plan:** TBD	2K87M
CPM96	**ATM Performance Monitoring with AAL1 CES** **Date Added:** 2/25/2003 **Description:** ATM Performance Monitoring with AAL1 CES Data in DPRAM is corrupted when performance monitoring is enabled in the receiver. **Workaround:** Impelement one of the following: 1. Disable Receive Performance Monitoring RCT[PMT]=0. 2. Use microcode patch available from Freescale. **Fix Plan:** TBD	0K40A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
CPM97	**MCC SS7 - No SUERM interrupt generated after an ABORT** **Date Added:** 2/25/2003 **Description:** Octet Count Mode is not entered properly when idles are received after an ABORT. Therefore N_Cnt is not decremented and no SUERM interrupt will be generated. This problem only affects the SS7 micro code in ITU-T / ANSI mode (SS7_OPT[STD]=0). **Workaround:** Use the latest RAM based SS7 micro code package available from Freescale. **Fix Plan:** TBD	2K87M
CPM98	**I^2C Erratic behavior can occur if extra clock pulse is detected on SCL** **Date Added:** 8/25/2003 **Description:** The I^2C controller has an internal counter that counts the number of bits sent. This counter is reset when the I^2C controller detects a START condition. When an extra SCL clock pulse is inserted in between transactions (before START and after STOP conditions), the internal counter may not get reset correctly. This could generate partial frames (less than 8 bits) in the next transaction. **Workaround:** Do not generate extra SCL pulses on the I^2C bus. In a noisy environment the digital filter I2MOD[FLT] and additional filtering capacitors should be used on SCL to eliminate clock spikes that may be misinterpreted as clock pulses. **System Number:** 9133 **Fix Plan:** TBD	0K42A 1K42A 2K42A 2K87M
CPM99	**ABR TCTE[ER-TA] Corruption** **Date Added:** 8/25/2003 **Description:** When using the AAL5 ABR ROM microcode it is possible for the TCTE[ER-TA] field to be overwritten with an erroneous value. This, in turn, will cause the TCTE[ER-BRM] to be updated with this value. As TCTE[ER-BRM] holds the maximum explicit rate value allowed for B-RM cells an erroneous value in this field could have a detrimental effect on the network performance. **Workaround:** Use the micro code patch available from Freescale. **System Number:** 9132 **Fix Plan:** TBD	0K42A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

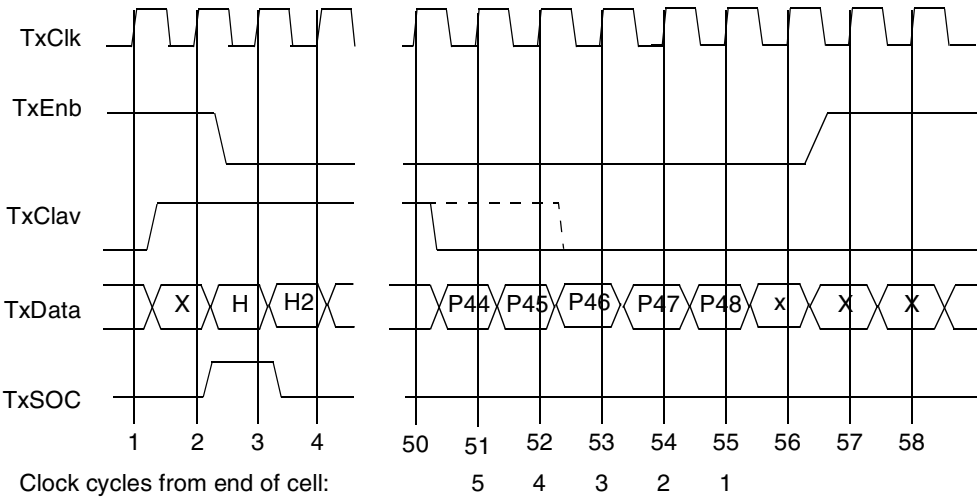
Errata Number	Errata Description	Applies to Mask
CPM101	**FCC RxClav Timing Violation (Slave)** **Date Added:** 1/15/2004 **Date Revised:** 11/05/2004 **Description:** FCC ATM Receive UTOPIA slave mode. When the RxFIFO is full, RxClav is negated 2 cycles before the end of the cell transfer, instead of 4. A master that polls RxClav or pauses 3 or 4 cycles before the end of the cell transfer may sample a false RxClav, and an overrun condition may occur. The dashed line in the timing diagram below depicts the actual RxClav negation (two cycles before the end of the cell transfer instead of four cycles). The signals in the timing diagram are with respect to the master, so the Tx interface is shown. **Workaround:** 1. The master should not poll RxClav or pause a cell transfer 4 cycles before the end of a cell transfer. The master should poll 2 cycles before the end of the current cell or later. This can be achieved by introducing cell-to-cell polling (and transfer) delay, which is equal or larger than one cell transfer time. If this can be achieved, the impact on performance is minimal. 2. Configuring ATM only on FCC1 and setting FPSMR[TPRI] ensures the highest priority to FCC1 Rx. In addition, for CPM usage lower then 80 percent (as reported by the CPM performance tool based on UTOPIA maximal bus rate), the CPM performance is enough to guarantee that the RxFIFO does not fill up. 	0K40A 1K42A 2K42A 1K87M
CPM111	**FCC Missing Reset** **Date Added:** 1/15/2004 **Description:** The TxBD may not close for the FCC in Half-Duplex 10BaseT Ethernet. There may be a mismatch between the actual transmitted BD and the BD for which the status is updated. As a result, the status of one to three BDs may not be updated. They appear to be “ready” although the associated frames have been transmitted (assuming a frame per BD). **Workaround:** Use microcode patch provided by Freescale. **System Number:** 11064 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
CPM112	**FCC Missing Reset at OverRun** **Date Added:** 12/19/2003 **Description:** TxBD may not be closed for FCC in Half-duplex 10BaseT Ethernet. There may be a mismatch between the actual transmitted BD and the BD for which status is updated. As a result, the status of one to three BDs may not be updated, and they would appear "Ready", although the associated frames have been transmitted (assuming a frame per BD). **Workaround:** Use microcode patch provided by Freescale. **System Number:** 11064, 11067 **Fix Plan:** N/A	0K40A 1K42A 2K42A 2K87M
CPM113	**Incorrect Return Value from Event Register Read (SCC, SPI, I^2C, and SMC)** **Date Added:** 12/19/2003 **Description:** When the Event Register is read while the SCC, SPI, I^2C, or SMC is active, it is sometimes read as 0, even though it has some bits set. **Workaround:** **System Number:** 11068 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M
CPM115	**APC Transmits Unwanted Idle Cells** **Date Added:** 12/19/2003 **Description:** In heavily loaded ATM applications, if the ATM pace controller (APC) is configured for multiple priority levels and a burst of traffic for transmission is sustained long enough on the highest priority APC table, then an unwanted idle cell can be trasmitted on the lower priority APC tables when there are cells available in lower priority APC scheduling table for transmission. The transmission of the unwanted idles could cause the valid ATM cells on lower-priority APC scheduling tables not to be transmitted. This transmission of unwanted idles can affect all ATM channels that are not located in the highest-priority APC scheduling table. **Workaround:** Increase the size of lower-priority APC scheduling tables so they are large enough to absorb any burst or back-to-back bursts on the highest-priority APC scheduling table. Otherwise, use the microcode patch available from Freescale. **System Number:** 11069 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M
