MSC8113 Reference Manual

Tri Core 16-Bit Digital Signal Processor

MSC8113RM
Rev 0, May 2008

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

MSC8113RM
Rev. 0
5/2008

Contents

About This Book

1 MSC8113 Overview

2 SC140 Core Overview

3 External Signals

4 System Interface Unit (SIU)

5 Reset

6 Boot Program

7 Clocks

8 Memory Map

9 Extended Core

13 System Bus

15 Hardware Semaphores

16 Direct Memory Access (DMA) Controller

17 Interrupt Processing

18 Debugging

19 Internal Peripheral Bus (IPBus)

20 TDM Interface

21 UART

22 Timers

23 GPIO

24 I²C Software Module

25 Ethernet Controller

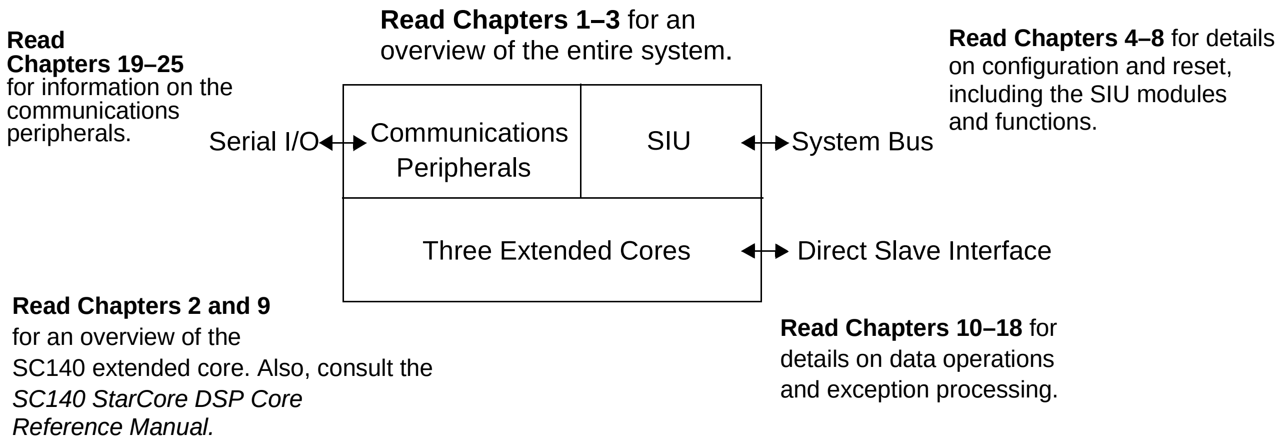
A Programming Reference

B MSC8113 Dictionary

C MSC8113 Boot Code

About This Book

The MSC8113 device is based on the StarCore® SC140 DSP core. It addresses the challenges of the networking market. The benefits of the MSC8113 include not only a very high level of performance but also a product design that enables effective software development and integration. Its tool suite provides a full-featured development environment for C/C++ and assembly languages as well as ease of integration with third-party software, such as off-the-shelf libraries and a real-time operating system. The MSC8113 is logically partitioned into three distinct blocks: three extended cores, a system interface unit (SIU), and communications peripherals.



Read Chapters 1–3 for an overview of the entire system.

Read Chapters 19–25 for information on the communications peripherals.

Read Chapters 4–8 for details on configuration and reset, including the SIU modules and functions.

Read Chapters 2 and 9 for an overview of the SC140 extended core. Also, consult the *SC140 StarCore DSP Core Reference Manual.*

Read Chapters 10–18 for details on data operations and exception processing.

Three Extended Cores

Each extended core contains an SC140 DSP core with internal memory for data and program storage, peripheral hardware, and two interrupt controllers. Memory includes 224 KB (896 KB total) of zero wait state SRAM and 16 KB (64 KB total) of instruction cache. The MSC8113 also includes 476 KB of shared memory (M2) and 4 KB of boot ROM. Minimum code density is achieved using a 16-bit instruction set that is grouped into execution sets by the compiler (or by the programmer) for high instruction parallelism.The DSI provides a glueless 32/64-bit interface to a host processor for data and command communication. The programmable interrupt controller (PIC) and local interrupt controller (LIC) process all internal interrupt requests, notifying the SC140 DSP cores or external devices of an interrupt event.

SIU

Supports internal and external system-related functions. The SIU includes hardware such as a direct memory access (DMA) controller, clocks, and reset configuration registers. It also includes the memory controllers, which interface to external memory devices and/or other devices such as a system host or other DSPs.

Communications Peripherals

Includes four TDM interfaces with 256 channels each, a UART, thirty-two 16-bit timers, thirty-two programmable GPIO signals, eight hardware semaphore registers, an I^2C software module, an Ethernet interface, and a global interrupt controller (GIC). The serial interfaces give additional functionality and flexibility. The semaphore registers provide resource control for external hosts. The GIC extends interrupt handling capability.

Before Using This Manual—Important Note

This manual describes the structure and function of the MSC8113 device. The information in this manual is subject to change without notice, as described in the disclaimers on the title page of this manual. As with any technical documentation, it is your responsibility as the reader to ensure that you are using the most recent version of the documentation. For more information, contact your sales representative.

Before using this manual, determine whether it is the latest revision and whether there are errata or addenda. To locate any published errata or updates associated with this manual or this product, refer to the Freescale web site. The address for the web site is listed on the back cover of this manual.

Audience and Helpful Hints

This manual is intended for software and hardware developers and applications programmers who want to develop products with the MSC8113. It is assumed that you have a working knowledge of DSP technology and that you may be familiar with Freescale products based on the Freescale DSP56000 or DSP56300 core. Familiarity with Freescale DSP products is not necessary.

For your convenience, the chapters of this manual are organized to make the information flow as predictably as possible. When feasible, the information in each chapter follows this general sequence:

- Features
- Architecture
- Signals
- Operation/operating modes
- Programming
- Programming Examples
- Programming Model (registers)

In chapters that include a Programming Model section, this section is the last one in the chapter, or module subsection for those chapters that include multiple modules, and describes all registers for the module discussed. The Programming Model section begins with a bulleted overview of the registers that includes the page number where the description of each register begins.

Notational Conventions and Definitions

This manual uses the following notational conventions:

mnemonics Instruction mnemonics appear in lowercase bold.

COMMAND names Command names are set in small caps, as follows: GRACEFUL STOP TRANSMIT or ENTER HUNT MODE.

italics Book titles in text are set in italics, as are cross-referenced section titles. Also, italics are used for emphasis and to highlight the main items in bulleted lists.

0x Prefix to denote a hexadecimal number.

0b Prefix to denote a binary number.

REG[FIELD] Abbreviations or acronyms for registers or buffer descriptors appear in uppercase text. Specific bits, fields, or numeric ranges appear in brackets. For example, ICR[INIT] refers to the Force Initialization bit in the host Interface Control Register.

ACTIVE HIGH SIGNALS Names of active high signals appear in sans serif capital letters, as follows: TT[04], TSIZ[0–3], and DP[0–7].

$\overline{\text{ACTIVE LOW}}$ SIGNALS Signal names of active low signals appear in sans serif capital letters with an overbar, as follows: $\overline{\text{DBG}}$, $\overline{\text{AACK}}$, and $\overline{\text{EXT\_BG[2]}}$.

x A lowercase italicized x in a register or signal name indicates that there are multiple registers or signals with this name. For example, BRCG*x* refers to BRCG[1–8], and M*x*MR refers to the MAMR/MBMR/MCMR registers.

On the MSC8113 device, the SC140 cores are 16-bit DSP processors. The following table shows the SC140 assembly language data types. For details, see the *StarCore SC140 DSP Core Reference Manual (MNSC140CORE/D)*.

Name	SC140
Byte/Octet	8 bits
Half Word	8 bits
Word	16 bits
Long/Long Word/2 Words	32 bits
Quad Word/4 Words	64 bits

The following table lists the SC140 C language data types recognized by the StarCore C compiler. For details, see the *StarCore SC100 C Compiler User's Manual (MNSC100CC/D)*.

Name	Size
char/unsigned char	8 bits
short/unsigned short	16 bits
int/unsigned int	16 bits

Name	Size
fractional short	16 bits
long/unsigned long	32 bits
fractional long	32 bits
pointer	32 bits

Conventions for Registers

The Programming Model section of each chapter includes a register bit table for each register in that module, as well as a table describing each bit in the register. The register bit table not only shows the names and positions of the bits/bit fields but also their reset value, value after boot, and their type (Read/Write). For registers that are not changed by the system boot, no boot line is listed. The register address is shown with the register name and mnemonic. Reserved bits/fields are indicated with a long dash (—). In the PPC_ALRH shown below, all of the bits are read/write (R/W). Other registers may include read-only (R) and write-only (W) bits. Notice that the most significant bit (MSB) is 0, or little-endian order.

PPC_ALRH System Bus Arbitration-Level Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	Priority Field 0				Priority Field 1				Priority Field 2				Priority Field 3			
Type	R/W															
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0
Boot	1	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	Priority Field 4				Priority Field 5				Priority Field 6				Priority Field 7			
Type	R/W															
Reset	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	1
Boot	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	0

Organization

Following is a summary and a brief description of the chapters of this manual:

- **Chapter 1, *MSC8113 Overview***. Features, descriptive overview of main modules, configurations, and application examples.
- **Chapter 2, *SC140 Core Overview***. Target markets, features, overview of development tools, descriptive overview of main modules.
- **Chapter 3, *External Signals***. Identifies the external signals, lists signal groupings, including the number of signal connections in each group, and describes each signal within a functional group.

- **Chapter 4,** ***System Interface Unit (SIU)***. Describes the modules and functions of the SIU, which controls system start-up and initialization as well as operation, protection, and the external 60x-compatible system bus.
- **Chapter 5,** ***Reset***. Covers reset sources, causes, and configurations; gives examples of different reset configuration scenarios, including systems with multiple MSC8113s.
- **Chapter 6,** ***Boot Program***. Describes the bootloader program, which loads and executes source code that initializes the MSC8113 after it completes a reset sequence and programs its registers for the required mode of operation. This chapter covers selection of bootloader modes, normal sequence of events for bootloading a source program, and booting in a multi-processor environment.
- **Chapter 7,** ***Clocks***. Contains an overview of the MSC8113 clock module. For complete clock information, refer to the *MSC8113 Technical Data* sheet. The data sheet is available in PDF format on the Freescale web site listed on the back cover of this manual.
- **Chapter 8,** ***Memory Map***. Defines the address spaces for all MSC8113 modules; includes cross references to all registers discussed.
- **Chapter 9,** ***Extended Core***. Describes the structure of the extended core, which includes the SC140 core, its internal memory (M1), the extended QBus structure (EQBS), the Instruction Cache (ICache), the programmable interrupt controller (PIC), and the local interrupt controller (LIC).
- **Chapter 10,** ***MQBus and M2 Memory***. Describes how the MQBus supports a multi-core environment by allowing all three SC140 cores to share the M2 memory through the MQBus. The MQBus ensures a low miss ratio for SC140 ICache accesses.
- **Chapter 11,** ***SQBus***. Explains the structure and function of the SQBus, which is available to all SC140 cores to fetch program code from external memory on the system bus.
- **Chapter 12,** ***Memory Controller***. Covers the features and basic architecture of the memory controller, which is part of the system interface unit (SIU). The memory controller provides an interface to internal DSP memory and DSP peripherals residing on the internal local bus and also to external memory and peripheral devices on the external 60x-compatible system bus. In addition to features and basic architecture, this chapter extensively covers the three basic machines that compose the memory controller: synchronous DRAM machine (SDRAM), general-purpose chip-select machine (GPCM), and the user-programmable machines (UPMs).
- **Chapter 13,** ***System Bus***. Discusses the system bus, which is a 60x-compatible bus that provides flexible support for the on-chip SC140 cores as well as other internal and external 60x-compatible bus masters.
- **Chapter 14,** ***Direct Slave Interface (DSI)***. Discusses the DSI host interface, which is a 32/64-bit wide, full-duplex, double-buffered, parallel port that can directly connect to the data bus of a host processor. The DSI supports a variety of buses and provides glueless connection with a number of industry-standard microcomputers, microprocessors, and DSPs.

- **Chapter 15,** ***Hardware Semaphores***. Describes the function and programming of the hardware semaphores, which control resource sharing.
- **Chapter 16,** ***Direct Memory Access (DMA) Controller***. Describes the different DMA operating modes, transfer types, and buffer types. The chapter also gives procedures for programming different types of transfers. The multi-channel DMA controller includes hardware support for up to 16 time-multiplexed channels including buffer alignment, connects to both the system bus and the local bus, and functions as a bridge between both buses. The DMA controller supports flyby transactions on either bus. and enables hot swaps between channels, by using time-multiplexed channels that impose no cost in clock cycles.
- **Chapter 17,** ***Interrupt Processing***. Discusses the three interrupt controllers that provide maximum flexibility in handling MSC8113 interrupts, enabling interrupts to be handled by the SC140 core internally, by an external host, or by a combination of the two; also discusses source priority schemes.
- **Chapter 18,** ***Debugging***. Includes aspects of the JTAG implementation that are specific to the SC140 and should be used with the supporting **IEEE**® Std. 1149.1™ documentation. The discussion covers the items that the standard requires to be defined and provides additional information specific to the MSC8113 implementation.
- **Chapter 19,** ***Internal Peripheral Bus (IPBus)***. Describes the internal peripheral buss (IPBus), the devices that connect to it, energy management capabilities for devices on the bus (Stop modes), and the programming model.
- ***Chapter 20, TDM Interface***. Describes the four TDM interfaces. Each can handle up to 256 channels. The interfaces support the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates.
- **Chapter 21,** ***UART***. Describes the UART interface, which is a full-duplex serial port used to communicate with other devices.
- **Chapter 22,** ***Timers*** Discusses the 32 identical 16-bit general-purpose timers residing in two timer modules (A and B) that each have their set of configuration registers.
- **Chapter 23,** ***GPIO***. Discusses the thirty-two GPIO signals. Each pin is multiplexed with a TDM, UART, or timer signal and can be configured as an input or output or a dedicated peripheral pin. Part of the pins can be configured as open-drain (that is, the pin can be configured in a wired-OR configuration on the board). The pin drives a zero voltage but tri-states when driving a high voltage.
- **Chapter 24,** ***I²C Software Module***. Describes the I^{2}C interface. which allows the MSC8113 to boot from a serial EEPROM device.
- **Chapter 25,** ***Ethernet Controller***. Discusses the Ethernet controller, which supports three modes of operation: MII, RMII, and SMII.

- **Appendixes:**
  - — **Appendix A,** ***Programming Reference***.
  - — **Appendix B,** ***MSC8113 Dictionary***.
  - — **Appendix C,** ***MSC8113 Boot Code***.

Other MSC8113 Documentation

You can find the following documents on the Freescale Semiconductor web site listed on the back cover of this manual.

- *MSC8113 Technical Data sheet (MSC8113)*. Details the signals, AC/DC characteristics, PLL/DLL performance issues, clock configuration and signal characteristics, package and pinout, and electrical design considerations of the MSC8113 device.
- *Application Notes*. Cover various programming topics related to the StarCore DSP core and the MSC8113 device.

Further Reading

- *StarCore SC140 DSP Core Reference Manual*. Covers the SC140 core architecture, control registers, phase lock loop (PLL), clock registers, hardware debug capabilities (EOnCE), program control, and instruction set.

MSC8113 Overview 1

The MSC8113 device is a highly integrated DSP that combines three StarCore SC140 cores with large internal memory spaces, an extended core, and several industry-standard peripherals and external interfaces to target highly computational DSP network and communication applications. The device is optimized for high-bandwidth wireless transcoding and a high-density packet telephony DSP farm, as well as high-bandwidth base station applications. The MSC8113 delivers enhanced performance while maintaining low power dissipation and greatly reducing overall system cost.

Each SC140 core has four ALUs that provide performance of up to 1600 DSP million multiply and accumulate commands per second (MMACS) using an internal 400 MHz clock. The MSC8113 three-core device therefore delivers a total performance of up to 4800 DSP MMACS.

Each core is part of an extended core that includes a level-1 224 KB internal memory (M1) for program and data storage, a 16 KB 16-way instruction cache (ICache), a fetch unit for the ICache, and a 4-entry write buffer queue for boosting core performance. Each extended core also includes a programmable interrupt controller (PIC), a local interrupt controller (LIC), and debugging registers in an Enhanced On-Chip Emulation (EOnCE) module and JTAG TAP controller. All the extended cores share an internal 476 KB level-2 memory (M2) and a general interrupt controller (GIC).

The external interfaces and peripherals include a system and local bus managed by a system interface unit (SIU) and memory controller, a 32/64-bit direct slave interface (DSI) port, four 256-channel TDM interfaces, a serial universal asynchronous receiver/transmitter (UART), timers, an Ethernet interface that can operate in any of three modes (MII, RMII, or SMII), an I^{2}C interface to allow booting from a serial EEPROM, and general-purpose input/output (GPIO) ports.The MSC8113 device is backward-compatible with the MSC8102; that is, it can replace a MSC8102 device and execute the same code with no modifications.

1.1 Features

The tables in this section list the features of the MSC8113 device.

Table 1-1. Extended SC140 Cores and Core Memories

Feature	Description
SC140 Core	Three SC140 cores: • Up to 4800 MMACS using 12 ALUs running at up to 400 MHz. • A total of 1196 KB of internal SRAM (224 KB per core). Each SC140 core provides the following: • Up to 1600 MMACS using an internal 400 MHz clock. A MAC operation includes a multiply-accumulate command with the associated data move and pointer update. • 4 ALUs per SC140 core. • 16 data registers, 40 bits each. • 27 address registers, 32 bits each. • Hardware support for fractional and integer data types. • Very rich 16-bit wide orthogonal instruction set. • Up to six instructions executed in a single clock cycle. • Variable-length execution set (VLES) that can be optimized for code density and performance. • **IEEE** Std. 1149.1 JTAG port. • Enhanced on-device emulation (EOnCE) with real-time debugging capabilities.
Extended Core	Each SC140 core is embedded within an extended core that provides the following: • 224 KB M1 memory that is accessed by the SC140 core with zero wait states. • Support for atomic accesses to the M1 memory. • 16 KB instruction cache, 16 ways. • A four-entry write buffer that frees the SC140 core from waiting for a write access to finish. • External cache support by asserting the global signal (GBL) when predefined memory banks are accessed. • Programmable Interrupt Controller (PIC). • Local Interrupt Controller (LIC).
Multi-Core Shared Memories	• M2 memory (shared memory): – A 476 KB memory working at the core frequency. – Accessible from the local bus – Accessible from all three SC140 cores using the MQBus. • 4 KB bootstrap ROM.
M2-Accessible Multi-Core Bus (MQBus)	• A QBus protocol multi-master bus connecting the three SC140 cores to the M2 memory. • Data bus access of up to 128-bit read and up to 64-bit write. • Operation at the SC140 core frequency. • A central efficient round-robin arbiter controlling SC140 core access on the MQBus. • Atomic operation control of access to M2 memory by the three SC140 cores and the local bus.

Table 1-2. Phase-Lock Loop (PLL)

Feature	Description
Internal PLL	• Generates up to 500 MHz core clock and up to 166 MHz bus clocks for the 60x-compatible local and system buses and other modules. • PLL values are determined at reset based on configuration signal values.

Table 1-3. Buses and Memory Controller

Feature	Description
System Bus	• 64/32-bit data and 32-bit address 60x bus. • Support for multiple-master designs. • Four-beat burst transfers (eight-beat in 32-bit wide mode). • Port size of 64, 32, 16, and 8 controlled by the internal memory controller. • Bus can access external memory expansion or off-device peripherals, or it can enable an external host device to access internal resources. • Slave support, direct access by an external host to internal resources including the M1 and M2 memories. • On-device arbitration between up to four master devices.
Direct Slave Interface (DSI)	A 32/64-bit wide slave host interface that operates only as a slave device under the control of an external host processor. • 21–25 bit address, 32/64-bit data. • Direct access by an external host to internal resources, including the M1 and the M2 memories as well as external devices on the system bus. • Synchronous and asynchronous accesses, with burst capability in the synchronous mode. • Dual or Single strobe modes. • Write and read buffers improve host bandwidth. • Byte enable signals enables 1, 2, 4, and 8 byte write access granularity. • Sliding window mode enables access with reduced number of address pins. • Chip ID decoding enables using one $\overline{\text{CS}}$ signal for multiple DSPs. • Broadcast $\overline{\text{CS}}$ signal enables parallel write to multiple DSPs. • Big-endian, little-endian, and munged little-endian support.
3-Mode Signal Multiplexing	• 64-bit DSI and 32-bit system bus. • 32-bit DSI and 64-bit system bus. • 32-bit DSI and 32-bit system bus.
Memory Controller	Flexible eight-bank memory controller: • Three user-programmable machines (UPMs), general-purpose chip-select machine (GPCM), and a page-mode SDRAM machine • Glueless interface to SRAM, page mode SDRAM, DRAM, EPROM, Flash memory, and other user-definable peripherals. • Byte enables for either 64-bit or 32-bit bus width mode. • Eight external memory banks (banks 0–7). Two additional memory banks (banks 9, 11) control IPBus peripherals and internal memories. Each bank has the following features: – 32-bit address decoding with programmable mask. – Variable block sizes (32 KB to 4 GB). – Selectable memory controller machine. – Two types of data errors check/correction: normal odd/even parity and read-modify-write (RMW) odd/even parity for single accesses. – Write-protection capability. – Control signal generation machine selection on a per-bank basis. – Support for internal or external masters on the system bus. – Data buffer controls activated on a per-bank basis. – Atomic operation. – RMW data parity check (on system bus only). – Extensive external memory-controller/bus-slave support. – Parity byte select pin, which enables a fast, glueless connection to RMW-parity devices (on system bus only). – Data pipeline to reduce data set-up time for synchronous devices.

Table 1-4. DMA Controller

Feature	Description
Multi-Channel DMA Controller	• 16 time-multiplexed unidirectional channels. • Services up to four external peripherals. • Supports $\overline{\text{DONE}}$ or $\overline{\text{DRACK}}$ protocol on two external peripherals. • Each channel group services 16 internal requests generated by eight internal FIFOs. Each FIFO generates: – A watermark request to indicate that the FIFO contains data for the DMA to empty and write to the destination. – A hungry request to indicate that the FIFO can accept more data. • Priority-based time-multiplexing between channels using 16 internal priority levels. • Round-robin time-multiplexing between channels. • A flexible channel configuration: – All channels support all features. – All channels connect to the system bus or local bus. • Flyby transfers in which a single data access is transferred directly from the source to the destination without using a DMA FIFO.

Table 1-5. Serial Interfaces

Feature	Description
Time-Division Multiplexing (TDM)	Up to four independent TDM modules, each with the following features: • Optional operating configurations: – Totally independent receive and transmit channels, each having one data line, one clock line, and one frame sync line – Four data lines with one clock and one frame sync shared among the transmit and receive lines. – Glueless interface to E1/T1 framers and switches, as well as to common buses, such as the ST-BUS. • Hardware A-law/µ-law conversion • Up to 62.5 Mbps per TDM (62.5 MHz bit clock if one data line is used, 31.25 MHz if two data lines are used, 15.625 MHz if four data lines are used). • Up to 256 channels. • Up to 16 MB per channel buffer (granularity 8 bytes), where A/µ law buffer size is double (granularity 16 byte). • Receive buffers share one global write offset pointer that is written to the same offset relative to their start address. • Transmit buffers share one global read offset pointer that is read from the same offset relative to their start address. • All channels share the same word size. • Two programmable receive and two programmable transmit threshold levels with interrupt generation that can be used, for example, to implement double buffering. • Each channel can be programmed to be active or inactive. • 2-, 4-, 8-, or 16-bit channels are stored in the internal memory as 2-, 4-, 8-, or 16-bit channels, respectively. • The TDM Transmitter Sync Signal (TxTSYN) can be configured as either input or output. • Frame Sync and Data signals can be programmed to be sampled either on the rising edge or on the falling edge of the clock. • Frame sync can be programmed as active low or active high. • Selectable delay (0–3 bits) between the Frame Sync signal and the beginning of the frame. • MSB or LSB first support.

Table 1-5. Serial Interfaces (Continued)

Feature	Description
UART	• Two signals for transmit data and receive data. • No clock, asynchronous mode. • Can be serviced either by the SC140 DSP cores or an external host on the system bus or the DSI. • Full-duplex operation. • Standard mark/space non-return-to-zero (NRZ) format. • 13-bit baud rate selection. • Programmable 8-bit or 9-bit data format. • Separately enabled transmitter and receiver. • Programmable transmitter output polarity. • Two receiver wake-up methods: – Idle line wake-up. – Address mark wake-up. • Separate receiver and transmitter interrupt requests. • Eight flags, the first five can generate interrupt request: – Transmitter empty. – Transmission complete. – Receiver full. – Idle receiver input. – Receiver overrun. – Noise error. – Framing error. – Parity error. • Receiver framing error detection. • Hardware parity checking. • 1/16 bit-time noise detection. • Maximum bit rate 6.25 Mbps. • Single-wire and loop operations.
General-Purpose I/O (GPIO) port	• 32 bidirectional signal lines that either serve the peripherals or act as programmable I/O ports. • Each port can be programmed separately to serve up to two dedicated peripherals, and each port supports open-drain output mode.
I^{2}C Software Module	• Supports booting from a serial EEPROM

Table 1-5. Serial Interfaces (Continued)

Feature	Description
Ethernet Controller	• Designed to comply with **IEEE** Std. 802.3™, 802.3u™, 802.3x™, and 802.3ac™ • Three Ethernet physical interfaces: – 10/100 Mbps **IEEE** Std. 802.3 MII. – 10/100 Mbps RMII. – 10/100 Mbps SMII. • Full and half-duplex support. • **IEEE** Std. 802.3® full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition). • Support of out-of-sequence transmit queue (for initiating flow-control). • Programmable maximum frame length supports jumbo frames (up to 9.6k) and **IEEE** Std. 802.1™ virtual local area network (VLAN) tags and priority. • Retransmission from transmit FIFO following a collision. • CRC generation and verification of inbound/outbound packets. • Address recognition: – Each exact match can be programmed to be accepted or rejected. – Broadcast address (accept/reject). – Exact match 48-bit individual (unicast) address. – Hash (256-bit hash) check of individual (unicast) addresses. – Hash (256-bit hash) check of group (multicast) addresses. – Promiscuous mode. • Pattern matching: – Up to 16 unique 4-byte patterns. – Pattern match on bit-basis. – Matching range up to 256 bytes deep into the frame. – Offsets to a maximum of 252 bytes. – Programmable pattern size in 4-byte increments up to 64 bytes. – Accept or reject frames if a match is detected. – Up to eight unicast addresses for exact matches. – Pattern matching accepts/rejects IP addresses. • Filing of receive frames based on pattern match; prioritization of frames. • Insertion with expansion or replacement for transmit frames; VLAN tag insertion. • RMON statistics. • Master DMA on the local bus for fetching descriptors and accessing the buffers. • Ethernet PHY can be exposed either on GPIO pins or on the high ms bits of the DSI/system when the DSI and the system bus are both 32 bits. • MPC8260(PQ2) 8 byte width buffer descriptor mode as well as 32 byte width buffer descriptor mode. • MII Bridge (MIIGSK): – Programmable selection of the 50 MHz RMII reference clock source (external or internal). – Independent 2 bit wide transmit and receive data paths. – Six operating modes. – Four general-purpose control signals. – Programmable transmitted inter-frame bits to support inter-frame gap for frames in the SMII domain. • SMII features: – Convey complete MII information between the PHY and MAC. – Allow direct MAC-to-MAC communication in SMII mode. – Can generate an interrupt request line while receiving inter-frame segments.

Table 1-6. Miscellaneous Modules

Feature	Description
Timers	Two modules of 16 timers each. Each timer has the following features: • Cyclic or one-shot. • Input clock polarity control. • Interrupt request when counting reaches a programmed threshold. • Pulse or level interrupts. • Dynamically updated programmed threshold. • Read counter any time. Watchdog mode for the timers that connect to the device.
Hardware Semaphores	Eight coded hardware semaphores, locked by simple write access without need for read-modify-write mechanism.
Global Interrupt Controller (GIC)	• Consolidation of chip maskable interrupt and non-maskable interrupt sources and routing to $\overline{\text{INT\_OUT}}$, $\overline{\text{NMI\_OUT}}$, and to the cores. • Generation of 32 virtual interrupts (eight to each SC140 core) by a simple write access. • Generation of virtual $\overline{\text{NMI}}$ (one to each SC140 core) by a simple write access.

Table 1-7. Power and Packaging

Feature	Description
Reduced Power Dissipation	• Low power CMOS design. • Separate power supply for internal logic and I/O. • Low-power standby modes. • Optimized power management circuitry (instruction-dependent, peripheral-dependent, and mode-dependent).
Packaging	• 0.8 mm pitch High Temperature Coefficient for Expansion Flip-Chip Ceramic Ball-Grid Array (CBGA (HCTE)). • 431-connection (ball). • 20 mm × 20 mm.

Table 1-8. Software Support

Feature	Description
Real-Time Operating System (RTOS)	The Real-Time Operating System (RTOS) fully supports device architecture (multi-core, memory hierarchy, ICache, timers, DMA controller, interrupts, peripherals), as follows: • High-performance and deterministic, delivering predictive response time. • Optimized to provide low interrupt latency with high data throughput. • Preemptive and priority-based multitasking. • Fully interrupt/event driven. • Small memory footprint. • Comprehensive set of APIs.
Multi-Core Support	• Use of one instance of kernel code in all three SC140 cores. • Dynamic and static memory allocation from local memory (M1) and shared memory (M2).
Distributed System Support	Enables transparent inter-task communications between tasks running inside the SC140 cores and the other tasks running in on-board devices or remote network devices: • Messaging mechanism between tasks using mailboxes and semaphores. • Networking support; data transfer between tasks running inside and outside the device using networking protocols. • Includes integrated device drivers for such peripherals as TDM, UART, and external buses.

Table 1-8. Software Support (Continued)

Feature	Description
Additional Features	• Incorporates task debugging utilities integrated with compilers and vendors. • Board support package (BSP) for the application development system (ADS). • Integrated Development Environment (IDE): – C/C++ compiler with in-line assembly. Enables the developer to generate highly optimized DSP code. It translates code written in C/C++ into parallel fetch sets and maintains high code density. – Librarian. Enables the user to create libraries for modularity. – C libraries. A collection of C/C++ functions for the developer's use. – Linker. Highly efficient linker to produce executables from object code. – Debugger. Seamlessly integrated real-time, non-intrusive multi-mode debugger that enables debugging of highly optimized DSP algorithms. The developer can choose to debug in source code, assembly code, or mixed mode. – Simulator. Device simulation models, enables design and simulation before the hardware arrival. – Profiler. An analysis tool using a patented Binary Code Instrumentation (BCI) technique that enables the developer to identify program design inefficiencies. – Version control. CodeWarrior™ includes plug-ins for ClearCase, Visual SourceSafe, and CVS.
Boot Options	• External memory. • External host. • UART. • TDM. • I^2C

Table 1-9. Application Development System (ADS) Board

Feature	Description
MSC8113ADS	• Host debug through single JTAG connector supports both processors. • MSC8103 as the MSC8113 host with both devices on the board. The MSC8103 system bus connects to the MSC8113 DSI. • Flash memory for stand-alone applications. • Communications ports: – 10/100Base-T. – 155 Mbps ATM over Optical. – T1/E1 TDM interface. – H.110. – Voice codec. – RS-232. – High-density (MICTOR) logic analyzer connectors to monitor MSC8113 signals – 6U cPCI form factor. • Emulates MSC8113 DSP farm by connecting to three other ADS boards.

1.2 Architecture

Figure 1-1 shows the MSC8113 block diagram. Note that the arrows on the buses describe the direction of the address flow; an arrow points from the master of the bus to the slave(s).

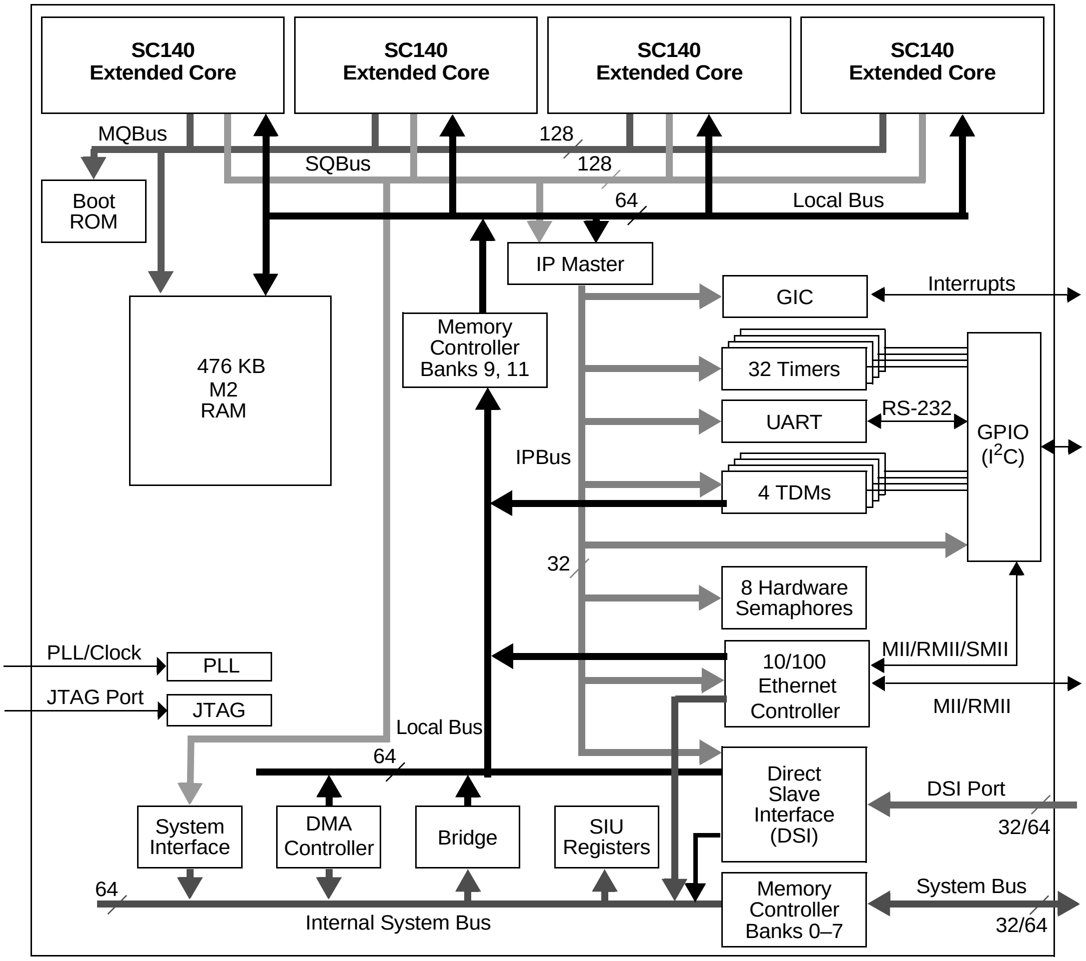


Figure 1-1. MSC8113 Block Diagram

Data is transferred to the MSC8113 device from either the system bus port, the DSI, the Ethernet, the TDM, the Ethernet interface. The SC140 core processes the data in the buffers and the result is transferred back to one of the ports.

The MSC8113 architecture is optimized so that applications can efficiently use the available 4800 MMACS for the three SC140 cores. For most applications:

- The data is accessed for a bounded number of times while the critical code is run in loops for many cycles. DSP applications have a high degree of code locality and a low degree of data locality.
- Different channels can share code but do not share data.
- A small portion of the code is run for most of the time (the “20–80” rule).

Since the instructions can be shared, a typical application stores them in the shared memory, M2. Since each DSP core typically spends most of the time running loops of selected routines, these routines can be stored either in the local M1 memory or automatically fetched to the local cache. Achieving high hit ratios on the cache prevents core stalls and thus boosts overall performance. During a miss, instructions are fetched from the M2 memory through the MQBus. Since the miss ratio is very low, the probability of a collision with another SC140 core on the MQBus is low. Therefore, the overall fetch latency is low. Since different channels do not typically share data, the data can be located in the local M1 memory. The architecture is flexible enough to enable storage of data in M2 as well. In fact, the powerful DMA can perform data overlays between the M2 and the M1 memories or between the M1 memory of one SC140 core to the M1 memory of another SC140 core. For example, while performing channel N, the DMA controller can bring in the data needed for channel N+1. To achieve the best transfer rate, these DMA transfers can be programmed as flyby transfers, also called “single access transactions.” For a flyby transfer, the data path is between a peripheral and memory with the same port size, located on the same bus. Flyby transactions can occur only between external peripherals and external memories located on the system bus, between internal peripherals and internal SRAM located on the local bus, and between internal memories.

The SC140 core accesses the M2 memory through the MQBus. All accesses to other internal peripherals and accesses external to the MSC8113 occur on a separate bus, the SQBus. This separation ensures that the latencies for SC140 core accesses to the M2 memory remain as low as possible. Write accesses with high latencies are typically routed through the write buffer. The write buffer can store the write access, release the SC140 core, and execute it at a later time.

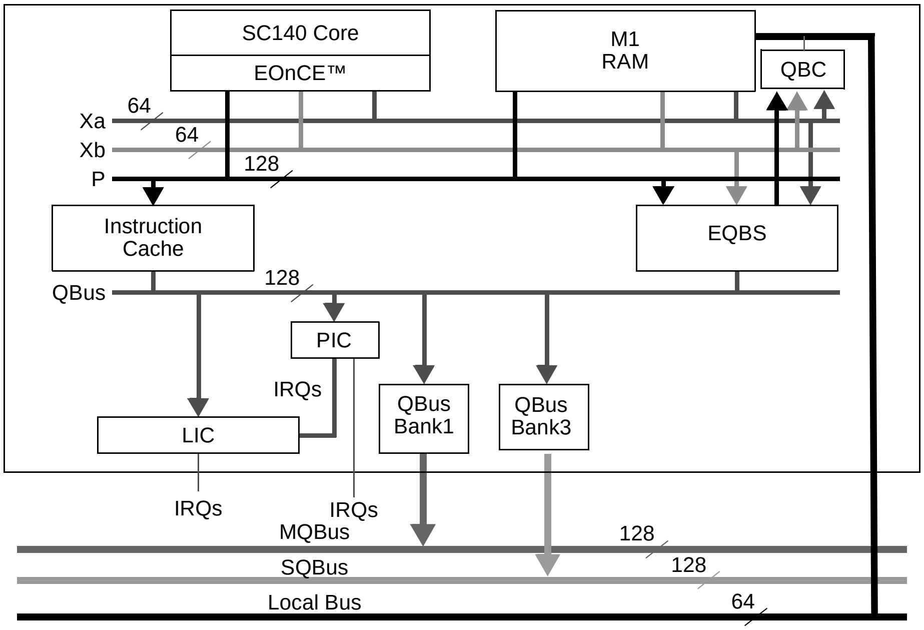
The SC140 cores should be focused on the intensive computational work and should not have to deal with bringing new data buffers. Data can be prepared in the M1 (or M2) memory in a “next” buffer while the SC140 core processes the ‘current’ buffer. The SC140 core can use the flexible DMA controller to transfer large blocks of data from the external memory to the internal memory and also between the internal memories. In some applications, data is written from an external host directly to the MSC8113 M1 and M2 memories through the DSI interface while the SC140 handles the computational work in parallel.

Note: For details on the SC140 core, see **Chapter 2**, *SC140 Core Overview*.

1.2.1 Extended Core

The extended core contains the SC140 core, its M1 memory, an instruction cache, a write buffer, a programmable interrupt controller (PIC), a local interrupt controller (LIC) and interfaces to the MQBus and the SQBus through which accesses are performed to addresses outside the extended core. See **Figure 1-2**.

Note: Details on extended core functionality are in **Chapter 9**, *Extended Core*.



Notes: 1. The arrows show the data transfer direction.
2. The QBus interface includes a bus switch, write buffer, fetch unit, and a controller that defines four QBus banks.

Figure 1-2. MSC8113 SC140 Extended Core

1.2.1.1 SC140 Core

The SC140 core is a flexible, programmable DSP core that handles compute-intensive communications applications, providing high performance, low power, and code density. It efficiently deploys a novel variable-length execution set (VLES), attaining maximum parallelism by allowing multiple address generation and data arithmetic logic units to execute multiple operations in a single clock cycle. The SC140 core contains four ALU units, each with a 16-bit × 16-bit MAC that results in a 40-bit wide and 40-bit parallel barrel shifter. Each ALU performs one MAC operation per clock cycle, so a single core running at 400 MHz can perform 1600 MMACS. Having three such cores, the MSC8113 can perform up to 4800 MMACS per second. An address generation unit includes two address arithmetic units and one bit mask unit. There are also 16 address registers, of which eight can serve as base address registers.

The main reason for the high code density of the SC140 is that all instructions are 16 bits wide. During each clock cycle, the SC140 core reads eight instruction words, referred to as a *fetch set*. The SC140 core identifies which instructions can be performed in parallel and runs them on the ALUs and address generation units. In one clock cycle, up to six instructions, four ALU operations, and two address generation operations can be performed. In the rich instruction set, special attention is given to control code, making the SC140 core ideal for applications embedding DSP and communications. Arithmetic operations are performed using both fractional and integer data types, enabling the user to choose a style of code development or use coding techniques derived from an application-specific standard. The programming model of the SC140 core is highly orthogonal, and both data and instructions reside in one unified memory. The powerful SC140 compiler translates code written in C/C++ into parallel fetch sets and maintains high code density and/or high performance by taking advantage of the core high code orthogonality and unified memory architecture. For details, see the *SC140 DSP Core Reference Manual*, MNSC140CORERM/D.

Note: For details, see **Chapter 2**, *SC140 Core Overview*.

1.2.1.2 M1 Memory

The 224 KB M1 memory can be accessed with zero wait states from the SC140 core. Three accesses are performed concurrently on every SC140 core clock cycle. The SC140 core accesses one 128-bit instruction fetch set and two 64-bit data words. In addition, an external host or the DMA controller can access a 64-bit word through the local bus at the bus clock rate. To reduce the size of the memory, M1 is a single-access memory and is hierarchically divided so that four accesses can be performed in parallel. An intelligent memory allocation significantly decreases the probability of collisions between an SC140 core bus and the DMA bus. For example, two accesses cannot collide if they belong to different 32 KB memory groups, which is usually the case since program code is stored in a different group than the data space of the program. The DMA stores the “next” buffers in yet a different group. Even in the same group, if two data elements are placed on a different 4 KB module, a collision between two SC140 core buses is prevented. When a collision does occur, the SC140 core stalls for one clock cycle. The overall memory size available for one SC140 core in both M1 and M2 memories and the partition between the memories is carefully designed as a trade-off between chip size and the memory requirements imposed by the bandwidth of the SC140 core. Typically the M1 memory contains critical routines and most of the channel data.

Note: For details, see **Section 9.2**, *Extended Core Memory (M1)*, on page 9-3.

1.2.1.3 Instruction Cache

The instruction cache is highly optimized for real-time DSP applications and minimizes miss ratios, latencies, bus bandwidth requirements, and silicon area. The 16 KB instruction cache is 16-way set associative. **Figure 1-3** illustrates its logic structure and demonstrates how an address is mapped to this structure. Each of the 16 ways contains four 256-byte lines and is divided into 16 fetch sets, each with an associated valid bit. The 2-bit index field of the address serves as an index to the line within the way. The line whose tag matches the tag field of the address is the selected line.

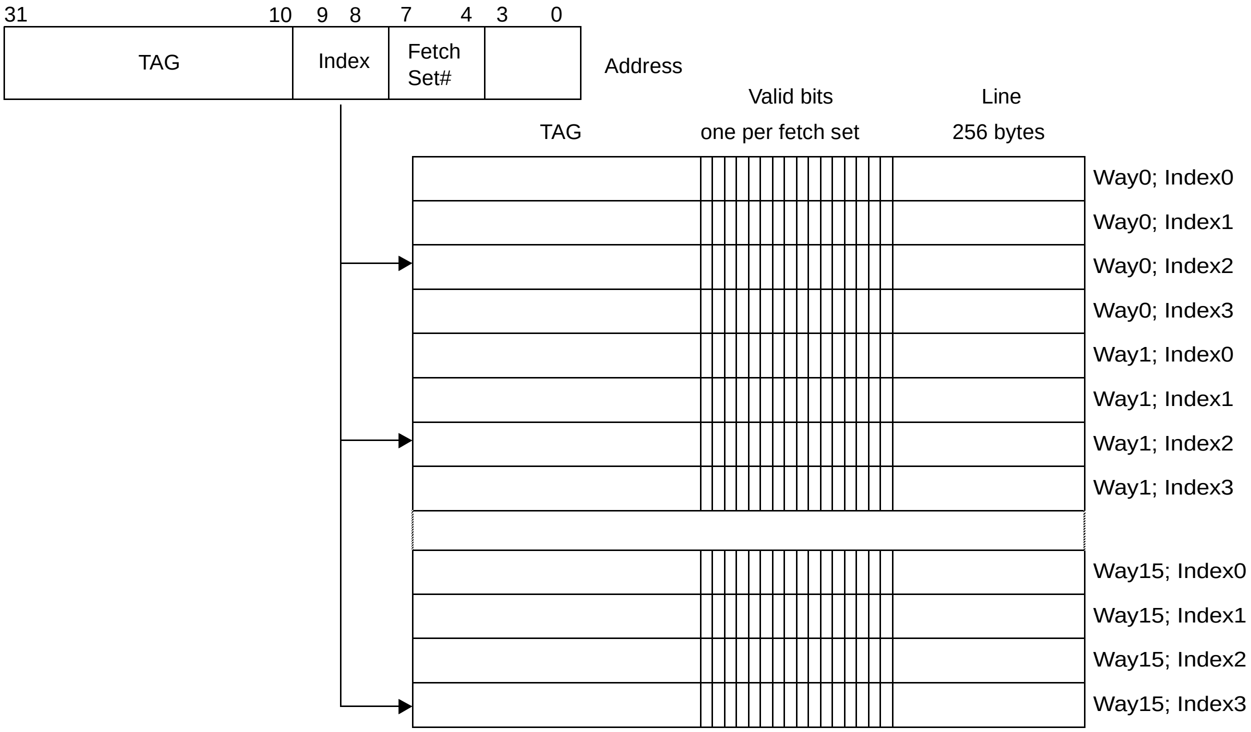


Figure 1-3. Mapping an Address to the Instruction Cache

When a cache miss occurs, the new data is fetched in bursts of 1, 2, or 4 fetch sets. There is also an option to fetch until the end of the line. This option, referred to as pre-fetch, takes advantage of the spatial locality of the code. When there is a need to fetch new data to the cache and the cache is full, one of the lines of the cache is thrashed using the least recently used (LRU) algorithm. The cache can be programmed so that only part of it is thrashed. For example, suppose task A needs to be preempted in favor of task B. While task B runs, the instructions of task A are thrashed from the instruction cache. When task B finishes and task A takes over, task A may not find its most recently used instructions in the cache. To prevent such a situation and thus keep task A’s most recently used instructions in the cache, the operating system can exclude the ways of task A from the part of the cache that can be thrashed. Another method of guaranteeing that the critical routines are always available for a task is to store them in the SC140 core M1 private memory. All the cache entries are flushed by issuing a cache flush command from the SC140 core, which is useful, for example, when new code is written to lines in the M2 memory that are already cached. The ICache has run-time debug support. A counter in the Emulation and Debug (EOnCE) module is incremented for cache hits and misses. When the SC140 core is in Debug mode, its fetch unit is in Debug mode and all the cache arrays can be read.

Note: For details, see **Section 9.4**, *Instruction Cache (ICache)*.

1.2.1.4 QBus System

The QBC is a bus controller that handles internal memory contentions. It snoops the activity on the buses connected to the internal memory and freezes the SC140 core and address bus activity. It creates the atomic instruction acknowledge to the SC140 core during the reservation process. The EQBS enables the SC140 core to communicate with external devices efficiently. It handles the switching between the three core buses and the QBus. SC140 core accesses that apply to memory space above the internal memory (QBus Base Line = 0x00F00000) are transferred to the QBus through the EQBS. The EQBS also connects to the instruction cache and initiates requests for cache updates in order to improve the hit ratio. The EQBS operates at the same frequency as the SC140 core. The module handles the SC140 core and the instruction cache requests, bringing the data on the QBus. As **Figure 1-4** shows, the EQBS consists of a bus switch to handle data read operations, a write buffer to handle data write operations, a fetch unit to handle all program read operations, a control unit, and the banks to handle the communication with the slaves and all EQBS registers.

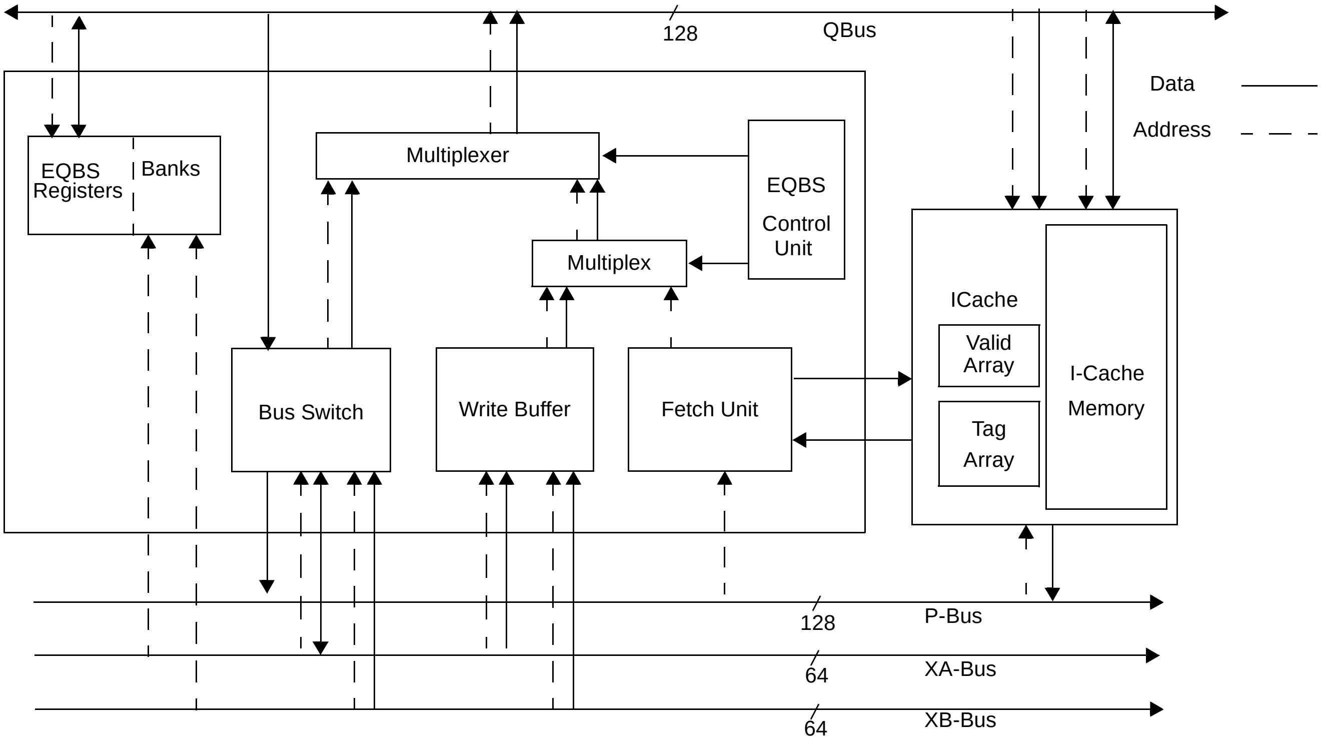


Figure 1-4. EQBS Block Diagram

Note: For details, see **Section 9.4**, *Instruction Cache (ICache)*, on page 9-24.

The QBus masters are the fetch unit, write buffer, and bus switch. The control unit is the arbiter for the QBus masters.The bus switch handles all data read above the QBus baseline, write operations when the write buffer is disabled, and atomic (read modify write) write operations. Read accesses of program (fetch) that are above the QBus baseline occur through the fetch unit.

This unit is triggered by a cache “miss” access. It brings the data into the SC140 core and continues to update the cache until the end of the cache line or until a new “miss” is accepted. This improves the overall performance of the cache in the system. The fetch unit initiates cache update requests for data of consecutive addresses after every “miss.” The block and burst sizes are configurable. When the SC140 core writes to an external address (that is, to an address beyond the M1 memory), it can stall while waiting for the access to complete. To prevent such stalls, all the external accesses are first written to a write buffer. The write buffer releases the SC140 core and then completes the access. Located on the SC140 core buses, the write buffer is a zero wait state client and thus boosts the SC140 core performance. The write buffer is a four-entry FIFO that automatically handles data coherency problems. For example, if the write buffer contains data to be written to address A, and a read access occurs before the buffer completes the write access, the contents of the write buffer are written to the destination before the read can be executed. Not all writes beyond the M1 memory are routed through the write buffer. Write accesses do not use the write buffer in the following cases:

- The address of the destination belongs to a bank that is defined as immediate.
- It is an atomic operation essentially writing to a semaphore.
- The write buffer is disabled.

The write buffer counts the number of clocks that elapse between the time data is written to the write buffer and the time it is emptied. When the counter exceeds a pre-programmed value, the contents of the write buffer are flushed so that the time for the write accesses through the write buffer can be limited.

1.2.2 Power Saving Modes

The MSC8113 device is a low-power CMOS design. Also, you can put unused modules into power saving modes. The TDM, DSI, timers, GPIO, GIC, UART, and Ethernet controller can be put into Stop mode, in which their clocks are frozen as described in **Section 19.10**, *Stop Options*. Each of the extended cores can be put into either Stop or Wait mode.

1.2.2.1 Extended Core Wait Mode

An extended core enters Wait mode when it issues the **wait** instruction. In Wait mode, the SC140 core consumes minimal power because its clocks are frozen. The clocks of other modules inside the extended core do not stop, so the QBus, PIC, LIC, and MQBus and SQBus controllers are functional. The extended core exits Wait mode when there is an interrupt or a reset or when the MSC8113 device enters Debug mode by either a JTAG DEBUG_REQUEST command or assertion of EE0.

Note: When multiple cores are in Wait mode, issuing a simultaneous virtual interrupt to all these cores does not guarantee that the cores exit Wait mode on the same clock cycle.

1.2.2.2 Extended Core Stop Mode

An extended core enters Stop mode when it issues the **stop** instruction. In Stop mode, the SC140 core and all the extended core peripherals except for the MQBus and SQBus controllers consume minimal power because their clocks are frozen. The extended core exits this mode when reset.

1.2.3 M2 Memory

The M2 is a 476 KB RAM and 4 KB ROM that is shared between the three SC140 cores. Up to 128 bits of data are accessed at up to 500 MHz. Each SC140 core treats the M2 as its secondary memory. Only one SC140 core can access this memory at a given time. When an SC140 core needs to access this memory, it arbitrates on the MQBus, and when access is granted it performs the access. The DMA controller or an external host can also access the M2 memory through the local bus. Enabling the DMA controller or an external host to write program and data directly to the M2 alleviates the load on the SC140 cores and keeps their focus on the intensive DSP processing. In a typical application that carefully considers memory allocation and uses the cache wisely, fewer SC140 core accesses occur to the M2 memory.

Note: For details, see **Section 8.4**, *SQBus Address Space*.

1.2.4 System Interface Unit (SIU)

The SIU is based on the MPC8260 SIU and is similar to the one used in the MSC8101. This unit controls the system bus and the internal local bus. It contains a flexible memory controller for accessing various memory devices both internally and externally. The SIU also controls the system start-up and initialization as well as operation and protection.

Note: For details, see **Chapter 4**, *System Interface Unit (SIU)*.

1.2.4.1 60x-Compatible System Bus Interface

The system bus interface can function as a master in a multi-master environment. It runs at up to 166 MHz and supports 32-bit addressing, a 32/64-bit data bus, and burst operations that transfer up to 256 bits of data per burst. The 60x-compatible data bus is accessible in 8-bit, 16-bit, 32-bit, and 64-bit data widths. In 32-bit mode, the system bus supports accesses of 1–4 bytes, aligned or unaligned, on 4-byte (word) boundaries and 1–8 bytes, aligned or unaligned on 8-byte (double word) boundaries. The address and data buses support synchronous, one-level pipelined transactions and non-pipelined SRAM-like accesses. Various applications can use this bus interface—for example, a system in which the MSC8113 uses a shared external memory. An external host can directly access the device internal memories and peripherals because the system bus is bridged to the internal local bus where the memories and peripherals are located. At reset, the system bus can also be configured in a single bus master mode so that it can connect gluelessly to slaves, typically memory devices, using only the memory controller. This mode is useful when the SC140 cores use an external memory private to the MSC8113 device.

Note: For details, see **Chapter 13**, *System Bus*.

1.2.4.2 Memory Controller

The memory controller controls up to 12 memory banks, four of which access internal modules (two of them reserved) and eight of which access external devices. These memory banks are shared by a high-performance SDRAM machine, a general-purpose chip-select machine (GPCM), and three user-programmable machines (UPMs). Internally, Bank 9 is assigned to the IPBus peripherals (four TDMs, DSI, UART, GPIO, GIC, HS, Ethernet controller, and Ethernet controller), and Bank 11 is assigned to the internal M1 and M2 memories. The memory controller supports a glueless interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. It allows the implementation of memory systems with very specific timing requirements. The SDRAM machine provides an interface to synchronous DRAMs using SDRAM pipelining, bank interleaving, and back-to-back page mode to achieve the highest performance. The GPCM provides interfacing for simpler, slow memory resources and memory-mapped devices. GPCM performance is inherently lower than that of the SDRAM machine because it does not support bursting. Therefore, GPCM-controlled banks are mainly used for boot-loading and access to low-performance memory-mapped peripherals. The UPMs support address multiplexing of the system bus and refresh timers as well as generation of programmable control signals for row address and column address strobes, providing a glueless interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The refresh timers allow refresh cycles to be initiated. The UPM can generate different timing patterns for the control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access request. Also, refresh timers can periodically generate user-defined refresh cycles.

Note: For details, see**Chapter 12**, *Memory Controller*.

1.2.5 Direct Slave Interface (DSI)

The DSI gives an external host direct access to the MSC8113 internal and external memory space, including internal memories and the registers of internal modules as well as access to the system bus. When a 21-bit address is used, the DSI can access all the internal 2 MB address space as well as the system bus through a 32 KB sliding window. When more address bits (between 22 to 25 bit address) are used, the DSI can directly access the system bus. The DSI data bus is 32/64-bit wide and provides the following slave interfaces to an external host:

- Asynchronous interface (no clock reference) enabling the host single accesses.
- Synchronous interface enabling host single or burst accesses of 256 bits (8 accesses of 32 bits or 4 accesses of 64 bits) with its external clock de-coupled from the internal bus clock.

Supporting various interfaces flexibly, the DSI interfaces to most available processors in the market. The DSI write buffer stores the address and the data of the accesses until they are performed. The external host can therefore perform multiple writes without waiting for those accesses to complete. Latencies that are typical during accesses to internal memories are greatly reduced by the DSI read prefetch mechanism. An external host addresses up to 16 MSC8113 devices using a single chip-select by which the most significant bits on the address bus identify the addressed MSC8113 device. The host can also write the same data to multiple MSC8113 devices simultaneously by asserting a dedicated broadcast chip select. This can be proved useful during boot from DSI and also for 3G uplink processing where the same chip rate data is being processed differently in each and every MSC8113 slave device.

Note: For details, see **Chapter 14**, *Direct Slave Interface (DSI)*.

1.2.6 Direct Memory Access (DMA) Controller

The multi-channel DMA controller connects to both the system bus and the local bus. Transfers on both buses can be performed in parallel. Transfers that occur on the same bus between clients with the same port size can use the flyby mode on which the data is read and written in the same cycle. Other transfers are dual accesses that occur in two phases so that data is first read to a DMA internal FIFO and then written from that FIFO. Eight internal DMA FIFOs support this mode. Flyby mode is used for data transfers between internal memories, all residing on the internal local bus. **Table 1-10** lists the possible DMA transfers.

Table 1-10. DMA Transfers

DMA client	DMA client	Flyby	Through FIFO in the DMA
Internal memory	Internal memory[1].	+[2]	+
Internal memory	Memory device on the system bus.	—	+
Internal memory	Peripheral on the system bus.	—	+
Memory device on the system bus	Peripheral on the system bus.	Only if on the same bus with the same port size.	+[3]
Memory device on the system bus	Memory device on the system bus.	—	+
Peripheral on the system bus	Peripheral on the system bus.	—	+
Notes: **1.** Source internal memory is different than the destination internal memory. **2.** Using start-address + a counter on the M1 memory. **3.** Not recommended if Flyby is possible.			

The DMA controller receives requests from the following clients:

- 8 requests from each flyby counter of each SC140 core.
- 4 requests from clients external to the MSC8113 device.

DMA requests are tied to up to 16 DMA channels that run concurrently. Each channel is programmed as either flyby or dual-access. The arbitration algorithm can be priority-based using 16 priority levels or round-robin-based. All clients connect to the DMA controller through the DREQ and DACK signals. A client uses the DREQ signal to request a DMA data transfer. This signal can be either level or edge. The DMA controller asserts the DACK signal to perform the data access. The DMA controller asserts DRACK to indicate that it has sampled the peripheral request. The bidirectional DONE signal indicates that the channel must be terminated. The DMA supports a flexible buffer configuration, including: simple buffers, cyclic buffers, single-address buffers (I/O device), incremental address buffers, chained buffers, and complex buffers by hardware.

Note: For details, see **Chapter 16**, *Direct Memory Access (DMA) Controller*.

1.2.7 Internal and External Bus Architecture

The SC140 cores and other MSC8113 modules interconnect via a variety of bus and interface structures that provide great flexibility for transferring and storing data both within the MSC8113 device and with external devices. The internal bus structures include the following:

- *SC140 core buses*. Each SC140 core can access its own M1 memory, ICache, and the write buffer with zero wait states using its internal 128-bit instruction bus and two 64-bit data buses. These buses include:
  - 32-bit program address bus (PAB) that allows the SC140 core to specify program addresses in the local unified memory (M1).
  - 128-bit program data bus (PDB) that transfers the program data to and from M1 or the ICache.
  - Two 32-bit address buses (XABA and XABB) to specify data locations in M1 for the two data streams required for DSP operations.
  - Two 64-bit data buses (XDBA and XDBB) to transfer data values to and from M1.
- *QBus*. A 128-bit wide, single-master, multi-slave bus within each extended core. The SC140 core is the master and all other modules on the bus are slaves: LIC, PIC, and QBus memory controller. The QBus memory controller directs QBus accesses by the slave devices and interfaces the MQBus and SQBus. It includes a Fetch Unit that moves program code into the ICache when required and a four-entry write buffer so that the SC140 core does not have to wait for a write access to finish before continuing instruction processing. When an SC140 core accesses an address beyond a programmable address referred to as the QBus base line, this access is forwarded to the QBus. The PIC is a slave on this bus and needs zero QBus wait states for an access. Accesses to the same bank can be pipelined.
- *MQBus*. A 128-bit wide bus that connects all the extended cores to the internal shared memory (M2) and Boot ROM. Each core accesses the MQBus through its own QBus Bank1. The SC140 core requires low latencies when it access the M2 memory. To minimize latencies when the M2 is accessed, M2 accesses occur on a dedicated separate

bus called the MQBus. The boot ROM can be accessed through the MQBus. The MQBus is a multi-master bus whose masters are the three SC140 cores and whose sole slave is the M2 memory. This bus can transfer up to 128 bits at 500 MHz.

— Data bus access of up to 128-bit read and up to 64-bit write.
— Operation at the SC140 core frequency.
— A central efficient round-robin arbiter controlling SC140 core access on the MQBus.
— Atomic operation control of access to M2 memory by the three SC140 cores and the local bus.

- *SQBus*. A 128-bit wide, multi-master, multi-slave bus that connects the SC140 cores to the system interface unit (SIU) and the IP master module. Each SC140 core accesses the SQBus through its own QBus Bank 3. The SC140 core does not require reduced latency when accessing the typically slower peripherals. The SC140 core access to other peripherals, including the system and the local interfaces, occurs through the SQBus. The SQBus is a multi-master bus whose masters are the three SC140 cores. This bus accesses either the external memory or other slaves through the IPBus.
- *IPBus*. A 32-bit wide bus controlled by an IP master that connects the local bus and SQBus to some system peripherals. This one-master multi-slave bus runs at up to 166 MHz and enables access to the control and the status registers of the DSI, the TDM interface, the ethernet controller, the timers, the UART, the hardware semaphores, the virtual interrupt registers, and the GPIOs. Either an external host or an SC140 core accesses the clients on this bus as follows:

— An SC140 core accesses the IPBus through the SQBus.
— An external host on the DSI accesses the IPBus through the IP master from the local bus.
— An external host on the system bus accesses the IPBus through the bridge and the IP master.

- *DSI*. A 32/64-bit slave host interface to connect an external host processor.

— 21-25 bit address, 32/64-bit data.
— Direct access by an external host to internal and external resources, including the M1 and the M2 memories as well as the system bus.
— Synchronous and asynchronous accesses, with burst capability in the synchronous mode.
— Dual or Single strobe modes.
— Write and read buffers improve host bandwidth.
— Byte enable signals enable 1-, 2-, 4-, and 8-byte write access granularity.
— Sliding Window mode enables accesses with a reduced number of address lines.
— Chip ID decoding enables the use of one $\overline{\text{CS}}$ signal for multiple DSPs.
— Broadcast $\overline{\text{CS}}$ signal enables parallel writes to multiple DSPs.
— Big-endian, little-endian, and munged little-endian support.

- *Local bus*. A 64-bit wide bus connecting the Ethernet Controller, TDM, DSI, DMA controller, and the local-to-system bus bridge to each other. The extended cores also share this bus, which connects to the M1 memory of each SC140 core. This multi-master multi-slave bus runs at 166 MHz. The DSI, TDM, DMA controller, and the bridge are the masters of this bus. The internal memories are slaves to this bus, which is primarily used for transferring data between the chip interfaces to the internal memories.
- *System bus*. A 64-bit wide bus controlled by the SIU that connects the DMA controller and system interface (from the SQBus) through the SIU memory controller to the external 32/64 bit system bus. It also connects to the local-to-system bus bridge to allow data transfers between the 60x-compatible local and internal system buses.
  - 64/32-bit data and 32-bit address bus.
  - Support for multiple-master designs.
  - Four-beat burst transfers (eight-beat in 32-bit wide mode).
  - Port size of 64, 32, 16, and 8 controlled by the internal memory controller.
  - Bus accesses external memory or peripherals, or an external host device uses it to access internal resources.
  - Slave support, direct access by an external host to internal resources including the M1 and M2 memories.
  - Internal arbitration between up to four master devices.
- The external buses can be configured during reset in three modes:
  - 64-bit data DSI and 32-bit data system bus.
  - 32-bit data DSI and 64-bit data system bus.
  - 32-bit data DSI, 32 bit data system, and Ethernet MII or RMII.

1.2.8 TDM Serial Interface

The TDM interface connects gluelessly to common telecommunication framing schemes, such as T1 and E1 lines. It can also connect to multiple framers and switches, as well as to commons buses such as the ST-BUS. The TDM contains four identical and independent engines. Each TDM engine can be configured in one of the following options:

- Two independent receive and transmit links.
  - The transmit has an input clock of up to 50 MHz, output data, and a frame sync that is configured as either input or output. Up to 256 transmit channels are supported.
  - The receive has an input clock of up to 50 MHz, input data, and an input frame sync. Up to 256 receive channels are supported.
- Two receive and two transmit links share the clock and the frame sync. The input clock runs up to 25 MHz, and the sync is configured as either input or output. Each of the two receive links supports up to 128 channels, and each transmit link supports up to 128 channels.

- One receive and one transmit link share the clock and the frame sync. The input clock runs at up to 50 MHz, and the sync is configured as either input or output. There are up to 256 channels for the receive link and up to 256 channels for the transmit link.

Each channel can be 2, 4, 8, or 16 bits wide. When the slot size is 8 bits wide, selected channels can be defined as A-law/μ-law. These channels are converted to 13–14 bits, which are padded into 16 bits and stored as such in memory. Each receive channel and each transmit channel can be active or not. An active channel has a buffer that is placed into the M1 and M2 memories or into memory devices on the system bus. All the buffers belonging to one TDM interface have the same size and are filled/emptied at the same rate. A-law/μ-law buffers are filled at twice the rate, so their buffer size is twice that of the non A-law/μ-law channels.

For receives, all the buffers belonging to a specific TDM interface fill at the same rate and therefore share the same write pointer relative to the beginning of the buffer. When the write pointer reaches a pre-determined threshold, an interrupt to the SC140 core is generated. The SC140 core empties the buffers while the TDM continues to fill the buffers until a second threshold line is reached, and an interrupt is generated to the SC140 core. The SC140 core empties the data between the first and the second threshold lines. Both the first and the second threshold lines are programmable. Using threshold lines, the SC140 core and the TDM can implement a double buffer handshake. In addition, the TDM generates an interrupt on frame start to the SC140 core, which helps synchronize to the TDM system.

For transmits, the SC140 core fills all the buffers belonging to a specific TDM interface, and the TDM empties them. A similar method employing two threshold line interrupts is used for a double-buffer handshake between the SC140 core and the TDM.

Note: For details, see **Chapter 20**, *TDM Interface*.

1.2.9 Ethernet Controller

The Ethernet controller complies with the **IEEE** Std. 802.3 standard and supports 10Mb/s and 100Mb/s operation as a media-independent interface (MII), a reduced media-independent interface (RMII), or a serial media-independent interface (SMII). The Ethernet controller works with minimal SC140 core intervention and operates in two modes:

- Full Duplex mode, for connecting the Ethernet to an on-board Ethernet switch.
- Half-Duplex mode, for connecting the Ethernet to an on-board physical layer (PHY).

On the receive side, the SC140 core prepares empty buffers and points to these buffers through up to four rings residing in memory (for example M2). The Ethernet controller reads the descriptors, fills their associated buffers with the received buffers, and interrupts the SC140 core when done. On the transmit side, the SC140 core prepares buffers in memory and prepares a descriptor ring that points to those buffers. The Ethernet controller reads these descriptors, reads the data from the buffers, and sends the data over the Ethernet. Enhanced pattern matching enables you to process received frames with a wide variety of tools to assist network applications. Features such as extraction of data and filing of frames in queues based on a pattern hit can accelerate post processing of data. It can further enhance the address recognition process by applying additional filtering to frames that pass the destination address check. Flexibility is built into pattern matching architecture to give you more control in manipulating receive frames. When the receiver detects the first bytes of a frame, the Ethernet controller begins to perform the frame recognition functions. It first tries to "filter" the frames based on matching a pattern in the frame, and if it fails it filters according to the MAC address. Pattern matching is performed using user-selected patterns of flexible length, up to 256 bytes into the frame. For example, if four patterns of 16 bytes are used, incoming frames can be filtered to four destination queues, with each queue dedicated to one DSP subsystem. Frames can be filtered not only by their MAC address but also by their IPv4 or IPv6 address and even their UDP port number. Based on these matches, the frame is accepted or rejected. Once a frame is accepted, the Ethernet controller processes it on the basis of user-defined attributes. The receiver can also receive physical (individual), group (multicast), and broadcast addresses. The Ethernet transmitter requires very little SC140 core intervention. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the MII/RMII/SMII interface to the physical media. Once initialized, the transmitter runs until the end-of-frame (EOF) condition is detected, unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered. In addition to the MAC-to-PHY interface, the Ethernet controller also supports a MAC-to-MAC interface with the SMII mode. In all three modes, the Ethernet controller can automatically gather network statistics required for RMON without the need to receive all addresses using promiscuous mode.

Note: For details, see **Chapter 25**, *Ethernet Controller*.

1.2.10 Universal Asynchronous Receiver/Transmitter (UART)

The UART is used mainly for debugging or booting. It provides a full-duplex port for serial communications via transmit data (TXD) and receive data (RXD) lines. During reception, the UART generates an interrupt request when a new character is available in the UART data register. During transmission, the UART generates an interrupt request when a new character can be written to its data register. When it accepts an interrupt request, an SC140 core or external host should read the UART status register to identify the interrupt source and service it accordingly.

Note: For details, see **Chapter 21**, *UART*.

1.2.11 Timers

The MSC8113 device contains 32 identical general-purpose 16-bit timers divided into two 16-timer groups. Within a group, each timer functions independently or as part of a programmable cascade of two timers. Each timer is programmable as either one-shot or cyclic. The SC140 cores can program the counters, read their updated values, and also be interrupted when the timers reach a predefined value. The timers are clocked by either the internal clock generator or one of four dedicated external signals or from the receive and transmit TDM clocks. When a timer reaches a predefined value, it either toggles or generates a pulse that can be directed to one of the four dedicated external signals or to other timers. In addition, it can generate an interrupt.

Note: For details, see **Chapter 22**, *Timers*.

1.2.12 GPIOs

The MSC8113 device has 32 general-purpose I/O (GPIO) signals. Each connection in the I/O ports is configured either as a GPIO signal or as a dedicated peripheral interface signal. In addition, fifteen of the GPIO signals can generate interrupts to the global interrupt controller (GIC). Each line is configured as an input or output (with a register for data output that is read or written at any time). All outputs can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, the signal drives a zero voltage but goes to tri-state (high impedance) when driving a high voltage. GPIO signals do not have internal pull-up resistors. Dedicated MSC8113 peripheral functions are multiplexed onto the shared external connections. The functions are grouped to maximize connection use for the greatest number of MSC8113 applications.

Note: For details, see **Chapter 23**, *GPIO*.

1.2.13 Reset and Boot

The Hard Reset Configuration Word (HRCW) contains the essential information for resetting the device, including the PLL divide ratio, signal configuration, and the DSI host endian mode. This configuration word is initialized by writing to it either from the system configuration source. When the MSC8113 is reset from the system bus, the configuration word is latched using a dedicated $\overline{\text{RSTCONF}}$ signal. In another reset procedure, a master DSP reads data from the ROM and latches it to the other DSPs. When resetting from an external host, the HRCW is latched from the 32 least significant bits (lsb) of the DSI bus. Immediately after reset, SC140 core 0 starts executing the code on the internal boot ROM. The value in the configuration register identifies the boot source. There are five possible boot sources:

- System port
- External host
- TDM interface
- UART
- I^2C software module

Note: For details, see **Chapter 5**, *Reset* and **Chapter 6**, *Boot Program*.

1.2.14 Interrupt Scheme

Each of the three extended cores contains two local interrupt modules, the programmable interrupt controller (PIC) and the local interrupt controller (LIC). The PIC has 24 maskable and 8 non maskable interrupts, and its SC140 core accesses it directly. The PIC handles interrupts from the SC140 DSP core EOnCE module and some external interrupts. The PIC also receives interrupts from the LIC, which in turn concentrates interrupts from the MSC8113 peripherals (TDMs, timers, DMA controller, UART, and virtual interrupts), the SIU, and other external interrupts, such the Ethernet interface. With interrupt controllers local to the SC140 core, each SC140 core can handle the relevant interrupts and either treat them as level sources or capture them as edge-triggered sources. For example, when the TDM generates an interrupt upon reaching the first threshold of its buffers, this interrupt could be useful for multiple cores. Therefore, the interrupt pulse can be captured, as well as an edge source in all of the LIC modules, and each SC140 core can process the interrupt and clear the local status bit separately, without unnecessary arbitration. A global interrupt controller (GIC) concentrates interrupts from the SIU, the UART, and external signals and drives the $\overline{\text{INT\_OUT}}$ signal. It also generates the virtual interrupts for core-to-core and external host-to-core interrupts.

Note: For details, see **Chapter 17**, *Interrupt Processing*.

1.2.15 Signal Multiplexing Options

The MSC8113 device allows various external signal multiplexing options to distribute the external signal lines among the system bus, DSI bus, TDM interfaces, Ethernet signals, and GPIO signals. **Table 1-11** summarizes the multiplexing options.

Table 1-11. External Signal Multiplexing Options

Configuration Setting				Configuration Options				
DSI64[1]	ETHSEL[2]	Ethernet Enable[3]	Ethernet Mode[4]	Available TDMs	Bus Width in Bits		Ethernet	
					DSI	System Bus	DSI/ System Bus	GPIO
0	0	0	None	0, 1, 2, 3	32	64	—	—
0	0	1	00 = MII	0,1	32	64	—	MII
0	0	1	01 = RMII	0,1, 3	32	64	—	RMII
0	0	1	10 = SMII	0,1, 3	32	64	—	SMII
0	1	0	None	0, 1, 2, 3	32	32	—	—
0	1	1	00 = MII	0, 1, 2, 3	32	32	MII	—
0	1	1	01 = RMII	0, 1, 2,3	32	32	RMII	—
0	1	1	10 = SMII	0, 1, 2, 3	32	32	—	—
1	0	0	None	0, 1, 2, 3	64	32	—	—

Table 1-11. External Signal Multiplexing Options

Configuration Setting				Configuration Options				
DSI64[1]	ETHSEL[2]	Ethernet Enable[3]	Ethernet Mode[4]	Available TDMs	Bus Width in Bits		Ethernet	
					DSI	System Bus	DSI/ System Bus	GPIO
1	0	1	00 = MII	0, 1	64	32	—	MII
1	0	1	01 = RMII	0, 1, 3	64	32	—	RMII
1	0	1	10 = SMII	0, 1, 3	64	32	—	SMII
1	1	X	None	0, 1, 2, 3	64	32	—	—

Notes:
1. Represents the sampled value when $\overline{\text{PORESET}}$ is deasserted.
2. The value of the ETHSEL bit in the Hard Reset Configuration Word.
3. The value of the EN bit in the MIIGSK Enable Register.
4. The value of the IFMODE bits in the MIIGSK Configuration Register.

1.3 Internal Communication and Semaphores

The MSC8113 device contains flexible mechanisms for communicating between SC140 cores and between an SC140 core and an external host. An SC140 core sends a message to another SC140 core either by accessing an agreed location (mailbox) in the shared M2 memory or by accessing any of the M1 memories and using an interrupt to indicate the access. Access to shared resources can be protected by semaphores.

1.3.1 Internal Communication

Each SC140 core or an external host can generate an interrupt to another SC140 core or to an external host by writing the destination core number and the virtual interrupt number to a virtual interrupt register. Each generated interrupt destination is programmable and can be forwarded to one or multiple destination SC140 cores.

Note: For details, see **Chapter 17**, *Interrupt Processing*.

1.3.2 Atomic Operations

When the SC140 core executes the **bmtset** instruction, it issues a read access followed by a write access to the semaphore address and then asserts the atomic signal. The MQBus and the SQBus prevent the SC140 cores from writing to the same semaphore address. A semaphore shared by an SC140 core and an external host on the system bus is protected by a snooper on the bus interface. When the system bus interface receives a read with atomic signal, the snooper starts to snoop the bus. The snooper returns a failure if the external host writes to the same location. Snoopers also protect the M1 and the M2 memories, which are accessible to both the SC140 cores and external hosts.

Note: For details, see **Section 9.3**, *Extended QBus System*.

1.3.3 Hardware Semaphores

There are eight coded hardware semaphores. Each semaphore is an 8-bit register with a selective write protection mechanism. When the register value is zero, it is writable to any new value. When the register value is not zero, it is writable only to zero. Each SC140 core/host/task has a unique pre-defined lock number (8-bit code). When trying to lock the semaphore, the SC140 core writes its lock number to the semaphore and then reads it. If the read value equals its lock number, the semaphore belongs to that host and is essentially locked. An SC140 core/host/task releases the semaphore by simply writing 0.

Note: For details, see **Chapter 15**, *Hardware Semaphores*.

SC140 Core Overview 2

The SC140 digital signal processing (DSP) core features an innovative architecture that addresses the key market needs of the next-generation DSP applications, mainly in the field of wireline and wireless infrastructure and subscriber communication. This flexible DSP core supports compute-intensive applications by providing high performance, low power, efficient compile, and high code density. The SC140 core efficiently deploys a novel variable-length execution set (VLES) execution model, maximizing parallelism by allowing multiple address generation and data arithmetic logic units to execute multiple operations in a single clock cycle. This section provides an overview of the key features and main modules of the SC140 core, as well as the programming model and instruction set list.

Note: The information in this chapter is based on Revision 3 of the *SC140 DSP Core Reference Manual*. To get the updates in later revisions of this manual, visit the Freescale Web site shown on the back cover of this manual.

The 16-bit SC140 core packs four data arithmetic-logic execution units (ALUs), each consisting of a multiply-accumulate unit (MAC), a logic unit, and a bit field unit (BFU), which also serves as a barrel shifter. In addition to the four data execution units, the core contains two address arithmetic units (AAUs), one bit manipulation unit (BMU) and one branch unit. Overall, the SC140 can issue and execute up to six instructions per clock—for example, four independent arithmetic instructions and two pointer-related instructions (such as moves or other operations on addresses). At a clock speed of 400 MHz, the SC140 can therefore execute 1600 true DSP MIPS—1600 million multiply-accumulate operations per second (MMACS), concurrent with associated data movement functions and pointer updates.

The SC140 core can sustain this high performance over time because of the flexibility of its data execution units and ability to transfer up to 128 data bits per cycle. The four data execution units can operate simultaneously in any combination. For example, the SC140 core can execute four multiply-accumulate operations in a single clock, or one MAC, two arithmetic/logical operations and one bit field operation. All four data ALUs are identical, permitting great flexibility in assigning and executing instructions, increasing the likelihood that four execution units can be kept busy on any given cycle and enabling programs to take better advantage of the SC140 core parallel architecture.

2.1 Architecture

This section discusses the main functional blocks of the SC140 core. **Figure 2-1** shows a block diagram of the core as used by the MSC8113.

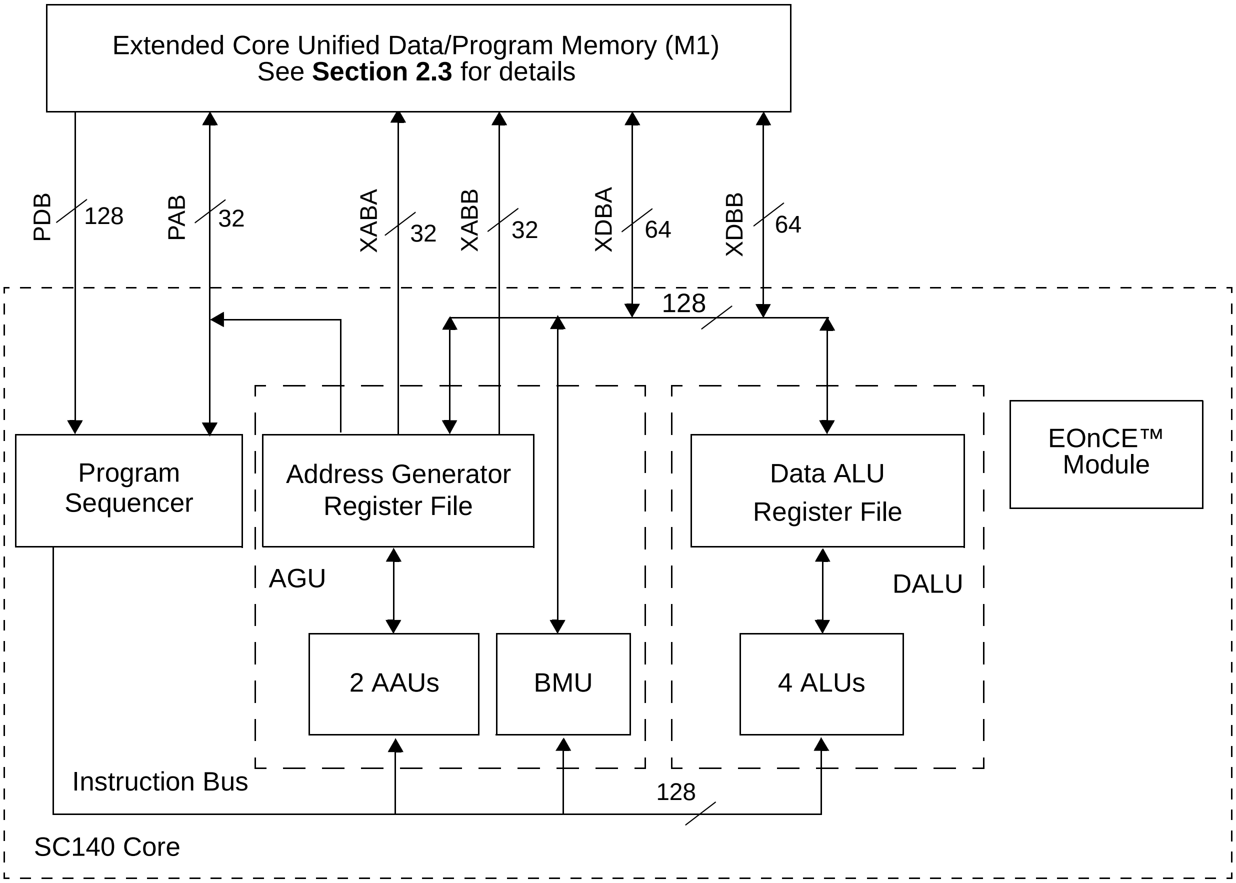


Figure 2-1. Block Diagram of the SC140 Core in the MSC8113

Note: The SC140 DSP core defines the PLL Control Registers 0–1 (PCTL[0–1]) for PLL and clock control. The MSC8113 does not use these registers in its design. In addition, the manual defines six debug modules and seven EE signal lines in the EOnCE module.

The MSC8113 device uses only two of these modules (0 and 1) and two signals (EE0 and EE1).

2.1.1 Data Arithmetic Logic Unit (Data ALU)

The Data ALU performs arithmetic and logical operations on data operands in the MSC8113. The data registers can be read or written to memory over the XDBA and the XDBB as 8-bit, 16-bit, or 32-bit operands. The 64-bit wide data buses XDBA and XDBB support the transfer of several operands on a single access. The source operands for the Data ALU, which may be 16, 32, or 40 bits, originate either from data registers or from immediate data. The results of all Data ALU operations are stored in the data registers. All Data ALU operations are performed in one clock cycle. Up to four parallel arithmetic operations can be performed in each cycle. The destination of every arithmetic operation can be used as a source operand for the operation immediately following, without any time penalty.

The components of the Data ALU are as follows:

- A bank of sixteen 40-bit registers
- Four parallel ALUs, each ALU containing a MAC unit and a BFU with a 40-bit barrel shifter
- Eight data bus shifter/limiter circuits, to allow limiting four 16-bit fractional words over each of the 64-bit data buses in a single cycle.

All the MAC units and BFUs can access all the Data ALU registers. Each register is partitioned into three portions: two 16-bit registers (low and high portion of the register) and one 8-bit register (extension portion). The 16-bit high and low register portions are typically used as an inputs for arithmetic operations. The full 40-bit register can be used as an input operand, but is generally used as an output operand for most instructions. The two 64-bit wide data buses that connect between the Data ALU register file and the memory enable a very high data bandwidth between memory and registers. Load and store instructions utilize the maximum width of the bus according to the application requirement because there are different versions of the instructions for different bandwidths:

- **move.b** loads or stores bytes (8-bit)
- **move.w** or **move.f** loads or stores integer or fractional words (16-bit)
- **move.l** loads or stores long words (32-bit)
- **move.2w** `or` **move.2f** loads or stores double-integers and double-fractions, respectively (32-bit)
- **move.4w** or **move.4f** loads or stores quad-integers and quad-fractions respectively (64-bit)
- **move.2l** loads or stores two long words (64-bits total)

Figure 2-2 shows the architecture of the Data ALU.