CPM116	**Pointer 93 in Partially Filled (PFM) Mode** **Date Added:** 1/15/2004 **Description:** In PFM mode, the pointer value of 93 is not generated, causing the loss of synchronization at the far end. Also, when the pointer value of 93 is received, the synchronization is lost, which causes a loss of data and the resynchronization routine. **Workaround:** Use microcode patch provided by Freescale. **System Number:** 11912 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
CPM117	**False Address Compression** **Date Added:** 11/05/2004 **Description:** If there are active AAL0 channels and a CRC-10 error has been received, VP-level address compression might have false results, which could lead to one of the following: • Wrong calculation of a VP pointer address • Cells might be falsely discarded as misinserted cells • Misidentification of misinserted cells (in CUAB mode) This is a statistical error, which is conditional on the reception of AAL0 cells with a CRC-10 error. The probability of false address compression is directly correlated with higher CPM bit rate and longer system bus latency. While the false address compression is possible only if there are active AAL0 channels, it may have an impact on all AAL types. However, it cannot occur unless AAL0 cells with CRC-10 error have been received beforehand. **Workaround:** Use the microcode patch supplied by Freescale. **System Number:** 17129	0K40A 1K42A 2K42A 1K87M
CPM118	**Aborted HDLC Frame Followed by a Good Frame** **Date Added:** 7/11/2004 **Description:** When an aborted HDLC frame is followed by a good frame, the receive data buffer may contain the data of the aborted frame followed by the data of the good frame. **Workaround:** Use the microcode patch provided by Freescale. **System Number:** 15905 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M
CPM119	**Ethernet Collision Occurs on the Line 125 Clocks after TX_EN Assertion** **Date Added:** 7/11/2004 **Description:** When an ethernet collision occurs on the line 125 clocks after TX_EN assertion, late collision will be reported even though this is only 63 bytes into the frame instead of 64. When a collision occurs 124 cycles after TX_EN assertion, no event is reported, the TxBD is not closed, and transmission halts. Retransmission behavior is correct for collisions occurring between assertion of TX_EN and 123 clocks. **Workaround:** Use the microcode patch provided by Freescale. **System Number:** 15907 **Fix Plan:**	0K40A 1K42A 2K42A 2K87M
CPM120	**SS7_OPT[FISU_PAD] parameter has no effect on the number of flags between FISUs** **Date Added:** 12/22/2004 **Description:** The SS7_OPT[FISU_PAD] parameter has no effect on the number of flags between FISUs. Regardless of the value of this field, one flag will be present between back-to-back FISUs. **Workaround:** Use the latest SS7 microcode package provided by Freescale. **System Number:** 18767 **Fix Plan:** None at this time.	2K87M

Table 50. 2K87M Errata

Errata Number	Errata Description	Applies to Mask
CPM121	**TDM Data Frame Corruption** **Date Added:** 11/05/2004 **Description:** During a write to one of the SI registers (GMR, AMR, BMR, CMR, DMR) while one or more TDMs are working, one data frame of a working TDM may become corrupted. **Workaround:** Work with the shadow RAM when changing data and do not disable and then enable the TDM. **System Number:** 17460	0K40A 1K42A 2K42A 1K87M

A Bootloader Program

This appendix lists the boot program for mask set 2K87M of the MSC8103.

Note: The first instruction in the boot code sets the stack address to 0x68000. In the event of a hard or soft reset, user data at this location is overwritten by the boot code. In addition, source code should not be bootloaded to this address due to corruption of the stack during the bootload program execution.

```
;/*************************************************************/
;/*                                                           */
;/* File : boot_code_revA.asm                                 */
;/*                                                           */
;/* (C) Copyright Freescale Inc, 2002.                         */
;/* All rights reserved                                       */
;/*                                                           */
;/*                                                           */
;/* Description:                                              */
;/* This file contains the MSC8103 RevA boot code as          */
;/* specified in the boot section of the Reference Manual.    */
;/*                                                           */
;/*                                                           */
;/* Modifications from Rev0:                                  */
;/*   - Host checksum fixed (Erratum Boot1)                   */
;/*   - SRAM base address is ISB dependent (Erratum Boot2)    */
;/*   - Software watchdog handled in host code instead of     */
;/*       disabled.                                           */
;/*   - Added I2C serial boot                                 */
;/*                                                           */
;/*************************************************************/

STACK_ADDR equ $68000
BOOT_BYPASS_ADD equ $0


; BANKS
MASK0_A equ     $00f0ff00
BASE0_A equ     $00f0ff02
BASE0_D equ     $00f00000
BASE1_A equ     $00f0ff06

BASE_ROM_ADDRESS equ $00f80000
BASE_EXEPTION_TABLE equ $00f80000



EXTERNAL_MEM_BOOT_TABLE equ $fe000110
; BOOT_REV_REG
BOOT_REV_REG equ BASE_ROM_ADDRESS+$ffa0



; Host Interface Registers
; Dsp Side
HCR          equ    BASE0_D+$0000
HPCR         equ    BASE0_D+$0020
```

```
HSR           equ    BASE0_D+$0040
HCVR          equ    BASE0_D+$0060
HOTX          equ    BASE0_D+$0080
HORX          equ    BASE0_D+$00a0
ELIRE         equ    BASE0_D+$1c20
ELIRF         equ    BASE0_D+$1c28


;the registers to be used are :
;      d4,d5 for reading from the host fifo
;      r3    for holding the block address
;      d6    for holding the size
;      d7    for holding the checksum
;      d2    for holding the ~checksum
;      d1,r6    for holding the IMMR
;      r5  for holding the SRAM BASE MEM
;      d12,r15 for sw watchdog handling


;      During the loading proces the checksum is calculated for the whole
;      long and at the end of the block the two words in the calculated
;      checksum are XORed to generate the real checksum.

;---------- NMI0 (HDI16) offset 0xe00 -----
       section nmi0
       org             p:$0e00+BASE_EXEPTION_TABLE

nmi0
       rte

       endsec

;---------- NMI1 (TEA) offset 0xe40 -----
       section nmi1
       org p:$0e40+BASE_EXEPTION_TABLE

nmi1
       rte

       endsec
;---------- NMI2 (bus controller , memory write errors) offset 0xe80 -----
       section nmi2
       org p:$0e80+BASE_EXEPTION_TABLE

nmi2
       rte

       endsec
;---------- NMI3 (bus controller non aligned error) offset 0xec0 -----
       section nmi3
       org p:$0ec0+BASE_EXEPTION_TABLE

nmi3
       rte
```

```
        endsec
;---------- NMI4 (bus controller bus error) offset 0xf00 -----
        section nmi4
        org p:$0f00+BASE_EXEPTION_TABLE

nmi4
        rte

        endsec
;---------- NMI5(reserved) offset 0xf40 -----
        section nmi5
        org p:$0f40+BASE_EXEPTION_TABLE

nmi5
        rte

        endsec
;---------- NMI6 (reserved) offset 0xf80 -----
        section nmi6
        org p:$0f80+BASE_EXEPTION_TABLE

nmi6
        rte

        endsec
;---------- NMI7 (sic nmi,s/w wd,external nmi, parity) offset 0xfc0 -----
        section nmi7
        org p:$0fc0+BASE_EXEPTION_TABLE

nmi7
        rte

        endsec




;---------- illegal exeption offset 0x80 ----------
        section illegal_exeption
        org p:$0080+BASE_EXEPTION_TABLE

illegal_exeption
        rte

        endsec

;---------- debug exeption offset 0xc0 ----------
        section debug_exeption
        org p:$00c0+BASE_EXEPTION_TABLE

debug_exeption
        rte

        endsec
```

```
;---------- overflow exeption offset 0x100 ----------
       section overflow_exeption
       org p:$0100+BASE_EXEPTION_TABLE

overflow_exeption
       rte

       endsec

;---------- auto nmi exeption offset 0x180 ----------
       section auto_nmi_exeption
       org p:$0180+BASE_EXEPTION_TABLE

auto_nmi_exeption
       rte

       endsec

;---------- auto ir exeption offset 0x1c0 ----------
       section auto_ir_exeption
       org p:$01c0+BASE_EXEPTION_TABLE

auto_ir_exeption
       rte

       endsec


;---------- in address 0 goto 0x1000 -------------
       section start
       org p:$0000+BASE_ROM_ADDRESS
start
       ; exeption stack pointer initialization
       move.l #BASE_EXEPTION_TABLE,vba;  init vba
       move.l #STACK_ADDR,r0
       nop
       tfra r0,sp   ; init ESP

       ; stack initialization
       move.l #0,d3
       move.l d3,(r0)


       jmp $1000+BASE_ROM_ADDRESS

       endsec

;-----------------------

       section boot
       org          p:$1000+BASE_ROM_ADDRESS

Fmain
       move.l emr,d1
```

```
        extractu #2,#17,d1,d3 ;get EE4,EE5 -> d3
        nop
        cmpeq.w #$3,d3
        jt to_boot_bypass



        move.l emr,d1
        extractu #3,#19,d1,d3
               ; d3 xor 3'b100 to recover original isb
        eor #$4,d3.l
        nop
        move.l d3,d13; Added for serial BOOT
        jmp find_siubase
return_find_siu
        ; SRAM INIT
        jmp sram_init
return_sram_init
        ; initialize ELIRE
        move.w #$8000,d0
        nop
        move.w d0,ELIRE
        ; initialize ELIRF
        move.w #$0008,d0
        nop
        move.w d0,ELIRF
        ;check the scmr to see which upm routine to load
        move.l (r6+$c88),d1     ;scmr,d1
        ; extract the busdf from scmr
        extractu #4,#20,d1,d3
        jmp upmc_init

check_boot
        move.l emr,d1
        ;get EE4,EE5 -> d3
        extractu #2,#17,d1,d3
        nop
        cmpeq.w #$0,d3
        jt external_memory
        cmpeq.w #$1,d3
        jt from_host
        cmpeq.w #$2,d3;added for serial boot
        jt from_serial
        cmpeq.w #$3,d3;added for serial boot
        jt to_boot_bypass

        stop

external_memory

        ;d3 <- isb[0,1,2]
        extractu #3,#19,d1,d3
        ; d3 xor 3'b100 to recover original isb
        eor #$4,d3.l
        ;r3 <- d3
```

```
        move.l d3,r3
        nop
        ; FIND IMMR FROM ISB
        ;r3 *= 4 so the offset will be in long instead of bytes
        ;r3 = r3<<2
        asl2a r3
        ;put the address of the external memory boot table in r4
        move.l #EXTERNAL_MEM_BOOT_TABLE,r4
        nop
        adda r4,r3
        nop
        ;move the address from the table into r3
        move.l (r3),r3
        nop
        ;jump to that address
        jmp r3


to_boot_bypass
        jmp BOOT_BYPASS_ADD
        nop

from_host
        ; software watch dog is not disabled
        ; it will be handled in the load_from_fifo routine if it is enabled
        move.l #$0,d15;clear d15, software watchdog is not enabled
        move.l (r6+$4),d1;get sypcr value
        bmtsts #$0004,d1.l
        jf HDI_en
        nop
        move.l d15,r15 ; clear watchdog handle counter
        move.l #$1,d15; indicate software watchdog enabled
        move.l #$0100,r15; initialize watchdog handle counter


        ;set the HEN bit in hpcr
HDI_en
        move.w HPCR,d3
               or #$0080,d3.l
        move.w d3,HPCR


        ;get the 8bit bit from hpcr ( which is in d3 )
        ;if (8bit ) goto load_8bit else goto load_16bit
        bmtsts #$0040,d3.l
        jt load_8bit
        jmp load_16bit

load_last_long_2_16
        ;if the size left to load is only 2 words ( 4 bytes)
        ;and the mode is 16-bit so every load action is of 4 words
        ;in this case the load action already loaded the checksum
        ;and the ~checksum, so it is a special case which needs handling
        ;by itself
        jsr load_from_fifo
```

```
        ;move the last 2 words to their address (which is kept in r3)
        move.l d4,(r3)
        adda #$4,r3
        ;write checksum , ~checksum at the and of the block
        move.l d5,(r3)
        ;calculate the checksum of these words
        eor d4,d7

        ;calculate ~checksum -> d2

        ;get the real checksum from d7 and
        ;get it into d7.l as explained in the beggining of the file
        ;d2 = (0xffff0000 & d7)>>16
        extractu #16,#16,d7,d2
        ;d2 = d2 & 0x0000ffff
        and #$0000ffff,d2,d2
        ;d7 = d7 & 0x0000ffff
        and #$0000ffff,d7,d7
        ;d7 = d7 ^ d2 = checksum
        eor d2,d7
        ;d2 = (~d7 & 0x0000ffff) = ~checksum
        not d7,d2
        and #$0000ffff,d2,d2


        ;get loaded checksum , ~checksum from d5
        ;get checksum into d4
        extractu #16,#16,d5,d4
        and #$0000ffff,d4,d4
        ;delete the checksum from d5
        and #$0000ffff,d5,d5
        ;if ( ~checksum_loaded |= ~Checksum_calculated ) goto set sticky bit
        cmpeq d5,d2
        nop
        iff jsr set_sticky_bit
        ;if ( checksum_loaded |= Checksum_calculated ) goto set sticky bit
        cmpeq d4,d7
        nop
        iff jsr set_sticky_bit
        ;goto loading the next block
        ;goto load_16bit

load_16bit

        ;d7 = checksm =0 , clear the calculated checksum register

        move.l #0,d7

              ;load the size and address
        jsr load_from_fifo
        ;move the size into d6
        move.l d4,d6
        ;move the address into r3
        move.l d5,r3
        ; should be calculated on data and address also
```

```
        eor d4,d7
        eor d5,d7


        ; check if the finished (HF4) bit is set ,if it is set clear it
        ; and the sticky bit .( it means there was an error and
        ; the blocks are being loaded for the second time
        move.w HCR,d4
        bmtsts.w #$8000,d4.l
        jf continue_loading_16
        move.w HCR,d4
        and #$6fff,d4.l
        move.w d4,HCR


continue_loading_16
        ;if (size == 0 ) in the beggining of a block it means no more blocks
        ;then jump to the address loaded in r3 (from d5 )
        tsteq  d6
        jt end_of_loading_16

load_loop_16
        ;if (size ==2 ) it is a special case so goto load_last_long_2_16
        move.l #$00000002,d4
        cmpeq d4,d6
        jt   load_last_long_2_16


        ;load 2 data words
        jsr load_from_fifo
        ;load the first 2 data words (4 bytes) to the address
        move.l d4,(r3)
        ;increment the address by 4 bytes
        adda #$4,r3
        ;load the second 2 data words (4 bytes) to the address
        move.l d5,(r3)
        ;increment the address by 4 bytes
        adda #$4,r3

        ;CALCULATE_CHECKSUM
        eor d4,d7
        eor d5,d7
        ;decrease the size by 4 words
        sub #$4,d6
        ;jump to loading the next words
        jmp load_loop_16



load_8bit
        move.l #0,d7 ; un-initialized dalu register.