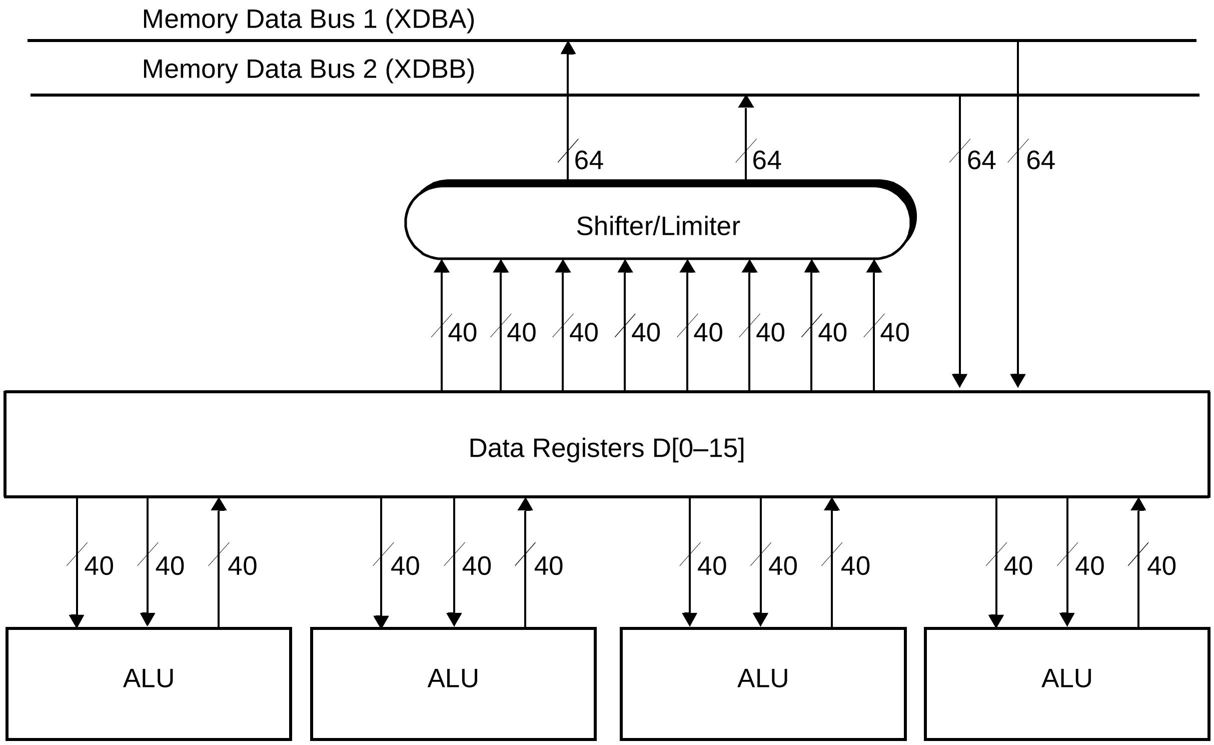


Figure 2-2. Data ALU Architecture

With the ability to execute any two **MOVE** instructions in parallel every clock cycle, a maximum data throughput of 6.4 GBps (at 400 MHz) can be achieved between the memory and the register file.

2.1.1.1 Data Registers

The Data ALU registers are read or written over the data buses (XDBA and XDBB). The source operands for Data ALU arithmetic instructions always originate from Data ALU registers. All the Data ALU operations are performed in one clock cycle so that a new instruction can be initiated in every clock, yielding a rate of up to four Data ALU instructions per clock cycle. The destination of every arithmetic operation can be used as a source operand for the operation immediately following, without any time penalty.

2.1.1.2 Multiply-Accumulate (MAC) Unit

The MAC unit comprises the main arithmetic processing unit of each SC140 core and performs all the calculations on data operands. The MAC unit outputs one 40-bit result in the form of [Extension:Most Significant Portion:Least Significant Portion] (EXT:MSP:LSP). The multiplier executes 16-bit × 16-bit fractional or integer multiplication between two's complement signed, unsigned, or mixed operands. The 32-bit product is right-justified and added to the 40-bit contents of one of the sixteen data registers.

2.1.1.3 Bit-Field Unit (BFU)

The BFU contains a 40-bit parallel bidirectional shifter with a 40-bit input and a 40-bit output, mask generation unit, and logic unit. The BFU is used in the following operations:

- Multi-bit left/right shift (arithmetic or logical)
- One-bit rotate (right or left)
- Bit-field insert and extract
- Count leading bits
- Logical operations
- Sign or zero extension operations

2.1.2 Address Generation Unit (AGU)

The AGU is one of the execution units in the SC140 core. The AGU performs effective address calculations using the integer arithmetic necessary to address data operands in memory, and it contains the registers to generate the addresses. It performs four types of arithmetic: linear, modulo, multiple wrap-around modulo, and reverse-carry. The AGU operates in parallel with other chip resources to minimize address generation overhead. The AGU also generates change-of-flow program addresses and manages the stack pointer (SP). The major components of the AGU are as follows:

- Eight address registers (R[0–7])
- Eight alternative address registers (R[8–15]) or eight base address registers (B[0–7])
- Two stack pointers (NSP, ESP), only one of which is active at a time (SP)
- Four offset registers (N[0–3])
- Four modifier registers (M[0–3])
- A Modifier Control Register (MCTL)
- Two Address Arithmetic Units (AAU)
- One Bit Mask Unit (BMU)

Figure 2-3 shows a block diagram of the AGU.

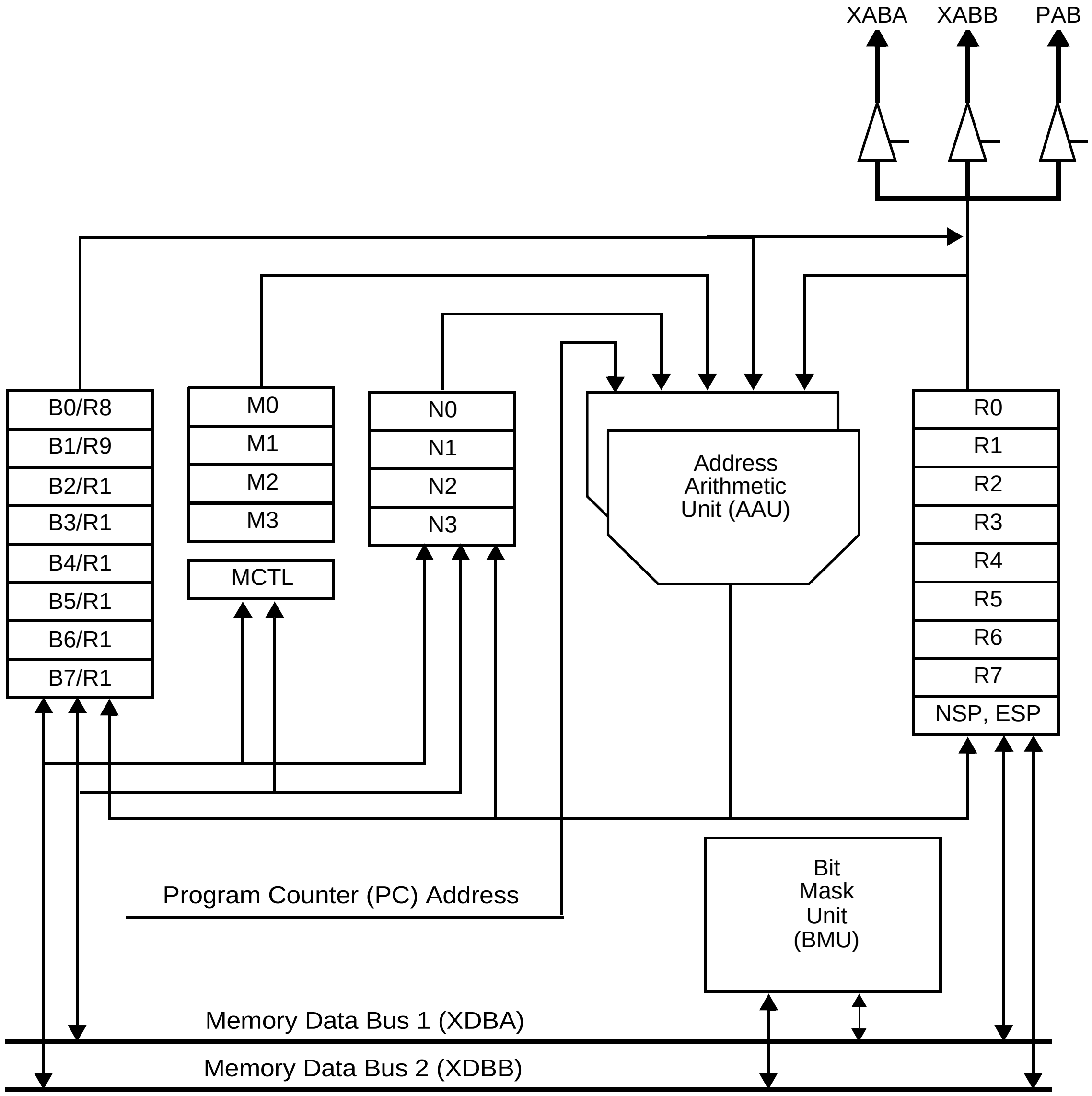


Figure 2-3. AGU Block Diagram

The two AAUs are identical. Each contains a 32-bit full adder called an offset adder and a 32-bit full adder called a modulo adder. The offset adder performs the following operations:

- Add or subtract an AGU registers or PC to/from an AGU register
- Add or subtract an immediate value to/from an AGU register
- Compare to or test an AGU register
- Logical and arithmetic shift operations on AGU registers
- Sign or zero-extend an AGU register
- Add with reverse carry

The offset values added in this adder are pre-shifted by 1, 2, or 3, according to the access width. In reverse-carry mode, the carry propagates in the opposite direction. The modulo adder adds the summed result of the first full adder to a modulo value, M or minus M, where M is stored in the selected modifier register. In modulo mode, the modulo comparator tests whether the result is inside the buffer by comparing the results to the B register and chooses the correct result from between the offset adder and the modulo adder.

2.1.2.1 Stack Pointer Registers

To facilitate use of a software stack, two special registers with special addressing modes are assigned to the AGU: the Normal Mode Stack Pointer (NSP) and the Exception Mode Stack Pointer (ESP). Both the ESP and the NSP are 32-bit read/write address registers with predecrement and post-increment updates, as well as offset with immediate values to allow random access to the software stack. Stack instructions use the ESP when the MSC8113 is in the Exception mode of operation, which it enters when exceptions occur. The NSP is used in Normal mode, while not servicing an exception. The two stack pointers make it easier to support multitasking systems and optimizes stack usage for these systems.

2.1.2.2 Bit Mask Unit (BMU)

The BMU performs bit mask operations, such as setting, clearing, changing, or testing a destination, according to an immediate mask operand. Data is loaded to the BMU over the data memory buses XDBA or XDBB. The result is written back over XDBA or XDBB to the destinations in the next cycle. All bit mask instructions typically execute in two cycles and work on 16-bit data. This data can be a memory location, or a portion (high or low) of a register. The BMU supports a set of bit mask instructions that operate on:

- All AGU pointers (R[0–15])
- All Data ALU registers (D[0–15])
- All control registers (EMR, VBA, SR, MCTL)
- Memory locations

Only a single bit mask instruction is allowed in any single execution set, since only one execution unit exists for these instructions. A subset of the bit mask instructions (BMTSET) allows support for hardware semaphores.

2.1.3 Program Sequencer Unit (PSEQ)

The PSEQ fetches and dispatches instructions, controls hardware loops, and controls exception processing. The PSEQ implements three out of the five stages of the pipeline and controls the different processing states of the MSC8113 core. It consists of three hardware blocks:

- *Program address generator (PAG)*. Generates the program counter (PC) for instruction fetch operations and controls the hardware loop functionality.

- *Program dispatch unit (PDU)*. Detects the execution set out of the fetch set and dispatches the various instructions of the execution set to their appropriate execution units.
- *Program control unit (PCU)*. Controls the overall pipeline behavior of the program flow.

The PSEQ implements its functions using the following registers:

- Program Counter Register (PC)
- Status Register (SR)
- Four Loop Start Address Registers (SA[0–3])
- Four Loop Counter Registers (LC[0–3])
- Exception and Mode Register (EMR)
- Vector Base Address Register (VBA)

2.1.4 Enhanced On-Chip Emulation (EOnCE)

The EOnCE module allows nonintrusive interaction with the MSC8113 and its peripherals so that you can examine registers, memory, or on-chip peripherals, define various breakpoints, and read the trace-FIFO. These interactions facilitate hardware and software development on the MSC8113 processor. The EOnCE module interfaces with the debugging system through on-chip JTAG TAP controller signals. For details, see the *SC140 DSP Core Reference Manual*.

2.2 Programming Model

The three main units of the SC140 DSP core programming model are the Address Generation Unit (AGU), the Data Arithmetic Logic Unit (Data ALU), and the PSEQ (see **Figure 2-4**). This section gives a brief overview of each of these units.

2.2.1 AGU Programming Model

The address registers can be programmed for linear, modulo (regular or multiple wrap-around), and bit-reverse addressing. Automatic updating of address registers is available when address register indirect addressing is used.

- *Address Registers (R[0–15])*. The sixteen 32-bit address registers R[0–15] contain addresses or general-purpose data. These are 32-bit read/write registers. The 32-bit address in a selected address register is used in calculating the effective address of an operand. The contents of an address register point directly to memory or are used as an offset. R[0–15] are composed of two separate banks, a lower bank (R[0–7]) and an upper bank (R[8–15]). The lower bank registers can be used for linear, modulo, or bit reverse addressing. An upper bank register can be used in linear addressing modes only if the respective register in the lower bank is not using modulo addressing mode. In modulo addressing mode, each lower bank register Rn is assigned a corresponding base address register Bn. Registers B[0–7] and R[8–15] are mapped to the same physical register, respectively. Therefore, for example, R8 is available only if R0 is not being used in modulo addressing, since this requires the base address register B0. See **Section 2.2.2**, *Data Arithmetic Logic Programming Model*, on page 2-11 for further information.

If an address register is updated, one of the modifier control registers (MCTL) specifies the type of update arithmetic. Offset registers (Ni) are used for post-addition and indexing by offset. The address register modification is performed by either of the two AAUs.

- *Stack Pointer Registers (NSP, ESP)*. The MSC8113 has two stack pointer registers: the Normal Stack Pointer (NSP) and the Exception Stack Pointer (ESP). These 32-bit registers are used implicitly in all PUSH and POP instructions. Only one stack pointer is active at a time, according to the mode:

  — In Normal mode, the NSP is used.
  — In Exception mode, the ESP is used.

  The Status Register EXP bit determines the active mode. The active stack pointer (SP) is used explicitly for memory references in the address register indirect modes. The stack pointers point to the next unoccupied location in the stacks. They are post-incremented on all the implicit PUSH operations and pre-decremented on all the implicit POP operations.

Note: You must explicitly initialize both stack pointer registers after reset.

- *Offset Registers (N[0–3])*. The 32-bit read/write offset registers N[0–3] contain offset values to increment or decrement address registers in address register update calculations. These registers are also used for 32-bit general-purpose storage. For example, the contents of an offset register specify the offset into a table or the base of the table for indexed addressing. An offset register can be used to step through a table at a specified rate—such as five locations per step for waveform generation. Each address register can be used with each offset register. For example, R0 can be used with N0, N1, N2, or N3 for offset address calculation.
- *Base Address Registers (B[0–7])*. The 32-bit read/write base address registers B[0–7] are used in modulo calculations. Each B register is associated with an R register (B0 with R0, and so on). When the modulo addressing mode is activated, the B register contains the lower boundary value of the modulo buffer. The upper boundary of the modulo buffer is calculated by B+M-1, where M is the modifier register associated with the register used. When not used for modulo accessing, these registers can function as alternative address registers (R[8–15]). Both Rx and B_{X-8} share the same physical register. For example, if R0 is not programmed for modulo addressing, the base address register B0 can serve as an additional address register R8.

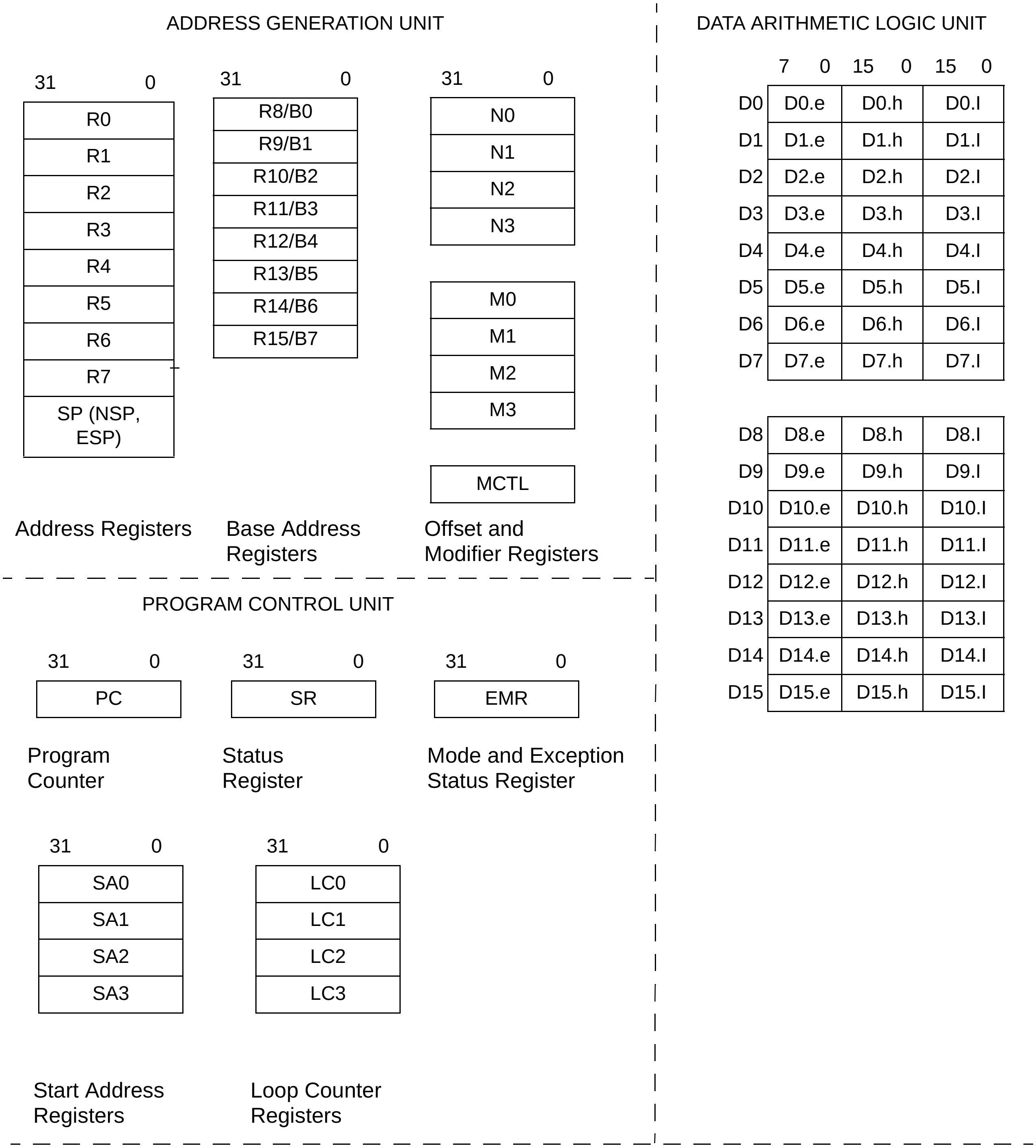


Figure 2-4. SC140 Programming Model

- *Modifier Registers (M[0–3])*. The 32-bit read/write modifier registers M[0–3] contain the value of the modulus modifier. These registers are also used for general-purpose storage. The address arithmetic unit (AAU) supports linear, modulo, multiple wrap-around modulo, and reverse-carry arithmetic types for most address register indirect addressing modes. When the modulo arithmetic is activated, the contents of Mj specify the modulus.

Each address register can be used with each modifier register, as programmed in the MCTL register.

- *Modifier Control Register (MCTL)*. The 32-bit read/write register to program the address mode (AM) for each of the eight address registers (R[0–7]). The addressing mode of the upper address register file (R[8–15]) cannot be programmed and functions in linear mode only.

2.2.2 Data Arithmetic Logic Programming Model

The Data ALU programming model is shown in **Figure 2-4**. Register D0 refers to the entire 40-bit register, whereas D0.e, D0.h, D0.l refer to the extension, most significant and least significant portions of the D0 register, respectively. The D[0–15] data registers, referred to as D*x*, give maximum flexibility, since they are used as source operands, destination storage, or accumulators.The registers serve as input buffer registers between the XDBA or XDBB and the ALUs. They are used as Data ALU source operands, allowing new operands to be loaded for the next instruction while the register contents are used by the current arithmetic instruction.

Each data register Dx has an additional associated flag bit, the limit tag bit Lx, to signify that limiting could occur when reading Dx over XDBA and XDBB. For saving and restoring, the limit tag bit Lx is coupled with the extension portion Dx.e, to form a 9-bit operand. The limit tag bit Lx is updated when a result is written from the ALU to the Dx register.

The data registers are accessed with three types of data width:

- A long-word type access, writing or reading 32-bit operands
- A word type access, writing or reading 16-bit operands
- A byte type access, writing or reading 8-bit operands

Fractional data in Dx registers that is transferred to memory over XDBA and XDBB is replaced by a limiting constant if the value cannot be represented by the number of bits in the access width. The contents of Dx are not affected if limiting occurs. Only the value transferred over XDBA or XDBB is limited. This process is commonly referred to as transfer saturation, and it should not be confused with the arithmetic saturation mode. The overflow protection is performed after the contents of the register are shifted according to the scaling mode. Shifting and limiting are performed only when a fractional operand is specified as the source for a data move over XDBA or XDBB. When an integer operand is specified as the source for a data move, shifting and limiting are not performed.

Automatic sign extension or zero extension of the data values into the 40-bit registers is provided when an operand is transferred from memory to a data register. If a fractional word operand is to be written to a data register, the MSP portion of the register is written with the word operand, the LSP portion is zero-extended, and the EXT portion is sign-extended from MSP. When an integer operand is to be written to a data register, the LSP portion of the register is written with the word operand, and the MSP portion and EXT are either zero-extended or sign-extended from the LSP.

Long-word operands are written into the MSP:LSP portions of the register, and the EXT portion is either zero- or sign-extended.

When a byte operand is to be written to a data register, the register's first eight bit portion of the LSP (Dx.1[7–0]) is written with the byte operand, and the remaining bits are either zero-extended or sign-extended from the LSP lower byte.

2.2.3 Program Control Unit Programming Model

The Program Control Unit (PCU) is part of the Program Sequencer Unit (PSEQ). The PCU controls the overall pipeline behavior of the program flow. The PCU implements its functions using the following registers:

- Program Counter Register (PC)
- Status Register (SR)
- Four Start Address Registers (SA[0–3])
- Four Loop Counter Registers (LC[0–3])
- Exception and Mode Register (EMR). The EMR reflects and controls exception situations in the core. It contains bits that reflect memory configuration, servicing of a non-maskable interrupt, and the following exception conditions: Data ALU overflow, illegal execution set, and illegal instruction flow.

The EMR GP[0–6] and BEM fields are initialized at reset as described in **Table 2-1**.

Table 2-1. EMR GP[6–0] and BEM Field Reset Values

Field	Reset Value
BEM	1
GP0	EE1
GP1	0
GP2	ISBSEL2 from Hard Reset Configuration Word (HRCW) bit 15
GP3	ISBSEL1 from HRCW bit 14
GP4	ISBSEL0 from HRCW bit 13
GP5	0
GP6	0
Note: GP4 equals the inversion of the HRCW bit 13.	

2.3 Instruction Set Overview

The SC140 instruction set is divided into the following functional groups:

- Data ALU arithmetic
- AGU arithmetic

- Move
- Stack support
- Bit mask
- Change-of-flow
- Program control

This following tables list the SC140 instructions alphabetically within the appropriate functional group.

Table 2-2. DALU Arithmetic Instructions

Instruction	Description
ABS	Absolute value
ADC	Add long with carry
ADD	Add
ADD2	Add two 16-bit values
ADDNC.W	Add without changing the carry bit in the status register
ADR	Add and round
ASL	Arithmetic shift left by one bit
ASR	Arithmetic shift right by one bit
CLR	Clear
CMPEQ	Compare data registers for equal
CMPEQ.W	Compare immediate value to data register for equal
CMPGT	Compare data registers for greater than
CMPGT.W	Compare data register to immediate for greater than
CMPHI	Compare for higher (unsigned)
DECEQ	Decrement a data register and set T if zero
DECGE	Decrement a data register and set T if greater than or equal to zero
DIV	Divide iteration
DMACSS	Multiply signed by signed and accumulate with data register right shifted by word size
DMACSU	Multiply signed by unsigned and accumulate with data register right shifted by word size
IADD	Integer addition - no saturation
IMAC	Signed integer multiply-accumulate
IMACLHUU	Integer multiply-accumulate unsigned times unsigned; first source from lower portion second from upper
IMACUS	Integer multiply-accumulate unsigned times signed
IMPY	Signed integer multiply
IMPYHLUU	Integer multiply unsigned times unsigned; first source from upper portion second from lower
IMPYSU	Integer multiply signed times unsigned
IMPYUU	Integer multiply unsigned times unsigned
INC	Increment a data register (as integer data)

Table 2-2. DALU Arithmetic Instructions (Continued)

Instruction	Description
INC.F	Increment a data register (as fractional data)
MAC	Signed fractional multiply-accumulate
MACR	Signed fractional multiply-accumulate and round
MACSU	Signed/unsigned fractional multiply-accumulate
MACUS	Unsigned/signed fractional multiply-accumulate
MACUU	Unsigned/unsigned fractional multiply-accumulate
MAX	Transfer maximum signed value
MAX2	Transfer two 16-bit maximum signed value
MAX2VIT	Specialized MAX2 version for Viterbi kernel
MAXM	Transfer maximum magnitude value
MIN	Transfer minimum signed value
MPY	Signed fractional multiply
MPYR	Signed fractional multiply and round
MPYSU	Signed/unsigned fractional multiply
MPYUS	Unsigned/signed fractional multiply
MPYUU	Unsigned/unsigned fractional multiply
NEG	Negate
RND	Round
SAT.F	Saturate value in data register to fit in top 16 bits
SAT.L	Saturate value in data register to fit in 32 bits
SBC	Subtract long with carry
SBR	Subtract and round
SUB	Subtract
SUB2	Subtract two 16-bit values
SUBL	Shift left and subtract
SUBNC.W	Subtract without changing the carry bit in the status register
TFR	Transfer data register to a data register
TFRF	Conditional data register transfer if the T bit is clear
TFRT	Conditional data register transfer if the T bit is set
TSTEQ	Test for equal to zero
TSTGE	Test for greater than or equal to zero
TSTGT	Test for greater than zero

Table 2-3. DALU Logical Instructions

Instruction	Description
AND	Logical AND
ASLL	Multi-bit arithmetic shift left
ASLW	Word arithmetic shift left (16-bit shift)
ASRR	Multi-bit arithmetic shift right
ASRW	Word arithmetic shift right (16-bit shift)
CLB	Count leading bits
EOR	Logical exclusive OR
EXTRACT	Extract signed bit field
EXTRACTU	Extract unsigned bit field
INSERT	Insert bit field
LSLL	Multi-bit logical shift left
LSR	Logical shift left by one bit
LSRR	Multi-bit logical shift right
LSRW	Word logical shift right (16-bit shift)
NOT	Logical complement
OR	Logical inclusive OR
ROL	Rotate one bit left through the carry bit
ROR	Rotate one bit right through the carry bit
SXT.B	Sign extend byte
SXT.L	Sign extend long
SXT.W	Sign extend word
ZXT.B	Zero extend byte
ZXT.L	Zero extend long
ZXT.W	Zero extend word

Table 2-4. AGU Arithmetic Instructions

Instruction	Description
ADDA	AGU add
ADDL1A	AGU add with 1-bit left shift of source operand
ADDL2A	AGU add with 2-bit left shift of source operand
ASL2A	AGU arithmetic shift left by 2 bits (32-bit)
ASLA	AGU arithmetic shift left (32-bit)
ASRA	AGU arithmetic shift right (32-bit)
CMPEQA	AGU compare for equal
CMPGTA	AGU compare for greater than
CMPHIA	AGU compare for higher (unsigned)
DECA	AGU decrement register

Table 2-4. AGU Arithmetic Instructions (Continued)

Instruction	Description
DECEQA	AGU decrement and set T if zero
DECGEA	AGU decrement and set T if equal or greater than zero
INCA	AGU increment register
LSRA	AGU logical shift right (32-bit)
SUBA	AGU subtract
SXTA.B	AGU sign extend byte
SXTA.W	AGU sign extend word
TFRA	AGU register transfer
TSTEQA.L	AGU test for equal on all 32 bits
TSTEQA.W	AGU test for equal on lower 16 bits
TSTGEA.L	AGU test for greater than or equal
TSTGTA	AGU test for greater than
ZXTA.B	AGU zero extend byte
ZXTA.W	AGU zero extend word

Table 2-5. Move Instructions

Instruction	Description
MOVE.2F	Move two fractional words from memory to a register pair
MOVE.2L	Move two longs to/from a register pair
MOVE.2W	Move two integer words to/from a register pair
MOVE.4F	Move four fractional words from memory to a register quadrant
MOVE.4W	Move four integer words to/from a register quadrant
MOVE.B	Move byte (sign-extended for memory reads)
MOVE.F	Move fractional word to and from memory
MOVE.L	Move long (sign extended for memory or register reads)
MOVE.W	Move integer word (sign extended for memory reads)
MOVEF	Move address register to address register, depending on T bit of SR
MOVES.F	Move fractional word to memory with saturation enabled
MOVES.L	Move long to memory with saturation enabled
MOVES.2F	Move two fractional words to memory with saturation enabled
MOVES.4F	Move four fractional words to memory with saturation enabled
MOVET	Move address register to address register, depending on T bit of SR
MOVEU.B	Move unsigned byte from memory
MOVEU.L	Move unsigned long from memory
MOVEU.W	Move unsigned integer word from memory
VSL.2F	Viterbi shift left: specialized move to support Viterbi kernel
VSL.2W	Viterbi shift left: specialized move to support Viterbi kernel

Table 2-5. Move Instructions (Continued)

Instruction	Description
VSL.4F	Viterbi shift left: specialized move to support Viterbi kernel
VSL.4W	Viterbi shift left: specialized move to support Viterbi kernel

Table 2-6. Stack Support Instructions

Instruction	Description
POP	Pop a register from the software stack
POPN	Pop a register from the software stack using the normal stack pointer
PUSH	Push a register into the software stack
PUSHN	Push a register into the software stack using the normal stack pointer
TFRA	OSP Move the “other” stack pointer to/from a register, inversely defined by the exception mode

Table 2-7. Bit Mask Instructions

Instruction	Description
AND	Logical AND on a 16-bit operand
AND.W	Logical AND on a 16-bit immediate value
BMCHG	Bit-mask change for a 16-bit operand
BMCHG.W	Bit-mask change for a 16-bit operand in memory
BMCLR	Bit-mask clear for a 16-bit operand
BMCLR.W	Bit-mask clear for a 16-bit operand in memory
BMSET	Bit-mask set for a 16-bit operand
BMSET.W	Bit-mask set for a 16-bit operand in memory
BMTSET	Bit mask test and set for a 16-bit operand
BMTSET.W	Bit mask test and set for a 16-bit operand in memory
BMTSTC	Bit-mask test if clear for a 16-bit operand
BMTSTC.W	Bit-mask test if clear for a 16-bit operand in memory
BMTSTS	Bit-mask test if set for a 16-bit operand
BMTSTS.W	Bit-mask test if set for a 16-bit operand in memory
EOR	Logical Exclusive OR on a 16-bit operand
EOR.W	Logical Exclusive OR on a 16-bit operand in memory
NOT	Binary inversion of a 16-bit operand
NOT.W	Binary inversion of a 16-bit operand in memory
OR	Logical OR on a 16-bit operand
OR.W	Logical OR on a 16-bit operand in memory

Table 2-8. AGU Non-Loop Change-of-Flow Instructions

Instruction	Description
BF	Branch if false
BFD	Branch if false (delayed)
BRA	Branch
BRAD	Branch (delayed)
BSR	Branch to subroutine
BSRD	Branch to subroutine (delayed)
BT	Branch if true
BTD	Branch if true (delayed)
JF	Jump if false
JFD	Jump if false (delayed)
JMP	Jump
JMPD	Jump (delayed)
JSR	Jump to subroutine
JSRD	Jump to subroutine (delayed)
JT	Jump if true
JTD	Jump if true (delayed)
RTE	Return from exception
RTED	Return from exception (delayed)
RTS	Return from subroutine
RTSD	Return from subroutine (delayed)
RTSTK	Force restore PC from the stack, updating SP
RTSTKD	Force restore PC from the stack, updating SP (delayed)
TRAP	Execute a precise software exception.

Table 2-9. AGU Loop Control (including Loop COF) Instructions

Instruction	Description
BREAK	Terminate the loop and branch to an address
CONT	Jump to the start of the loop to start the next iteration
CONTD	Jump to the start of the loop to start the next iteration (delayed)
DOENn	Do enable - set the “nth” loop counter and enable the loop as a long loop
DOENSHn	Do enable short - set the “nth” loop counter and enable the loop as a short loop
DOSETUPn	Setup the “nth” hardware loop start address
SKIPLS	Test the active LC and skip the loop if LCn is equal or smaller than zero

Table 2-10. AGU Program Control Instructions

Instruction	Description
DEBUG	Enter debug mode
DEBUGEV	Signal debug event
DI	Disable interrupts (sets the DI bit in the status register)
EI	Enable interrupts (clears the DI bit in the status register)
ILLEGAL	Triggers an illegal instruction exception
MARK	Push the PC into the trace buffer
STOP	Stop processing (lowest power stand-by)
WAIT	Wait for interrupt (low power stand-by)

Table 2-11. Prefix Instructions

Instruction	Description
IFA	Execute current execution set or subset unconditionally
IFF	Execute current execution set or subset if the T bit is clear
IFT	Execute current execution set or subset if the T bit is set
NOP	No operation, not dispatched to an execution unit

2.4 Additional Programming Considerations

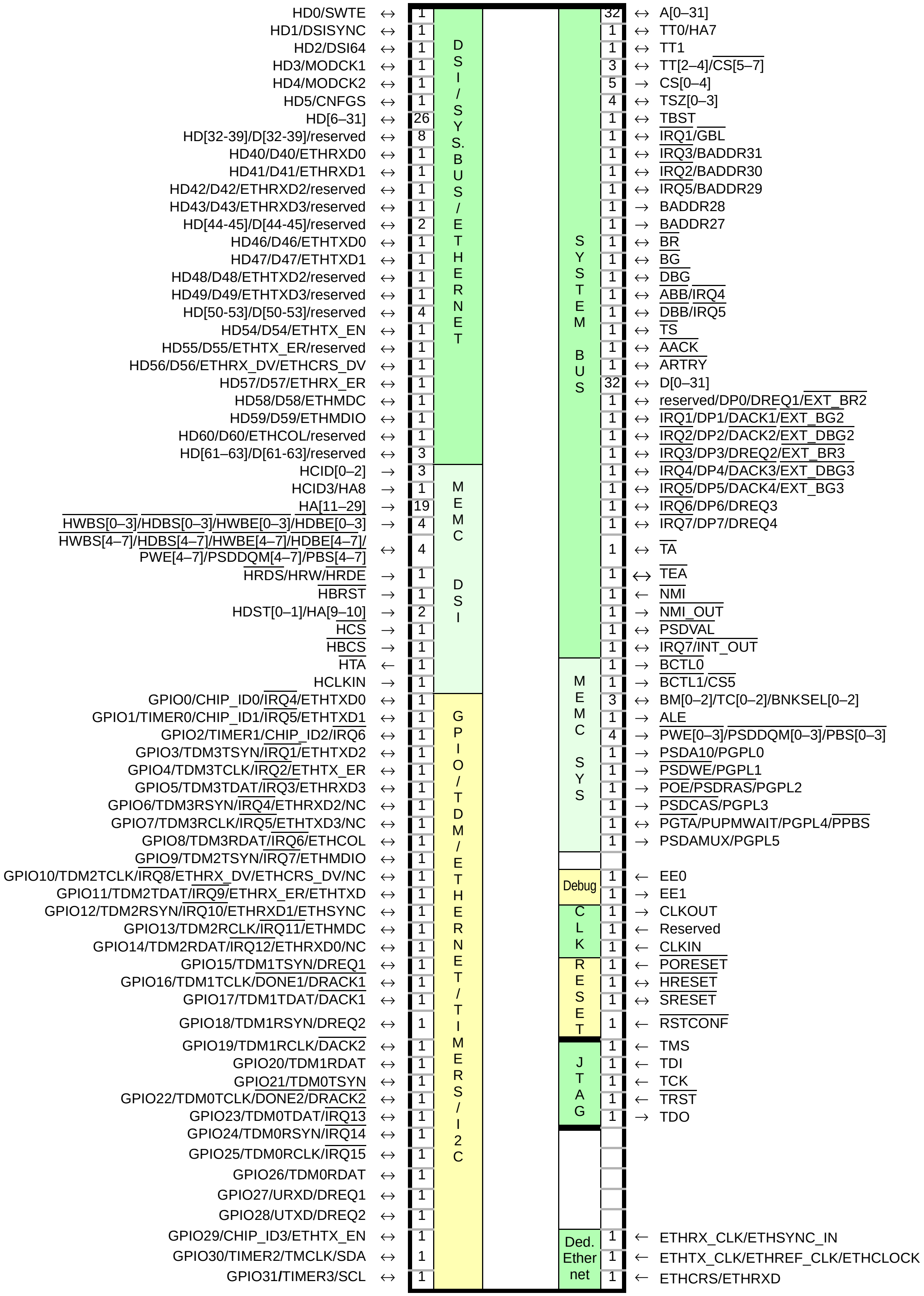
- Use the last 64 bytes of M1 memory for data only. Because of system pipelining, code fetches from this area by the cores can result in an attempt to access areas beyond the end of the M1 memory. Such fetches may cause the system to stop operation. To prevent this occurrence, do not store instruction code in the range 0x00037FC0–0x00037FFF.
- In some rare instances, accesses to illegal addresses may not generate the correct illegal address exception. If this occurs, program execution does not continue beyond the illegal address access.
- In rare situations, an illegal execution set fetched by the SC140 core can alter the settings of system registers or cause the SC140 core to enter a freeze state that can only be released by reset. The SC140 illegal instruction trap does not provide 100 percent protection against illegal instruction execution.

External Signals 3

The MSC8113 external signals are organized into functional groups, as shown in **Table 3-1** and **Figure 3-1**. **Table 3-1** lists the functional groups, the number of signal connections in each group, and references the table that gives a detailed listing of multiplexed signals within each group. **Figure 3-1** shows MSC8113 external signals organized by function.

Table 3-1. MSC8113 Functional Signal Groupings

Functional Group	Number of Signal Connections	Detailed Description
Power (V_{DD}, V_{CC}, and GND)	155	**Table 3-2** on page 3-3
Clock	3	**Table 3-3** on page 3-3
Reset and Configuration	4	**Table 3-4** on page 3-3
DSI, System Bus, Ethernet, and Interrupts	210	**Table 3-1** on page 3-4
Memory Controller	16	**Table 3-2** on page 3-16
General-Purpose Input/Output (GPIO), Time-Division Multiplexed (TDM) Interface, Universal Asynchronous Receiver/ Transmitter (UART), Ethernet, and Timers	32	**Table 3-3** on page 3-19
Ethernet signals	3	**Table 3-4** on page 3-28
EOnCE module and JTAG Test Access Port	7	**Table 3-5** on page 3-28
Reserved (denotes connections that are always reserved)	1	**Table 3-6** on page 3-29


HD0/SWTE ↔ 1
HD1/DSISYNC ↔ 1
HD2/DSI64 ↔ 1
HD3/MODCK1 ↔ 1
HD4/MODCK2 ↔ 1
HD5/CNFGS ↔ 1
HD[6–31] ↔ 26
HD[32-39]/D[32-39]/reserved ↔ 8
HD40/D40/ETHRXD0 ↔ 1
HD41/D41/ETHRXD1 ↔ 1
HD42/D42/ETHRXD2/reserved ↔ 1
HD43/D43/ETHRXD3/reserved ↔ 1
HD[44-45]/D[44-45]/reserved ↔ 2
HD46/D46/ETHTXD0 ↔ 1
HD47/D47/ETHTXD1 ↔ 1
HD48/D48/ETHTXD2/reserved ↔ 1
HD49/D49/ETHTXD3/reserved ↔ 1
HD[50-53]/D[50-53]/reserved ↔ 4
HD54/D54/ETHTX_EN ↔ 1
HD55/D55/ETHTX_ER/reserved ↔ 1
HD56/D56/ETHRX_DV/ETHCRS_DV ↔ 1
HD57/D57/ETHRX_ER ↔ 1
HD58/D58/ETHMDC ↔ 1
HD59/D59/ETHMDIO ↔ 1
HD60/D60/ETHCOL/reserved ↔ 1
HD[61–63]/D[61-63]/reserved ↔ 3
HCID[0–2] → 3
HCID3/HA8 → 1
HA[11–29] → 19
HWBS[0–3]/HDBS[0–3]/HWBE[0–3]/HDBE[0–3] → 4
HWBS[4–7]/HDBS[4–7]/HWBE[4–7]/HDBE[4–7]/
PWE[4–7]/PSDDQM[4–7]/PBS[4–7] ↔ 4
HRDS/HRW/HRDE → 1
HBRST → 1
HDST[0–1]/HA[9–10] → 2
HCS → 1
HBCS → 1
HTA ← 1
HCLKIN → 1
GPIO0/CHIP_ID0/IRQ4/ETHTXD0 ↔ 1
GPIO1/TIMER0/CHIP_ID1/IRQ5/ETHTXD1 ↔ 1
GPIO2/TIMER1/CHIP_ID2/IRQ6 ↔ 1
GPIO3/TDM3TSYN/IRQ1/ETHTXD2 ↔ 1
GPIO4/TDM3TCLK/IRQ2/ETHTX_ER ↔ 1
GPIO5/TDM3TDAT/IRQ3/ETHRXD3 ↔ 1
GPIO6/TDM3RSYN/IRQ4/ETHRXD2/NC ↔ 1
GPIO7/TDM3RCLK/IRQ5/ETHTXD3/NC ↔ 1
GPIO8/TDM3RDAT/IRQ6/ETHCOL ↔ 1
GPIO9/TDM2TSYN/IRQ7/ETHMDIO ↔ 1
GPIO10/TDM2TCLK/IRQ8/ETHRX_DV/ETHCRS_DV/NC ↔ 1
GPIO11/TDM2TDAT/IRQ9/ETHRX_ER/ETHTXD ↔ 1
GPIO12/TDM2RSYN/IRQ10/ETHRXD1/ETHSYNC ↔ 1
GPIO13/TDM2RCLK/IRQ11/ETHMDC ↔ 1
GPIO14/TDM2RDAT/IRQ12/ETHRXD0/NC ↔ 1
GPIO15/TDM1TSYN/DREQ1 ↔ 1
GPIO16/TDM1TCLK/DONE1/DRACK1 ↔ 1
GPIO17/TDM1TDAT/DACK1 ↔ 1
GPIO18/TDM1RSYN/DREQ2 ↔ 1
GPIO19/TDM1RCLK/DACK2 ↔ 1
GPIO20/TDM1RDAT ↔ 1
GPIO21/TDM0TSYN ↔ 1
GPIO22/TDM0TCLK/DONE2/DRACK2 ↔ 1
GPIO23/TDM0TDAT/IRQ13 ↔ 1
GPIO24/TDM0RSYN/IRQ14 ↔ 1
GPIO25/TDM0RCLK/IRQ15 ↔ 1
GPIO26/TDM0RDAT ↔ 1
GPIO27/URXD/DREQ1 ↔ 1
GPIO28/UTXD/DREQ2 ↔ 1
GPIO29/CHIP_ID3/ETHTX_EN ↔ 1
GPIO30/TIMER2/TMCLK/SDA ↔ 1
GPIO31/TIMER3/SCL ↔ 1
DSI/SYS.BUS/ETHERNET
MEMC
DSI
GPIO/TDM/ETHERNET/TIMERS/I2C
SYSTEM BUS
MEMC
SYS
Debug
CLK
RESET
JTAG
Ded. Ethernet
32 ↔ A[0–31]
1 ↔ TT0/HA7
1 ↔ TT1
3 ↔ TT[2–4]/CS[5–7]
5 → CS[0–4]
4 ↔ TSZ[0–3]
1 ↔ TBST
1 ↔ IRQ1/GBL
1 ↔ IRQ3/BADDR31
1 ↔ IRQ2/BADDR30
1 ↔ IRQ5/BADDR29
1 → BADDR28
1 → BADDR27
1 ↔ BR
1 ↔ BG
1 ↔ DBG
1 ↔ ABB/IRQ4
1 ↔ DBB/IRQ5
1 ↔ TS
1 ↔ AACK
1 ↔ ARTRY
32 ↔ D[0–31]
1 ↔ reserved/DP0/DREQ1/EXT_BR2
1 ↔ IRQ1/DP1/DACK1/EXT_BG2
1 ↔ IRQ2/DP2/DACK2/EXT_DBG2
1 ↔ IRQ3/DP3/DREQ2/EXT_BR3
1 ↔ IRQ4/DP4/DACK3/EXT_DBG3
1 ↔ IRQ5/DP5/DACK4/EXT_BG3
1 ↔ IRQ6/DP6/DREQ3
1 ↔ IRQ7/DP7/DREQ4
1 ↔ TA
1 ↔ TEA
1 ← NMI
1 → NMI_OUT
1 ↔ PSDVAL
1 ↔ IRQ7/INT_OUT
1 → BCTL0
1 → BCTL1/CS5
3 ↔ BM[0–2]/TC[0–2]/BNKSEL[0–2]
1 → ALE
4 → PWE[0–3]/PSDDQM[0–3]/PBS[0–3]
1 → PSDA10/PGPL0
1 → PSDWE/PGPL1
1 → POE/PSDRAS/PGPL2
1 → PSDCAS/PGPL3
1 ↔ PGTA/PUPMWAIT/PGPL4/PPBS
1 → PSDAMUX/PGPL5
1 ← EE0
1 → EE1
1 → CLKOUT
1 ← Reserved
1 ← CLKIN
1 ← PORESET
1 ↔ HRESET
1 ↔ SRESET
1 ← RSTCONF
1 ← TMS
1 ← TDI
1 ← TCK
1 ← TRST
1 → TDO
1 ← ETHRX_CLK/ETHSYNC_IN
1 ← ETHTX_CLK/ETHREF_CLK/ETHCLOCK
1 ← ETHCRS/ETHRXD

Note: Power signals are: V_{DD}, V_{DDH}, V_{CCSYN}, GND, GND_H, and GND_{SYN}. Reserved signals can be left unconnected. NC signals must not be connected.