        ;load the size into d6
        jsr load_from_fifo
```

```
        ; check if the finished bit (HF4)is set ,if it is set clear it
        ; and the sticky bit .( it means there was an error and
        ; the blocks are being loaded for the second time
        move.w HCR,d6
        bmtsts.w #$8000,d6.1
        jf continue_loading_8
        move.w HCR,d6
        and #$6fff,d6.1
        move.w d6,HCR

continue_loading_8
        eor d5,d7   ;checksum is calculated  on size

        move.l d5,d6
        ;if (size == 0) it means the last block was loaded
        ;so goto end_of_loading
        tsteq  d6
        jt     end_of_loading
        ;load the address
        jsr load_from_fifo
        move.l d5,r3
        eor d5,d7 ;checksum is calculated on address
load_loop_8
        ;load data word
        jsr load_from_fifo
        move.l d5,(r3)
        ;add 4 bytes to the address
        adda #$4,r3

        ;CALCULATE_CHECKSUM d7 = d7 ^ d5
        eor d5,d7

        ;subtract 2 words from the size
        sub #$2,d6
        ;if ( size | = 0) go to load_loop_8
        tsteq  d6
        jf     load_loop_8

        ;get the checksum into d7.1
        ;d2 = (0xffff0000 & d7)>>16
        extractu #16,#16,d7,d2
        ;d2 = d2 & 0x0000ffff
        and #$0000ffff,d2,d2
        ;d7 = d7 & 0x0000ffff
        and #$0000ffff,d7,d7
        ;d7 = d7 ^ d2
        eor d2,d7
        ;d2 = (~d7 & 0x0000ffff) = ~checksum
        not d7,d2
        and #$0000ffff,d2,d2
        ;load the checksum ,~checksum
        jsr load_from_fifo
        ;get ~checksum into d4
        extractu #16,#16,d5,d4
        ;delete the ~checksum from d4
```

```
        and #$0000ffff,d5,d5
        ;if ( checksum_loaded |= Checksum_calculated ) goto set sticky bit
        cmpeq d5,d2
        nop
        iff jsr set_sticky_bit
        ;if ( ~checksum_loaded |= ~Checksum_calculated ) goto set sticky bit
        ;clean d4.h
        and #$0000ffff,d4,d4
        cmpeq d4,d7   ; d4,d7 contain checksum
        nop
        iff jsr set_sticky_bit
        ;goto load the next block
        jmp load_8bit


end_of_loading_16
        jsr load_from_fifo

        ;get the checksum into d7.l
        ;d2 = (0xffff0000 & d7)>>16
        extractu #16,#16,d7,d2
        ;d2 = d2 & 0x0000ffff
        and #$0000ffff,d2,d2
        ;d7 = d7 & 0x0000ffff
        and #$0000ffff,d7,d7
        ;d7 = d7 ^ d2
        eor d2,d7
        ;d2 = (~d7 & 0x0000ffff) = ~checksum
        not d7,d2
        and #$0000ffff,d2,d2

        ;get checksum into d4
        extractu #16,#16,d5,d4
        ;delete the checksum from d5
        and #$0000ffff,d5,d5
        ;if ( ~checksum_loaded |= ~Checksum_calculated ) goto set sticky bit
        cmpeq d5,d2   ;d5 and d2 contain ~checksum
        nop
        iff jsr set_sticky_bit
        ;clean d4.h
        and #$0000ffff,d4,d4
        ;if ( checksum_loaded |= Checksum_calculated ) goto set sticky bit
        cmpeq d4,d7
        nop
        iff jsr set_sticky_bit


        ;set HF4 bit in HCR to show that loading is finished
        move.w HCR,d6
        or #$00008000,d6.l
        move.w d6,HCR

        ;check if the checksum bit is set ( HF3 in HSR (HSR[3] )
        move.w HSR,d6
        and #$00001000,d6,d6
```

```
        ;check if the sticky bit is set ( HF7 in HCR (HCR[3]) )
        move.w HCR,d4
        and #$00001000,d4,d4
        ;if both of the flags are set start the loading again
        and d4,d6
        tsteq d6
        jf from_host
        ;if everything is OK jump to the address in r3
        jmp     r3


end_of_loading
              ;load the destination address into r3
        jsr load_from_fifo
        move.l d5,r3

        eor d5,d7

        jsr load_from_fifo  ; reads 0
        jsr load_from_fifo  ; reads checksum to d5

        ;get the checksum into d7.l
        ;d2 = (0xffff0000 & d7)>>16
        extractu #16,#16,d7,d2
        ;d2 = d2 & 0x0000ffff
        and #$0000ffff,d2,d2
        ;d7 = d7 & 0x0000ffff
        and #$0000ffff,d7,d7
        ;d7 = d7 ^ d2
        eor d2,d7
        ;d2 = (~d7 & 0x0000ffff) = ~checksum
        not d7,d2
        and #$0000ffff,d2,d2

        ;get ~checksum into d4
        extractu #16,#16,d5,d4
        ;delete the ~checksum from d5
        and #$0000ffff,d5,d5
        ;if ( checksum_loaded |= Checksum_calculated ) goto set sticky bit
        cmpeq d5,d2
        nop
        iff jsr set_sticky_bit
        ;clean d4.h
        and #$0000ffff,d4,d4
        ;if ( ~checksum_loaded |= ~Checksum_calculated ) goto set sticky bit
        cmpeq d4,d7
        nop
        iff jsr set_sticky_bit


        ;set HF4 bit in HCR to show that loading is finished
        move.w HCR,d6
        or #$8000,d6.l
        move.w d6,HCR
```

```
        ;check the check sticky bit is set ( HF3 in HSR  (HSR[3) )
        move.w HSR,d6
        and #$00001000,d6,d6
        ;check if the sticky bit is set ( HF7 in HCR(HCR[3]) )
        move.w HCR,d4
        and #$00001000,d4,d4
        ;if both of the flags are set start the loading again
        and d4,d6
        tsteq d6
        jf from_host
        nop
        ; jump to the destination address
        jmp  r3

load_from_fifo
        tsteq d15 ; check flag if watchdog disabled
        jt load_no_wd
load_wd deceqa r15
        nop
        ift jsr watchdog_handle
HSR_read move.w HSR,d4
        bmtsts.w #$0001,d4.1
        jf    load_wd
        nop
        jmp read_fifo
        nop
load_no_wd
        ;if the host is empty wait for it to fill
              move.w HSR,d4
        bmtsts.w #$0001,d4.1
        jf    load_no_wd
        ; the host is not empty so load 8 bytes from it
read_fifo move.l #HORX,r0
        nop
        move.2l (r0),d4:d5
        rts


; Set the sticky bit (HF7) if there is an error in loading the program
set_sticky_bit
        ;set the HF7 bit in HCR
        move.w HCR,d6
        or #$00001000,d6.1
        move.w d6,HCR
        rts

watchdog_handle
        move.l #$0100,r15
        move.w #$556c,d12
        move.w d12,(r6+$e) ;write $556c to swsr
        move.w #$aa39,d12
        move.w d12,(r6+$e) ;write $aa39 to swsr
        rts
```

```
find_siubase
       ;read IMMR using siureg1
       move.l #$00f8ffc0,r7;move Address of IMMR (using siureg1) into d9
       nop
       move.l (r7),d1;read IMMR into d1
       nop
       extractu #$20,#0,d1,d1;remove the extend bits
       nop
       and #$ffff0000,d1,d1;Leave only ISB bits on IMMR
       move.l d1,d9        ;Added for serial boot
                           ;d9 gets the IMMR value
       inc.f d1     ; d1+$10000->d1
       jmp return_find_siu


sram_init
       move.l d1,r6
       move.l #$00000200,d1; base address for bank 10
       move.l #$00000100,d2; delta/2
       imac d2,d3,d1; (d1+d3.l*d2.l)->d1
       aslw d1,d1   ; d1<<16
       move.l #$01f00000,d0; offset 2 (for bank 11)
       aslw d3,d2   ; multiply d3 by 0x10,000
       add d2,d0,d0

       move.l d1,r5 ;
       bmset #$00c1,d1.l
       move.l #$fff80000,d7
       move.l d7,(r6+$154); SET OR10
       move.l d1,(r6+$150); SET BR10
       ; gpcm_init
       move.l #$ffff0000,d1; SET OR11 MASK 64 KB
       bmset #$0021,d0.l; changed the machine select to gpcm.