Figure 3-1. MSC8113 External Signals

3.1 Power Signals

Table 3-2. Power and Ground Signal Inputs

Signal Name	Description
V_{DD}	**Internal Logic Power** V_{DD} dedicated for use with the device core. The voltage should be well-regulated and the input should be provided with an extremely low impedance path to the V_{DD} power rail.
V_{DDH}	**Input/Output Power** This source supplies power for the I/O buffers. The user must provide adequate external decoupling capacitors.
V_{CCSYN}	**System PLL Power** V_{CC} dedicated for use with the system Phase Lock Loop (PLL). The voltage should be well-regulated and the input should be provided with an extremely low impedance path to the V_{CC} power rail.
GND	**System Ground** An isolated ground for the internal processing logic and I/O buffers. This connection must be tied externally to all chip ground connections, except GND_{SYN}. The user must provide adequate external decoupling capacitors.
GND_{SYN}	**System PLL Ground** Ground dedicated for system PLL use. The connection should be provided with an extremely low-impedance path to ground.

3.2 Clock Signals

Table 3-3. Clock Signals

Signal Name	Type	Signal Description
CLKIN	Input	**Clock In** Primary clock input to the MSC8113 PLL.
CLKOUT	Output	**Clock Out** The bus clock.
Reserved	Input	Pull down.

3.3 Reset and Configuration Signals

Table 3-4. Reset and Configuration Signals

Signal Name	Type	Signal Description
$\overline{\text{PORESET}}$	Input	**Power-On Reset** When asserted, this line causes the MSC8113 to enter power-on reset state.
$\overline{\text{RSTCONF}}$	Input	**Reset Configuration** Used during reset configuration sequence of the chip. A detailed explanation of its function is provided in the *MSC8113 Reference Manual*. This signal is sampled upon deassertion of $\overline{\text{PORESET}}$. **Note:** When $\overline{\text{PORESET}}$ is deasserted, the MSC8113 also samples the following signals: • BM[0–2]—Selects the boot mode. • MODCK[1–2]—Selects the clock configuration. • SWTE—Enables the software watchdog timer. • DSISYNC, DSI64, CNFGS, and CHIP_ID[0–3]—Configures the DSI. Refer to **Table 3-5** for details on these signals.

Table 3-4. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
$\overline{\text{HRESET}}$	Input/ Output	**Hard Reset** When asserted as an input, this signal causes the MSC8113 to enter the hard reset state. After the device enters a hard reset state, it drives the signal as an open-drain output.
$\overline{\text{SRESET}}$	Input/ Output	**Soft Reset** When asserted as an input, this signal causes the MSC8113 to enter the soft reset state. After the device enters a soft reset state, it drives the signal as an open-drain output.

3.4 Direct Slave Interface, System Bus, Ethernet, and Interrupt Signals

The direct slave interface (DSI) is combined with the system bus because they share some common signal lines. Individual assignment of a signal to a specific signal line is configured through internal registers. **Table 3-5** describes the signals in this group. Although there are fifteen interrupt request (IRQ) connections to the core processors, there are multiple external lines that can connect to these internal signal lines. After reset, the default configuration enables only $\overline{\text{IRQ[1–7]}}$, but includes two input lines each for $\overline{\text{IRQ[1–3]}}$ and $\overline{\text{IRQ7}}$. The designer must select one line for each required interrupt and reconfigure the other external signal line or lines for alternate functions. Additional alternate IRQ lines and $\overline{\text{IRQ[8–15]}}$ are enabled through the GPIO signal lines.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals

Signal Name	Type	Description
HD0	Input/ Output	**Host Data Bus 0** Bit 0 of the DSI data bus.
SWTE	Input	**Software Watchdog Timer Disable**. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
HD1	Input/ Output	**Host Data Bus 1** Bit 1 of the DSI data bus.
DSISYNC	Input	**DSI Synchronous** Distinguishes between synchronous and asynchronous operation of the DSI. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
HD2	Input/ Output	**Host Data Bus 2** Bit 2 of the DSI data bus.
DSI64	Input	**DSI 64** Defines the width of the DSI and SYSTEM Data buses. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
HD3	Input/ Output	**Host Data Bus 3** Bit 3 of the DSI data bus.
MODCK1	Input	**Clock Mode 1** Defines the clock frequencies. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
HD4	Input/ Output	**Host Data Bus 4** Bit 4 of the DSI data bus.
MODCK2	Input	**Clock Mode 2** Defines the clock frequencies. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
HD5	Input/ Output	**Host Data Bus 5** Bit 5 of the DSI data bus.
CNFGS	Input	**Configuration Source** One signal out of two that indicates reset configuration mode. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
HD[6–31]	Input/ Output	**Host Data Bus 6–31** Bits 6–31 of the DSI data bus.
HD[32–39]	Input/ Output	**Host Data Bus 32–39** Bits 32–39 of the DSI data bus.
D[32–39]	Input/ Output	**System Bus Data 32–39** For write transactions, the bus master drives valid data on this bus. For read transactions, the slave drives valid data on this bus.
Reserved	Input	If the Ethernet port is enabled and multiplexed with the DSI/System bus, these pins are reserved and can be left unconnected.
HD40	Input/ Output	**Host Data Bus 40** Bit 40 of the DSI data bus.
D40	Input/ Output	**System Bus Data 40** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHRXD0	Input	**Ethernet Receive Data 0** In MII and RMII modes, bit 0 of the Ethernet receive data.
HD41	Input/ Output	**Host Data Bus 41** Bit 41 of the DSI data bus.
D41	Input/ Output	**System Bus Data 41** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHRXD1	Input	**Ethernet Receive Data 1** In MII and RMII modes, bit 1 of the Ethernet receive data.
HD42	Input/ Output	**Host Data Bus 42** Bit 42 of the DSI data bus.
D42	Input/ Output	**System Bus Data 42** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHRXD2	Input	**Ethernet Receive Data 2** In MII mode only, bit 2 of the Ethernet receive data.
Reserved	Input	In RMII mode, this pin is reserved and can be left unconnected.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
HD43	Input/ Output	**Host Data Bus 43** Bit 43 of the DSI data bus.
D43	Input/ Output	**System Bus Data 43** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHRXD3	Input	**Ethernet Receive Data 3** In MII mode only, bit 3 of the Ethernet receive data.
Reserved	Input	In RMII mode, this pin is reserved and can be left unconnected.
HD[44–45]	Input/ Output	**Host Data Bus 44–45** Bits 44–45 of the DSI data bus.
D[44–56]	Input/ Output	**System Bus Data 44–45** For write transactions, the bus master drives valid data on this bus. For read transactions, the slave drives valid data on this bus.
Reserved	Input	If the Ethernet port is enabled and multiplexed with the DSI/System bus, these pins are reserved and can be left unconnected.
HD46	Input/ Output	**Host Data Bus 46** Bit 46 of the DSI data bus.
D46	Input/ Output	**System Bus Data 46** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTXD0	Output	**Ethernet Transmit Data 0** In MII and RMII modes, bit 0 of the Ethernet transmit data.
HD47	Input/ Output	**Host Data Bus 47** Bit 47 of the DSI data bus.
D47	Input/ Output	**System Bus Data 47** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTXD1	Output	**Ethernet Transmit Data 1** In MII and RMII modes, bit 1 of the Ethernet transmit data.
HD48	Input/ Output	**Host Data Bus 48** Bit 48 of the DSI data bus.
D48	Input/ Output	**System Bus Data 48** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTXD2	Output	**Ethernet Transmit Data 2** In MII mode only, bit 2 of the Ethernet transmit data.
Reserved	Input	In RMII mode, this pin is reserved and can be left unconnected.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
HD49	Input/ Output	**Host Data Bus 49** Bit 49 of the DSI data bus.
D49	Input/ Output	**System Bus Data 49** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTXD3	Output	**Ethernet Transmit Data 3** In MII mode only, bit 3 of the Ethernet transmit data.
Reserved	Input	In RMII mode, this pin is reserved and can be left unconnected.
HD[50–53]	Input/ Output	**Host Data Bus 50–53** Bits 50–53 of the DSI data bus.
D[50–53]	Input/ Output	**System Bus Data 50–53** For write transactions, the bus master drives valid data on this bus. For read transactions, the slave drives valid data on this bus.
Reserved	Input	If the Ethernet port is enabled and multiplexed with the DSI/System bus, these pins are reserved and can be left unconnected.
HD54	Input/ Output	**Host Data Bus 54** Bit 54 of the DSI data bus.
D54	Input/ Output	**System Bus Data 54** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTX_EN	Output	**Ethernet Transmit Data Enable** In MII and RMII modes, indicates that the transmit data is valid.
HD55	Input/ Output	**Host Data Bus 55** Bit 55 of the DSI data bus.
D55	Input/ Output	**System Bus Data 55** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHTX_ER	Output	**Ethernet Transmit Data Error** In MII mode only, indicates a transmit data error.
Reserved	Input	In RMII mode, this pin is reserved and can be left unconnected.
HD56	Input/ Output	**Host Data Bus 56** Bit 56 of the DSI data bus.
D56	Input/ Output	**System Bus Data 56** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus.
ETHRX_DV	Input	**Ethernet Receive Data Valid** Indicates that the receive data is valid.
ETHCRS_DV	Input	**Ethernet Carrier Sense/Receive Data Valid** In RMII mode, indicates that a carrier is detected and after the connection is established that the receive data is valid.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
HD57 D57 ETHRX_ER	Input/ Output Input/ Output Input	**Host Data Bus 57** Bit 57 of the DSI data bus. **System Bus Data 57** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus. **Ethernet Receive Data Error** In MII and RMII modes, indicates a receive data error.
HD58 D58 ETHMDC	Input/ Output Input/ Output Output	**Host Data Bus 58** Bit 58 of the DSI data bus. **System Bus Data 58** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus. **Ethernet Management Clock** In MII and RMII modes, used for the MDIO reference clock.
HD59 D59 ETHMDIO	Input/ Output Input/ Output Input/ Output	**Host Data Bus 59** Bit 59 of the DSI data bus. **System Bus Data 59** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus. **Ethernet Management Data** In MII and RMII modes, used for station management data input/output.
HD60 D60 ETHCOL Reserved	Input/ Output Input/ Output Input/ Output Input	**Host Data Bus 60** Bit 60 of the DSI data bus. **System Bus Data 60** For write transactions, the bus master drives valid data on this line. For read transactions, the slave drives valid data on this bus. **Ethernet Collision** In MII mode only, indicates that a collision was detected. In RMII mode, this pin is reserved and can be left unconnected.
HD[61–63] D[61–63] Reserved	Input/ Output Input/ Output Input	**Host Data Bus 61–63** Bits 61–63 of the DSI data bus. **System Bus Data 61–63** For write transactions, the bus master drives valid data on this bus. For read transactions, the slave drives valid data on this bus. If the Ethernet port is enabled and multiplexed with the DSI/System bus, these pins are reserved and can be left unconnected.
HCID[0–2]	Input	**Host Chip ID 0–2** With HCID3, carries the chip ID of the DSI. The DSI is accessed only if $\overline{\text{HCS}}$ is asserted and HCID[0–3] matches the Chip_ID, or if $\overline{\text{HBCS}}$ is asserted.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
HCID3	Input	**Host Chip ID 3** With HCI[0–2], carries the chip ID of the DSI. The DSI is accessed only if HCS is asserted and HCID[0–3] matches the Chip_ID, or if HBCS is asserted.
HA8	Input	**Host Bus Address 8** Used by an external host to access the internal address space.
HA[11–29]	Input	**Host Bus Address 11–29** Used by external host to access the internal address space.
HWBS[0–3]	Input	**Host Write Byte Strobes** (In Asynchronous dual mode) One bit per byte is used as a strobe for host write accesses.
HDBS[0–3]	Input	**Host Data Byte Strobe** (in Asynchronous single mode) One bit per byte is used as a strobe for host read or write accesses
HWBE[0–3]	Input	**Host Write Byte Enable** (In Synchronous dual mode) One bit per byte is used to indicate a valid data byte for host read or write accesses.
HDBE[0–3]	Input	**Host Data Byte Enable** (in Synchronous single mode) One bit per byte is used as a strobe enable for host write accesses
HWBS[4–7]	Input	**Host Write Byte Strobes** (In Asynchronous dual mode) One bit per byte is used as a strobe for host write accesses.
HDBS[4–7]	Input	**Host Data Byte Strobe** (in Asynchronous single mode) One bit per byte is used as a strobe for host read or write accesses
HWBE[4–7]	Input	**Host Write Byte Enable** (In Synchronous dual mode) One bit per byte is used to indicate a valid data byte for host write accesses.
HDBE[4–7]	Input	**Host Data Byte Enable** (in Synchronous single mode) One bit per byte is used as a strobe enable for host read or write accesses
PWE[4–7]	Output	**System Bus Write Enable** Outputs of the bus general-purpose chip-select machine (GPCM). These pins select byte lanes for write operations.
PSDDQM[4–7]	Output	**System Bus SDRAM DQM** From the SDRAM control machine. These pins select specific byte lanes of SDRAM devices.
PBS[4–7]	Output	**System Bus UPM Byte Select** From the UPM in the memory controller, these signals select specific byte lanes during memory operations. The timing of these pins is programmed in the UPM. The actual driven value depends on the address and size of the transaction and the port size of the accessed device.
HRDS	Input	**Host Read Data Strobe** (In Asynchronous dual mode) Used as a strobe for host read accesses.
HRW	Input	**Host Read/Write Select** (in Asynchronous/Synchronous single mode) Host read/write select.
HRDE	Input	**Host Read Data Enable** (In Synchronous dual mode) Indicates valid data for host read accesses.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
$\overline{\text{HBRST}}$	Input	**Host Burst** The host asserts this pin to indicate that the current transaction is a burst transaction in synchronous mode only.
HDST[0–1] HA[9–10]	Input	**Host Data Structure 0–1** Defines the data structure of the host access in DSI little-endian mode. **Host Bus Address 9–10** Used by an external host to access the internal address space.
$\overline{\text{HCS}}$	Input	**Host Chip Select** DSI chip select. The DSI is accessed only if $\overline{\text{HCS}}$ is asserted and HCID[0–3] matches the Chip_ID.
$\overline{\text{HBCS}}$	Input	**Host Broadcast Chip Select** DSI chip select for broadcast mode. Enables more than one DSI to share the same host chip-select pin for broadcast write accesses.
$\overline{\text{HTA}}$	Output	**Host Transfer Acknowledge** Upon a read access, indicates to the host when the data on the data bus is valid. Upon a write access, indicates to the host that the data on the data bus was written to the DSI write buffer.
HCLKIN	Input	**Host Clock Input** Host clock signal for DSI synchronous mode.
A[0–31]	Input/ Output	**Address Bus** When the MSC8113 is in external master bus mode, these pins function as the system address bus. The MSC8113 drives the address of its internal bus masters and responds to addresses generated by external bus masters. When the MSC8113 is in internal master bus mode, these pins are used as address lines connected to memory devices and are controlled by the MSC8113 memory controller.
TT0 HA7	Input/ Output	**Bus Transfer Type 0** The bus master drives this pins during the address tenure to specify the type of the transaction. **Host Bus Address 7** Used by an external host to access the internal address space.
TT1	Input/ Output	**Bus Transfer Type 1** The bus master drives this pins during the address tenure to specify the type of the transaction. Some applications use only the TT1 signal, for example, from MSC8113 to MSC8113 or MSC8113 to MSC8101 and *vice versa*. In these applications, TT1 functions as read/write signal.
TT[2–4] CS[5–7]	Input/ Output Output	**Bus Transfer Type 2–4** The bus master drives these pins during the address tenure to specify the type of the transaction. **Chip Select 5–7** Enables specific memory devices or peripherals connected to the system bus.
CS[0–4]	Output	**Chip Select 0–4** Enables specific memory devices or peripherals connected to the system bus.
TSZ[0–3]	Input/ Output	**Transfer Size 0–3** The bus master drives these pins with a value indicating the number of bytes transferred in the current transaction.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
$\overline{\text{TBST}}$	Input/ Output	**Bus Transfer Burst** The bus master asserts this pin to indicate that the current transaction is a burst transaction (transfers eight words).
$\overline{\textbf{IRQ1}}$ $\overline{\text{GBL}}$	Input Output	**Interrupt Request 1**[1] One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140. **Global**[1] When a master within the MSC8113 initiates a bus transaction, it drives this pin. Assertion of this pin indicates that the transfer is global and should be snooped by caches in the system.
$\overline{\textbf{IRQ3}}$ BADDR31	Input Output	**Interrupt Request 3**[1] One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140. **Burst Address 31**[1] There are five burst address output pins, which are outputs of the memory controller. These pins connect directly to burstable memory devices without internal address incrementors controlled by the MSC8113 memory controller.
$\overline{\textbf{IRQ2}}$ BADDR30	Input Output	**Interrupt Request 2**[1] One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140. **Burst Address 30**[1] There are five burst address output pins, which are outputs of the memory controller. These pins connect directly to burstable memory devices without internal address incrementors controlled by the MSC8113 memory controller.
$\overline{\textbf{IRQ5}}$ BADDR29	Input Output	**Interrupt Request 5**[1] One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140. **Bus Burst Address 29**[1] There are five burst address output pins, which are outputs of the memory controller. These pins connect directly to burstable memory devices without internal address incrementors controlled by the MSC8113 memory controller.
BADDR28	Output	**Burst Address 28** There are five burst address output pins, which are outputs of the memory controller. These pins connect directly to burstable memory devices without internal address incrementors controlled by the MSC8113 memory controller.
BADDR27	Output	**Burst Address 27** There are five burst address output pins, which are outputs of the memory controller. These pins connect directly to burstable memory devices without internal address incrementors controlled by the MSC8113 memory controller.
$\overline{\text{BR}}$	Input/ Output	**Bus Request**[2] When an external arbiter is used, the MSC8113 asserts this pin as an output to request ownership of the bus. When the MSC8113 controller is used as an internal arbiter, an external master asserts this pin as an input to request bus ownership.
$\overline{\text{BG}}$	Input/ Output	**Bus Grant**[2] When the MSC8113 acts as an internal arbiter, it asserts this pin as an output to grant bus ownership to an external bus master. When an external arbiter is used, it asserts this pin as an input to grant bus ownership to the MSC8113.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
$\overline{\text{DBG}}$	Input/ Output	**Data Bus Grant**[2] When the MSC8113 acts as an internal arbiter, it asserts this pin as an output to grant data bus ownership to an external bus master. When an external arbiter is used, it asserts this pin as an input to grant data bus ownership to the MSC8113.
$\overline{\textbf{ABB}}$ $\overline{\text{IRQ4}}$	Input/ Output Input	**Address Bus Busy**[1] The MSC8113 asserts this pin as an output for the duration of the address bus tenure. Following an $\overline{\text{AACK}}$, which terminates the address bus tenure, the MSC8113 deasserts $\overline{\text{ABB}}$ for a fraction of a bus cycle and then stops driving this pin. The MSC8113 does not assume bus ownership as long as it senses this pin is asserted as an input by an external bus master. **Interrupt Request 4** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
$\overline{\textbf{DBB}}$ $\overline{\text{IRQ5}}$	Input/ Output Input	**Data Bus Busy**[1] The MSC8113 asserts this pin as an output for the duration of the data bus tenure. Following a $\overline{\text{TA}}$, which terminates the data bus tenure, the MSC8113 deasserts $\overline{\text{DBB}}$ for a fraction of a bus cycle and then stops driving this pin. The MSC8113 does not assume data bus ownership as long as it senses that this pin is asserted as an input by an external bus master. **Interrupt Request 5** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
$\overline{\text{TS}}$	Input/ Output	**Bus Transfer Start** Assertion of this pin signals the beginning of a new address bus tenure. The MSC8113 asserts this signal when one of its internal bus masters begins an address tenure. When the MSC8113 senses that this pin is asserted by an external bus master, it responds to the address bus tenure as required (snoop if enabled, access internal MSC8113 resources, memory controller support).
$\overline{\text{AACK}}$	Input/ Output	**Address Acknowledge** A bus slave asserts this signal to indicate that it has identified the address tenure. Assertion of this signal terminates the address tenure.
$\overline{\text{ARTRY}}$	Input/ Output	**Address Retry** Assertion of this signal indicates that the bus master should retry the bus transaction. An external master asserts this signal to enforce data coherency with its caches and to prevent deadlock situations.
D[0–31]	Input/ Output	**Data Bus Bits 0–31** In write transactions, the bus master drives the valid data on this bus. In read transactions, the slave drives the valid data on this bus.
Reserved DP0 DREQ1 EXT_BR2	Input Input/ Output Input Input	The primary configuration selection (default after reset) is reserved. **System Bus Data Parity 0** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 0 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 0 and D[0–7]. **DMA Request 1** Used by an external peripheral to request DMA service. **External Bus Request 2** An external master asserts this pin to request bus ownership from the internal arbiter.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
$\overline{\text{IRQ1}}$	Input	**Interrupt Request 1** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP1	Input/ Output	**System Bus Data Parity** 1 The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 1 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 1 and D[8–15].
$\overline{\text{DACK1}}$	Output	**DMA Acknowledge 1** The DMA drives this output to acknowledge the DMA transaction on the bus.
$\overline{\text{EXT\_BG2}}$	Output	**External Bus Grant 2**[2] The MSC8113 asserts this pin to grant bus ownership to an external bus master.
$\overline{\text{IRQ2}}$	Input	**Interrupt Request 2** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP2	Input/ Output	**System Bus Data Parity 2** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 2 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 2 and D[16–23].
$\overline{\text{DACK2}}$	Output	**DMA Acknowledge 2** The DMA drives this output to acknowledge the DMA transaction on the bus.
$\overline{\text{EXT\_DBG2}}$	Output	**External Data Bus Grant 2**[2] The MSC8113 asserts this pin to grant data bus ownership to an external bus master.
$\overline{\text{IRQ3}}$	Input	**Interrupt Request 3** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP3	Input/ Output	**System Bus Data Parity 3** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 3 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 3 and D[24–31].
DREQ2	Input	**DMA Request 2** Used by an external peripheral to request DMA service.
EXT_BR3	Input	**External Bus Request 3**[2] An external master should assert this pin to request bus ownership from the internal arbiter.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
$\overline{\mathrm{IRQ4}}$	Input	**Interrupt Request 4** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP4	Input/ Output	**System Bus Data Parity 4** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 4 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 4 and D[32–39].
$\overline{\mathrm{DACK3}}$	Output	**DMA Acknowledge 3** The DMA drives this output to acknowledge the DMA transaction on the bus.
$\overline{\mathrm{EXT\_DBG3}}$	Output	**External Data Bus Grant 3**[2] The MSC8113 asserts this pin to grant data bus ownership to an external bus master.
$\overline{\mathrm{IRQ5}}$	Input	**Interrupt Request 5** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP5	Input/ Output	**System Bus Data Parity 5** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 5 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 5 and D[40–47].
$\overline{\mathrm{DACK4}}$	Output	**DMA Acknowledge 4** The DMA drives this output to acknowledge the DMA transaction on the bus.
$\overline{\mathrm{EXT\_BG3}}$	Output	**External Bus Grant 3**[2] The MSC8113 asserts this pin to grant bus ownership to an external bus.
$\overline{\mathrm{IRQ6}}$	Input	**Interrupt Request 6** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP6	Input/ Output	**System Bus Data Parity 6** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 6 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 6 and D[48–55].
DREQ3	Input	**DMA Request 3** Used by an external peripheral to request DMA service.
$\overline{\mathrm{IRQ7}}$	Input	**Interrupt Request 7** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
DP7	Input/ Output	**System Bus Data Parity 7** The agent that drives the data bus also drives the data parity signals. The value driven on the data parity 7 pin should give odd parity (odd number of ones) on the group of signals that includes data parity 7 and D[56–63].
DREQ4	Input	**DMA Request 4** Used by an external peripheral to request DMA service.

Table 3-5. DSI, System Bus, Ethernet, and Interrupt Signals (Continued)

Signal Name	Type	Description
TA	Input/ Output	**Transfer Acknowledge** Indicates that a data beat is valid on the data bus. For single-beat transfers, TA assertion indicates the termination of the transfer. For burst transfers, TA is asserted eight times to indicate the transfer of eight data beats, with the last assertion indicating the termination of the burst transfer.
TEA	Input/ Output	**Transfer Error Acknowledge** Assertion indicates a failure of the data tenure transaction.The masters within the MSC8113 monitor the state of this pin. The MSC8113 internal bus monitor can assert this pin if it identifies a bus transfer that does not complete.
NMI	Input	**Non-Maskable Interrupt** When an external device asserts this line, it generates an non-maskable interrupt in the MSC8113, which is processed internally (default) or is directed to an external host for processing (see NMI_OUT).
NMI_OUT	Output	**Non-Maskable Interrupt Output** An open-drain pin driven from the MSC8113 internal interrupt controller. Assertion of this output indicates that a non-maskable interrupt is pending in the MSC8113 internal interrupt controller, waiting to be handled by an external host.
PSDVAL	Input/ Output	**Port Size Data Valid** Indicates that a data beat is valid on the data bus. The difference between the TA pin and the PSDVAL pin is that the TA pin is asserted to indicate data transfer terminations, while the PSDVAL signal is asserted with each data beat movement. When TA is asserted, PSDVAL is always asserted. However, when PSDVAL is asserted, TA is not necessarily asserted. For example, if the DMA initiates a double word (2 × 64 bits) transaction to a memory device with a 32-bit port size, PSDVAL is asserted three times without TA and, finally, both pins are asserted to terminate the transfer.
IRQ7 INT_OUT	Input Output	**Interrupt Request 7** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140. **Interrupt Output** Assertion of this output indicates that an unmasked interrupt is pending in the MSC8113 internal interrupt controller.

Notes:
1. See the *System Interface Unit (SIU)* chapter in the *MSC8113 Reference Manual* for details on how to configure these pins.
2. When used as the bus control arbiter, the MSC8113 can support up to three external bus masters. Each master uses its own set of Bus Request, Bus Grant, and Data Bus Grant signals (BR/BG/DBG, EXT_BR2/EXT_BG2/EXT_DBG2, and EXT_BR3/EXT_BG3/EXT_DBG3). Each of these signal sets must be configured to indicate whether the external master is or is not a MSC8113 master device. See the Bus Configuration Register (BCR) description in the System Interface Unit (SIU) chapter in the *MSC8113 Reference Manual* for details on how to configure these pins. The second and third set of pins is defined by EXT_xxx to indicate that they can only be used with external master devices. The first set of pins (BR/BG/DBG) have a dual function. When the MSC8113 is not the bus arbiter, it uses these signals (BR/BG/DBG) to obtain master control of the bus.

3.5 Memory Controller Signals

Refer to the Memory Controller chapter in the *MSC8113 Reference Manual* for details on configuring these signals.

Table 3-6. Memory Controller Signals

Signal Name	Type	Description
$\overline{\text{BCTL0}}$	Output	**System Bus Buffer Control 0** Controls buffers on the data bus. Usually used with $\overline{\text{BCTL1}}$. The exact function of this pin is defined by the value of SIUMCR[BCTLC].
$\overline{\text{BCTL1}}$	Output	**System Bus Buffer Control 1** Controls buffers on the data bus. Usually used with $\overline{\text{BCTL0}}$. The exact function of this pin is defined by the value of SIUMCR[BCTLC].
$\overline{\text{CS5}}$	Output	**System and Local Bus Chip Select 5** Enables specific memory devices or peripherals connected to MSC8113 buses.
BM[0–2]	Input	**Boot Mode 0–2** Defines the boot mode of the MSC8113. This signal is sampled on $\overline{\text{PORESET}}$ deassertion.
TC[0–2]	Input/ Output	**Transfer Code 0–2** The bus master drives these pins during the address tenure to specify the type of the code.
BNKSEL[0–2]	Output	**Bank Select 0–2** Selects the SDRAM bank when the MSC8113 is in 60x-compatible bus mode.
ALE	Output	**Address Latch Enable** Controls the external address latch used in an external master bus.
$\overline{\text{PWE[0–3]}}$	Output	**System Bus Write Enable** Outputs of the bus general-purpose chip-select machine (GPCM). These pins select byte lanes for write operations.
$\overline{\text{PSDDQM[0–3]}}$	Output	**System Bus SDRAM DQM** From the SDRAM control machine. These pins select specific byte lanes of SDRAM devices.
$\overline{\text{PBS[0–3]}}$	Output	**System Bus UPM Byte Select** From the UPM in the memory controller, these signals select specific byte lanes during memory operations. The timing of these pins is programmed in the UPM. The actual driven value depends on the address and size of the transaction and the port size of the accessed device.
PSDA10	Output	**System Bus SDRAM A10** From the bus SDRAM controller. The precharge command defines which bank is precharged. When the row address is driven, it is a part of the row address. When column address is driven, it is a part of column address.
PGPL0	Output	**System Bus UPM General-Purpose Line 0** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM.
$\overline{\text{PSDWE}}$	Output	**System Bus SDRAM Write Enable** From the bus SDRAM controller. Should connect to SDRAM WE input.
PGPL1	Output	**System Bus UPM General-Purpose Line 1** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM.

Table 3-6. Memory Controller Signals (Continued)

Signal Name	Type	Description
$\overline{\text{POE}}$ $\overline{\text{PSDRAS}}$ PGPL2	Output Output Output	**System Bus Output Enable** From the bus GPCM. Controls the output buffer of memory devices during read operations. **System Bus SDRAM** $\overline{\text{RAS}}$ From the bus SDRAM controller. Should connect to SDRAM $\overline{\text{RAS}}$ input. **System Bus UPM General-Purpose Line 2** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM.
$\overline{\text{PSDCAS}}$ PGPL3	Output Output	**System Bus SDRAM** $\overline{\text{CAS}}$ From the bus SDRAM controller. Should connect to SDRAM $\overline{\text{CAS}}$ input. **System Bus UPM General-Purpose Line 3** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM.
$\overline{\text{PGTA}}$ PUPMWAIT PGPL4 $\overline{\text{PPBS}}$	Input Input Output Output	**System GPCM TA** Terminates external transactions during GPCM operation. Requires an external pull-up resistor for proper operation. **System Bus UPM Wait** An external device holds this pin low to force the UPM to wait until the device is ready to continue the operation. **System Bus UPM General-Purpose Line 4** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM. **System Bus Parity Byte Select** In systems that store data parity in a separate chip, this output is used as the byte-select for that chip.
PSDAMUX PGPL5	Output Output	**System Bus SDRAM Address Multiplexer** Controls the system bus SDRAM address multiplexer when the MSC8113 is in external master mode. **System Bus UPM General-Purpose Line 5** One of six general-purpose output lines from the UPM. The values and timing of this pin are programmed in the UPM.

3.6 GPIO, TDM, UART, and Timer Signals

The general-purpose input/output (GPIO), time-division multiplexed (TDM), universal asynchronous receiver/transmitter (UART), and timer signals are grouped together because they use a common set of signal lines. Individual assignment of a signal to a specific signal line is configured through internal registers. **Table 3-7** describes the signals in this group.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals

Signal Name	Type	Description
GPIO0	Input/ Output	**General-Purpose Input Output 0** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
CHIP_ID0	Input	**Chip ID 0** Determines the chip ID of the MSC8113 DSI. It is sampled on the rising edge of $\overline{\mathrm{PORESET}}$ signal.
$\overline{\mathrm{IRQ4}}$	Input	**Interrupt Request 4** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140 core.
ETHTXD0	Output	**Ethernet Transmit Data 0** For MII or RMII mode, bit 0 of the Ethernet transmit data.
GPIO1	Input/ Output	**General-Purpose Input Output 1** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TIMER0	Input/ Output	**Timer 0** Each signal is configured as either input to or output from the counter. For details, see **Chapter 22**, *Timers*.
CHIP_ID1	Input	**Chip ID 1** Determines the chip ID of the MSC8113 DSI. It is sampled on the rising edge of $\overline{\mathrm{PORESET}}$ signal.
$\overline{\mathrm{IRQ5}}$	Input	**Interrupt Request 5** One of the fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140 core.
ETHTXD1	Output	**Ethernet Transmit Data 1** For MII or RMII mode, bit 1 of the Ethernet transmit data.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO2	Input/ Output	**General-Purpose Input Output 2** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TIMER1	Input/ Output	**Timer 1** Each signal is configured as either input to or output from the counter. For details, see **Chapter 22**, *Timers*.
CHIP_ID2	Input	**Chip ID 2** Determines the chip ID of the MSC8113 DSI. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
$\overline{\text{IRQ6}}$	Input	**Interrupt Request 6** One of the fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140 core.
GPIO3	Input/ Output	**General-Purpose Input Output 3** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3TSYN	Input/ Output	**TDM3 Transmit Frame Sync** Transmit frame sync for TDM 3. See **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ1}}$	Input	**Interrupt Request 1** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHTXD2	Output	**Ethernet Transmit Data 2** For MII mode only, bit 2 of the Ethernet transmit data.
Reserved	Output	In RMII or SMII mode, this signal is reserved and can be left unconnected.
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3TCLK	Input	**TDM3 Transmit Clock** Transmit Clock for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ2}}$	Input	**Interrupt Request 2** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHTX_ER	Output	**Ethernet Transmit Data Error** For MII mode only, indicates whether a transmit data error occurred.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO5	Input/ Output	**General-Purpose Input/Output 5** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3TDAT	Input/ Output	**TDM3 Serial Transmitter Data** The serial transmit data signal for TDM 3. As an output, it provides the DATA_D signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ3}}$	Input	**Interrupt Request 3** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRXD3	Input	**Ethernet Receive Data 3** For MII mode only, bit 3 of the Ethernet receive data.
Reserved	Input	For RMII or SMII mode, this pin is reserved and can be left unconnected.
GPIO6	Input/ Output	**General-Purpose Input Output 6** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3RSYN	Input/ Output	**TDM3 Receive Frame Sync** The receive sync signal for TDM 3. As an input, this can be the DATA_B data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ4}}$	Input	**Interrupt Request 4** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRXD2	Input	**Ethernet Receive Data 2** For MII mode only, bit 2 of the Ethernet receive data.
Reserved	Input	For RMII or SMII mode, this pin is reserved and can be left unconnected.
GPIO7	Input/ Output	**General-Purpose Input Output 7** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3RCLK	Input/ Output	**TDM3 Receive Clock** The receive clock signal for TDM 3. As an output, this can be the DATA_C data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ5}}$	Input	**Interrupt Request 5** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHTXD3	Output	**Ethernet Transmit Data 3** For MII mode only, bit 3 of the Ethernet transmit data.
Reserved	Output	For RMII or SMII mode, this pin is reserved and can be left unconnected.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO8	Input/ Output	**General-Purpose Input Output 8** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM3RDAT	Input/ Output	**TDM3 Serial Receiver Data** The receive data signal for TDM 3. As an input, this can be the DATA_A data signal for TDM 3. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ6}}$	Input	**Interrupt Request 6** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHCOL	Input	**Ethernet Collision** For MII mode only, indicates whether a collision was detected.
Reserved	Input	For RMII or SMII mode, this pin is reserved and can be left unconnected.
GPIO9	Input/ Output	**General-Purpose Input Output 9** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2TSYN	Input/ Output	**TDM2 Transmit frame Sync** Transmit Frame Sync for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ7}}$	Input	**Interrupt Request 7** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHMDIO	Input/ Output	**Ethernet Management Data** Station management data input/output line in MII, RMII, and SMII modes.
GPIO10	Input/ Output	**General-Purpose Input Output 10** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2TCLK	Input	**TDM 2 Transmit Clock** Transmit Clock for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ8}}$	Input	**Interrupt Request 8** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRX_DV	Input	**Ethernet Receive Data Valid** In MII mode, this signal indicates that the receive data is valid.
ETHCRS_DV	Input	**Ethernet Carrier Sense/Receive Data Valid** In RMII mode, this signal indicates that a carrier is sense or that the receive data is valid.
NC	Input	**Not Connected** For SMII mode, this signal must be left unconnected.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO11	Input/ Output	**General-Purpose Input Output 11** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2TDAT	Input/ Output	**TDM2 Serial Transmitter Data** The transmit data signal for TDM 2. As an output, this can be the DATA_D data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ9}}$	Input	**Interrupt Request 9** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRX_ER	Input	**Ethernet Receive Data Error** In MII and RMII modes indicates that a receive data error occurred.
ETHTXD	Output	**Ethernet Transmit Data** In SMII, used as the Ethernet transmit data line.
GPIO12	Input/ Output	**General-Purpose Input Output 12** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2RSYN	Input/ Output	**TDM2 Receive Frame Sync** The receive sync signal for TDM 2. As an input, this can be the DATA_B data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ10}}$	Input	**Interrupt Request 10** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRXD1	Input	**Ethernet Receive Data 1** In MII or RMII mode, bit 1 of the Ethernet receive data.
ETHSYNC	Output	**Ethernet Sync Signal** In SMII mode, this is the SMII sync signal.
GPIO13	Input/ Output	**General-Purpose Input Output 13** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2RCLK	Input/ Output	**TDM2 Receive Clock** The receive clock signal for TDM 2. As an input, this can be the DATA_C data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ11}}$	Input	**Interrupt Request 11** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHMDC	Output	**Ethernet Management Clock** Used for the MDIO reference clock for MII, RMII, and SMII modes.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM2RDAT	Input/ Output Input	**TDM2 Serial Receiver Data** The receive data signal for TDM 2. As an input, this can be the DATA_A data signal for TDM 2. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ12}}$	Input	**Interrupt Request 12** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
ETHRXD0	Input	**Ethernet Receive Data 0** Bit 0 of the Ethernet receive data (MII and RMII).
NC	Input	**Not Connected** For SMII mode, this signal must be left unconnected.
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1TSYN	Input/ Output	**TDM1 Transmit frame Sync** Transmit Frame Sync for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
DREQ1	Input	**DMA Request 1** Used by an external peripheral to request DMA service.
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1TCLK	Input	**TDM1 Transmit Clock** Transmit Clock for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{DONE1}}$	Input/ Output	**DMA Done 1** Signifies that the channel must be terminated. If the DMA generates $\overline{\text{DONE}}$, the channel handling this peripheral is inactive. As an input to the DMA, $\overline{\text{DONE}}$ closes the channel much like a normal channel closing. See the *MSC8113 Reference Manual* chapters on DMA and GPIO for information on configuring the $\overline{\text{DRACK}}$ or $\overline{\text{DONE}}$ mode and pin direction.
$\overline{\text{DRACK1}}$	Output	**DMA Data Request Acknowledge 1** Asserted by the DMA controller to indicate that the DMA controller has sampled the peripheral request.
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1TDAT	Input/ Output	**TDM1 Serial Transmitter Data** The transmit data signal for TDM 1. As an output, this can be the DATA_D data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{DACK1}}$	Output	**DMA Acknowledge 1** The DMA controller drives this output to acknowledge the DMA transaction on the bus.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1RSYN	Input/ Output	**TDM1 Receive Frame Sync** The receive sync signal for TDM 1. As an input, this can be the DATA_B data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
DREQ2	Input	**DMA Request 1** Used by an external peripheral to request DMA service.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1RCLK	Input/ Output	**TDM1 Receive Clock** The receive clock signal for TDM 1. As an input, this can be the DATA_C data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{DACK2}}$	Output	**DMA Acknowledge 2** The DMA controller drives this output to acknowledge the DMA transaction on the bus.
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM1RDAT	Input/ Output	**TDM1 Serial Receiver Data** The receive data signal for TDM 1. As an input, this can be the DATA_A data signal for TDM 1. For configuration details, see **Chapter 20**, *TDM Interface*.
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0TSYN	Input/ Output	**TDM0 Transmit frame Sync** Transmit Frame Sync for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
GPIO22	Input/ Output	**General-Purpose Input Output 22** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0TCLK	Input	**TDM 0 Transmit Clock** Transmit Clock for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{DONE2}}$	Input/ Output	**DMA Done 2** Signifies that the channel must be terminated. If the DMA generates DONE, the channel handling this peripheral is inactive. As an input to the DMA, DONE closes the channel much like a normal channel closing. **Note:** See the *MSC8113 Reference Manual* chapters on DMA and GPIO for information on configuring the $\overline{\text{DRACK}}$ or $\overline{\text{DONE}}$ mode and pin direction.
$\overline{\text{DRACK2}}$	Output	**DMA Data Request Acknowledge 2** Asserted by the DMA controller to indicate that the DMA controller has sampled the peripheral request.