       move.l d1,(r6+$15c); SET OR11 MASK 64 KB
       move.l d0,(r6+$158); SET BR11 TO $01f0_0000
       jmp return_sram_init

upmc_init
; -------------- READ SINGLE -------------------------------------
       move.l #$90051240,d7
       move.l d7,(r6+$178) ;
       move.l #$00030040,d7
       move.l d7,(r6+$188) ;

       cmpeq.w #$2,d3
       jf continue_upmc1

       move.w #$0,(r5) ;
       move.l #$00030040,d7
       move.l d7,(r6+$188) ;

continue_upmc1
       move.w #$0,(r5) ;
       move.l #$00030045,d7
       move.l d7,(r6+$188) ;
```

```
        move.w #$0,(r5) ;
; -------------- READ BURST -------------------------------------
        move.l #$90051248,d7
        move.l d7,(r6+$178) ;
        move.l #$00030c48,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00030c4c,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00030c4c,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00030044,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00030045,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
; -------------- WRITE SINGLE -----------------------------------
        move.l #$90051258,d7
        move.l d7,(r6+$178) ;
        move.l #$00000040,d7
        move.l d7,(r6+$188) ;

        cmpeq.w #$2,d3
        jf continue_upmc2

        move.w #$0,(r5) ;
        move.l #$00000040,d7
        move.l d7,(r6+$188) ;

continue_upmc2
        move.w #$0,(r5) ;
        move.l #$00000045,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
; -------------- WRITE BURST ------------------------------------
        move.l #$90051260,d7
        move.l d7,(r6+$178) ;
        move.l #$00000c48,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00000c4c,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
```

```
        move.l #$00000c4c,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00000044,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
        move.l #$00000045,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
; -------------- EXCEPTION ------------------------------------
        move.l #$9005127c,d7
        move.l d7,(r6+$178) ;
        move.l #$ff000001,d7
        move.l d7,(r6+$188) ;

        move.w #$0,(r5) ;
; -------------- RESUME NORMAL OPERATION -------------------------
        move.l #$80011240,d7
        move.l d7,(r6+$178) ;
        jmp check_boot




;serial boot code here
;*********************************************************

TXB   equ $2160 ;TX Buffer Pointer
RXB   equ $2170;Rx Buffer Pointer

TIMEOUT equ $80000 ;Time Out till system is consider dead
SERBIT16 equ $1000 ;
;I2c Parameter Ram
I2C_BASE     equ $8afc
I2C_BASE_VAL equ $3e00


;To these
RBASE  equ    $3e00
RBASEVAL equ$3e50
TBASEVAL equ$3e40
RTBASEVAL equ   $3e503e40
RFCR   equ    $3e04
RSTATE  equ     $3e08
RPTR   equ    $3e0c
RBPTR   equ     $3e10
RTEMP  equ    $3e14
TSTATE  equ$3e18
TPTR   equ    $3e1c
TBPTR  equ    $3e20
TTEMP  equ    $3e24
```

```
DEBUGPTR equ    $3ea0
DEBUGEND equ    $3ef0

; Backup of registers
SCCRB    equ $3f00
PODRBB   equ$3f04
PSORBB   equ$3f08
PPARBB  equ   $3f0c
PDIRBB  equ  $3f10


; SYPCR   equ$0004
SWSR    equ $000e

SCCR    equ     $0c80
CPCR   equ    $19c0


;I2C Register
I2MOD  equ    $1860
I2ADD   equ$1864
I2BRG  equ    $1868
I2CER  equ    $1870
I2CMR  equ    $1874
I2COM  equ    $186c
;PB Regs
PODRB   equ$0d2c
PSORB   equ$0d28
PPARB  equ    $0d24
PDIRB  equ    $0d20
PDATB   equ$0d30

PDATC   equ$0d50



I2MODVAL  equ$001a;GCD=1, FLT=1 , PDIV = 01
I2MODVALE equ$001b;GCD=1, FLT=1 , PDIV = 01 , EN  = 1



I2BRGVAL  equ$0007

ARSLAVEADDRESS equ  $af000000;Aligned Read  Slave address
AWSLAVEADDRESS equ  $ae000000;Aligned Write Slave address



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;RESERVED REGISTER:
;      d5 : The address for read_bd function (Byte Shifted )
;      d7 : The Rx Buffer Lenght for read_bd function
;      d8 :
;      d9 : IMMR - Setted by previous boot_code
;      d10: Block counter ( count the numer of loaded blocks )
```

```
;   d12: Used for watchdog handler function.
;   d14: Tx buffer address, for read_db function.
;      r2 : Pointer to address of Block data on EEPROM
;      r5 : The Rx Buffer pointer for read_bd function
;      r6 : IMMR + $0001_0000
;      r7 : IMMR
;
;The follow registers were initiated by the previous code (boot_code_revA1. )
;      d13:Contain the ISB Bits
;      d9 :IMMR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; ---------------------------------------------------------------------
; -----------------  SERIAL BOOT CODE START HERE  -------------------
; ---------------------------------------------------------------------


from_serial

       move.l d9,r7        ;d9&r7 will have always IMMR value
       move.l r7,d11
       move.l #$10000,d12
       add    d11,d12,d11
       move.l d11,r6;r6<-IMMR+$10000 for registers access


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     Configure I2 Register &
;     Parameter RAM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

       move.l d9,r3        ;r3 <-  IMMR
       bmset   #I2C_BASE,r3.l;r3 receive I2C page base address

       ;sw watchdog
       move.l #$0,d11;clear d11, software watchdog is not enabled
       move.l (r6+$4),d12;get sypcr value
       bmtsts #$0004,d12.l
       jf init_
       nop
       move.l d11,r15 ; clear watchdog handle counter
       move.l #$1,d11; indicate software watchdog enabled
       move.l #$0100,r15  ; initialize watchdog handle counter


;Initiate I2C_BASE
init_  move.w #I2C_BASE_VAL,d4
       nop                       ;inserted due to restriction 6.4.4.a.2
       move.w   d4,(r3);Save d4 into I2C_BASE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Initiate Parameter RAM
       move.l  #RTBASEVAL,r1 ;RBASE=3e50,TBASE=3e40
       nop                       ;due to restriction 6.4.4.a.2
       move.l r1,(r7+RBASE);$00
       move.l #$1212ff00,r1 ;RFCR=12(use local BUS),TFCR=12,MRLBR=ff00
```

```
        nop                         ;due to restriction 6.4.4.a.2
        move.l r1,(r7+RFCR) ;$04
        move.l #$00000000,r1 ;RSTATE
        nop                         ;due to restriction 6.4.4.a.2
        move.l r1,(r7+RSTATE);$08
        move.l r1,(r7+RPTR) ;$0c
        move.l r1,(r7+RBPTR);$10
        move.l r1,(r7+RTEMP);$14
        move.l r1,(r7+TSTATE);$18
        move.l r1,(r7+TPTR) ;$1c
        move.l r1,(r7+TBPTR);$20
        move.l r1,(r7+TTEMP);$24
        move.l #$10000,d1
        nop                         ;due to restriction 6.4.4.a.2
        add    d1,d9,d4     ;d4 receive  IMMR +1_0000
        nop                         ;due to restriction 6.4.4.a.2
        move.l d4,r6                ;R6 <-IMMR +1_0000
        ;Initiate I2C command - Page = 01010,code=01011
        move.l #$29610000,r1
        nop
        move.l r1,(r6+CPCR);Write the cmd to CPCR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Save registers in the backup area of DP ram
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        move.l (r6+SCCR),r1
        nop
        move.l r1,(r7+SCCRB)

        move.l (r6+PODRB),r1
        nop
        move.l r1,(r7+PODRBB)

        move.l (r6+PSORB),r1
        nop
        move.l r1,(r7+PSORBB)

        move.l (r6+PPARB),r1
        nop
        move.l r1,(r7+PPARBB)

        move.l (r6+PDIRB),r1
        nop
        move.l r1,(r7+PDIRBB)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Initiate SCCR to 01 -> divide by 16
        move.l   #1,r1
        move.l  r1,(r6+SCCR)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;Initiate IO Port
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ; Configure port B SCL=PB18,SDA=PB19
        move.l #$00003000,r1
        nop                 ;due to restriction 6.4.4.a.2
```

```
        move.l r1,(r6+PODRB); PODRB[18,19]=1
        move.l r1,(r6+PSORB); PSORB[18,19]=1
        move.l r1,(r6+PPARB);
        move.l #$00000000,r1
        nop                 ;due to restriction 6.4.4.a.2
        move.l r1,(r6+PDIRB);

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;Initiate I2C Registers
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        move.w #$0,r1
        move.b r1,(r6+I2MOD);Clear i2mod
        move.b r1,(r6+I2CMR);Disable All Interrupt
        move.w #$ec,r1
        move.b r1,(r6+I2ADD);Slave Address= 00 ( optional )
        move.w #$0017,r1
        move.b r1,(r6+I2CER)   ;Clear all previous events
        move.w #I2MODVAL,r1
        move.b r1,(r6+I2MOD);I2MOD[GCD]=1,[FLT]=1
        move.w   #I2BRGVAL,r1
        move.b r1,(r6+I2BRG)   ;I2BRG[DIV]=6
        move.w #$0001,r1
        move.b r1,(r6+I2COM)   ;I2COM[M/s]=1 ( Master mode )
        move.w #I2MODVALE,r1
        move.b r1,(r6+I2MOD);Set I2MO[EN]

;********************************
;SRAM base moved to r7 due to register usage
;********************************
        move.l r5,(r5)
        nop
        move.l (r5),r7



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Initiate debug area at IMMR + Debug Pointer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        move.l #DEBUGPTR,d10  ;
        add    d9,d10,d10;d10 <- ptr to debug area
        move.l  d10,r1;r1  <- ptr to debug area
        add    #$4,d10      ;d10 <- ptr to empty space in debug area
        nop
        move.l d10,(r1);save the the ptr in r1
        move.l d10,r1       ;move empty ptr to r1

        ; This lines will move a empty pattern ( a5 )
        ; to all the debug area
        move.w  #$a5a5,d10;d10 receive empty pattern
iloop   move.b  d10,(r1)+;save empty pattern in r1 and increment r1
        nop                 ;added due to restriction
        move.l  r1,d8;Move r1 to d8
        nop                 ;added due to restriction
```

```
       extractu #$f,#0,d8,d8 ;Leave only offset in d8
       nop                ;added due to restriction
       cmpeq.w  #DEBUGEND,d8;compare if end of debug
       jf  iloop    ;if not jmp to the loop
       nop                ;added due to restriction
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    Calculate Table Address &
;     Get the code Address
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

       move.l d13,d5   ; d5 receive ISB
       move.l #$0,d10  ; reset the Blocks counter ( d10 )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    Read first Block Address
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ;D5 <- D5 x 2 + byte shift left
       asll #$9,d5  ;d5 <- Address(1 Byte shifted )
       move.l #$2,d7;d7 <- Rx BD lenght
       move.l #TXB,d4;prepare Tx buffer address in d14
       move.l r5,d14;r5 holds SRAM base address
       nop
       add d4,d14,d14;add sram base
       move.l #RXB,r5 ;r5 <-Rx Buffer Ofset from SRAM
       move.l r7,d4 ;d4 <-SRAM base

       bsr read_bd
       ;In the read_bd routine ,the start of code address ( 2 bytes )
       ;was read to the RX Buffer at address RXB
       move.w (RXB),d5;d5 Receive the code start address
       bmclr #$FFFF,d5.h;clear unreaded upper bits
       move.l  d5,d8;Back d5 it into d8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Prepare for read First Block Header
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ;D5 <- D5 x 2 + byte shift left
       asll #$8,d5  ;d5 <- Address(1 Byte shifted )
       move.l #$8,d7;d7 <- Header lenght(4 words)
       ; Calculate the address for Head at DPRAM
       move.l #RXB,d2  ; d2<- Block Header (Ofset from IMMR)
       nop                ;due to restriction 6.4.4.a.2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    Get 4 first word
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Get4FWord
       move.l  d2,r5; r5<-Head Pointer ( RXB )
       move.l #$8,d7; d7 <- Header lenght(4 words)
       bsr read_bd  ; Call read_bd to read the first 4 words
;   Read the Block  Header Data
       move.l  r5,d6; d6<-Head Pointer Offset
       nop                ;due to restriction 6.4.4.a.2
       add    d4,d6,d6; d6<-Head Pointer Offset + RAMptr
       move.l  d6,r8; r8<-Head Pointer Offset + RAMptr
       nop                ;due to restriction 6.4.4.a.2
       move.w (r8),d7;d7 <- CS Enabled bit | Block size
```

```
        extractu #$e,#0,d7,d7 ;remove the extend bits and CS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; if Block size == 0 goto Boot_loaded ( end )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        tsteq  d7           ;compare d7 to 0
        jt     Boot_loaded;if d7 = 0->jmp to Boot Loaded
        nop                 ;due to restriction 6.4.4.a.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Read Block Data
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        move.l (r8+$4),r5;Address to save the data on DPRAM (offset from SRAM )
        move.l d8,d5        ;d5<-Block data address on seeprom
        nop                 ;due to restriction 6.4.4.a.2
        add    #$8,d5       ;increment d5 so it points to data (first 8 bytes = Header )
        asll #$8,d5  ;Byte shift d5 ( prepare for read_bd)
read_block
        move.l #$0,d0       ;Reset Status Bit in d0
read_block2
        move.l r7,d4
        bsr read_bd
        move.l  d2,r4;r4<-Head Pointer
        nop                 ;due to restriction 6.4.4.a.2
        move.w (r4),d8;d8<- CSE | Block size
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Checksum Enabled ????