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0TDAT	Input/ Output	**TDM0 Serial Transmitter Data** The transmit data signal for TDM 0. As an output, this can be the DATA_D data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ13}}$	Input	**Interrupt Request 13** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0RSYN	Input/ Output	**TDM0 Receive Frame Sync** The receive sync signal for TDM 0. As an input, this can be the DATA_B data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0RCLK	Input/ Output	**TDM0 Receive Clock** The receive clock signal for TDM 0. As an input, this can be the DATA_C data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** One of fifteen external lines that can request a service routine, via the internal interrupt controller, from the SC140.
GPIO26	Input/ Output	**General-Purpose Input Output 26** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TDM0RDAT	Input/ Output	**TDM0 Serial Receiver Data** The receive data signal for TDM 0. As an input, this can be the DATA_A data signal for TDM 0. For configuration details, see **Chapter 20**, *TDM Interface*.
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
URXD	Input	**UART Receive Data**
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
UTXD	Output	**UART Transmit Data**

Table 3-7. GPIO, TDM, UART, Ethernet, and Timer Signals (Continued)

Signal Name	Type	Description
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
CHIP_ID3	Input	**Chip ID 3** Determines the chip ID of the MSC8113 DSI. It is sampled on the rising edge of $\overline{\text{PORESET}}$ signal.
ETHTX_EN	Output	**Ethernet Transmit Enable** Enables the Ethernet transmit controller for MII and RMII modes.
GPIO30	Input/ Output	**General-Purpose Input Output 30** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TIMER2	Input/ Output	**Timer 2** Each signal is configured as either input to the counter or output from the counter. For the configuration of the pin direction, refer to the *MSC8113 Reference Manual*.
TMCLK	Input	**External TIMER Clock** An external timer can connect directly to the SIU as the SIU Clock.
SDA	Input/ Output	**I^{2}C-Bus Data Line** This is the data line for the I^{2}C bus.
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIO pins used as GPIO or as one of two dedicated inputs or one of two dedicated outputs. For details, see **Chapter 23**, *GPIO*.
TIMER3	Input/ Output	**Timer 3** Each signal is configured as either input to or output from the counter. For configuration details, see **Chapter 22**, *Timers*.
SCL	Input/ Output	**I^{2}C-Bus Clock Line** This the clock line for the I^{2}C bus.

3.7 Dedicated Ethernet Signals

Most of the Ethernet signals are multiplexed with the DSI/System Bus and the GPIO ports. In addition to the multiplexed signals, there are three additional dedicated Ethernet signals that are described in **Table 3-4.**,

Table 3-8. Dedicated Ethernet Signals

Signal Name	Type	Signal Description
ETHRX_CLK ETHSYNC_IN	Input Input	**Receive Clock** In MII and RMII modes, provides the timing reference for the receive signals in MII mode. **Sync Input** In SMII mode, is the sync signal input line.
ETHTX_CLK ETHREF_CLK ETHCLOCK	Input Input Input	**Transmit Clock** In MII mode, provides the timing reference for transmit signals. **Reference Clock** In RMII mode, provides the timing reference. **Ethernet Clock** In SMII mode, provides the Ethernet clock signal.
ETHCRS ETHRXD	Input Input	**Carrier Sense** In MII mode, indicates that either the transmit or receive medium is non-idle in MII mode. **Ethernet Receive Data** In SMII mode, used for the Ethernet receive data.

3.8 EOnCE Event and JTAG Test Access Port Signals

The MSC8113 uses two sets of debugging signals for the two types of internal debugging modules: EOnCE and the JTAG TAP controller. Each internal SC140 core has an EOnce module, but they are all accessed externally by the same two signals EE0 and EE1. The MSC8113 supports the standard set of test access port (TAP) signals defined by **IEEE**® Std. 1149.1™ Test Access Port and Boundary-Scan Architecture specification and described in **Table 3-9**.

Table 3-9. JTAG TAP Signals

Signal Name	Type	Signal Description
EE0	Input	**EOnCE Event Bit 0** Used for putting the internal SC140 cores into Debug mode.
EE1	Output	**EOnCE Event Bit 1** Indicates that at least one on-chip SC140 core is in Debug mode.
TCK	Input	**Test Clock** A test clock signal for synchronizing JTAG test logic.
TDI	Input	**Test Data Input** A test data serial signal for test instructions and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TDO	Output	**Test Data Output** A test data serial signal for test instructions and data. TDO can be tri-stated. The signal is actively driven in the shift-IR and shift-DR controller states and changes on the falling edge of TCK.

Table 3-9. JTAG TAP Signals (Continued)

Signal Name	Type	Signal Description
TMS	Input	**Test Mode Select** Sequences the test controller’s state machine, is sampled on the rising edge of TCK, and has an internal pull-up resistor.
$\overline{\text{TRST}}$	Input	**Test Reset** Asynchronously initializes the test controller; must be asserted during power up.

3.9 Reserved Signals

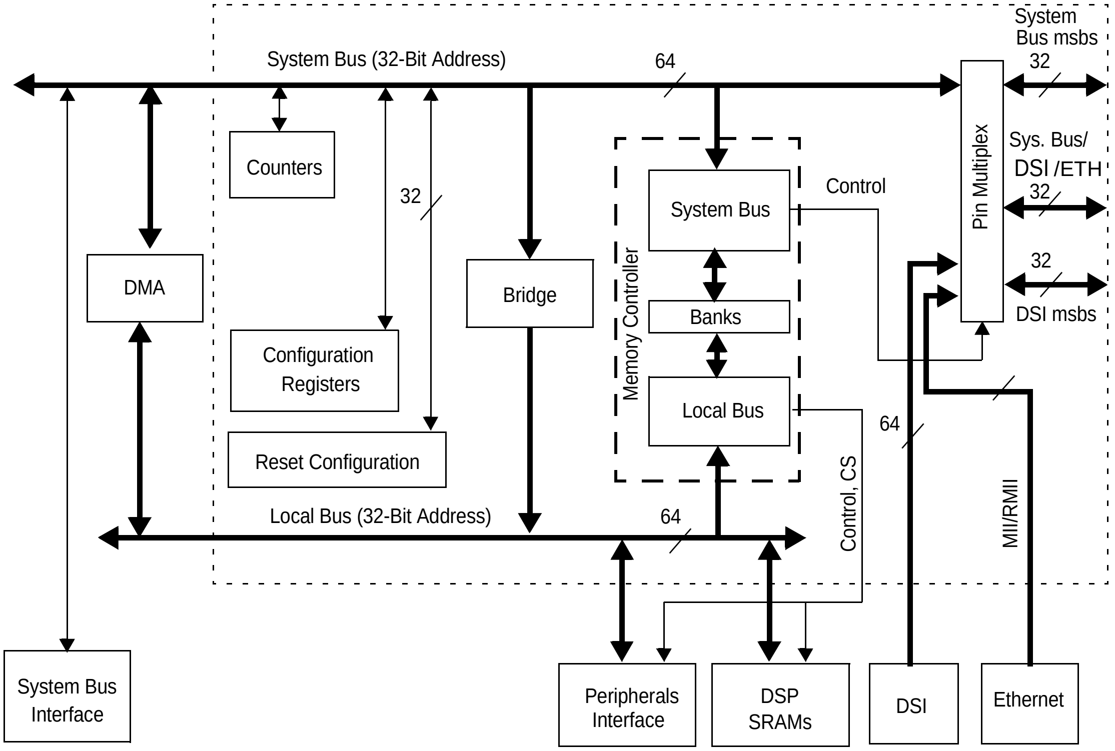
Table 3-10. Reserved Signals

Signal Name	Type	Signal Description
TEST	Input	**Test** Used for manufacturing testing. You *must* connect this pin to GND.

System Interface Unit (SIU) 4

The system interface unit (SIU) controls system start-up and initialization, as well as operation, protection, and the external system bus. Key functions of the SIU include the following (see **Figure 4-1**):

- System configuration and protection
- System reset monitoring and generation
- Clock synthesizer
- Power management
- 60x-compatible system bus interface
- Flexible, high-performance memory controller



Note: msbs = most significant bits; lsbs = least significant bits.

Figure 4-1. SIU Block Diagram

The system configuration and protection functions provide various monitors and timers, including the bus monitor, software watchdog timer, periodic interrupt timer, and time counter. The clock synthesizer generates the clock signals for the SIU and other modules. The system bus interface is a standard pipelined bus. The SIU allows external bus masters to request and obtain system bus mastership. **Chapter 13**, *System Bus* describes system bus operation and configuration. The memory controller module, described in **Chapter 12**, *Memory Controller*, provides a seamless interface to many types of memory devices and peripherals. It supports up to eight external memory banks, each with its own device and timing attributes. Two additional memory banks support internal SRAM and IPBus peripherals.

4.1 Architecture

The SIU incorporates many system functions that normally must be provided in external circuits. In addition, it provides maximum system safeguards against hardware and/or software faults. **Table 4-1** describes the functions in the system configuration and protection sub-module. **Figure 4-2** shows a block diagram of the system configuration and protection logic.

Table 4-1. System Configuration and Protection Functions

Function	Description
System configuration	The SIU allows configuration of the system according to your requirements. The functions include part and mask number constants.
System bus monitor	Monitors the transfer acknowledge ($\overline{\text{TA}}$) and address acknowledge ($\overline{\text{AACK}}$) response time for all bus accesses initiated by internal or external masters. A transfer error acknowledge ($\overline{\text{TEA}}$) is asserted if the $\overline{\text{TA}}$/$\overline{\text{AACK}}$ response limit is exceeded. This function can be disabled. See **Section 4.1.1**, *Bus Monitors*.
Local bus monitor	Monitors transfers between local bus internal masters and local bus slaves. An internal $\overline{\text{TEA}}$ assertion occurs if the transfer time limit is exceeded. This function can be disabled. See **Section 4.1.1**, *Bus Monitors*.
Software watchdog timer	The SIU watchdog timer can be associated with one of the SC140s. Three other watchdog timers can be implemented by the timer block. The SIU watchdog timer asserts a reset or $\overline{\text{NMI}}$ interrupt, selected by the System Protection Control Register (SYPCR), if the software fails to service the software watchdog timer for a certain period of time (for example, because software is lost or trapped in a loop). After a system reset, this function can be enabled or disabled according to a reset configuration pin. When enabled, it selects a maximum time-out period, and asserts a system reset if the time-out is reached. The software watchdog timer can be disabled, or its time-out period can be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset occurs. See **Section 4.1.5**, *SIU and General Software Watchdog Timers*.
Periodic Interrupt Timer (PIT)	Generates periodic interrupts for use with a real-time operating system or the application software. The periodic interrupt timer (PIT) is clocked by the TIMERSCLK clock, providing a period from 122 µs to 8 seconds. The PIT function can be disabled. See **Section 4.1.4**, *Periodic Interrupt Timer (PIT)*.
Time counter	Provides time-of-day information to the operating system/application software. It is composed of a 32-bit counter and an alarm register. A maskable interrupt is generated when the counter is equals the value programmed in the alarm register. The time counter (TMCNT) is clocked by the TIMERSCLK clock. See **Section 4.1.3**, *Time Counter (TMCNT)*.

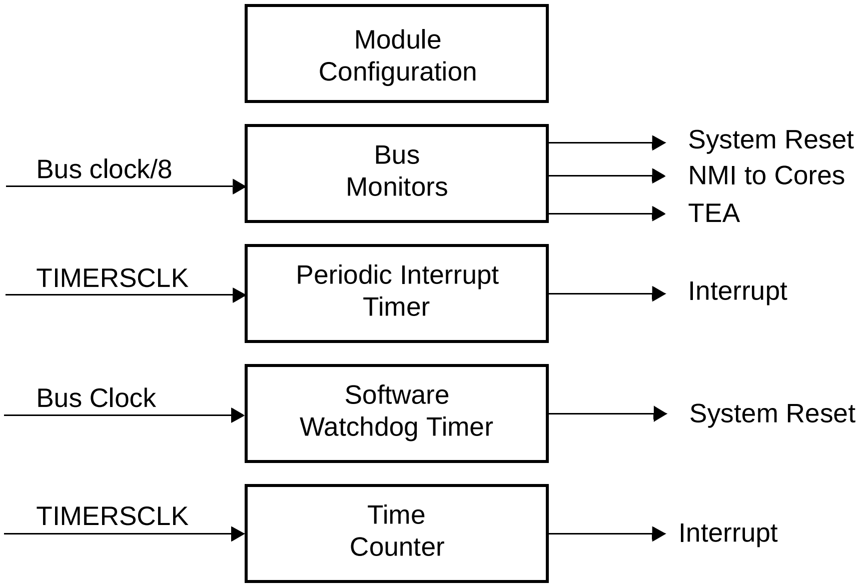


Figure 4-2. System Configuration and Protection Logic

Many aspects of system configuration are controlled by several SIU module configuration registers described in **Section 4.2.1**, *System Configuration and Protection Registers*.

4.1.1 Bus Monitors

There are two bus monitors, one for the system bus and one for the local bus. The bus monitor ensures that each bus cycle terminates within a reasonable period. The bus monitor does not count when the bus is idle. When a transaction starts ($\overline{\text{TS}}$ asserted), the bus monitor starts counting down from the time-out value.

For standard bus transactions with an address tenure and a data tenure, the bus monitor counts until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the count down. This process continues until the whole data tenure is completed. Following the data tenure, the bus monitor idles if there is no pending transaction; otherwise, it reloads the time-out value and resumes counting. For address-only transactions, the bus monitor counts until $\overline{\text{AACK}}$ is asserted.

If the monitor times out for a standard bus transaction, transfer error acknowledge ($\overline{\text{TEA}}$) is asserted. If the monitor times out for an address-only transaction, the bus monitor asserts $\overline{\text{AACK}}$ and a core machine check interrupt or reset is generated, depending on SYPCR[SWRI]. Note that the device does not generate address-only transactions.

To allow variation in system peripheral response times, SYPCR[BMT] defines the time-out period, whose maximum value can be 2,040 system bus clocks. The timing mechanism is clocked by the system bus clock divided by eight.

4.1.2 Timers Clock

The two SIU timers (the time counter and the periodic interrupt timer) use the same clock source, TIMERSCLK, which is derived from several sources, as described in **Figure 4-3**.

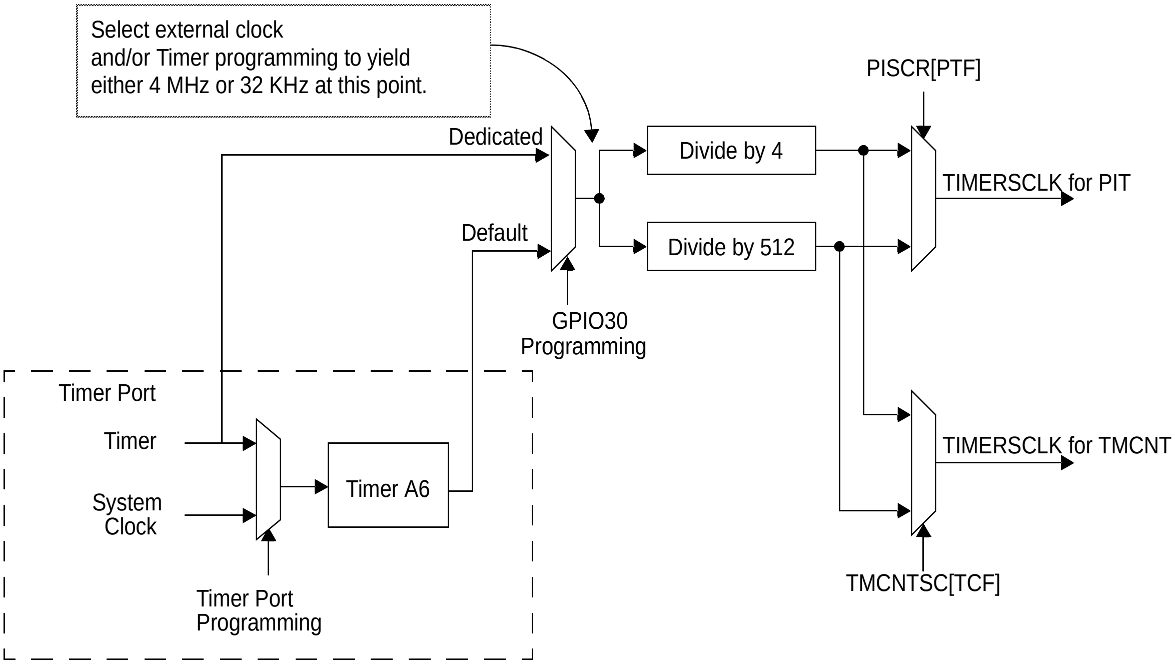


Figure 4-3. Timers Clock Generation

Refer to **Section 22.1**, *Timers Programming Model*, for details on timer programming. For proper time counter operation, you must ensure that the frequency of TIMERSCLK for TMCNT is 8,192 Hz by properly selecting the clock and programming the timer and the prescaler control bits in the Time Counter Status and Control Register (TMCNTSC[TCF]) and periodic interrupt status and control register (PISCR[PTF]).

4.1.3 Time Counter (TMCNT)

TMCNT is a 32-bit counter that is clocked by TIMERSCLK. It indicates time-of-day for the operating system and application software. The counter is reset to zero on $\overline{\text{PORESET}}$ or a hard reset but is unaffected by a soft reset. Software initializes the time counter; you should set the TIMERSCLK frequency to 8,192 Hz, as explained in **Section 4.1.2**. TMCNT can be programmed to generate a maskable interrupt when the time value matches the value programmed in its associated alarm register. The interrupt is generated on the last TMCNT clock before it transitions to a value greater than the alarm register. It can also be programmed to generate an interrupt every second. The TMCNTSC enables or disables the various timer functions and reports the interrupt source. **Figure 4-4** shows a block diagram of TMCNT, which is described on **page 4-27** in **Section 4.2.1**, *System Configuration and Protection Registers*.

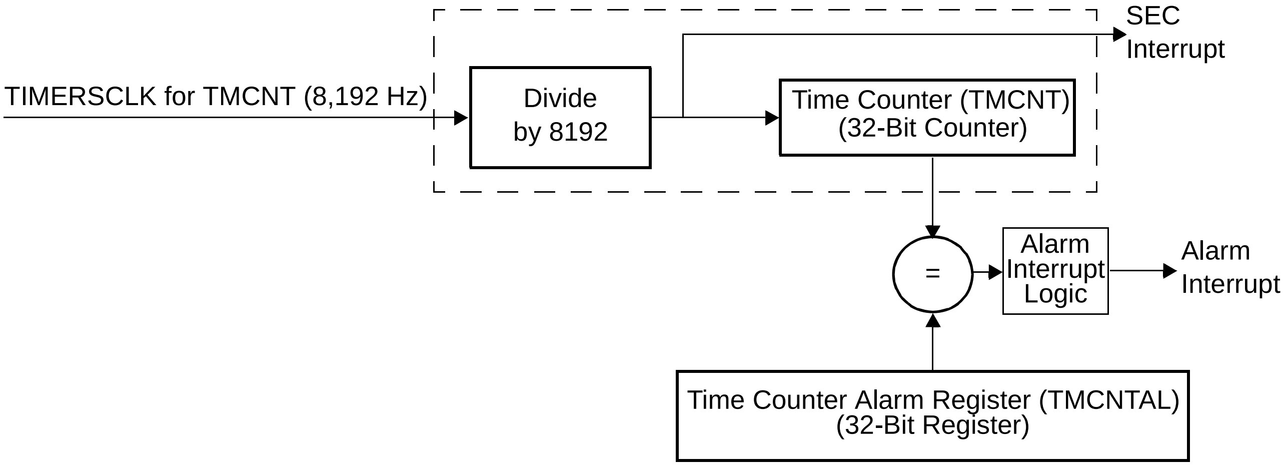


Figure 4-4. TMCNT Block Diagram

4.1.4 Periodic Interrupt Timer (PIT)

The periodic interrupt timer consists of a 16-bit counter clocked by TIMERSCLK. The 16-bit counter decrements to zero when loaded with a value from the Periodic Interrupt Timer Count Register (PITC). After the timer reaches zero, PISCR[PS] is set and an interrupt is generated if PISCR[PIE] = 1. At the next input clock edge, the value in the PITC is loaded into the counter and the process repeats. When a new value is loaded into the PITC, the PIT is updated, the divider is reset, and the counter begins counting. Setting PISCR[PS] creates a pending interrupt that remains pending until PS is cleared. If PS is set again before being cleared, the interrupt remains pending until PS is cleared. Any write to the PITC stops the current countdown, and the count resumes with the new value in PITC. If PISCR[PTE] = 0, the PIT cannot count and retains the old count value. The PIT is unaffected by reads. **Figure 4-5** shows a block diagram of the PIT.

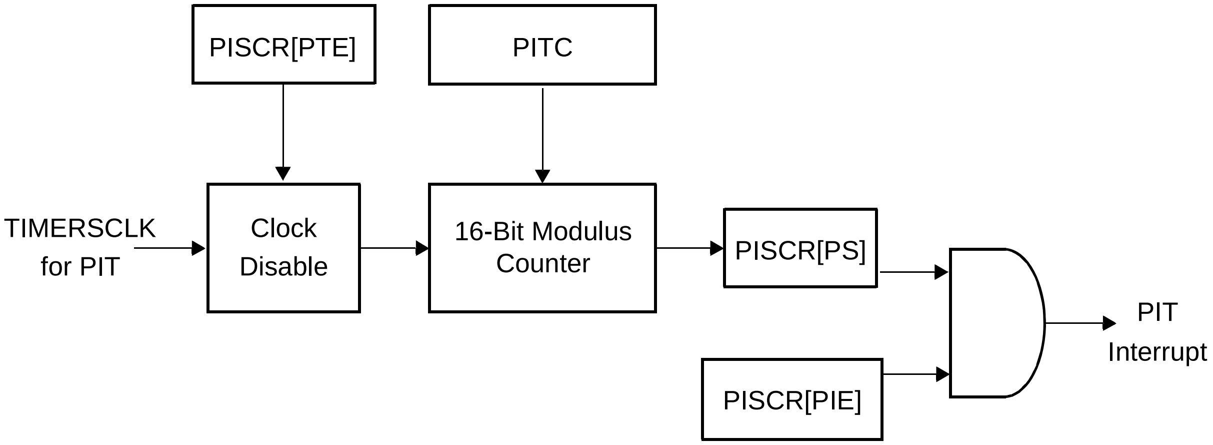


Figure 4-5. PIT Block Diagram

The time-out period is calculated as follows:

$$PIT_{period} = \frac{PITC + 1}{F_{timersclk}} = \frac{PITC + 1}{8192}$$

This calculation gives a range from 122 µs (PITC = 0x0000) to 8 seconds (PITC = 0xFFFF).

4.1.5 SIU and General Software Watchdog Timers

The SIU software watchdog timer (SWT) should be associated by the application with only one of the SC140 cores, called here the *primary core*. The SWT prevents system lock if the software becomes trapped in loops with no controlled exit. Three additional watchdog timers can be implemented by general timers in the Timers block, each associated with one SC140. The general timer interrupt lines are routed and distributed between the SC140 cores and can be programmed to assert an interrupt to the appropriate core local interrupt controller (LIC). This interrupt should be mapped to a high priority input of the PIC. Thus, regular timers achieve watchdog timer functionality. Since the hardware does not protect these timers from corruption, the primary SC140 core must protect them and ensure their proper operation.

The SWT operations are configured in the SYPCR register, as described in **Section 4.2**, *SIU Programming Model*, and at power-on reset. At power-on reset deassertion, SWTE (Software Watchdog Timer Enable) is sampled. After reset, if SWTE is sampled high, the SWT is enabled to cause a hard reset if it times out. Otherwise, it wakes up disabled. If the SWT is not needed, but the SWTE bit is sampled high, you must clear SYPCR[SWE] to disable the SWT before it times out. If the SWT is used and the SWTE bit is sampled low, you should set SYPCR[SWE] to re-enable the SWT. This option allows you to achieve an unlimited time-out period for the boot sequence with no chance of hard reset caused by SWT time-out.

When enabled, the SWT requires a special service sequence to execute periodically. Software enables the remaining three general-purpose timers used as watchdog timers and their associated SC140 cores periodically service them via a simple preload operation. Without the periodic servicing, the SWT times out and issues a reset or a non-maskable interrupt, programmed in SYPCR[SWRI]. The SYPCR register can be written once after reset. Therefore, once software programs any SYPCR bit, the state of the bits cannot be changed.

When a general timer that is used as watchdog timer times out, it asserts an interrupt line. The timer can generate an interrupt either to its associated SC140 core or to the primary core LIC. Since the general-purpose timers and their interrupts are not protected from corruption by software errors, the primary core, which is protected by the SWT, can periodically monitor their valid programming, or a virtual interrupt can be used as a life-sign from the other SC140 cores. The primary core coordinates the overall system protection and detects situations in which other SC140 cores are not responding and their general watchdog timer is not functioning (either timer or LIC/PIC programming is corrupted). The primary core may use various options to wake a non-responding core or all cores, such as a virtual $\overline{\text{NMI}}$ or letting the SIU SWT issue $\overline{\text{NMI}}$ to all the cores or a cause a hard reset.

The SIU SWT is programmed either to reset the device or to generate a dedicated $\overline{\text{NMI}}$ to all SC140 cores. The SWT service sequence consists of the following two steps:

1. Write 0x556C to the Software Service Register (SWSR).
2. Write 0xAA39 to the SWSR.

The service sequence clears the watchdog timer, and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSR, the entire sequence must start over. Although the writes must occur in the correct order before a time-out, any number of instructions can execute between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. **Figure 4-6** shows a state diagram used by the SIU watchdog timer.

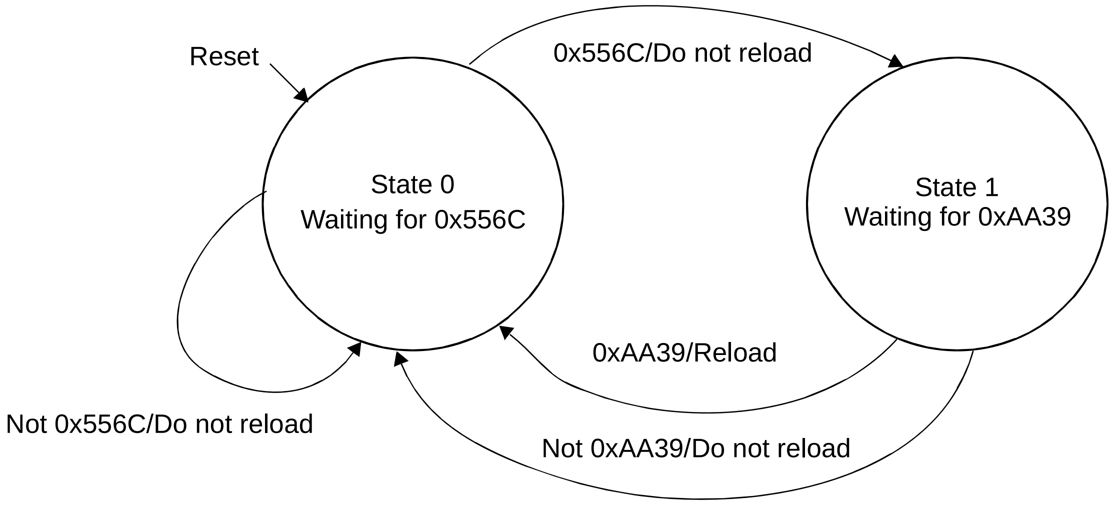


Figure 4-6. SIU Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of time-out periods. Therefore, each software watchdog timer must provide a selectable range for the time-out period. **Figure 4-7** shows how to meet this need in the SWT. On the general timers, it is met by programming the required timer period value.

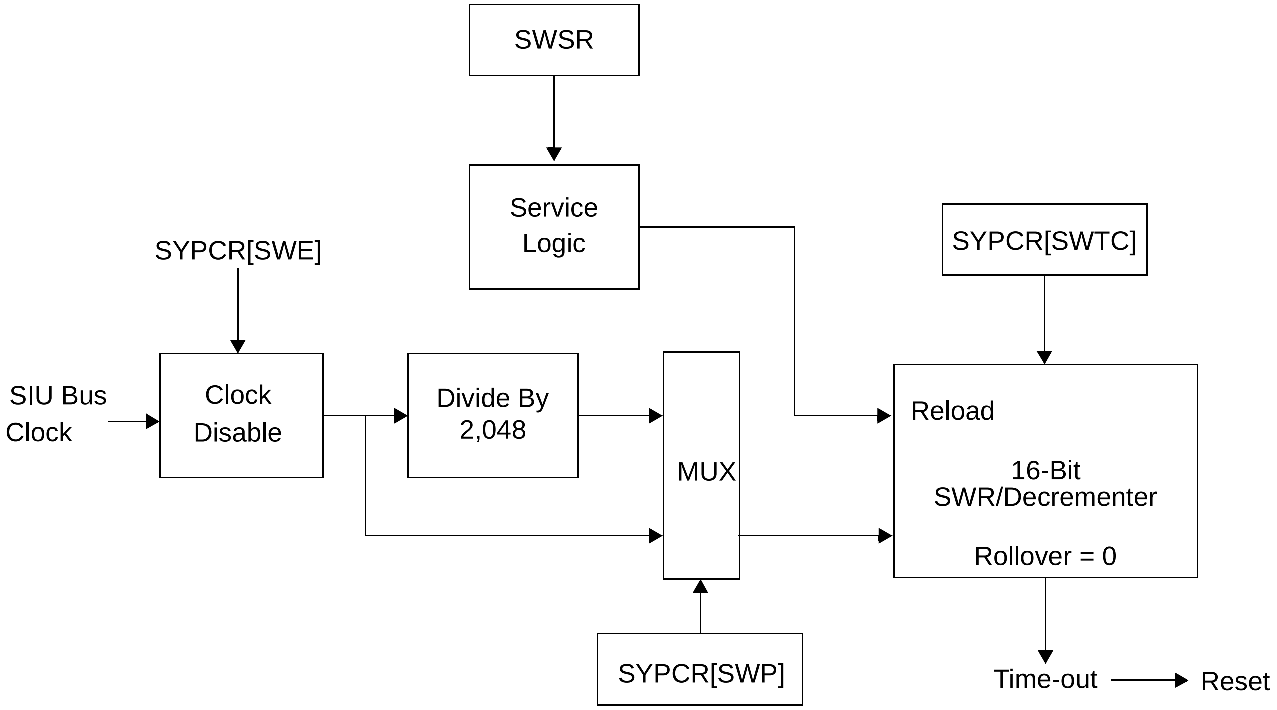


Figure 4-7. SIU Software Watchdog Timer Block Diagram

In **Figure 4-7**, the range is determined by SYPCR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the system clock. An additional divide-by-2,048 prescaler is used when needed. The decrementer begins counting when loaded with a value from SWTC. After the timer reaches 0x0, a software watchdog expiration request is issued to the reset control logic. Upon reset, SWTC is set to the maximum value and is again loaded into the Software Watchdog Register (SWR), restarting the process. When a new value is loaded into SWTC, the software watchdog timer is not updated until the servicing sequence is written to the SWSR. If SYPCR[SWE] has a value of zero (0) at power-on reset, the modulus counter does not count.

General timers used as watchdog timers should be configured to work in periodic mode and count on the system bus clock or its derivative. The time-out period is programmed in the timer period register. The interrupt lines are routed to the appropriate SC140 LIC. For details on timer programming, refer to **Section 22.1**, *Timers Programming Model*. For details on interrupt routing, refer to **Section 17.1.2.5**, *LIC Interrupt Sources*.

4.1.6 SIU Multiplexing

Some functions share signal connections. The pinout of the MSC8113 is shown in **Chapter 3**, *External Signals*. The control of the functionality used on a specific connection is shown in **Table 4-2**.

Table 4-2. SIU Signal Multiplexing Control

Pin Name	Configuration Control
TT0/HA7 TT2/CS5 TT3/CS6 TT4/CS7 GBL/IRQ1 BADDR29/IRQ5 BADDR30/IRQ2 BADDR31/IRQ3 ABB/IRQ4 DBB/IRQ5 INT_OUT/IRQ7 NC/DP0/DREQ1/EXT_BR2 IRQ1/DP1/DACK1/EXT_BG2 IRQ2/DP2/DACK2/EXT_DBG2 IRQ3/DP3/DREQ2/EXT_BR3 IRQ4/DP4/DACK3/EXT_DBG3 IRQ5/DP5/DACK4/EXT_BG3 IRQ6/DP6/DREQ3 IRQ7/DP7/DREQ4 BCTL1/CS5 BM0/TC0/BNKSEL0 BM1/TC1/BNKSEL1 BM2/TC2/BNKSEL2	Controlled by SIUMCR programming during the reset configuration sequence. For details, see **Section 4.2**, *SIU Programming Model*.
PWE[0–3]/PSDDQM[0–3]/PBS[0–3] PWE[4–7]/PSDDQM[4–7]/PBS[4–7]/ HWBS[4-7]/HDBS[4-7]/HWBE[4-7]/HDBE[4-7] PSDA10/PGPL0 PSDWE/PGPL1 POE/PSDRAS/PGPL2 PSDCAS/PGPL3 PGTA/PUPMWAIT/PGPL4/PPBS PSDAMUX/PGPL5	System bus signals are controlled dynamically according to the specific memory controller machine that handles the current bus transaction. All functions starting with H belong to the host port. Its data bus width is selected at power-on reset by the value of DSI64. Multiplexing between host port functions is selected by the DSI control registers. For details on DSI interface refer to **Chapter 14**, *Direct Slave Interface (DSI)*.
HD[32–63]/D[32–63] HD[32–39]/D[32–39]/NC HD[40]/D[40]/ETHRXD0 HD[41]/D[41]/ETHRXD1 HD[42]/D[42]/ETHRXD2/NC HD[43]/D[43]/ETHRXD3/NC HD[44–45]/D[44–45]/NC HD[46]/D[46]/ETHTXD0 HD[47]/D[47]/ETHTXD1 HD[48]/D[48]/ETHTXD2/NC HD[49]/D[49]/ETHTXD3/NC HD[50–53]/D[50–53]/NC HD[54]/D[54]/ETHTX_EN HD[55]/D[55]/ETHTX_ER HD[56]/D[56]/ETHRX_DV/ETHCRS_DV HD[57]/D[57]/ETHRX_ER HD[58]/D[58]/ETHMDC HD[59]/D[59]/ETHMDIO HD[60]/D[60]/ETHCOL HD[61–63]/D[61–63]/NC	The least significant 32 data lines on this shared bus are configured during the power-on reset configuration sequence to serve as the least significant system bus data lines, the Ethernet MII/RMII lines (only the used lines are connected), or the least significant DSI data lines. The signal line assignment is determined at power-on reset by the value of the DSI64 reset configuration pin and, if DSI64 = 0, by the ETHSEL bit in the hard reset configuration word (HRCW). If DSI64 = 0 during power-on reset and HRCW[ETHSEL]= 0 (the default), the signal lines are assigned to system bus data and designated as Dn. If DSI64 = 0 and HRCW[ETHSEL] = 1, the signal lines are assigned to the Ethernet interface (MII or RMII) and are designated as ETHxxx. If DSI64 = 1, the pins are assigned to DSI data bus and are designated as HDn. Multiplexing between Ethernet functions is selected by the Ethernet block control registers according to the mode (disabled, MII, or RMII). Unused pins in any mode should be left unconnected. For details on the Ethernet interface, refer to **Chapter 25**, *Ethernet Controller*. Multiplexing between host port functions is selected by the DSI control registers. See **Chapter 14**, *Direct Slave Interface (DSI)*. All functions starting with ETH belong to the Ethernet block.

4.2 SIU Programming Model

This section discusses the SIU registers in detail. The SIU programming model comprises three groups of registers:

- System Configuration and Protection registers and periodic interrupt registers.
- Memory controller configuration registers. For details refer to **Section 12.8**, *Memory Controller Programming Model*.

4.2.1 System Configuration and Protection Registers

The SIU system configuration and protection registers are as follows:

- Bus Configuration Register (BCR), **page 4-10**
- System Bus Arbiter Configuration Register (PPC_ACR), **page 4-13**
- System Bus Arbitration-Level Registers (PPC_ALRH/PPC_ALRL), **page 4-14**
- Local Bus Arbiter Configuration Register (LCL_ACR), **page 4-16**
- Local Bus Arbitration Level Registers (LCL_ALRH and LCL_ACRL), **page 4-17**
- SIU Module Configuration Register (SIUMCR), **page 4-17**
- Internal Memory Map Register (IMMR), **page 4-20**
- System Protection Control Register (SYPCR), **page 4-21**
- Software Service Register (SWSR), **page 4-22**
- System Bus Transfer Error Status and Control Register 1 (TESCR1), **page 4-22**
- System Transfer Error Status and Control Register 2 (TESCR2), **page 4-24**
- Local Bus Transfer Error Status and Control Register 1 (L_TESCR1), **page 4-25**
- Time Counter Status and Control Register (TMCNTSC), **page 4-26**
- Time Counter Register (TMCNT), **page 4-27**
- Time Counter Alarm Register (TMCNTAL), **page 4-27**

BCR Bus Configuration Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	EBM	APD			—	—	—	—	PLDP	DSBI	—	EAV	ETM	LETM	EPAR	—
Type	R/W															
Reset	EBM	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**
	NPQM			—		EXDD	—	—	—	—	—	ISPS	—	—	—	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	ISPS	0	0	0	0

BCR contains configuration bits for various features and wait states on the system bus.

Reconfiguring BCR modifies system and local bus behavior and can result in unexpected results if there are currently active accesses. Therefore, execute code that modifies the BCR only from internal memory. Also, execute a write to the BCR when only one core is active and there are no active DMA transfers.

Table 4-3. BCR Bit Descriptions

Name	Description	Settings
EBM 0	**External Bus Mode** Selects the bus mode. For details, refer to **Section 13.2.1**, *System Bus Operating Modes*. The initial value is selected by the Hard Reset Configuration Word (HRCW) EBM bit. See **Section 5.6.1** for details.	0 Single MSC8113 bus mode. 1 Multi-master bus mode.
APD 1–3	**Address Phase Delay** Specifies the minimum number of address tenure wait states for address operations initiated by a 60x-compatible system bus master. APD indicates how many cycles the MSC8113 should wait for ARTRY, but because ARTRY can be asserted (by other masters) only on cacheable address spaces, APD is considered only on transactions that hit one of the 60x-assigned memory controller banks and have the GBL signal asserted during address phase.	
— 4–7	Reserved. Write to zero for future compatibility.	
PLDP 8	**Pipeline Maximum Depth** See **Section 13.2.3.12**, *Pipeline Control*.	0 Pipeline maximum depth is one. 1 Pipeline maximum depth is zero.
DSBI 9	**Disable System Bus on Internal Access** Determines which internal system bus lines are reflected on the external system bus. **Note:** Address attribute lines are always reflected on the external system bus.	0 Data and address lines for internal system bus transfers are reflected on the external system bus. 1 Depends on the value of EBM, as follows: • If EBM = 0, neither data nor address lines for internal system bus transfers are reflected on the external system bus. • If EBM = 1, only address lines for internal system bus transfers are reflected on the external system bus.
10	Reserved. Write to zero for future compatibility.	
EAV 11	**Enable Address Visibility** Normally, when the MSC8113 is in single-MSC8113 bus mode, the bank select signals for SDRAM accesses are multiplexed on the system bus address lines. Therefore, for SDRAM accesses, the internal address is not visible for debug purposes. However the bank select signals can also be driven on dedicated lines (see SIUMCR[TCPC]). In this case, EAV is used to force address visibility.	0 Bank select signals are driven on system bus address lines. There is no full address visibility. 1 Bank select signals are not driven on address bus. During READ and WRITE commands to SDRAM devices, the full address is driven on system bus address lines.
ETM 12	**System Bus Compatibility Mode Enable** See **Section 13.2.3.8**, *Extended Transfer Mode*. This bit also enables data streaming mode; see **Section 13.2.4.2**, *Data Streaming Mode* for details.	0 Strict 60x system bus mode. Extended transfer mode is disabled. 1 Extended transfer mode is enabled.

Table 4-3. BCR Bit Descriptions (Continued)

Name	Description	Settings
LETM 13	**Local Bus Compatibility Mode Enable** Enables/disables extended transfer mode. See **Section 13.2.3.8**, *Extended Transfer Mode*. This bit also enables data streaming mode; see **Section 13.2.4.2**, *Data Streaming Mode* for details. **Note:** If the local bus memory controller is configured to work with read-modify-write parity, LETM must be cleared.	0 Extended transfer mode is disabled on the local bus. 1 Extended transfer mode is enabled on the local bus.
EPAR 14	**Even Parity** Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing.	0 Odd Parity. 1 Even Parity.
— 15	Reserved. Write to zero for future compatibility.	
NPQM 16–18	**Non-MSC8113 Type Master** Identifies the type of bus masters that connect to the arbitration lines when the MSC8113 is in Internal Arbiter mode. Possible types are an MSC8113 type of master (such as MSC8101 and MPC8260) or a non-MSC8113 type master. This field is related to the data pipelining bits (BRx[DR]) in the memory controller. Because a non-MSC8101 external bus master cannot use the data pipelining feature, the MSC8113, which controls the memory, must be notified when a non-MSC8113 type master is accessing the memory so it can handle the transaction differently. NPQM0 designates the type of master connected to the set of $\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{DBG}}$. NPQM1 designates the type of master connected to the set of $\overline{\text{EXT\_BR2}}$, $\overline{\text{EXT\_BG2}}$, and $\overline{\text{EXT\_DBG2}}$. NPQM2 designates the type of master that is connected to the set of $\overline{\text{EXT\_BR3}}$, $\overline{\text{EXT\_BG3}}$, and $\overline{\text{EXT\_DBG3}}$.	0 The bus master connected to the arbitration lines is an MSC8113. 1 The bus master connected to the arbitration lines is not an MSC8113.
— 19–20	Reserved. Write to zero for future compatibility.	
EXDD 21	**External Master Delay Disable** Generally, the MSC8113 adds a delay of one clock cycle for each external master access to a region controlled by the memory controller. This occurs because the external master drives the address externally (compared to an internal master, such as the MSC8113 DMA, which drives the address on an internal bus in the device). Thus, it is assumed that an additional cycle is needed for the memory controllers banks to complete the address match. However in some cases (when the bus is operated in low frequency), this extra cycle is not needed. You can disable the extra cycle by setting EXDD.	0 The memory controller inserts one wait state between the assertion of $\overline{\text{TS}}$ and the assertion of $\overline{\text{CS}}$ when an external master accesses an address space controlled by the memory controller. 1 The memory controller asserts $\overline{\text{CS}}$ on the cycle following the assertion of $\overline{\text{TS}}$ by an external master accessing an address space controlled by the memory controller.
— 22-26	Reserved. Write to zero for future compatibility.	

Table 4-3. BCR Bit Descriptions (Continued)

Name	Description	Settings
ISPS 27	**Internal Space Port Size** Defines the port size of the MSC8113 internal space region as seen by external masters. Setting ISPS enables a 32-bit master to access the MSC8113 internal space. The initial value is selected by the Hard Reset Configuration Word (HRCW) ISPS bit. See **Section 5.6.1** for details. **Note:** When the ISPS bit is set, an external master can only access the MSC8113 internal space using 32-bit single accesses.	0 MSC8113 acts as a 64-bit slave to external masters access to its internal space. 1 MSC8113 acts as a 32-bit slave to external masters access to its internal space.
— 28–31	Reserved. Write to zero for future compatibility.	