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        bmtsts  #$8000,d8.l; Test with CS bit is setted
        jt     calc_cs      ;if checksum  enable jump to calc CS
        nop                 ;;due to restriction 6.4.4.a.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Prepare Next Block Address
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PrepareNextBAdd
        add #$1,d10  ;increment the BD counter
        move.w (r4+$2),d8;d8<- Next Block Address
        move.l  d8,d5;Restore address from d8
        asll #$8,d5  ;d5 <- Address for next Block(1 Byte shifted )
        jmp Get4FWord;jmp to Get 4 First words
        nop


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Start Boot Execution
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Boot_loaded
        move.l #$80,d0      ;Set Boot Loaded flag on d0
        move.l (r8+$4),r3;r3 receive Boot Start Address


        ; Reset all I2C Parameter to their default value
        move.l #$0,r1
        move.b r1,(r6+I2MOD);Clear i2mod
        move.b r1,(r6+I2ADD);Slave Address= 00 ( optional )
```

```
       move.b r1,(r6+I2CMR);Disable All Interrupt
       move.b r1,(r6+I2BRG)   ;I2BRG[DIV]=6
       move.b r1,(r6+I2COM)   ;I2COM[M/s]=1 ( Master mode )
       move.w #$0017,r1
       move.b r1,(r6+I2CER)   ;Clear all previous events
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Reload saved registers values from the backup area of DP ram
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move.l  d9,r7
       nop
       move.l (r7+SCCRB),r1
       nop
       move.l r1,(r6+SCCR)

       move.l (r7+PODRBB),r1
       nop
       move.l r1,(r6+PODRB)

       move.l (r7+PSORBB),r1
       nop
       move.l r1,(r6+PSORB)

       move.l (r7+PPARBB),r1
       nop
       move.l r1,(r6+PPARB)

       move.l (r7+PDIRBB),r1
       nop
       move.l r1,(r6+PDIRB)

       jmp  end_

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;      END OF MAIN CODE FLOW
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
calc_cs
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Calculate checksum
;For this we will use the follow register
;      d1 : Calculated CS
;      d6 : words Count
;      r1 : Adddress of word to be read
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move   r5,d6
       nop                ;due to restriction 6.4.4.a.2
       add    d4,d6,d6;d6<-First RX Buffer Pointer
       nop                ;due to restriction 6.4.4.a.2
       move.l d6,r1       ;r1 receive the address of first word
       move.l (r4),d1     ;start xor with getting 1'st 2 words of block
       move.l (r4+4),d6 ;get 2'nd 2 words
       eor    d6,d1       ;xor with previous value
```

```
        move.l #$0,d6       ;reset the word count

cs_loop
        move.l (r1)+,d3;read long data into d3
        add    #$4,d6
        cmpeq  d6,d7        ;if d6 = d7 ->Last Block word ( CS )
        jt     calc_rchecksum;  then jump to calc_checksum
        nop                 ;due to restriction 6.4.4.a.3
        eor    d3,d1        ;     else CS <- ( CS xor data )
        jmp    cs_loop
        nop
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
calc_rchecksum
        aslw   d1,d6        ;shift left word d1 into d6
        eor    d1,d6
        extractu #$20,#0,d6,d6;remove the extend bits
        bmclr  #$FFFF,d6.l;Leave only the checksum on d6.h
        asrw   d6,d1        ;Move it (word shift right) to d1
        not    d1.l         ;Obtain inverted CS (CS_b)  on d1
        add    d1,d6,d6;Obtain CS|CS_b   on d6
        extractu #$20,#0,d3,d3;remove the extend bits
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  CheckSum Ok ?
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        cmpeq  d6,d3        ;Compare to received CS
        jt     PrepareNextBAdd;if equal jump to received BD
        nop                 ;due to restriction 6.4.4.a.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Got a CS Error
;If  this is the first CS error for this BD , Set Flag and Retry
;    Else jmp to CS error
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        bmtsts  #$0001,d0.l;Test with previous CS error ocurr
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First error
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        jt     cs_error;If yes jump to CS error routine
        nop                 ; due to restriction 6.4.4.a.3
        bmset #$0001,d0.l;set CS Fail flag
        jmp    read_block2;retry to read the block
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Set bit 1 in d0 ( to indicate a CS error) and abort
cs_error
        bmset #$0002,d0.l;set CS Error flag
        jmp reset

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;***********************************************
read_bd
;This Routine will read a BD
```

```
;This routine will create 2 Tx BD and one Rx BD That receive the data
;Before enter these routine the registers must be setted as follow:
;      d9 : IMMR
;      d4 : Rx Buffer Pointer base (SRAM base)
;      d5 : The address of EEPORM to be read (Shifted 1 byte)
;      d14: Tx buffer address
;            left, e.g add=$4 -> d5 = $0400)
;      d7 : The lenght for RX BD
;      r6 : IMMR + $1_0000
;      r5 : Rx Buffer Pointer offset
;Corrupted registers:
;      d1,d3,d6 ,r0,r1,r3
;************************************************
retry

       move.w #$0017,r1
       move.b r1,(r6+I2CER)   ;Clear all previous events
       ;Prepare the 2 TX & 1 Rx BD's  to read 2 bytes from add8ess at d5
       move.l #TBASEVAL,d1    ; d1<-TBASE
       nop                 ;due to restriction 6.4.4.a.2
       add    d9,d1,d1; d1<-TX BD ptr
       move.l d1,r0        ; r0<-TX BD ptr
;************************************************
;Prepare 2 Tx & 1 Rx BD
;************************************************

       ; First Tx BD
       move.l  #$84000003,d1;BD STATUS(R=1,I=1,L=0,S=1)|DATA Lenght = 3
       move.l d1,(r0)
       move.l d14,(r0+$4)  ;Tx buffer address
       ; Write TX Buffer DATA
       move.l d14,r3       ;r3 receive the Tx Buffer ptr
       move.l #AWSLAVEADDRESS,d6
       extractu #$20,#0,d6,d6;remove the extend bits
       add    d5,d6,d6        ;d5=SLAVE ADDRESS | ADDRESS WORD
       move.l d6,(r3)

       ; 2nd TX BD
       move.l  #$bc000001,d3;BD STATUS(R=1,I=1,w=1,L=1,S=1)|Lenght  = 1
       add    d7,d3,d3;set  Lenght = d7 + 1
       move.l d3,(r0+$8)
       move.l d14,d6
       add    #$4,d6       ; d6<-2nd TX Buffer Pointer
       move.l d6,(r0+$c)
       ; Write TX Buffer DATA for second Buffer
       move.l #ARSLAVEADDRESS,d6
       extractu #$20,#0,d6,d6;remove the extend bits
       move.l d6,(r3+4)

       ;Prepare the Rx BD
       move.l #RBASEVAL,d1    ; d1<- RBASE
       add    d9,d1,d1; d1<- BD ptr
       nop                 ;due to restriction 6.4.4.a.2
       move.l d1,r1        ; r1<- BD ptr
       nop                 ;due to restriction 6.4.4.a.2
```

```
        move.l  #$b8000000,d1;BD STATUS(E=1,W=1,I=1,L=1)|DATA Lenght = 0
        nop                 ;due to restriction 6.4.4.a.2
        move.l d1,(r1)      ;Write BD STATUS + lenght
        move   r5,d1        ;d1 receive Buffer ptr offset
        nop
        add    d1,d4,d1; added buffer base to d1
        nop
        move   d1,r3
        nop                 ;due to restriction 6.4.4.a.2
        move.l r3,(r1+$4);Write RX Buffer Pointer




;************************************************
;Set Start on i2COM
;************************************************

i2com_ move.w #$8181,r3
       move.b r3,(r6+I2COM)   ;I2COM[STR]=1 => Start transmission

       move.l r6,d6
       move.l #I2CER,d1
       add d1,d6,d6
       move.l d6,r3        ;  r3<-I2CER Adrress
       move.l #TIMEOUT,d1

read_i2cer
       move.l #SERBIT16,d15
       ;************************************************
       ; We insert a delay here in order to have
       ; less BUS activity
       ;************************************************

ser_cnt  sub#$1,d15
       tsteq d11; check flag if watchdog disabled
       jt tst_no_wd
       deceqa r15
       nop
       ift jsr watchdog_handle
tst_no_wd tsteqd15
       jf ser_cnt
       nop

       ;;Decrement Counter if zero goto dead_i2cer
       sub    #$1,d1
       tsteq  d1
       jt dead_i2cer
       nop          ; due to restriction 6.4.4.a.3

       move.b (r3),d6   ;Read I2CER
       nop
       cmpeq.w  #$0000,d6
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;     I2cer == 0 ??