PPC_ACR System Bus Arbiter Configuration Register

Bit	0	1	2	3	4	5	6	7
	—		DBGD	EARB	PRKM			
Type	R/W							
Reset	0	0	0	(EARB)	0	0	1	0
Boot	0	0	0	(EARB)	0	1	0	1

PPC_ACR defines the arbiter modes and parked master on the system bus.

Table 4-4. PPC_ACR Bit Descriptions

Name	Reset	Description	Settings
— 0–1	00	Reserved. Write to zero for future compatibility.	
DBGD 2	0	**Data Bus Grant Delay** Specifies the minimum number of data tenure wait states for system bus master-initiated data operations. This is the minimum delay between $\overline{\text{TS}}$ and $\overline{\text{DBG}}$. See **Section 13.2.4.1**, *Data Bus Arbitration*.	0 $\overline{\text{DBG}}$ is asserted with $\overline{\text{TS}}$ if the data bus is free. 1 $\overline{\text{DBG}}$ is asserted one cycle after $\overline{\text{TS}}$ if the data bus is not busy.
EARB 3	Set by HRCW	**External Arbitration** See **Section 13.2.4.1**, *Data Bus Arbitration*. This bit reset value is determined on power on reset by the Hard Reset Configuration Word (HRCW). See **Section 5.6.1**, *Hard Reset Configuration Word*.	0 Internal arbitration is performed. 1 External arbitration is assumed.

Table 4-4. PPC_ACR Bit Descriptions (Continued)

Name	Reset	Description	Settings	
			PRKM	**Bus Master Index**
PRKM 4–7	0101	**Parking Master** Defines the parked master.	0000	Reserved
			0001	Reserved
			0010	Reserved
			0011	Reserved
			0100	DSI
			0101	SC140 interface
			0110	Reserved
			0111	External master 1
			1000	External master 2
			1001	External master 3
			1010	DMA high priority
			1011	DMA middle priority
			1100	DMA low priority
			1101	ETH high priority
			1110	ETH medium priority
			1111	ETH low priority

PPC_ALRH System Bus Arbitration-Level Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	Priority Field 0				Priority Field 1				Priority Field 2				Priority Field 3			
Type	R/W															
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0
Boot	1	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	Priority Field 4				Priority Field 5				Priority Field 6				Priority Field 7			
Type	R/W															
Reset	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	1
Boot	0	1	1	1	1	0	0	0	1	0	0	1	1	1	1	0

PPC_ALRH defines the arbitration priority of MSC8113 bus masters 0–7. Priority field 0 has the highest priority. The content of each priority field is the index number one bus master. For information on MSC8113 bus master indexes, see the description of PPC_ACR[PRKM] in **Table 4-4**.

PPC_ALRL System Bus Arbitration-Level Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Priority Field 8				Priority Field 9				Priority Field 10				Priority Field 11			
Type	R/W															
Reset	1	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1
Boot	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Priority Field 12				Priority Field 13				Priority Field 14				Priority Field 15			
Type	R/W															
Reset	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
Boot	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0

PPC_ALRL defines the arbitration priority of MSC8113 bus masters 8–15. Priority field 0 has the highest priority.

LCL_ACR Local Bus Arbiter Configuration Register

Bit	0	1	2	3	4	5	6	7
	—		DBGD	—	PRKM			
Type	R/W							
Reset	0	0	0	0	0	0	1	0
Boot	0	0	0	0	0	0	1	1

LCL_ACR defines the arbiter modes and the parked master on the local bus.

Table 4-5. LCL_ACR Bit Descriptions

Name	Reset	Description	Settings
— 0–1	00	Reserved. Write to zero for future compatibility.	
DBGD 2	0	**Data Bus Grant Delay** Specifies the minimum number of data tenure wait states for MSC8113 master-initiated data operations. This is the minimum delay between $\overline{\text{TS}}$ and $\overline{\text{DBG}}$. See **Section 13.2.4.1**, *Data Bus Arbitration*.	0 $\overline{\text{DBG}}$ is asserted with $\overline{\text{TS}}$ if the data bus is free. 1 $\overline{\text{DBG}}$ is asserted one cycle after $\overline{\text{TS}}$ if the data bus is not busy.
— 3	0	Reserved. Write to zero.	

Table 4-5. LCL_ACR Bit Descriptions (Continued)

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th><th>Settings</th></tr>
<tr><td><b>PRKM</b>
4–7</td><td>0011</td><td><b>Parking Master</b>
Defines the parked master.</td><td>
<table>
<tr><th>PRKM</th><th>Bus Master Index</th></tr>
<tr><td>0000</td><td>Reserved</td></tr>
<tr><td>0001</td><td>Reserved</td></tr>
<tr><td>0010</td><td>TDM</td></tr>
<tr><td>0011</td><td>Host Bridge</td></tr>
<tr><td>0100</td><td>DSI</td></tr>
<tr><td>0101–1001</td><td>Reserved</td></tr>
<tr><td>1010</td><td>DMA high priority</td></tr>
<tr><td>1011</td><td>DMA middle priority</td></tr>
<tr><td>1100</td><td>DMA low priority</td></tr>
<tr><td>1101</td><td>ETH high priority</td></tr>
<tr><td>1110</td><td>ETH medium priority</td></tr>
<tr><td>1111</td><td>ETH low priority</td></tr>
</table>
</td></tr>
</table>

LCL_ALRH Local Bus Arbitration Level Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	Priority Field 0				Priority Field 1				Priority Field 2				Priority Field 3			
Type	R/W															
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0
Boot	0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	Priority Field 4				Priority Field 5				Priority Field 6				Priority Field 7			
Type	R/W															
Reset	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	1
Boot	1	1	1	0	0	0	1	1	1	0	1	1	0	0	1	0

LCL_ALRH defines the arbitration priority for MSC8113 local bus masters 0–7. Priority field 0 has the highest priority. The content of each priority field is the index number one bus master. For information about the MSC8113 local bus master indexes see LCL_ACR[PRKM] in **Table 4-5**.

LCL_ALRL Local Bus Arbitration Level Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Priority Field 8				Priority Field 9				Priority Field 10				Priority Field 11			
Type	R/W															
Reset	1	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1
Boot	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Priority Field 12				Priority Field 13				Priority Field 14				Priority Field 15			
Type	R/W															
Reset	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
Boot	1	0	0	1	0	0	0	1	0	1	0	1	0	1	1	0

LCL_ALRL defines the arbitration priority of MSC8113 local bus masters 8–15. Priority field 0 has highest priority. The reset value is the recommended arbitration priority configuration for most applications.

SIUMCR SIU Module Configuration Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BBD	ESE	PBSE	INTOUT	DPPC		IRPC	BM			TCPC		CS5PC	TTPC	BCTLC	
Type	R/W															
Reset	BBD	0	0	INTOUT	DPPC		IRPC	BM			TCPC		CS5PC	TTPC	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	MMR	INTODC	—	CLKOD	—	—	—	—	—	—	—	—	—	—	—	—
Type	R/W															
Reset	MMR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUMCR configures various SIU features. Its default reset value is 0, but the value of some fields can be overwritten by the reset configuration sequence, according to the HRCW value or by reset configuration signal lines. See **Section 5.6.1**, *Hard Reset Configuration Word*.

Table 4-6. SIUMCR Bit Descriptions

Name	Description	Settings
BBD 0	**Bus Busy Disable**	0 $\overline{\text{ABB}}$/$\overline{\text{IRQ4}}$ is $\overline{\text{ABB}}$, $\overline{\text{DBB}}$/$\overline{\text{IRQ5}}$ is $\overline{\text{DBB}}$. 1 $\overline{\text{ABB}}$/$\overline{\text{IRQ4}}$ is $\overline{\text{IRQ4}}$, $\overline{\text{DBB}}$/$\overline{\text{IRQ5}}$ is $\overline{\text{IRQ5}}$.
ESE 1	**External Snoop Enable** Configures $\overline{\text{GBL}}$/$\overline{\text{IRQ1}}$.	0 External snooping disabled. ($\overline{\text{GBL}}$/$\overline{\text{IRQ1}}$ is $\overline{\text{IRQ1}}$) 1 External snooping enabled. ($\overline{\text{GBL}}$/$\overline{\text{IRQ1}}$ is $\overline{\text{GBL}}$)

Table 4-6. SIUMCR Bit Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td><b>PBSE</b>
2</td><td><b>Parity Byte Select Enable</b>
Enables/disables parity byte select. When PBSE is cleared, GPL4 output of the UPM is available for memory control. When PBSE is set, GPL4 is used as parity byte select output from the MSC8113.</td><td>0 Parity byte select is disabled.
1 Parity byte select is enabled.</td></tr>
<tr><td><b>INTOUT</b>
3</td><td><b>IRQ7 or INT_OUT Selection</b></td><td>0 IRQ7/INT_OUT is IRQ7
1 IRQ7/INT_OUT is INT_OUT</td></tr>
<tr><td><b>DPPC</b>
4–5</td><td><b>Data Parity Pin Configuration</b>
The additional arbitration lines (EXT_BR2, EXT_BG2, EXT_DBG2, EXT_BR3, EXT_BG3, and EXT_DBG3) are operational only when ACR[EARB] = 0. Setting EARB (to choose external arbiter) combined with programming DPPC to 11 deactivates these lines.</td><td>
<table>
<tr><th rowspan="2">Pin</th><th colspan="4">DPPC</th></tr>
<tr><th>00</th><th>01</th><th>10</th><th>11</th></tr>
<tr><td>NC/DP0/DREQ1/ EXT_BR2</td><td>NC</td><td>DP0</td><td>DREQ1</td><td>EXT_BR2</td></tr>
<tr><td>IRQ1/DP1/DACK1/ EXT_BG2</td><td>IRQ1</td><td>DP1</td><td>DACK1</td><td>EXT_BG2</td></tr>
<tr><td>IRQ2/DP2/DACK2/ EXT_DBG2</td><td>IRQ2</td><td>DP2</td><td>DACK2</td><td>EXT_DBG2</td></tr>
<tr><td>IRQ3/DP3/DREQ2/ EXT_BR3</td><td>IRQ3</td><td>DP3</td><td>DREQ2</td><td>EXT_BR3</td></tr>
<tr><td>IRQ4/DP4/DACK3/ EXT_DBG3</td><td>IRQ4</td><td>DP4</td><td>DACK3</td><td>EXT_DBG3</td></tr>
<tr><td>IRQ5/DP5/DACK4/ EXT_BG3</td><td>IRQ5</td><td>DP5</td><td>DACK4</td><td>EXT_BG3</td></tr>
<tr><td>IRQ6/DP6/DREQ3</td><td>IRQ6</td><td>DP6</td><td>DREQ3</td><td>IRQ6</td></tr>
<tr><td>IRQ7/DP7/DREQ4</td><td>IRQ7</td><td>DP7</td><td>DREQ4</td><td>IRQ7</td></tr>
</table>
</td></tr>
<tr><td><b>IRPC</b>
6</td><td><b>Interrupt Pin Configuration</b>
During reset configuration sequence the BADDR lines are driven regardless on the IRPC selection, in order to provide correct address to an external configuration EPROM.</td><td>
<table>
<tr><th rowspan="2">Pin</th><th colspan="2">IRPC</th></tr>
<tr><th>0</th><th>1</th></tr>
<tr><td>BADDR29/ IRQ5</td><td>IRQ5</td><td>BADDR29</td></tr>
<tr><td>BADDR30/ IRQ3</td><td>IRQ2</td><td>BADDR30</td></tr>
<tr><td>BADDR31/ IRQ2</td><td>IRQ3</td><td>BADDR31</td></tr>
</table>
</td></tr>
<tr><td><b>BM</b>
7–9</td><td><b>Boot Mode Indication</b>
This field is copied by power on reset from BM as part of the reset configuration sequence. It reflects the selected boot mode. For details on different boot mode options refer to <b>Table 6-1<i>Boot Mode Selection,</i></b> on page 6&gt;-1.</td><td>000 MSC8113 boots from system external memory.
001 MSC8113 boots from External host (DSI or Power PC).
010 MSC8113 boots from TDM.
011 MSC8113 boots from UART.
100MSC8113 boots from I^{2}C.
101Reserved.
110 Reserved.
111 Reserved.</td></tr>
<tr><td><b>TCPC</b>
10-11</td><td><b>Transfer Code Pin Configuration</b>
These bits select the functionality of the TC/BNKSEL[0–2] as either Transfer Code or Bank Select.</td><td>
<table>
<tr><th rowspan="2">Pin</th><th colspan="4">TCPC</th></tr>
<tr><th>00</th><th>01</th><th>10</th><th>11</th></tr>
<tr><td>TC0/BNKSEL0</td><td>TC0</td><td>Reserved</td><td>BNKSEL0</td><td>Reserved</td></tr>
<tr><td>TC1/BNKSEL1</td><td>TC1</td><td>Reserved</td><td>BNKSEL1</td><td>Reserved</td></tr>
<tr><td>TC2/BNKSEL2</td><td>TC2</td><td>Reserved</td><td>BNKSEL2</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

Table 4-6. SIUMCR Bit Descriptions (Continued)

<table>
<tr><th>Name</th><th>Description</th><th>Settings</th></tr>
<tr><td><b>CS5PC</b>
12</td><td><b>BCTL1 or CS5 Pin Configuration</b></td><td>0 CS5/BCTL1 pin is CS5
1 CS5/BCTL1 pin is BCTL1.</td></tr>
<tr><td><b>TTPC</b>
13</td><td><b>Transfer Type Pin Configuration</b>
The TTPC field can assign unused TT lines to chip select lines and bit 7 (MSB) of the DSI address (selected by the DSI configuration). In this case the chip internally complements the TT field to valid five bit value and TT1 indicates read/write access (TT=00010 for write, TT=01010 for read).</td><td><table>
<tr><th rowspan="2">Pin</th><th colspan="2">TTPC</th></tr>
<tr><th>0</th><th>1</th></tr>
<tr><td>TT0/HA7</td><td>TT0</td><td>HA7</td></tr>
<tr><td>TT2/CS5</td><td>TT2</td><td>CS5</td></tr>
<tr><td>TT3/CS6</td><td>TT3</td><td>CS6</td></tr>
<tr><td>TT4/CS7</td><td>TT4</td><td>CS7</td></tr>
</table></td></tr>
<tr><td><b>BCTLC</b>
14–15</td><td><b>Buffer Control Configuration</b></td><td>00 BCTL0 is used as W/R control. BCTL1 is used as OE control.
01 BCTL0 is used as W/R control. BCTL1 is used as OE control.
10 BCTL0 is used as WE control. BCTL1 is used as RE control.
11 Reserved.</td></tr>
<tr><td><b>MMR</b>
16</td><td><b>Mask Masters Requests</b>
Masks the selected master’s bus requests. In some systems, several bus masters are active during normal operation; only one bus master should be active during boot sequence. The active master, which is the boot device, initializes system memories and devices and enables all other masters. MMR facilitates such a boot scheme. MMR is configured through the hard reset configuration sequence. See <b>Chapter 5</b>, <i>Reset</i>. Typically, system configuration identifies only one master as the boot device. It initializes the system and then enables all other devices by writing 00 to MMR.

<b>Note</b>: Do not mask the request of a master that is defined as the parked master in the arbiter, since masking the bus request does not prevent this master from getting a bus grant.</td><td>0 No masking on bus request lines.
1 All external bus requests masked (boot master is the internal core).</td></tr>
<tr><td>INTODC
17</td><td><b>INT_OUT Drive Control</b>
Selects the drive method of the INT_OUT line as open-drain of full drive. This field has effect only if the INTOUT configuration bit selects INT_OUT functionality.</td><td>0 INT_OUT line is an open-drain output, enabling wired connection of multiple drivers with shared pull-up resistor.
1 INT_OUT line is driven with full drive.</td></tr>
<tr><td>—
18</td><td colspan="2">Reserved. Write to zero for future compatibility.</td></tr>
</table>

Table 4-6. SIUMCR Bit Descriptions (Continued)

Name	Description	Settings
CLKOD 19	**CLKOUT Disable** Disables the driving of CLKOUT. Use it only in external clock mode.	0 CLKOUT is driven with the System clock. 1 CLKOUT is not driven.
— 20–31	Reserved. Write to zero for future compatibility.	

IMMR Internal Memory Map Register

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 ISB field. See **Section 5.6.1**, *Hard Reset Configuration Word*.															

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-device system resources from the values in IMMR. PARTNUM and MASKNUM are mask programmed and cannot be changed for any particular device.

Table 4-7. 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 six addresses; the software can then change it to any value. The default configuration maps ISB to address 0xF0000000 (when ISBSEL bits in the HRCW are zero). 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-MSC8113 systems. The number of programmable bits in this field, and hence the resolution of the location of internal space, depends on the internal memory space of a specific implementation. In the MSC8113, all 15 bits can be programmed. See **Chapter 8**, *Memory Map*, for details on the device’s internal memory map and to **Chapter 5**, *Reset*, for the available default initial ISB values depending on ISBSEL.	Implementation-dependent
— 15	Reserved. Write to zero for future compatibility.	

Table 4-7. IMMR Bit Descriptions (Continued)

Bits	Description	Settings
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 MSC8113 part number is 0x62.
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 MSC8113 initial mask number is 0x00.

SYPCR System Protection Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SWTC															
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	BMT								PBME	LBME	—			SWE	SWRI	SWP
Type	R/W															
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	SWTE	1	1

SYPCR controls the system monitors, SIU software watchdog period, and bus monitor timing. The SYPCR is readable at any time but is writable only once after system hard reset.

Table 4-8. SYPCR Bit Descriptions

Name	Reset	Description	Settings
SWTC 0–15	1	**Software Watchdog Timer Count** Contains the count value for the SIU software watchdog timer.	
BMT 16–23	1	**Bus Monitor Timing** Defines the time-out period for the bus monitor. The granularity of this field is 8 bus clocks. (BMT = 0xFF is translated to 0x7F8 clock cycles). BMT is used in both the system and local bus monitors. **Note:** The value 0 is invalid; an error is generated for each bus transaction.	
PBME 24	0	**System Bus Monitor Enable**	0 System bus monitor is disabled. 1 System bus monitor is enabled.
LBME 25	0	**Local Bus Monitor Enable**	0 Local bus monitor is disabled. 1 Local bus monitor is enabled.

Table 4-8. SYPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 26–29	0	Reserved. Write to zero for future compatibility.	
SWE 29	Sampled at power-on reset from SWTE.	**Software Watchdog Enable** Enables the operation of the software watchdog timer. If enabled after reset, software can clear SWE after a system reset to disable the software watchdog timer.	0 Watchdog timer disabled. 1 Watchdog timer enabled.
SWRI 30	1	**Software Watchdog Reset/Interrupt Select**	0 Software watchdog timer causes a machine check interrupt to the core. 1 Software watchdog timer causes a hard reset (this is the default value after hard reset).
SWP 31	1	**Software Watchdog Prescale** Controls the divide-by-2048 software watchdog timer prescaler.	0 Software watchdog timer is not prescaled. 1 Software watchdog timer clock is prescaled.

SWSR Software Service Register

SWSR is the location to which the software watchdog timer servicing sequence is written. To prevent software watchdog timer time-out, you should assign a value of 0x556C to this register followed by 0xAA39. SWSR is written at any time, but it returns all zeros when read. For details, see **Section 4.1.5**.

TESCR1 System Bus Transfer Error Status and Control Register 1

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	BM	ISBE	PAR	—	—	WP	EXT	TC			—	TT				
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**
	—	DMD	—						—							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The system bus transfer error status and control register 1 (TESCR1) holds status bits which indicate the reason of bus error or SIU NMI caused by access on the system bus. It also holds control fields which configure the data error and correction detection.

Table 4-9. TESCR1 Bit Descriptions

Name	Name	Description
BM 0	0	**System Bus Monitor Time-Out** Set when TEA is asserted due to the system bus monitor time-out.
ISBE 1	0	**Internal Space Bus Error** Indicates that TEA was asserted due to error on a transaction to MSC8113 internal memory space. TESCR2[REGS] indicates that the internal access is to the SIU Registers address space (address hit IMMR).
PAR 2	0	**System Bus Parity Error** Indicates that Error (NMI) was asserted due to a parity error on the system bus. TESCR2[PB] indicates which byte lane caused the error; TESCR2[BNK] indicates which memory controller bank was accessed.
— 3–4	0	Reserved. Write to zero for future compatibility.
WP 5	0	**Write Protect Error** Indicates an attempted write to a system bus memory region defined as read-only in the memory controller. Note that this alone does not cause TEA assertion. TEA is asserted by bus monitor time out.
EXT 6	0	**External Error** Indicates that TEA was asserted by an external bus slave.
TC 7–9	0	**Transfer Code** Indicates the transfer code of the system bus transaction that caused the TEA. See **Table 13-11** on page 13-22.
— 10	0	Reserved. Write to zero for future compatibility.
TT 11–15		**Transfer Type** Indicates the transfer type of the system bus transaction that caused the TEA. See **Table 13-10** on page 13-22, for a description of the various transfer types.
— 16	0	Reserved. Write to zero for future compatibility.
DMD 17		**Data Errors Disable** 0 Errors are enabled. 1 All data errors on the system bus are disabled.
18–31	—	Reserved. Write to zero for future compatibility.

TESCR2 System Bus Transfer Error Status and Control Register 2

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	REGS	—					LCL	PB							
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
	BNK								—							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TESCR2 provides additional information on bus error caused by access on the system bus.

Table 4-10. TESCR2 Bit Descriptions

Name	Reset	Description
— 0	0	Reserved. Write to zero for future compatibility.
REGS 1	0	**SEB1 Internal Registers Error** Indicates that an error occurred in a transaction to the MSC8113 internal registers.
— 2–6	0	Reserved. Write to zero for future compatibility.
LCL 7	0	**Local Bus Bridge Error** An error occurred in a transaction from the MSC8113 system bus to the local bus bridge. The bridge is non-burstable.
PB 8–15	0	**Parity Error on Byte** There are eight parity error status bits, one per 8-bit lane. A bit is set for the byte with a parity error.
BNK 16–23	0	**Memory Controller Bank** There are eight error status bits, one per memory controller external bank. A bit is set for the system bus memory controller bank with an error. Note that this field is invalid if the error was not caused by parity checks.
— 24–31	0	Reserved. Write to zero for future compatibility.

L_TESCR1 Local Bus Transfer Error Status and Control Register 1

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BM	—	—	—	—	WP	—	TC			—	TT				
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
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

L_TESCR1 holds status bits which indicate the reason of bus error caused by access on the local bus.

Table 4-11. L_TESCR1 Bit Descriptions

Name	Reset	Description
BM 0	0	**Bus Monitor Time-Out** Indicates that $\overline{\text{TEA}}$ was asserted due to the local bus monitor time-out.
— 1–4	0	Reserved. Write to zero for future compatibility.
WP 5	0	**Write Protect Error** Indicates that a write was attempted to a local bus memory region that was defined as read-only in the memory controller. Note that this alone does not cause $\overline{\text{TEA}}$ assertion. Usually, in this case, the bus monitor times out.
— 6	0	Reserved. Write to zero for future compatibility.
TC 7–9	0	**Transfer Code** Indicates the transfer code of the local bus transaction that caused the $\overline{\text{TEA}}$. **Table 13-11** describes the transfer codes.
— 10	0	Reserved. Write to zero for future compatibility.
TT 11–15	0	**Transfer Type** Indicates the transfer type of the local bus transaction that caused the $\overline{\text{TEA}}$. **Table 13-10** describes the transfer types.
— 16–31	0	Reserved. Write to zero for future compatibility.

TMCNTSC Time Counter Status and Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								SEC	ALR	—		SIE	ALE	TCF	TCE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMCNTSC enables the different TMCNT functions and reports the source of the interrupts. The register can be read at any time. Status bits are cleared by writing ones; writing zeros does not affect the value of a status bit.

Table 4-12. TMCNTSC Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
SEC 8	0	**Once Per Second Interrupt** This status bit is set every second and should be cleared by software.	
ALR 9	0	**Alarm Interrupt** This status bit is set when the value of the TMCNT equals the TMCNTAL value, on the clock when TMCNT counts to ALARM + 1.	
— 10–11	0	Reserved. Write to zero for future compatibility.	
SIE 12	0	**Second Interrupt Enable** Specifies whether the time counter generates an interrupt when SEC is set.	0 The time counter does not generate an interrupt. 1 The time counter generates an interrupt.
ALE 13	0	**Alarm Interrupt Enable** Enables or disables an alarm interrupt.	0 No alarm interrupt. 1 The time counter generates an interrupt when ALR is set.
TCF 14	0	**Time Counter Frequency** The input clock to the time counter may be either 4 MHz or 32 KHz. The TCF bit should be set according to the frequency of this clock. See **Section 4.1.2**, *Timers Clock*.	0 The input clock to the time counter is 4 MHz. 1 The input clock to the time counter is 32 KHz.
TCE 15	0	**Time Counter Enable** Is not affected by soft or hard reset.	0 The time counter is disabled. 1 The time counter is enabled.

TMCNT Time Counter Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	TMCNT															
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
	TMCNT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMCNTAL Time Counter Alarm Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ALARM															
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
	ALARM															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMCNTAL holds a value (ALARM). When the value of TMCNT equals ALARM, a maskable interrupt is generated at the point in time when the TMCNT counts to ALARM + 1. The resolution of the alarm is one second.

Table 4-13. TMCNTAL Bit Descriptions

Name	Reset	Description
ALARM 0–31	0	**Alarm Interrupt** The alarm interrupt is generated when the ALARM field equals the corresponding TMCNT bits. The interrupt is generated when TMCNT counts to ALARM + 1. The resolution of the alarm is one second.

4.2.2 Periodic Interrupt Registers

The periodic interrupt registers described in this section are listed as follows:

- Periodic Interrupt Status and Control Register (PISCR), **page 4-28**
- Periodic Interrupt Timer Count Register (PITC), **page 4-28**
- Periodic Interrupt Timer Register (PITR), **page 4-29**

PISCR Periodic Interrupt Status and Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								PS	—				PIE	PTF	PTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PISCR contains the interrupt request level and the interrupt status bit. It also contains the controls for the 16 bits to be loaded in a modulus counter.

Table 4-14. PISCR Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
PS 8	0	**Periodic Interrupt Status** The PIT issues an interrupt after the modulus counter counts to zero.	0 No effect on PS. 1 Deasserts PS.
— 9–12	0	Reserved. Write to zero for future compatibility.	
PIE 13	0	**Periodic Interrupt Enable** Enables or disables a periodic interrupt.	0 The period interrupt timer does not generate an interrupt. 1 The periodic interrupt timer generates an interrupt when PS = 1.
PTF 14	0	**Periodic Interrupt Frequency** The input clock to the periodic interrupt timer may be either 4 MHz or 32 KHz. You should set the PTF bit according to the frequency of this clock. See **Section 4.1.2**, *Timers Clock*.	0 The input clock to the periodic interrupt timer is 4 MHz. 1 The input clock to the periodic interrupt timer is 32 KHz.
PTE 15	0	**Periodic Timer Enable** Controls the counting of the periodic interrupt timer. When the timer is disabled, it maintains its old value. When the counter is enabled, it continues counting using the previous value.	0 Disable counter. 1 Enable counter.

PITC Periodic Interrupt Timer Count Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PITC															
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**
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PITC contains the 16 bits to be loaded in a modulus counter.

Table 4-15. PITC Bit Descriptions

Name	Reset	Description
PITC 0–15	0	**Periodic Interrupt Timing Count** Contains the count for the periodic timer. Setting PITC to 0xFFFF selects the maximum count period.
— 16–31	0	Reserved. Write to zero for future compatibility.

PITR Periodic Interrupt Timer Register

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">PIT</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

PITR shows the current value in the periodic interrupt down counter. The PITR counter is not affected by reads or writes to it.

Table 4-16. PITR Bit Descriptions

Name	Reset	Description
PIT 0–15	0	**Periodic Interrupt Timing Count** Contains the current count remaining for the periodic timer. Writes have no effect on this field.
— 16–31	0	Reserved. Write to zero for future compatibility.

Reset 5

All MSC8113 reset sources are fed into the reset controller, which takes different actions depending on the source of the reset. The Reset Status Register indicates the most recent sources to cause a reset. **Table 5-1** describes the reset sources.

Table 5-1. Reset Sources

Name	Direction	Description
Power-on reset (PORESET)	Input	Initiates the power-on reset flow that resets the MSC8113 and configures various attributes of the device. On PORESET, the entire MSC8113 device is reset. SPLL state is reset, HRESET and SRESET are driven, the SC140 extended cores are reset, and system configuration is sampled. The clock mode (MODCK bits), reset configuration mode, boot mode, Chip ID, DSI sync or a-sync mode, Software Watch Dog Timer Enable, and use of either a DSI 64-bit port or a 60x-compatible system bus 64-bit port are configured only at the rising edge of PORESET.
External hard reset (HRESET)	I/O	Initiates the hard reset flow that configures various attributes of the MSC8113. During HRESET, SRESET is asserted. HRESET is an open-drain output. Upon hard reset, HRESET and SRESET are driven, the SC140 extended cores are reset, and system configuration is sampled. The most configurable features are reconfigure. These features are defined in the 32-bit HRCW described in **Section 5.6.1**, *Hard Reset Configuration Word,* on page 5-13.
External soft reset (SRESET)	I/O	Initiates the soft reset flow. The MSC8113 device detects an external assertion of SRESET only if it occurs while the MSC8113 is not asserting reset. SRESET is an open-drain output. Upon soft reset, SRESET is driven, the SC140 extended cores are reset, and system configuration is maintained.
Software watchdog reset		When the MSC8113 watchdog count reaches zero, a software watchdog reset is signalled. The enabled software watchdog event then generates an internal hard reset sequence.
Bus monitor reset		When the MSC8113 bus monitor count reaches zero, a bus monitor hard reset is asserted. The enabled bus monitor event then generates an internal hard reset sequence.
JTAG Commands: EXTEST, CLAMP, or HIGH-Z		When one of JTAG commands EXTEST, CLAMP or HIGHZ is executed, JTAG logic asserts the JTAG soft reset signal and an internal soft reset sequence is generated.

Table 5-2 summarizes the reset actions that occur as a result of the different reset sources.

Table 5-2. Reset Actions for Each Reset Source

Reset Action \ Reset Source	Power-On Reset: External Power-On Reset	Hard Reset: External Hard Reset, Software Watchdog, Bus Monitor	Soft Reset:	
			External Soft Reset	JTAG Commands: EXTEST, CLAMP or HIGHZ
Configuration Signals Sampled (See Section 5.1)	Yes	No	No	No
SPLL State Reset	Yes	No	No	No
System Reset Configuration write through the DSI	Yes	No	No	No
System Reset Configuration write through the System Bus	Yes	Yes	No	No
HRESET Driven	Yes	Yes	No	No
SIU Registers	Yes	Yes	No	No
IPBus Modules Reset (ETH,TDM, UART, Timers, DSI, IPBus Master, GIC, HS, and GPIO)	Yes	Yes	Yes	Yes
SRESET Driven	Yes	Yes	Yes	Depend on JTAG
SC140 Extended Cores Reset (PC points to boot starting address)	Yes	Yes	Yes	Yes
MQBS Reset	Yes	Yes	Yes	Yes

5.1 Power-On Reset (PORESET)

Asserting PORESET initiates the power-on reset flow. PORESET must be asserted externally for at least 16 input clock cycles after MSC8113 V_{DD} and V_{DDH} reach their nominal values. **Table 5-3** shows the MSC8113 configuration signals. These signals are sampled at the rising edge of PORESET, which determines different MSC8113 configuration features.

Table 5-3. PORESET External Configuration Signals

Signal	Description	Settings
CNFGS	**Configuration Source** The MSC8113 device samples this signal and RSTCONF on the rising edge of PORESET to determine the MSC8113 reset configuration mode.	For details on MSC8113 reset configuration modes, refer to **Table 5-5, *Reset Configuration Modes,*** on page 5-4.
RSTCONF	**Reset Configuration Mode** The MSC8113 device samples this signal and CNFGS on the rising edge of PORESET to determine the MSC8113 reset configuration mode. In a multi-MSC8113 system, RSTCONF is used after the rising edge of PORESET to time the writing of the HRCW when the MSC8113 reset configuration is written through the system bus.	For details on MSC8113 reset configuration modes, refer to **Table 5-5, *Reset Configuration Modes,*** on page 5-4.

Table 5-3. PORESET External Configuration Signals (Continued)

Signal	Description	Settings
BM[0–2]	**Boot Mode** Input lines sampled at the rising edge of $\overline{\text{PORESET}}$, which determine the MSC8113 boot mode.	Refer to **Table 6-1*, Boot Mode Selection,*** on page 6-1.
SWTE	**Software Watchdog Timer Enable** Input line sampled at the rising edge of $\overline{\text{PORESET}}$. This bit defines whether the software watchdog timer is enabled or disabled. For details on how this signal functions, refer to the discussion of the System Protection Control Register (SYPCR) in **Section 4.2**, *SIU Programming Model*.	0 Watchdog timer disabled. 1 Watchdog timer enabled.
DSI64	**DSI 64-Bit Data Bus** Input line sampled at the rising edge of $\overline{\text{PORESET}}$. This input combined with ETHSEL hard reset configuration bit, define the pin multiplexing of the low part of the MSC8113 DSI/system data bus with the Ethernet. For details on how the DSI64 signal functions, refer to **Section 14.5.2**, *Status Registers,* on page 14-35). For details on how the ETHSEL bit functions, refer to **Section 5.6.1**, *Hard Reset Configuration Word,* on page 5-13.	For details on DSI/system bus/Ethernet pin multiplexing configuration, please refer to **Table 5-4** on page 5-3.
DSISYNC	**DSI Synchronous Mode** Input line sampled at the rising edge of $\overline{\text{PORESET}}$. This bit defines whether the DSI works in Synchronous or Asynchronous Mode. For details, refer to **Section 14.3.2**, *Synchronous Versus Asynchronous Access Mode*.	0 Asynchronous mode. 1 Synchronous mode.
CHIP_ID[0–3]	**Chip ID** Input line sampled at the rising edge of $\overline{\text{PORESET}}$. These bits define the unique number for each MSC8113 in a multi-MSC8113 system (up to 16). The DSI compares these bits to the HCID[0–3] input bus to identify access to the specific MSC8113. For details on how these signals function, refer to **Section 14.2.3**, *Host Chip ID Signals (HCID[0–3])*).	Any value between 0b0000–0b1111.
MODCK[1–2]	**Clock Mode** Input line sampled at the rising edge of $\overline{\text{PORESET}}$. For details on how these signals function, refer to **Section 7.3**, *Clock Configuration*.	

Table 5-4. DSI/System Bus/Ethernet Signal Multiplexing Configurations

DSI64 (Po-Reset Configuration Pin)	ETHSEL (Hard Reset Configuration Bit)	DSI bus width	System bus width	ETH is exposed on the low part of the DSI/System data bus (HD[32–63]/D[32–63])
0	0	32 bit	64 bit	No
0	1	32 bit	32 bit	Yes
1	x	64 bit	32 bit	No

5.2 Reset Configuration

The MSC8113 device has two mechanisms for reset configuration: reset configuration write through the direct slave interface (DSI) and reset configuration write through the system bus. When reset configuration is written through the system bus, the MSC8113 is a configuration master or slave. If a configuration slave is selected, but no special Hard Reset Configuration Word (HRCW) is written, a default HRCW is applied.

Note: Do not use the default HRCW which clears the HRCW[DLLDIS] bit. Because the MSC8113 does not support DLL operation, make sure that the HRCW[DLLDIS] bit is always set after reset.

Two signals (CNFGS and $\overline{\text{RSTCONF}}$), which are sampled on $\overline{\text{PORESET}}$ deassertion, define the reset configuration modes. See the summary in **Table 5-5**.

Table 5-5. Reset Configuration Modes

CNFGS, $\overline{\text{RSTCONF}}$	HRCW Source
00	Reset configuration write through the 60x-compatible system bus. MSC8113 is a configuration master.
01	Reset configuration write through the 60x-compatible system bus. MSC8113 is a configuration slave. If the HRCW is not written during 1024 CLKIN cycles, it gets a default value of all zeros. **Note:** Always ensure that a valid configuration is written to the HRCW through the system bus. The default configuration is not valid.
10	Reset configuration write through the DSI.
11	Reserved.

5.2.1 Reset Configuration Through the DSI

When reset configuration is written through the DSI, the host can program the HRCW (via the DSI) after $\overline{\text{PORESET}}$ is deasserted. The value driven on the $\overline{\text{CNFGS}}$ and $\overline{\text{RSTCONF}}$ signals at the rising edge of $\overline{\text{PORESET}}$ determines the MSC8113 reset configuration mode. If the value is 10, the HRCW must be written through the DSI port. The device extends the internal $\overline{\text{PORESET}}$ until the host writes the HRCW. The host must write 32 bits to the HRCW, which is 32 bits wide. For details, see **Section 14.4**, *DSI Configuration*, on page 14-27.

Note: The hard reset sequence that is initiated by asserting the external $\overline{\text{HRESET}}$ or internal source (bus monitor or software watch dog) does not restart the reset configuration sequence through the DSI. In this case, the previous HRCW is kept. See **Section 5.3**, *Hard Reset*, on page 5-8.

Next, the MSC8113 halts until the SPLL locks. As described in **Chapter 7**, *Clocks*, the SPLL locks according to MODCK[1–2], which are sampled on $\overline{\text{PORESET}}$ deassertion, and to the MODCK[3–5] bits taken from the HRCW. The SPLL locking time is 6400 reference clocks, which is the clock at the output of the SPLL pre-divider. After the SPLL is locked, all the clocks to the MSC8113 are enabled. During SPLL locking, $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ are asserted. After the SPLL is locked, HRESET remains asserted for another 512 bus clocks and is then released. The SRESET is released three bus clocks later.

Note: Because the MSC8113 does not support DLL operation, make sure that the HRCW[DLLDIS] bit is set after reset.

Figure 5-1 shows reset configuration write through the DSI timing diagram (including power-on reset flow).

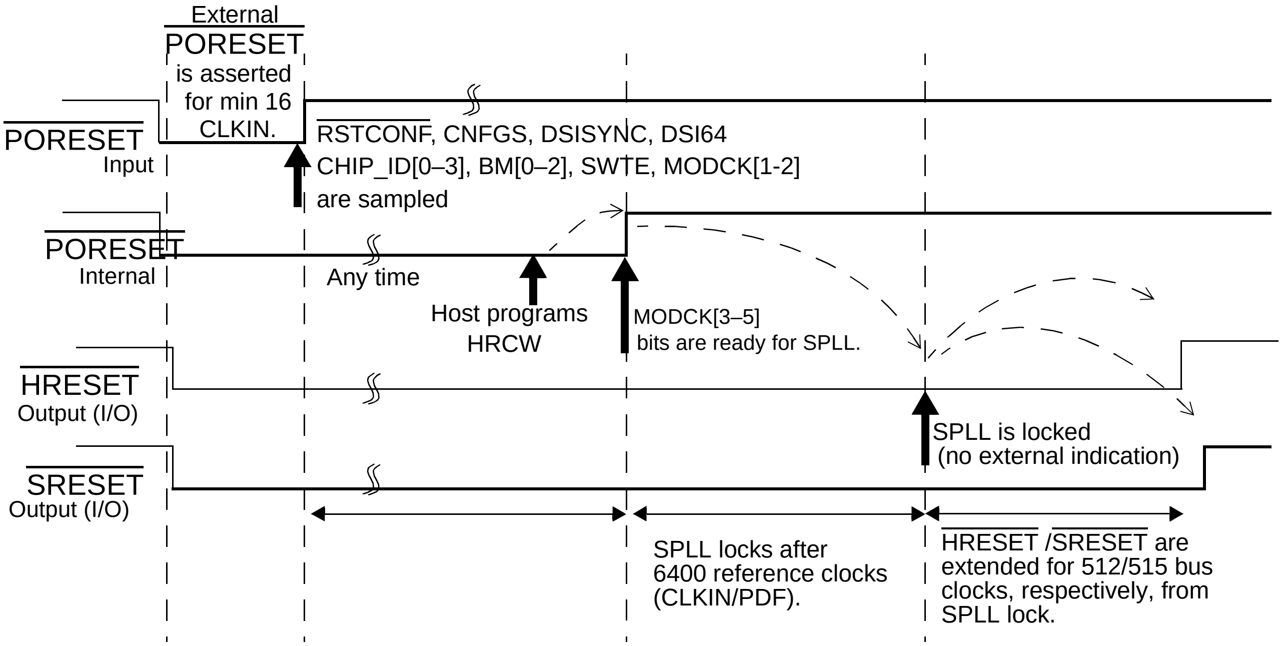


Figure 5-1. Reset Configuration Write Through the DSI, Timing Diagram

Figure 5-2 shows how to program the MSC8113 HRCW to multiple MSC8113 devices through the DSI port. The HRCW can be written separately for each device using a common CS signal and different Chip ID values, using a different CS for each device, or broadcasting the same HRCW to all the devices using a common CS signal connected to the DSI HBCS signal. In a system with only one master and one slave, the master should use only one CS signal connected to the DSI HCS signal and connect all the HCID signals of the DSI to a hard wired value that is the same as the value sampled during PORESET flow on the CHIP_ID signals. The host should finish its own wake-up reset sequence before waking up the slave. For example, when the host is an MSC8113, the RESET_OUT signal is HRESET. Connecting all the HRESET signals of all the slaves ensures that all MSC8113 devices exit reset together.

Note: Tying the host HRESET (for an MSC8113, MSC8126, MSC8101, MSC8103, or MPC8260 device) together with the slave HRESET causes a deadlock.

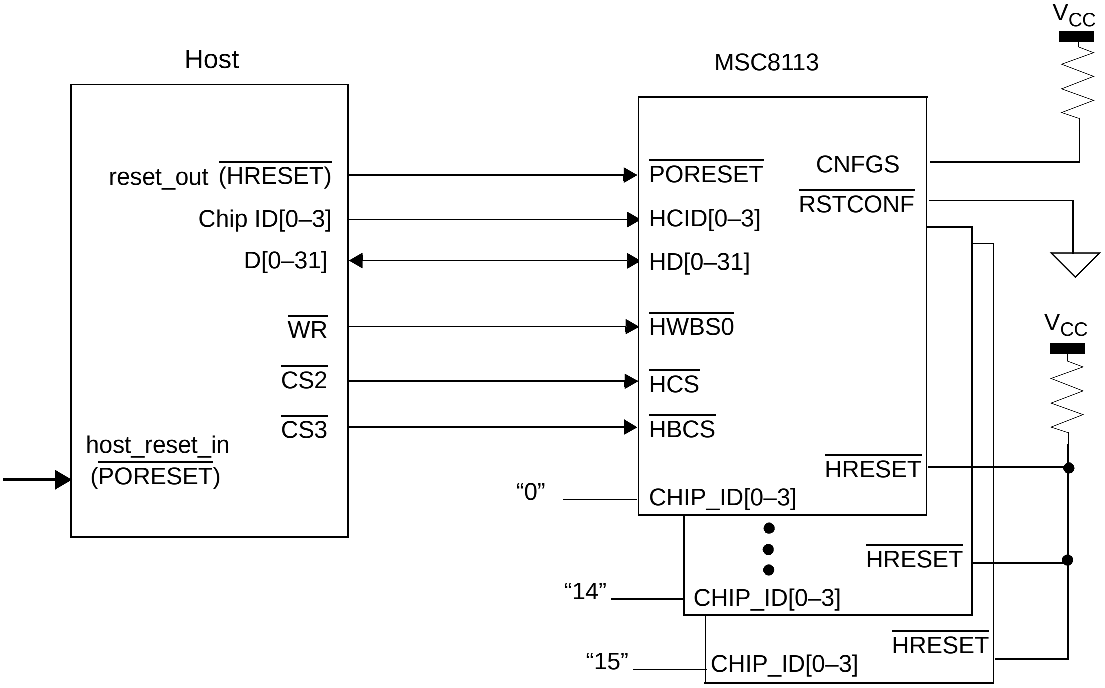


Figure 5-2. Configuring Multiple MSC8113s From the DSI Port

5.2.2 Reset Configuration Through the System Bus

The reset configuration write through the system bus allows external hardware to program the Hard Reset Configuration Word (via the system data bus) after $\overline{\text{PORESET}}$ is deasserted. The values driven on $\overline{\text{CNFGS}}$ and $\overline{\text{RSTCONF}}$ at the rising edge of $\overline{\text{PORESET}}$ determine the MSC8113 configuration mode. If the value is “01” the MSC8113 acts as a configuration slave. If the value is “00” the MSC8113 acts as a configuration master. Immediately after $\overline{\text{PORESET}}$ is deasserted and the reset operation mode is designated as configuration master or slave, the MSC8113 starts the configuration process by asserting $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout internal power-on reset. The reset configuration sequence requires 1024 CLKIN cycles.

In a typical multi- MSC8113 system, one MSC8113 acts as the configuration master and all other MSC8113 devices act as configuration slaves. The configuration master reads the various HRCWs from external memory and uses them to configure itself as well as the configuration slaves. If the MSC8113 is a configuration slave and the HRCW is not written during the 1024 CLKIN cycles, it gets a default value of all zeros, which is the reset value of the HRCW described in **Section 5.6.1**, *Hard Reset Configuration Word*. Examples of various MSC8113 system bus configurations are given in **Section 5.5**, *Reset Configuration Writes Through the System Bus*.

Note: During the reset sequence, any initiation of hard reset (that is, assertion of HRESET externally or a bus monitor event or software watch dog time-out internally) restarts the Reset Configuration sequence through the system bus. The reset configuration mode, which is determined by CNFGS and RSTCONF at the rising edge of PORESET, remains unchanged. When the MSC8113 is a configuration slave, if the HRCW is not written during 1024 CLKIN cycles, the previous HRCW is kept. See **Section 5.3**, *Hard Reset*, on page 5-8.

After configuration, the MSC8113 device halts until the SPLL locks. As described in **Chapter 7**, *Clocks*, the SPLL locks according to MODCK[1–2], which are sampled on PORESET deassertion, and to the MODCK[3–5] bits taken from the HRCW. SPLL locking time is 6400 reference clocks, which is the clock at the output of the SPLL pre-divider. After the SPLL is locked, all the clocks to the MSC8113 are enabled.

Note: Because the MSC8113 does not support DLL operation, make sure that the HRCW[DLLDIS] bit is set.

During SPLL locking, HRESET and SRESET are asserted. After the SPLL is locked, HRESET remains asserted for another 512 bus clocks and is then released. The SRESET is released three bus clocks later. The timing diagram in **Figure 5-3** shows a Reset Configuration write through the system bus (including the power-on reset flow).

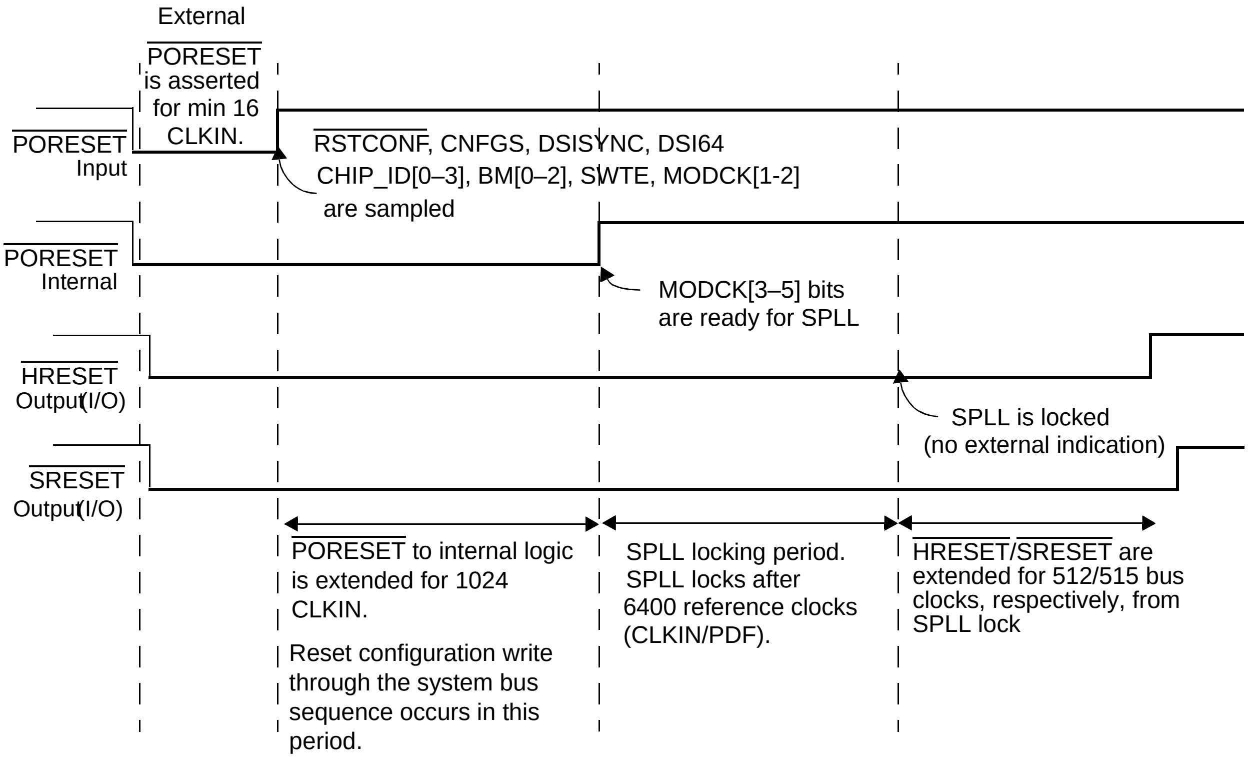

External PORESET is asserted for min 16 CLKIN.
PORESET Input
RSTCONF, CNFGS, DSISYNC, DSI64
CHIP_ID[0–3], BM[0–2], SWTE, MODCK[1-2]
are sampled
PORESET Internal
MODCK[3–5] bits
are ready for SPLL
HRESET Output(I/O)
SPLL is locked
(no external indication)
SRESET Output(I/O)
PORESET to internal logic is extended for 1024 CLKIN.
Reset configuration write through the system bus sequence occurs in this period.
SPLL locking period. SPLL locks after 6400 reference clocks (CLKIN/PDF).
HRESET/SRESET are extended for 512/515 bus clocks, respectively, from SPLL lock

Figure 5-3. Reset Configuration Write Through the System Bus

5.3 Hard Reset

A hard reset sequence is initiated externally when $\overline{\text{HRESET}}$ is asserted or internally when the MSC8113 detects a reason to start the hard reset sequence (a software watch dog timer or a bus monitor timer expires). In both cases, the MSC8113 continuously asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the hard reset sequence.

A hard reset sequence starts the reset configuration sequence through the system bus, as described in **Section 5.2.2**, *Reset Configuration Through the System Bus*, on page 5-6. The reset configuration mode (determined by $\overline{\text{CNFGS}}$ and $\overline{\text{RSTCONF}}$ at the rising edge of $\overline{\text{PORESET}}$), as well as other configuration modes determined by the $\overline{\text{PORESET}}$-sampled signals, do not change. When the hard reset sequence is not caused by asserting $\overline{\text{PORESET}}$, a reset configuration write through the DSI does not occur. The host cannot reprogram the HRCW, so the value of the HRCW set by the last configuration remains unchanged.

After the MSC8113 asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ for 512 bus clock cycles, it releases both signals and exits the hard reset sequence. An external pull-up resistor should deassert the signals. After deassertion is detected, a 16-bus cycle period is taken before testing for an external (hard/soft) reset.

5.4 Soft Reset

A soft reset sequence is initiated externally when $\overline{\text{SRESET}}$ is asserted or internally when the MSC8113 detects a cause to start the soft reset sequence (JTAG commands: EXTEST, CLAMP, or HIGHZ). In either case, the MSC8113 asserts $\overline{\text{SRESET}}$ for 512 bus clock cycles, after which the MSC8113 releases $\overline{\text{SRESET}}$ and exits soft reset. An external pull-up resistor should deassert $\overline{\text{SRESET}}$; after deassertion is detected, a 16-bus cycle period is taken before testing for an external (hard/soft) reset. While $\overline{\text{SRESET}}$ is asserted, internal hardware is reset, but the hard reset configuration as well as the SIU registers remain unchanged.

Note: If your application changes the ISB field in the IMMR, you must restore the initial ISB value (that is, the value in the field after power-up or hard reset), by writing the initial value to the ISB field before invoking a soft reset.

5.5 Reset Configuration Writes Through the System Bus

This section presents some examples of a reset configuration write through the system bus in different systems.

5.5.1 Single MSC8113 System Configuration From EPROM

If the value of CNFGS and RSTCONF is “00” on the rising edge of PORESET, the MSC8113 comes up as a configuration master. After PORESET is deasserted, RSTCONF is tied to GND as shown in **Figure 5-4**. The MSC8113 can then access the EPROM. The HRCW is assumed to reside in an EPROM connected to CS0 of the configuration master. Because the port size of this EPROM is unknown to the configuration master before the HRCWs are read, the configuration master reads the HRCW byte-by-byte only from locations that are independent of port size. The values of the bytes in **Table 5-7** are always read on byte lane D[0–7], regardless of port size. The configuration sequence (read from EPROM) occurs during a hard reset. When HRESET is deasserted (exited), the devices is assumed to be configured according to the EPROM.

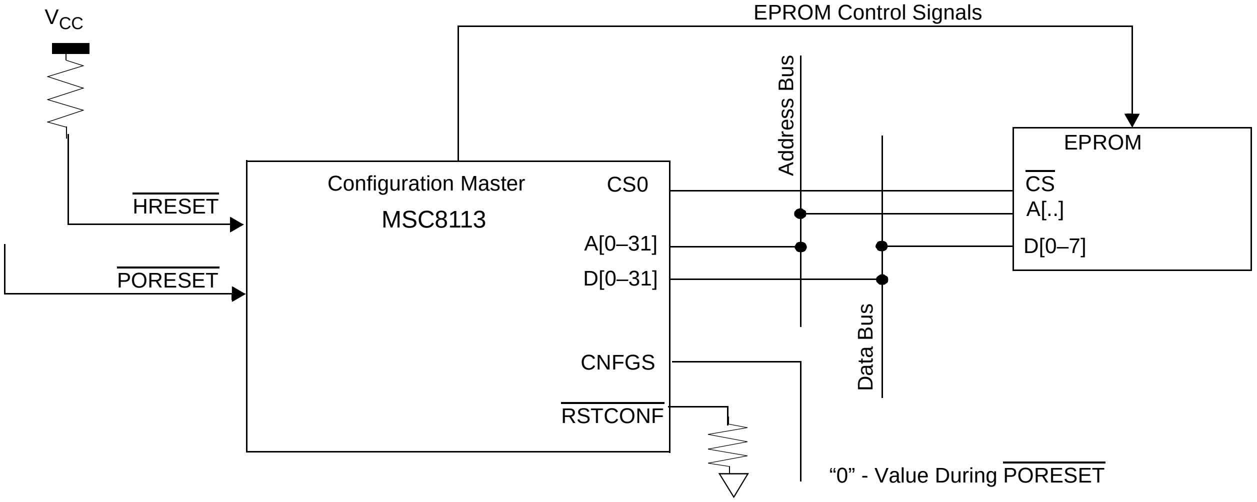


Figure 5-4. Configuring a Single MSC8113 Device From EPROM

Note: An EPROM that is accessed for system reset configuration should connect using one of the following methods:

- Connect directly to the MSC8113 without external buffer or glue logic.
- Use a data buffer that drives the MSC8113 signal lines only when external glue logic indicates that it is being accessed. Because the CS5/BCTL1 default functionality immediately after reset is CS5, this signal cannot enable the data drive from an external buffer to the MSC8113. An example of such glue logic would be represented by the equation OE = CS0 && BCTL1; that is, during boot, whenever the boot chip select (CS0) is asserted, it asserts the output enable (OE) of the buffer; after boot, the buffer is enabled as a function of BCTL1, which indicates whether it is a read or write.

5.5.2 Single Slave MSC8113 Configuration by System Bus Host

For a single-MSC8113 system with no EPROM, you can configure the MSC8113 as a configuration slave by driving a value of “01” on CNFGS and RSTCONF during PORESET assertion and then applying a negative pulse on RSTCONF and an appropriate HRCW on the MSC8113 system data bus D[0–31]. The negative pulse must occur within the 1024 configuration slave CLKIN cycles after the configuration slave PORESET ends. **Figure 5-5** shows an example an arbitrary configuration master and an MSC8113 device configured as configuration slave. The host should finish its own wake-up reset sequence before waking the slave. The host WE is used to apply the negative pulse on RSTCONF.

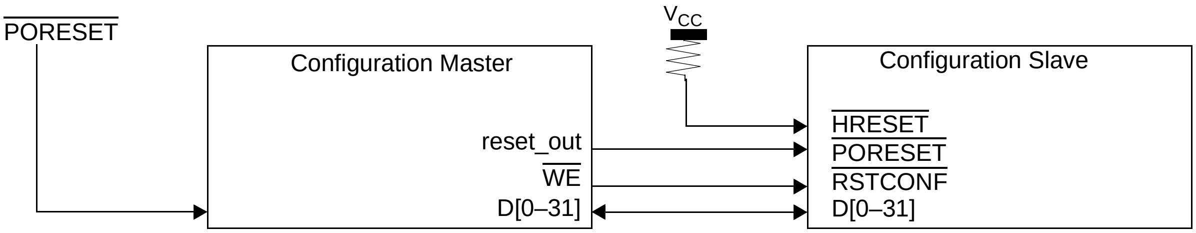


Figure 5-5. Single MSC8113 Slave Configuration

5.5.3 Multi-MSC8113 System Configuration

The sequence of reset configuration write through the system bus, which occurs during hard reset, supports a system that uses up to eight MSC8113 devices, each configured differently. It needs no additional glue logic for reset configuration. In a typical multi-MSC8113 system, one MSC8113 device acts as the configuration master and all other MSC8113 devices act as configuration slaves. The configuration master reads the various HRCWs from EPROM and uses them to configure itself as well as the configuration slaves. The reset mode that determines the MSC8113 behavior during reset configuration write through the system bus is specified by the value of CNFGS and RSTCONF on PORESET deassertion. During system bus configuration, which occurs after PORESET deassertion, the RSTCONF input of the configuration master is tied to ground, and the RSTCONF inputs of other devices connect to the high-order address bits of the configuration master, as shown in **Table 5-6**.

Note: The value of RSTCONF during PORESET may differ from its value after PORESET deassertion, which is then used for the system bus configuration sequence.

Table 5-6. RSTCONF Connections in Multi-MSC8113 Systems

Configured Device	RSTCONF Connection
Configuration master	GND
First configuration slave	A0
Second configuration slave	A1
Third configuration slave	A2
Fourth configuration slave	A3

Table 5-6. $\overline{\text{RSTCONF}}$ Connections in Multi-MSC8113 Systems (Continued)

Configured Device	$\overline{\text{RSTCONF}}$ Connection
Fifth configuration slave	A4
Sixth configuration slave	A5
Seventh configuration slave	A6

Table 5-7 shows addresses at which to configure the various MSC8113s. Byte addresses that do not appear in this table have no effect on the configuration of the MSC8113s. The values of the bytes in **Table 5-7** are always read on byte lane D[0–7], regardless of port size.

Table 5-7. Configuration EPROM Addresses

Configured Device	Byte 0 Address	Byte 1 Address	Byte 2 Address	Byte 3 Address
Configuration master	0x00	0x08	0x10	0x18
First configuration slave	0x20	0x28	0x30	0x38
Second configuration slave	0x40	0x48	0x50	0x58
Third configuration slave	0x60	0x68	0x70	0x78
Fourth configuration slave	0x80	0x88	0x90	0x98
Fifth configuration slave	0xA0	0xA8	0xB0	0xB8
Sixth configuration slave	0xC0	0xC8	0xD0	0xD8
Seventh configuration slave	0xE0	0xE8	0xF0	0xF8

The configuration master reads a value from address 0x00 and then reads a value from addresses 0x08, 0x10, and 0x18. These four bytes form the HRCW of the configuration master, which then proceeds to read the bytes that form the HRCW of the first slave device. The master drives the whole HRCW on D[0–31] and toggles its A0 address line. Each configuration slave uses its $\overline{\text{RSTCONF}}$ input as a strobe for latching the HRCW during $\overline{\text{HRESET}}$ assertion time. Thus, the first slave whose $\overline{\text{RSTCONF}}$ input connects to the master A0 output latches the 32 bits driven on D[0–31] as its HRCW. Then, the master continues to configure all MSC8113 devices in the system. The configuration master always reads eight HRCWs, regardless of the number of MSC8113 devices in the system. **Figure 5-6** shows a multi-device configuration. In this system, the configuration master initially reads its own HRCW. It then reads other HRCWs and drives them to the configuration slaves by asserting $\overline{\text{RSTCONF}}$. As **Figure 5-6** shows, this complex configuration is done without additional glue logic. The configuration master controls the whole process by asserting the EPROM control signals and the system address signals as needed. Connecting all the $\overline{\text{HRESET}}$ signals of the configuration master and all the configuration slaves ensures that all MSC8113 devices exit reset together.

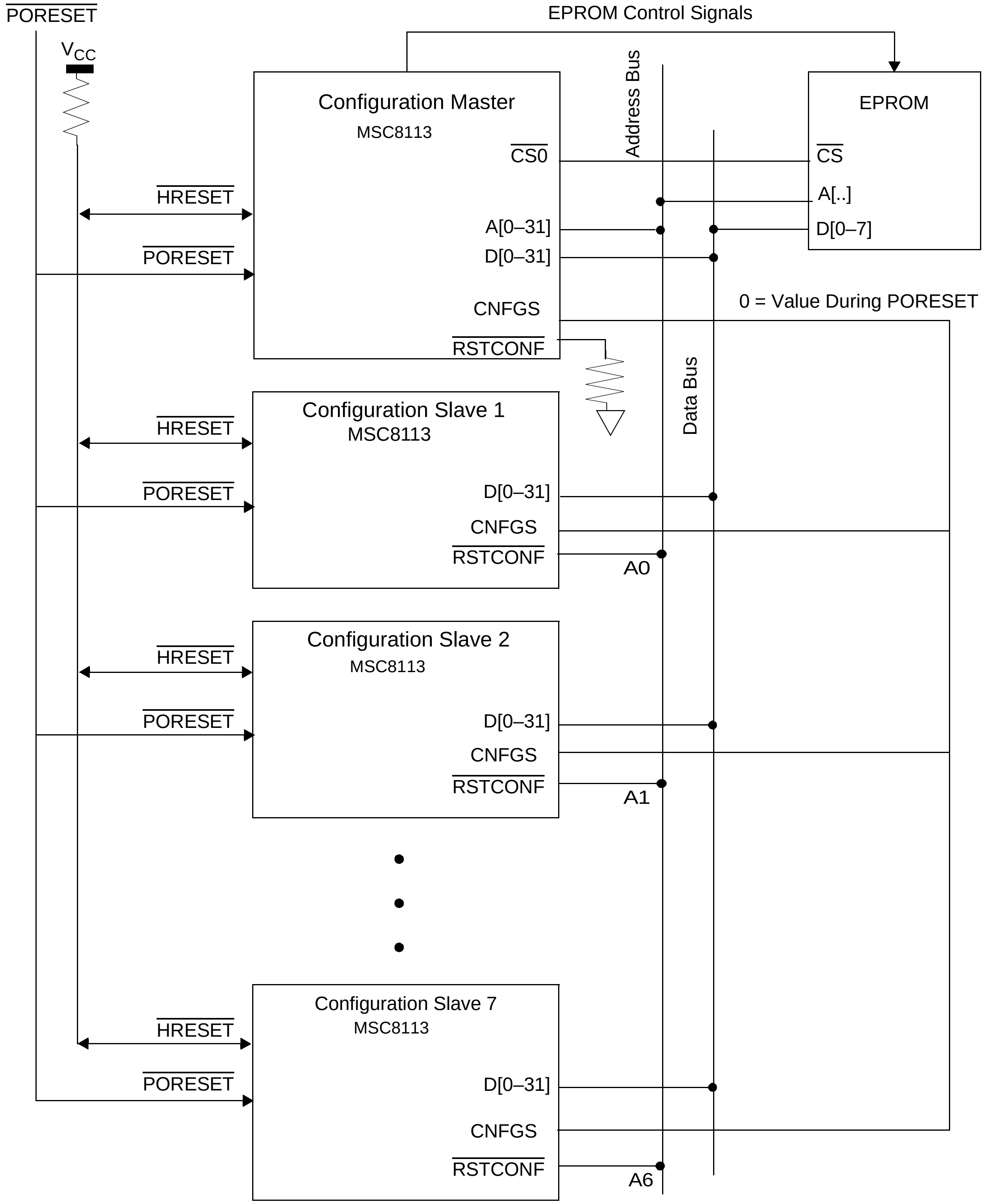


Figure 5-6. Configuring Multiple MSC8113 Devices

5.5.4 Multiple MSC8113 Devices in a System With No EPROM

In some cases, the configuration master capabilities of the MSC8113 cannot be used. This can happen, for example, if there is no EPROM in the system or if the EPROM is not controlled by an MSC8113. If this occurs, the configuration master actions must be emulated in external logic. The external hardware connects to all $\overline{\text{RSTCONF}}$ signals of the different devices and to the 32 bits of the data bus. During the rising edge of $\overline{\text{PORESET}}$, the external hardware puts all the devices in configuration slave mode. For 1024 input clocks after $\overline{\text{PORESET}}$ deassertion, the external hardware can configure the different devices by driving appropriate HRCWs on the data bus and asserting $\overline{\text{RSTCONF}}$ for each device to strobe the data received.

5.6 Reset Programming Model

This section describes the following reset registers in detail:

- Hard Reset Configuration Word (HRCW), **page 5-13**
- Reset Status Register (RSR), **page 5-16**

5.6.1 Hard Reset Configuration Word

HRCW Hard Reset Configuration Word

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EARB	EXMC	INTOUT	EBM	BPS		SCDIS	ISPS	IRPC	—	DPPC		NMIOUT	ISBSEL		
Type	Written by the hard reset configuration mechanism through the System Bus or by an external host through DSI															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	—	BBD	MMR	ETH SEL	TTPC	CS5PC	TCPC		LTLEND	PPCLE	—	DLLDIS	MODCK_H			—
Type	Written by the hard reset configuration mechanism through the System Bus or by an external host through DSI															

When reset configuration is written through the DSI, the host programs this register via the host port (DSI), as described in **Section 14.4**, *DSI Configuration*, on page 14-27. When reset configuration is written through the system bus, the reset configuration mechanism programs this register via the system bus port. This register is not directly accessible to the SC140 cores. Some bits programmed in this register affect bits in various registers that are accessible to the SC140 cores (SIUMCR, ACR, BR0, BCR, IMMR) and can be reprogrammed after reset.

Table 5-8. Hard Reset Configuration Word Bit Descriptions

Name	Reset	Description	Settings
EARB 0	0	**External Arbitration** Defines the initial value for ACR[EARB]. See **Section 4.2**, *SIU Programming Model*.	0 Internal arbitration is performed. 1 External arbitration is assumed.
EXMC 1	0	**External MEMC** Defines the initial value of BR0[EMEMC]. See **Section 12.8**, *Memory Controller Programming Model*.	0 No external memory controller is assumed. 1 External memory controller is assumed.

Table 5-8. Hard Reset Configuration Word Bit Descriptions (Continued)

Name	Reset	Description	Settings
INTOUT 2	0	**INT_OUT or IRQ7 Selection** Defines the initial value of SIUMCR[INTOUT]. See **Section 4.2**, *SIU Programming Model*.	0 IRQ7/INT_OUT is IRQ7. 1 IRQ7/INT_OUT is INT_OUT.
EBM 3	0	**External Bus Mode** Defines the initial value of BCR[EBM]. See **Section 4.2**, *SIU Programming Model*.	0 Single MSC8113 bus mode. 1 60x-compatible bus mode.
BPS 4–5	00	**Boot Port Size** Defines the initial value of BR0[PS], the port size for memory controller bank 0. See **Section 12.8**, *Memory Controller Programming Model*.	00 64-bit port size. 01 8-bit port size. 10 16-bit port size. 11 32-bit port size.
SCDIS 6	0	**SC140 Cores Disabled** Enables/disables the SC140 cores. See **Chapter 9**, *Extended Core*.	0 SC140 cores enabled. 1 SC140 cores disabled.
ISPS 7	0	**Internal Space Port Size** Defines the initial value of BCR[ISPS]. Setting ISPS enables a 32-bit master to access the MSC8113 internal space. See **Section 4.2**, *SIU Programming Model*. **Note:** When the ISPS bit is set, an external master can only access the MSC8113 internal space using 32-bit single accesses.	0 MSC8113 acts as a 64-bit slave to external masters access to its internal space. 1 MSC8113 acts as a 32-bit slave to external masters access to its internal space.
IRPC 8	0	**Interrupt Pin Configuration** Defines the initial value of SIUMCR[IRPC] and burst address pin functionality. See **Section 4.2**, *SIU Programming Model*.	0 IRQ5/IRQ2/IRQ3. 1 BADDR29/BADDR30/BADDR31.
— 9	0	Reserved. Write to zero for future compatibility.	
DPPC 10–11	00	**Data Parity Pin Configuration** Defines the initial value of SIUMCR[DPPC] and DMA channel request/acknowledge pin functionality. See **Section 4.2**, *SIU Programming Model*.	00 NC/IRQ[1–7]. 01 DP[0–7]. 10 DREQ[1–4]/DACK[1–4]. 11 EXT_BR[2–3]/EXT_BG[2–3]/ EXT_DBG[2–3]/IRQ[6–7].
NMI OUT 12	0	**NMI OUT** Defines the host core to handle a non-maskable interrupt (NMI) event.	0 NMI is serviced by SC140s. 1 NMI is routed to NMI_OUT and serviced by the external host.
ISBSEL 13–15	000	**Initial Internal Space Base Select** Defines the initial value of IMMR[ISB], which determines the base address of the internal memory space. See **Section 4.2**, *SIU Programming Model*. The SC140 internal address space spans from 0x00000000–0x00FFFFFF (16 MB). Therefore it is not advisable to map the IMMR in this space, since the SC140s cannot access the SIU registers. See **Chapter 12**, *Memory Controller*.	000 0xF0000000 001 0xF0F00000 010 0xFF000000 011 0xFFF00000 100 Reserved. Do not use this option. 101 Reserved. Do not use this option. 110 0x0F000000 111 0x0FF00000
— 16	0	Reserved. Write to zero for future compatibility.	

Table 5-8. Hard Reset Configuration Word Bit Descriptions (Continued)

Name	Reset	Description	Settings
BBD 17	0	**Bus Busy Disable** Defines the initial value of SIUMCR[BBD] and $\overline{ABB}$ and $\overline{DBB}$ pin functionality. See **Section 4.2**, *SIU Programming Model*.	0 $\overline{ABB}$/$\overline{DBB}$. 1 $\overline{IRQ4}$/$\overline{IRQ5}$.
MMR 18	0	**Mask Masters Requests** Defines the initial value of SIUMCR[MMR]. See **Section 4.2**, *SIU Programming Model*.	0 No masking on bus request lines. 1 All external bus requests masked (boot master is the one of the internal cores).
ETHSEL 19	0	**Ethernet Select** Defines whether the Ethernet is exposed on the low part of the DSI/system data bus lines (when ETHSEL is set and the DSI64 line is sampled low at reset) or on the GPIO lines (when ETHSEL is clear).	For details on DSI/system bus/Ethernet Pin multiplexing, refer to **Table 23-1, *GPIO/Dedicated Functionality Versus Ethernet Functionality,*** on page 23-4
TTPC 20	0	**Transfer Type Pin Configuration** Defines the initial value of SIUMCR[TTPC] and TT pin functionality. See **Section 4.2**, *SIU Programming Model*.	0 TT0/TT[2–4]. 1 HA7/$\overline{CS}$[5–7].
CS5PC 21	0	**Chip Select 5 Pin Configuration** Defines the initial value of SIUMCR[CS5PC] and $\overline{CS5}$ functionality. See **Section 4.2**, *SIU Programming Model*.	0 $\overline{CS5}$/$\overline{BCTL1}$ pin is $\overline{CS5}$ 1 $\overline{CS5}$/$\overline{BCTL1}$ pin is $\overline{BCTL1}$.
TCPC 22-23	00	**Transfer Code Pin Configuration** Defines the initial value of SIUMCR[TCPC] and TC pin functionality. See **Section 4.2**, *SIU Programming Model*.	00 TC[0–2]. 01 Reserved. 10 BNKSEL[0–2]. 11 Reserved.
LTLEND 24	0	**Little Endian** Defines the host Endian mode of operation. See **Section 14.2.4**, *DSI Endian Modes*.	0 Big Endian. 1 Little Endian.
PPCLE 25	0	**Munged Little Endian** When the LTLEND bit is set, PPCLE specifies whether the host is a Little-Endian host or a host that works in munged Little-Endian mode. See **Section 4.2**, *SIU Programming Model*.	0 True little-endian host. 1 Munged little-endian host.
— 26	0	Reserved. Write to zero for future compatibility.	
DLLDIS 27	0	**DLL Disable** Defines whether the DLL mechanism is disabled. See **Section 7.3**, *Clock Configuration*. **Note:** The MSC8113 does not support DLL operation. Always write a 1 to this bit to configure the device correctly.	0 No DLL bypass. 1 DLL bypass.
MODCK[3–5] 28–30	0	**MODCK High Order Bits** High-order bits of the MODCK bus, which determine the clock reset configuration. See **Section 7.3**, *Clock Configuration*.	
— 31	0	Reserved. Write to zero for future compatibility.	

5.6.2 Reset Status Registers

RSR Reset Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	—										JTRS	—	SWRS	BMRS	ESRS	EHRS
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

The RSR records reset events. For example, because software watchdog expiration results in a hard reset, which in turn results in a soft reset, RSR[SWRS], RSR[ESRS], and RSR[EHRS] are all set after a software watchdog reset. All bits are cleared by writing a 1 (writing zero has no effect). RSR is memory-mapped into the MSC8113 SIU register map.

Table 5-9. RSR Bit Descriptions

Name	Reset	Description	Settings
— 0–25	0	Reserved. Write to zero for future compatibility.	
JTRS 26	0	**JTAG Reset Status** When a host reset command is written, through JTAG logic (“JTAG reset request”), JTRS is set and remains set until software clears it.	0 No host reset command through JTAG occurred. 1 A host reset command through JTAG occurred.
— 27	0	Reserved. Write to zero for future compatibility.	
SWRS 28	0	**Software Watchdog Reset Status** When a software watchdog expire event (which causes a reset) is detected, the SWRS bit is set and remains set until the software clears it.	0 No software watchdog reset event occurred. 1 A software watchdog reset event occurred.
BMRS 29	0	**Bus Monitor Reset Status** When a bus monitor expire event (which causes a reset) is detected, BMRS is set and remains set until the software clears it.	0 No bus monitor reset event occurred. 1 A bus monitor reset event occurred.
ESRS 30	1	**External Soft Reset Status** When an external soft reset event is detected, ESRS is set and it remains set until software clears it.	0 No external soft reset event. 1 An external soft reset event.
EHRS 31	1	**External hard reset status** When an external hard reset event is detected, EHRS is set and it remains set until software clears it.	0 No external hard reset event. 1 An external hard reset event.

Boot Program 6

The boot program, which resides in the internal ROM, initializes the MSC8113 after it completes a reset sequence. The MSC8113 device can boot from an external host through the DSI or the 60x-compatible system ports, execute a user boot program located on an external memory device (such as EPROM, SDRAM), or download a user boot program through the I^{2}C, TDM, or UART ports. The boot operating mode is set by configuring BM[0–2], which are sampled on the rising edge of $\overline{\text{PORESET}}$. **Table 6-1** shows the mode options for BM[0–2].

Table 6-1. Boot Mode Selection

External Connection			Boot Sequence
BM0	**BM1**	**BM2**	
0	0	0	External Memory (from the system bus)
0	0	1	External Host: DSI or system bus
0	1	0	TDM
0	1	1	UART
1	0	0	I^{2}C
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

This chapter begins with booting basics, including the default values programmed by the boot program and interrupt handling during the boot process. Then it considers different ways to boot the MSC8113: from an external memory device on the system bus, from an external host located on the DSI or on the system bus port, from the time-division multiplexing (TDM) interface, from the UART, or from the I^{2}C software module interface.

6.1 Boot Basics

The boot program initializes the MSC8113 with default values shown in **Table 6-2**.

Table 6-2. Default MSC8113 Initialization Values of the Boot Program

Module or Register Initialized	Where Discussed
UPMC and the GPCM as required to support the MSC8113 M1 and M2 memories	**Section 12.7**, *Internal SRAM and IPBus Peripherals Support,* on page 12-92 **Table 8-7*Banks 9 and 11 Address Space,*** on page 8-28
Memory Controller Option Registers (OR[9,11])	**Table 12.8*Memory Controller Programming Model,*** on page 12-95
Memory Controller Base Registers (BR[9,11])	
System Bus and Local Bus Arbiter Configuration	**Section 4.2**, *SIU Programming Model,* on page 4-10
System bus and Local Bus Arbitration-Level	
QBus Mask Register 1 (QBUSMR1) is initialized to 0xFF80	**Section 9.3.9**, *EQBS Programming Model,* on page 9-18
EE Signals Control Register, EE1[DEF] field is initialized to '01'	EONCE chapter of *SC140 DSP Core Reference Manual*
Direct Slave Interface (DSI), DSI Internal Address Mask Register (DIAMR[9, 11]) and DSI Internal Base Address Register (DIBAR[9, 11])	**Section 14.5**, *DSI Programming Model,* on page 14-29
Edge/Level-Triggered Interrupt Register F (ELIRF) is initialized such that the IRQ20 (EOnCE interrupt) is edge-triggered mode. The LIC is initialized to edge-triggered mode accordingly to the TDM and timers initialization at reset. The LIC and GIC are also initialized such that virtual interrupts are referred to as edge.	**Section 17.3.3.2**, *Interrupt Priority Structure and Mode,* on page 17-40 **Section 20.7**, *TDM Programming Model,* on page 20-34 **Section 22.1**, *Timers Programming Model,* on page 22-8 **Section 17.3**, *Interrupts Programming Model,* on page 17-23
TDMxRIR, TDMxTIR	**Section 20.7**, *TDM Programming Model,* on page 20-34
Ethernet Threshold and Priority Registers	**Section 25.17**, *Ethernet Controller Programming Model,* on page 25-49

The boot program initializes the interrupt handler table base address (VBA register of each SC140 core) at its first instruction execution. Until this base address is initialized, no Non-Maskable Interrupt (NMI) should occur. After the VBA is initialized, all the SC140 cores enter Debug mode if any NMI is asserted. An SC140 core also enters Debug mode if the TRAP, ILLEGAL, DEBUG, or OVERFLOW interrupt is asserted. You must load code that handles any interrupts, and you typically change the location of the interrupt handler table as soon as possible in the user boot program. Refer to **Section 17.1.5**, *Interrupt Routing*, on page 17-18. Addresses 0x01076E00–0x01076FFF are reserved and cannot be written or used while the MSC8113 boot program is running.

If the RSR[EHRS] bit is cleared (see **Section 5.6.2**, *Reset Status Registers*) and the external soft reset signal is asserted, only QBUSMR1, EE_CTRL, ELIRF, TDMxRIR, TDMxTIR, PPC_ACR, PPC_ALRH, PPC_ALRL, LCL_ACR, LCL_ALRH, LCL_ALRL, LIC, and GIC registers are initialized and all the SC140 cores jump to address 0x0.

6.2 Booting From an External Memory Device

The MSC8113 device boots from an external memory device on the system bus. The MSC8113 boot program retrieves an address from the external memory and jumps to that address. Typically, the user boot program located at the retrieved address writes a loader program to the internal memory. That loader program reads code and data from the external memory and writes it to the internal memory. It is faster to run a loader program in the internal memory than to run code located in external memory.

The MSC8113 boot chip-select operation allows address decoding for a user boot ROM before system initialization for external memory boot operation. The $\overline{\text{CS0}}$ signal is the boot chip-select output, and the boot external memory should connect to it. The MSC8113 boot chip-select operation also provides a programmable port size during system reset, by writing the BPS field in the Hard Reset Configuration Word (HRCW).

Note: For details on MSC8113 chip-select operation, see **Section 12.3.3**. See also **Section 5.6.1**, *Hard Reset Configuration Word*.

The boot program accesses an address table that resides at address 0xFE000110 (see **Table 6-3**). This address table holds the 32-bit address of the user program in big-endian format. The retrieved address is user-programmable, and the target user program can be placed in any address in the space controlled by the chip-select. The MSC8113 device retrieves the user program address from the table according to the ISBSEL field in the HRCW.

Table 6-3. External Memory Address Table (32-Bit Wide EPROM)

Address (Big Endian Format)	+0	+1	+2	+3
ISBSEL= 0 — 0xFE000110	A(0-7)	A(8-15)	A(16-23)	A(24-31)
ISBSEL= 1 — 0xFE000114	A(0-7)	A(8-15)	A(16-23)	A(24-31)
ISBSEL= 2 — 0xFE000118	A(0-7)	A(8-15)	A(16-23)	A(24-31)
ISBSEL= 3 — 0xFE00011C	A(0-7)	A(8-15)	A(16-23)	A(24-31)
ISBSEL= 4	Reserved. Do not use.			
ISBSEL= 5				
ISBSEL=6 — 0xFE000128	A(0-7)	A(8-15)	A(16-23)	A(24-31)
ISBSEL=7 — 0xFE00012C	A(0-7)	A(8-15)	A(16-23)	A(24-31)

In a multi-device environment, multiple MSC8113 devices are initialized from the system bus. Each device can access the external memory, and optionally, a master device can initialize the other devices. The master device loads code and data for all slaves devices through the system bus. To reduce traffic on the system bus, the slave device user program should not access the bus.

The master device is identified by one of the following fields:

- DSI Chip ID Register (DCIR), Chip ID Value field (see **Section 14.5**)
- ISBSEL field in the Exception and Mode Register (EMR), which resides in the SC140 core Program Control Unit (see the *SC140 DSP Core Reference Manual* and **Table 2-1**).
- ISB field in the SIU IMMR (see **Section 4.2.1**).

When the MSC8113 device boots from an external memory device, the user boot program typically does the following:

- Writes a loader program to the internal memory. This program loads code and data to the internal RAM, thus enabling faster loading of code and data to the internal memory.
- Signals SC140 cores 1, 2, and 3 to initiate a jump to address 0x0 for the M1 memory of SC140 cores 1, 2, and 3 by asserting VIRQ[9, 17, 25] for those SC140 cores (see the discussion of the Virtual Interrupt Generation Register (VIGR) in **Section 17.3**).

6.3 Booting from an External Host (DSI or System Bus)

When the MSC8113 is booted from an external host, the host waits for the MSC8113 boot program to finish its default initialization and then initializes the device by typically loading code and data to the internal memory according to the memory map shown in **Figure 8-2**, *Host on the System Bus Memory Map View Example*, on page 8-4. The external host should poll the Valid bit (V) of the BR10 register. The valid bit is set when the MSC8113 boot code finishes the default initialization and the external host can access the internal resources, including internal memory. When the external host finishes its initialization sequence, it should notify the MSC8113 by asserting the virtual interrupt 1 (VIRQ1) to SC140 core 0, which in turn signals all the other SC140 cores to jump to address 0x0 of their M1 memory. The user boot program running from the external host typically does the following:

- Waits for Valid bit of Bank 10 (BR10) to be set, see **Section 12.8**.
- Loads code and data to internal RAM.
- Signals SC140 core 0 to initiate a jump to address 0x0 for all SC140 cores M1. Memory by asserting VIRQ1 for Core 0 (see VIGR in **Section 17.3**).

6.4 Booting From the TDM Interface

In a system that boots from the TDM interface, a TDM boot master device writes blocks of code and data into the memories of multiple MSC8113 devices as illustrated in **Figure 6-1** and according to the memory map shown in **Figure 8-6**. Two layered protocols are defined: a TDM physical layer, and a TDM logical layer handshake. The valid bit of Bank 10 (BR10) is asserted at the end of the TDM session.

6.4.1 Initializing the TDM Physical Layer

The boot master transmits messages to multiple MSC8113 devices on TDM channel 0. Each MSC8113 transmits back on a different TDM channel that equals the MSC8113 CHIP_ID as defined in the DCIR. The MSC8113configures the size and type (T1 or non T1) of the received and transmitted frame by synchronizing to the TDM Clock and Sync signals of the master boot device.

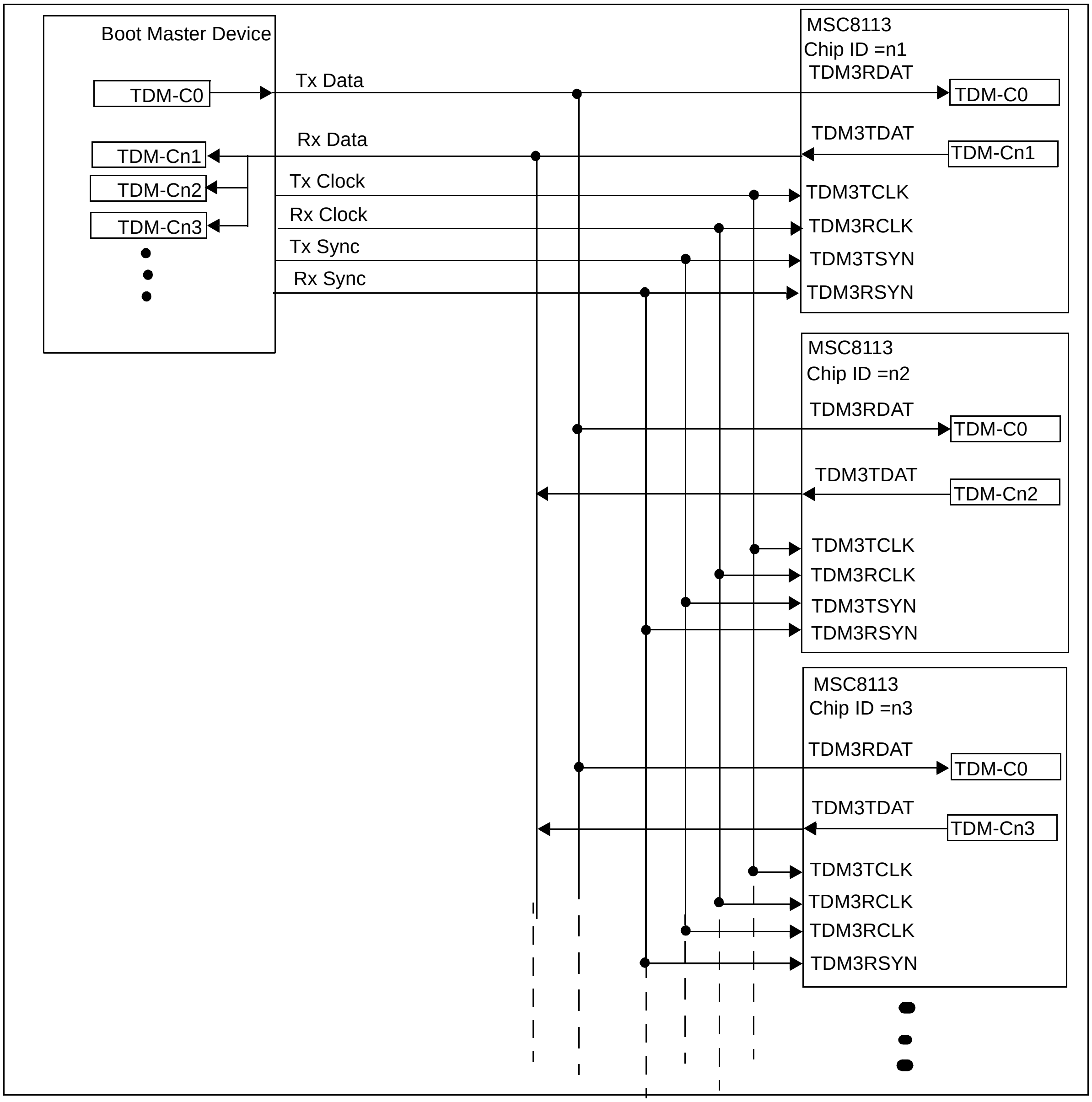


Figure 6-1. TDM Boot System

6.4.1.1 Receiver Initialization

For a non-T1 receive operation, the TDM3RDAT is sampled for eight consecutive clock cycles starting one clock after each first clock on which the TDM3RSYN is detected high. See **Figure 6-2**. For a 16-bit non-T1 receive frame operation, the TDM3RDAT is sampled for 16 consecutive clock cycles starting one clock after each first clock on which the TDM3RSYN is detected high. See **Figure 6-3**. D0 is the MSB and the first to be received. The number of channels at the receiver frame is limited to 128.