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       jt read_i2cer;if I2CER =0 => Read Again
       nop                ;due to restriction 6.4.4.a.3


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;     BD & I2CER OK ???
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The following lines test BD and I2CER
; To check that we received data correctly.
; If not, we will retry if possible or abort ( depending on error type )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

       bmtsts  #$04,d6.l;Test BSY bit
       jt     BSYError;If true Abort
       nop                ;due to restriction 6.4.4.a.3

       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move.w (r0),d3      ;Read the BD status of the 1st TXBD
       nop
       bmtsts  #$01,d3.l;Test Colision BIt
       jt     colision;If true retry
       nop                ;due to restriction 6.4.4.a.3

       bmtsts  #$04,d3.l;Test NAK BIt
       jt     nAK          ;If true abort
       nop                ;due to restriction 6.4.4.a.3

       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move.w (r0+$8),d3 ;Read the BD status of the 2nd TXBD
       nop
       bmtsts  #$01,d3.l;Test Colision BIt
       jt     colision2;If true retry
       nop                ;due to restriction 6.4.4.a.3

       bmtsts  #$04,d3.l;Test NAK BIt
       jt     nAK          ;If true abort
       nop                ;due to restriction 6.4.4.a.3

              cmpeq.w  #$0003,d6;If not I2cer[TXB]=1,I2CER[RXB]=1
       jf read_i2cer;read again
       nop                ;due to restriction 6.4.4.a.3

       move.w (r1),d3      ;Read the BD status of the RxBD ( r1 contain RX BD ptr )
       nop
       cmpeq.w  #$3800,d3
       jf read_i2cer
       nop                ;due to restriction 6.4.4.a.3


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; write to debug area at IMMR + Debug Pointer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
wdebug
       move.l #DEBUGPTR,d3
```

```
        add    d9,d3,d3;d3 <- ptr to debug area
        nop
        move.l  d3,r1;r1 receive d13
        nop                 ;due to restriction
        move.l (r1),d3      ;d3 receive available address on debug area
        move.l  d3,r0;r0 receive available address on debug area


        ; Check with end of debug area
        move.l  r0,d6
        nop
        extractu #$f,#0,d6,d6 ;remove the extend bits and CS
        nop
        cmpeq.w  #DEBUGEND,d6
        jt   wdebugend;if end of debug area jmp to end of debug
        nop
        ;;;;;;;;;;


        move.w  #0,d6
        move.b  d6,(r0)+;write 0 and increment d3
        nop
        move.l r0,(r1)
wdebugend nop

return_rts                ;return from sub routine

;************************************************
; End of read_bd Function
;************************************************




;************************************************

dead_i2cer
        move.l #$deadead,d0

reset


        move.l #$0,r1
        ; Reset port B SCL=PB18,SDA=PB19
        move.l r1,(r6+PODRB)
        move.l r1,(r6+PSORB)
        move.l r1,(r6+PPARB)
        move.l r1,(r6+PDIRB)


        debug

colision
colision2
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; write to debug area at IMMR + Debug Pointer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move.l #DEBUGPTR,d3
       add    d9,d3,d3;d3 <- ptr to debug area
       nop
       move.l  d3,r1;r1 receive d13
       nop                ;due to restriction
       move.l (r1),d3     ;d3 receive available address on debug area
       move.l  d3,r0;r0 receive available address on debug area

       ; Check with end of debug area
       move.l  r0,d6
       nop
       extractu #$f,#0,d6,d6 ;remove the extend bits and CS
       nop
       cmpeq.w  #DEBUGEND,d6
       jt   wdebugend2
       nop
       ;;;;;;;;;;;
       move.w  #1,d6
       move.b  d6,(r0)+;write 0 and increment d3

       nop
       move.l r0,(r1)

wdebugend2

restart

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Close RX BD Command routine + wait
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       move.l #$29610007,r1
       nop
       move.l r1,(r6+CPCR);Close BD command - Page = 01010,code=01011
       ; ***wait for command flag to be clear
wait   move.l (r6+CPCR),d1
       nop
       bmtsts  #$0001,d1.h; Test with CS bit is setted
       jt     wait         ;if flag is still setted goback to wait
       nop                 ;due to restriction 6.4.4.a.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Initiate Parameter RAM
       move.l  #RTBASEVAL,r1 ;RBASE=3e50,TBASE=3e40
       nop                 ;due to restriction 6.4.4.a.2
       move.l r1,(r7+RBASE);$00
       move.l #$1212ff00,r1 ;RFCR=12(use local BUS),TFCR=12,MRLBR=ff00
       nop                 ;due to restriction 6.4.4.a.2
```

```
        move.l r1,(r7+RFCR);$04
        move.l #$00000000,r1 ;RSTATE
        nop                  ;due to restriction 6.4.4.a.2
        move.l r1,(r7+RSTATE);$08
        move.l r1,(r7+RPTR);$0c
        move.l r1,(r7+RBPTR);$10
        move.l r1,(r7+RTEMP);$14
        move.l r1,(r7+TSTATE);$18
        move.l r1,(r7+TPTR);$1c
        move.l r1,(r7+TBPTR);$20
        move.l r1,(r7+TTEMP);$24
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Init Page Command routine + wait
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        move.l #$29610000,r1
        nop
        move.l r1,(r6+CPCR);Init command - Page = 01010,code=01011
        ; ***wait for command flag to be clear
wait2   move.l (r6+CPCR),d1
        nop
        bmtsts  #$0001,d1.h; Test with CS bit is setted
        jt     wait2        ;if flag is still setted goback to wait
        nop                 ;due to restriction 6.4.4.a.3

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


        jmp retry
        nop
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

nAK     move.l #$a1caaaa,d0
        debug

BSYError
        move.l #$bbbbbbbb,d0
        debug
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

end_    nop

        jmp    r3           ;jmp to Boot start Address

        endsec
```

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 München, 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

EB632
Rev. 2
1/2005

freescale™
semiconductor