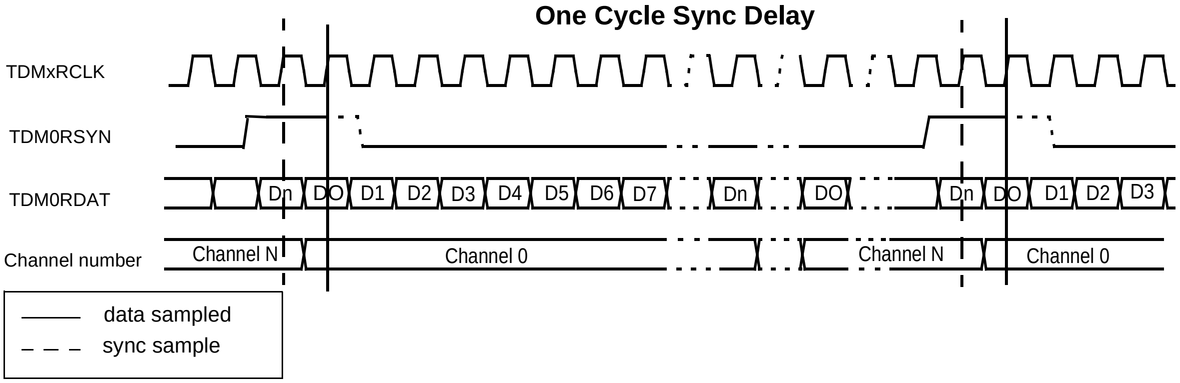


Figure 6-2. Receive Frame Non-T1 Configuration

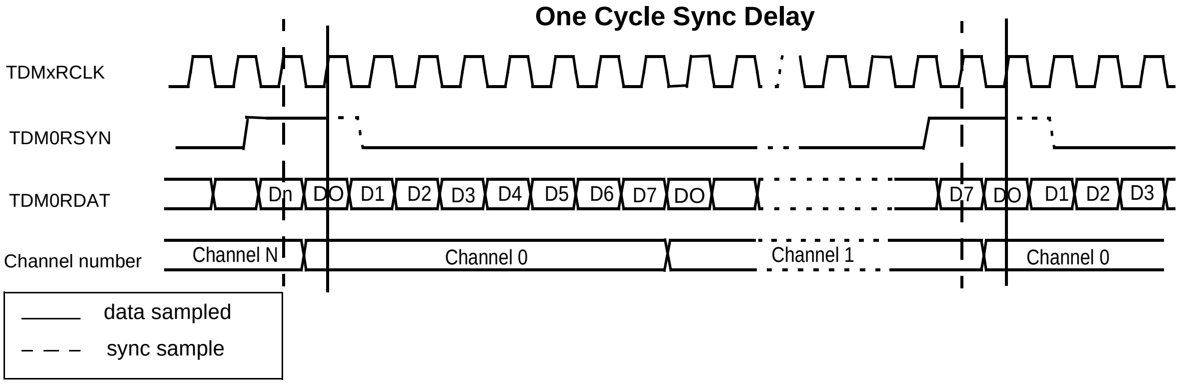


Figure 6-3. 16 bit Receive Frame Non-T1 Configuration

For a T1 receive operation, the TDM0RDAT is sampled for eight consecutive clock cycles starting two clocks after each first clock on which the TDM3RSYN is detected high. See **Figure 6-4**.

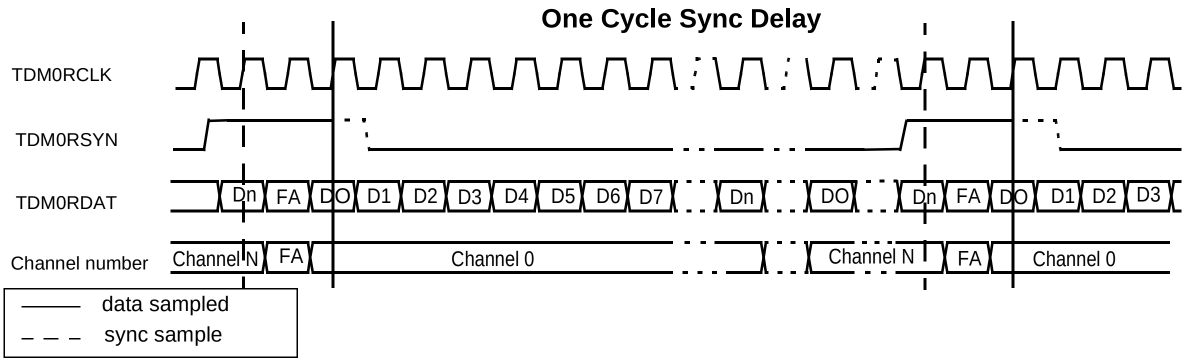


Figure 6-4. Receive Frame T1 Configuration

6.4.1.2 Transmitter Initialization

For a non-T1 transmit operation, 8-bit channels are transmitted on the TDM3TDAT signal in consecutive clock cycles starting on the negative edge of the first clock after which the TDM3TSYN is detected high. See **Figure 6-5**. Each MSC8113 transmits on a channel that equals its chip ID.

Note: D0 is the msb and first to be sent.

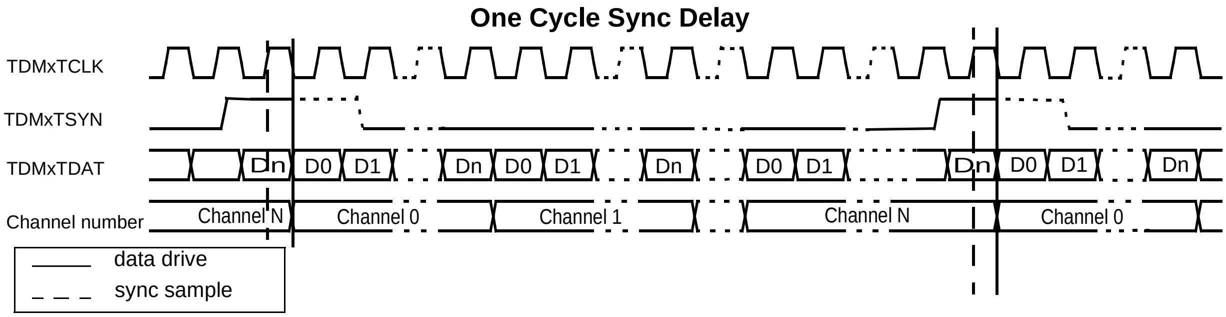


Figure 6-5. Transmit Frame Non-T1 Configuration

For a T1 transmit operation, 8-bit channels are transmitted on the TDM3TDAT signal in consecutive clock cycles starting on the negative edge of the second clock after which the

TDM3TSYN is detected high. See **Figure 6-6**. Each MSC8113 transmits on a channel that equals its chip ID.

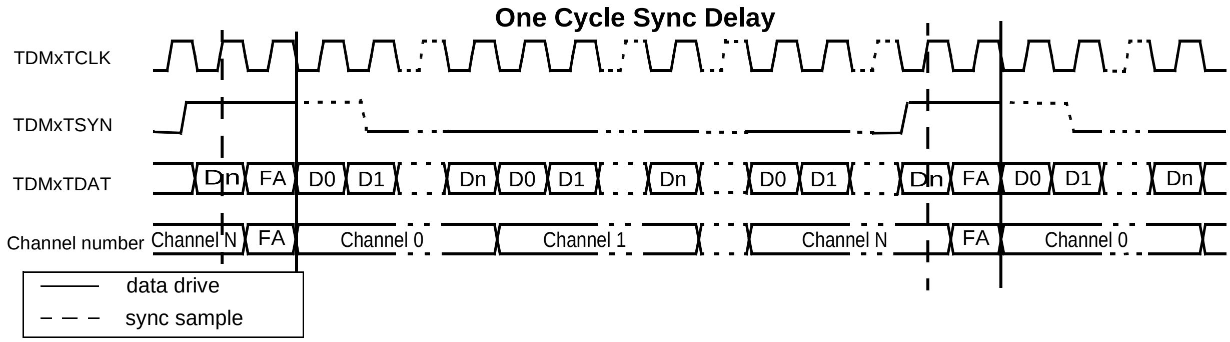


Figure 6-6. Transmit Frame T1 Configuration

6.4.2 TDM Logical Layer Handshake

The logical layer works directly with the physical layer to implement a block transfer protocol that is defined here. While the physical layer ensure that the data is sent to all the TDM devices, the logical layer sends the data from the TDM boot master device to one or all of the TDM boot slave devices and specifies the destination address and ensure its correct transmission. The TDM boot master device message structure used to exchange data is described in **Section 6.4.2.1**. The block transfer protocol operation is explained in **Section 6.4.2.2**.

6.4.2.1 Messages Structure

The TDM boot master device message contains the fields described in **Table 6-4**. The MSC8113 slave device acknowledge message contains the fields described in **Table 6-5**.

Table 6-4. Block Transfer Message

Block Transfer Message			Direction: from Master Boot Chip
Field Size	**Field Name**	**Field Value**	**Description**
4 bytes	PRM	0x44332211	*Preamble*. Indicates the start of the message. A value of 0x44332211 (first byte sent is 0x11) is assigned to the block transfer message (BTM), and a value of 0x6655 (first byte sent is 0x55) is assigned to the block transfer acknowledge message (BTAM).
1 byte	DCID		*Destination CHIP-ID/Broadcast=0xFF*. Identifies the target MSC8113 slave device to accept this message.
1 bytes	SN		*Send Sequence Number modulo 256*. The sequence number of the BTM.
1 bytes	EB		*End Block Flag*. A value of 0xFF in the EB field indicates the last message. After the last message, all SC140 cores jumps to address 0x0 of their M1 Memory.
3 bytes	PLDS		Payload field size in bytes, 0 = 2^{24} bytes.

Table 6-4. Block Transfer Message (Continued)

Block Transfer Message			Direction: from Master Boot Chip
Field Size	**Field Name**	**Field Value**	**Description**
4 bytes	DA		*Destination Address of Data Block*. The destination address for the data field of the slave MSC8113 internal memory as determined by the MSC8113 memory map (SC140 Core 0). See **Figure 8-1**, *SC140 Core View Memory Map Example,* on page 8-3. Addresses 0x01076E00–01076FFF are reserved and cannot be used.
2 bytes	HCRC		CRC-16 of PRM,DCID,SN,EB,PLDS and DA fields.
Up to 2^{24} bytes	PLD		*PayLoad*. Specifies the size of the payload field in bytes. A value of 0 indicates a size of 2^{24} bytes. The size of the payload data should be divisible by two.
2 bytes	CRC		*CRC-16 of PLD field*. All CRC fields are 16-bit CRC represented by $x^{16} + x^{15} + x^2 + 1$. The HCRC field is a CRC-16 calculation of the BTM headers fields (PRM, PLDS, SN, EB, DCID, and DA fields). The CRC field is a CRC-16 calculation of the PLD field in the BTM message. The ACRC field is a CRC-16 calculation of the APRM, SCID, and RN fields.

Table 6-5. Block Transfer Acknowledge Message

Block Transfer Acknowledge Message			Direction: from MSC8113 Slave Chip
Field Size	**Field Name**	**Field value**	**Description**
2 bytes	APRM	0x6655	*Preamble*. Indicates the start of the message. A value of 0x44332211 (first byte sent is 0x11) is assigned to the Block Transfer Message (BTM), and a value of 0x6655 (first byte sent is 0x55) is assigned to the Block Transfer Acknowledge Message (BTAM).
1 byte	SCID		Source CHIP-ID
1 bytes	RN		*Receive Sequence Number modulo 256*. The expected sequence number to receive next.
2 bytes	ACRC		CRC-16 of APRM,SCID and RN fields

Note: Fields larger than one byte send their LSB first.

6.4.2.2 Operation

The MSC8113 slave device logic layer implements the following algorithm:

1. Synchronize to the preamble (PRM) field of the BTM.
2. If an error is detected in the HCRC field, return to step 1.
3. If the DCID field identifies the MSC8113 slave device CHIP-ID or a broadcast message, write the payload data (PLD) to the destination address (DA).
4. Send a BTAM as follows:
   a. If the CRC field is received with an error, send BTAM with the current RN value and the MSC8113 slave device CHIP-ID in the SCID field.
   b. If the CRC field is received with no error and the RN value does not equal the that of the received SN field, send BTAM with the current RN value and the MSC8113 slave device CHIP-ID in the SCID field.
   c. If the CRC field is received with no error and the RN value equals that of the received SN field, update RN to be RN plus one modulo 256 and send BTAM with the updated RN value and the MSC8113 slave device CHIP-ID in the SCID field.
5. If the CRC field is received with no error, the RN value is correct, and the end block (EB field) flag is set, the MSC8113 slave device finishes the TDM boot session, and all its SC140 cores jump to address 0x0 of their M1 memory. Otherwise, return to step 1.

The TDM boot master device works in two modes:

- *Handshake mode*. Use the stop-and-wait technique to send the BTM messages and wait for the BTAM message or to time out equal to a 32-frame time of the TDM Tx port.
- *Non-Handshake mode*. Do not wait for the BTAM message because BTM messages can be sent in a sequence without any wait time. The BTAM messages are sent by the MSC8113 slave devices, but their correctness is not guaranteed.

Note: The MSC8113 slave device RN value is initialized to zero at the start of the TDM boot session. When the HCRC field is received with no error and the CRC field is received with error, corrupt data is written to the MSC8113 slave device memory. When a broadcast message is sent by the TDM boot master device, all MSC8113 slave devices send back acknowledge messages.

Figure 6-7 shows the MSC8113 slave device logic layer algorithm. **Figure 6-7** shows how a TDM master sends blocks of data to MSC8113 devices.

Note: In 16-bit frame mode, the maximum rate allowed at the TDM port (TDM3RCLK) is half the maximum rate available at the TDM port.

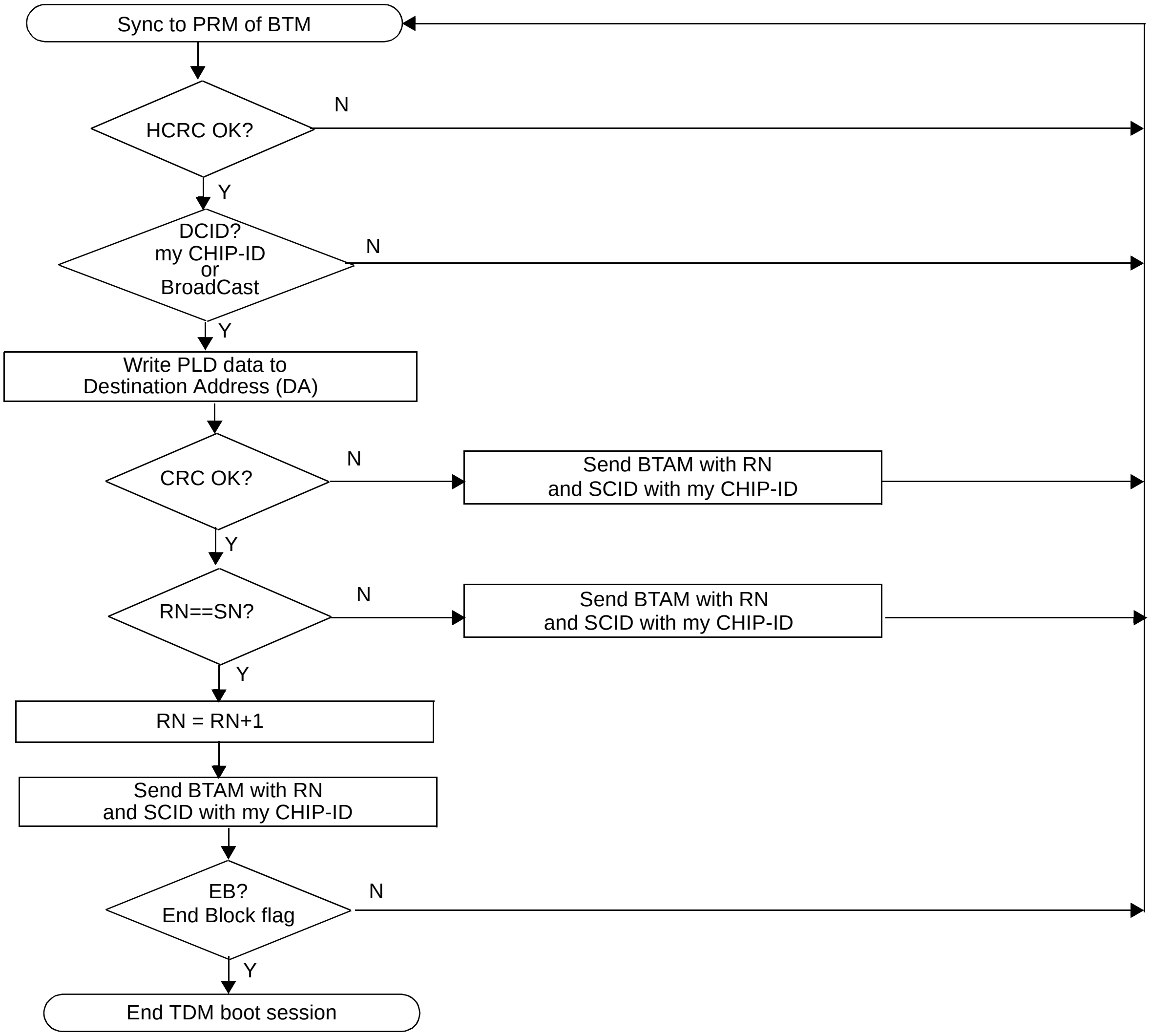


Figure 6-7. MSC8113 Logic Layer Algorithm

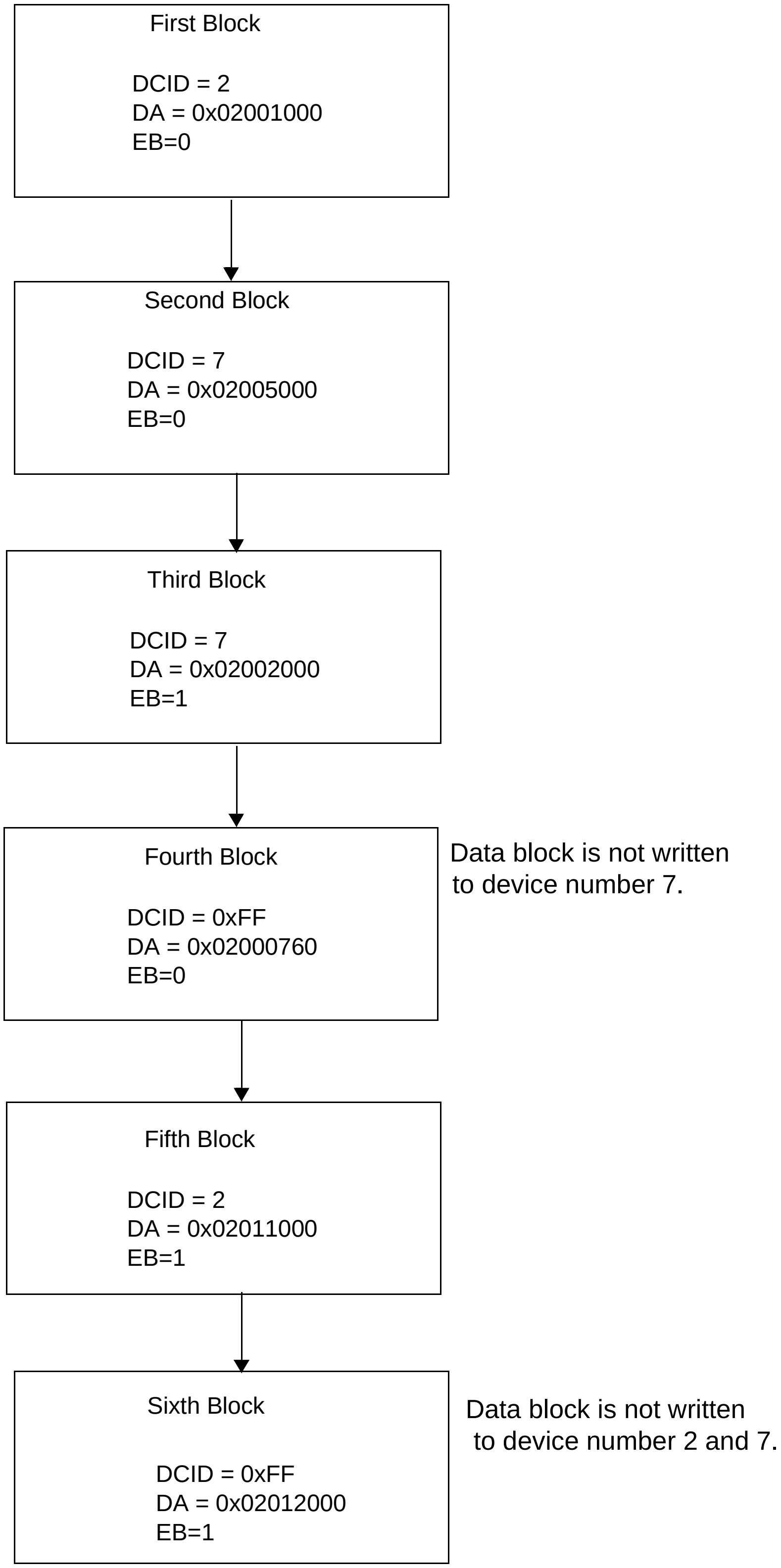
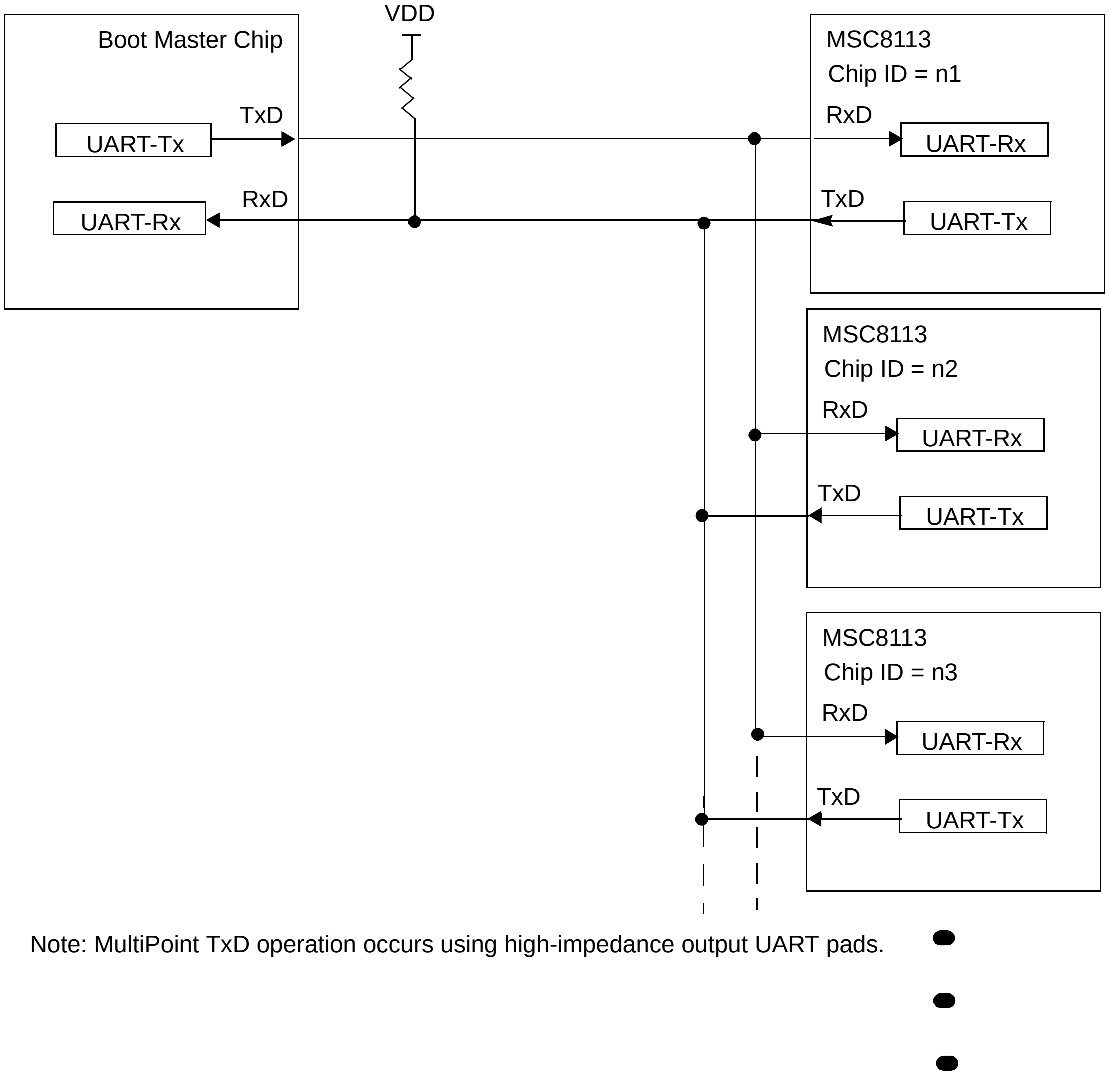


Figure 6-8. TDM Block Stream Structure Example from TDM Master Boot Device

6.5 Booting From a UART Device

In a system that boots from a UART device, a UART boot master device writes blocks of code and data into the memories of multiple MSC8113 devices (see **Figure 6-8**) and according to the memory map shown in **Figure 8-6**. UART booting occurs in a two-layer protocol: a UART physical layer and a UART logical layer handshake (see **Section 6.4.2**, *TDM Logical Layer Handshake,* on page 6-8). The broadcast message handshake capability of the TDM logical layer is not allowed in the UART logical layer. The Valid bit of Bank 10 (BR10) is asserted at the end of the UART session.



Note: MultiPoint TxD operation occurs using high-impedance output UART pads.

Figure 6-9. UART Boot System

The UART is initialized as follows:

- 9600 baud rate (at an IPBus rate of 100 MHz):
- One start bit, eight data bits, one stop bit.
- Wake up by idle line.
- Parity function disabled.
- TxD actively driven as an output.
- Full-duplex operation.

At the end of the UART loading process, all SC140 cores jump to address 0x0 of their M1 memory.

6.6 Booting from I²C Slave Memory Device

In a system that boots from an I^2C slave memory device, when the MSC8113 boot program finishes its default initialization, it starts to retrieve blocks from an external I^2C-slave memory device such as a serial EPROM, using the I^2C SM (see **Chapter 24**, *I²C Software Module*) that is implemented in the boot code. The I^2C slave device address is $1010A_0A_1A_2b$ where the $A_0A_1A_2$ bits are the high bits of the address being retrieved. The address field is 19 bit(3 additional bits from the I^2C slave device address), thus enable accesses of up to 1 MB of memory array. In a multi-master environment, the I^2C SM allows concurrent starts of block retrieves, so multiple masters concurrently load code and data (using the loose arbitration scheme of the I^2C protocol), thus reducing the loading time of any number of masters to the loading time of one master. Blocks are retrieved and written to internal memory until End Block is acknowledged. The first block resides at address 0x70020 of the I^2C slave memory. At the end of the I^2C loading process, all SC140 cores jump to address 0x0 of their M1 memory. See **Table 6-6** for details.

Table 6-6. Block Structure

Field Name	Field Size	Field Address	Description
Block Control	1 byte	Block_Address + 0x0	A 1-byte control field.
Block Size	3 bytes	Block_Address + 0x1	The block size is a 3-byte field that specifies the number of bytes in the PayLoad Data field. If the number of bytes does not align to the Data Structure Size, the last written value is padded with zeros.
Next Block Address	4 bytes	Block_Address + 0x4	The Next Block Address is a 4-byte field that holds the next block address in the serial memory. If the Next Block Address equals 0x0, the bootloader assumes that the next block is sequential. If Next Block Address equals 0xFFFFFFFF, the block is the End Block.
Destination Address	4 bytes	Block_Address + 0x8	The Destination Address is used by the boot program to locate where to write the I^{2}C data. Use **Figure 8-7** to determine the correct address.
Payload Data	up to 2^{24} bytes	Block_Address + 0xc	The PayLoad Data holds up to 2^{24} bytes of data to be written to on-device memory according to the DSS field in the Block Control field. To write to the internal memory in Big-Endian mode, the most significant byte of the data structure must be stored at the lower address.
Checksum	2 bytes	Block_Address + Block_Size + 0xc	Checksum is a 2-byte field that holds the XOR of all previous data. The boot code XORs each received 2 bytes with the previous checksum value and verifies the validation by comparing it to this field. If the CSE bit is set and the first block retrieve fails, a second retrieve is performed. If the second retrieve fails, all cores enter debug-halt mode.
Checksum	2 bytes	Block_Address + Block_Size + 0xc + 0x2	Checksum is a 2 bytes field that hold the XOR of all previous data. The boot code XORs each received 2 bytes to the previous checksum value and verifies its validation by comparing it to this field. If the CSE bit is set and the first block retrieve fails, a second retrieve is performed. If the second retrieve fails, all cores enter debug-halt mode.
Note: For the field size, the most significant byte is at the lower address.			

6.6.1 Procedure Flow

The flow of the I^2C boot procedure is shown in **Figure 6-10**.

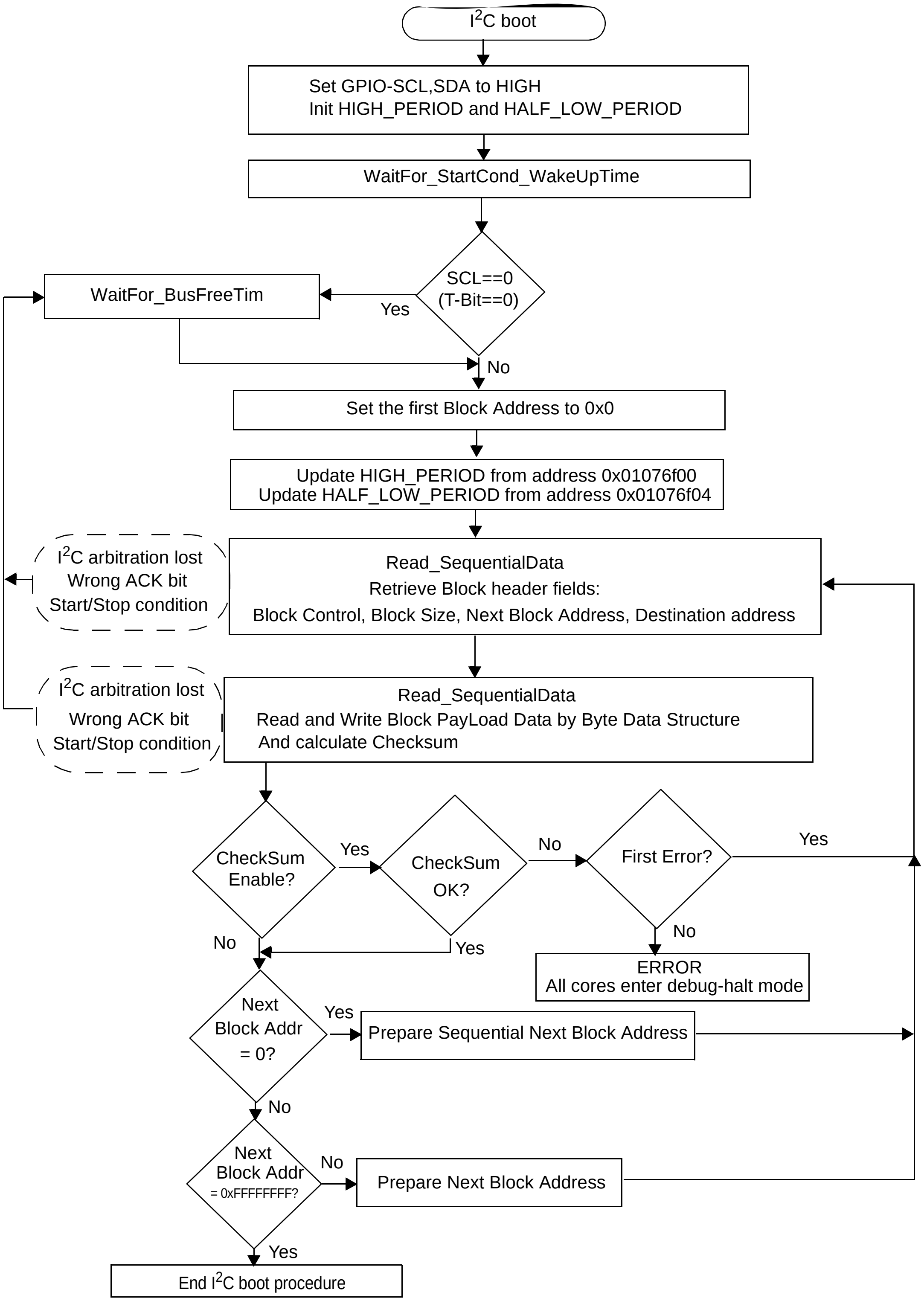

I2C boot
Set GPIO-SCL,SDA to HIGH
Init HIGH_PERIOD and HALF_LOW_PERIOD
WaitFor_StartCond_WakeUpTime
SCL==0
(T-Bit==0)
Yes
WaitFor_BusFreeTim
No
Set the first Block Address to 0x0
Update HIGH_PERIOD from address 0x01076f00
Update HALF_LOW_PERIOD from address 0x01076f04
I2C arbitration lost
Wrong ACK bit
Start/Stop condition
Read_SequentialData
Retrieve Block header fields:
Block Control, Block Size, Next Block Address, Destination address
I2C arbitration lost
Wrong ACK bit
Start/Stop condition
Read_SequentialData
Read and Write Block PayLoad Data by Byte Data Structure
And calculate Checksum
CheckSum
Enable?
Yes
CheckSum
OK?
No
First Error?
Yes
No
Yes
No
ERROR
All cores enter debug-halt mode
Next
Block Addr
= 0?
Yes
Prepare Sequential Next Block Address
No
Next
Block Addr
= 0xFFFFFFFF?
No
Prepare Next Block Address
Yes
End I2C boot procedure

Figure 6-10. I^2C Boot Procedure Flow

The I^{2}C loading procedure restarts under three conditions:

1. Assertion of a start or stop condition in a byte read or write session.
2. I^{2}C arbitration is lost because the bit transmitted on SDA when SCL is low does not equal the bit received on SDA when SCL is high. Each I^{2}C master checks this only when it transmits a bit.
3. The ACK bit is not as expected. On read session, ACK is low for all except the last byte.

6.6.2 I^{2}C System

Figure 6-11 shows the system connectivity for I^{2}C devices.

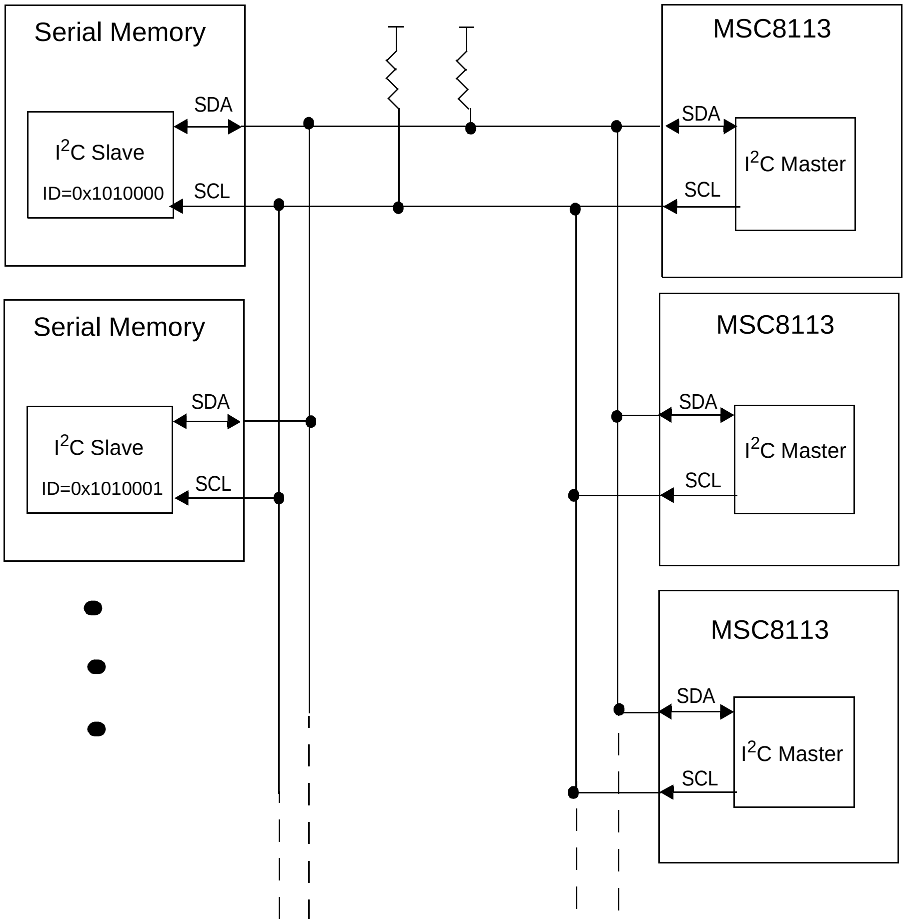


Figure 6-11. I^{2}C Boot System Example

Clocks 7

The MSC8113 device has two main clocks, CORES_CLOCK and BUSES_CLOCK, both of which are synchronized and phase aligned. The CORES_CLOCK supplies a timing signal for the extended core, including:

- SC140 cores
- M1 and M2 memories
- Instruction cache
- Write buffer
- PIC
- LIC

The BUSES_CLOCK clocks:

- SIU
- DMA
- DSI
- TDM
- Timers
- UART
- GIC
- Ethernet

Some MSC8113 subsystems are clocked by other special clocks, as follows:

- The direct slave interface (DSI) has two clearly separated clock zones:
  - It interfaces with the external host asynchronously or via a synchronous interface clocked by the HCLKIN signal.
  - It interfaces with the internal local bus via the BUSES_CLOCK.
- Each TDM has three clock zones:
  - The receiver is clocked by RCLKx.
  - The transmitter is clocked by TCLKx.
  - The interface to the local bus is clocked by the BUSES_CLOCK.
- The timers can also be clocked by the GPIO signals, which are clocked separately from the timer interface to the IPBus.

- The Ethernet interface has two clock zones:
  - — The serial interface:
    - MII mode. The receiver clock is ETHRX_CLK; the transmitter clock is ETHTX_CLK.
    - RMII mode. The receiver and transmitter are clocked by ETHREF_CLK.
    - SMII mode. The receiver and transmitter are clocked by ETHCLOCK.
  - — Local bus interface is clocked by the BUSES_CLOCK.

7.1 Clock Generation

The CLKOUT signal, the internal CORE_CLOCK, and the BUSES_CLOCK are generated as a function of the CLKIN signal, guaranteeing minimum skew between the CLKOUT and BUSES_CLOCK of all the MSC8113 devices. **Figure 7-1** shows the MSC8113 clock scheme. CLKIN is generated by an external oscillator and is fed to the SPLL that divides and multiplies its frequency according to the PLLRDF, PLLFDF, PLLODF, and the BUSDF factors as configured by the System Clock Mode Status Register (SCMSR) (see **Section 7.4**, *Clocks Programming Model*). The PLLPDF field divides the frequency by 1 or 2. The PLL VCO clock is generated by multiplying the PLL predivider clock by: 2 × PLLFDF × PLLODF × BUSDF. The PLL output clock is generated by dividing the PLL VCO clock by: 2 × PLLODF. The SCMSR[BUSDF] bit value controls the frequency ratio between the BUSES_CLOCK and the CORES_CLOCK. The CLKOUT is typically the system bus clock or the local bus clock.

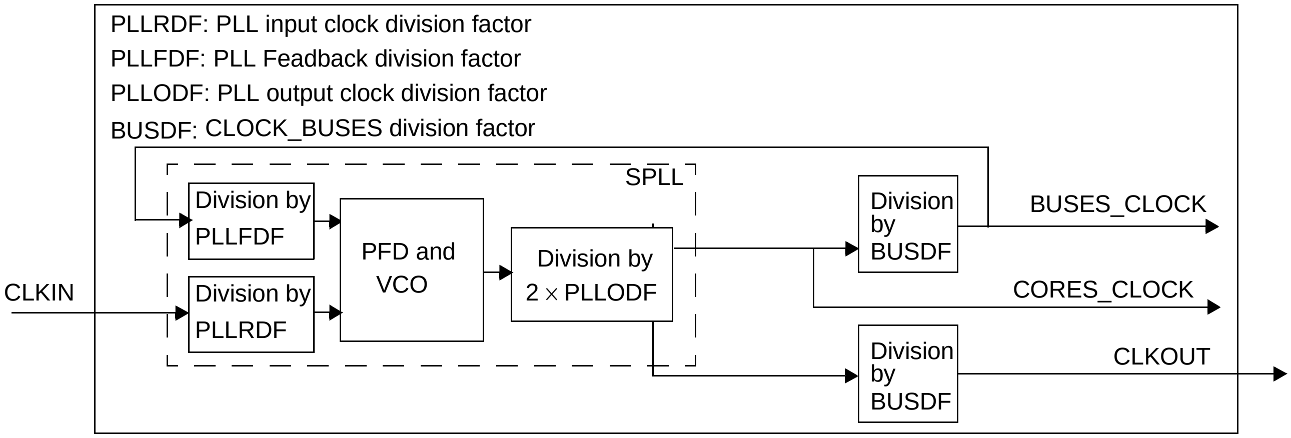

PLLRDF: PLL input clock division factor
PLLFDF: PLL Feadback division factor
PLLODF: PLL output clock division factor
BUSDF: CLOCK_BUSES division factor
SPLL
Division by PLLFDF
Division by PLLRDF
PFD and VCO
Division by 2 × PLLODF
Division by BUSDF
Division by BUSDF
CLKIN
BUSES_CLOCK
CORES_CLOCK
CLKOUT

Figure 7-1. CORES_CLOCK, BUSES_CLOCK, and CLKCOUT Generation

In **Figure 7-2**, the CORES_CLOCK and BUSES_CLOCK frequency ratio is 1:3.

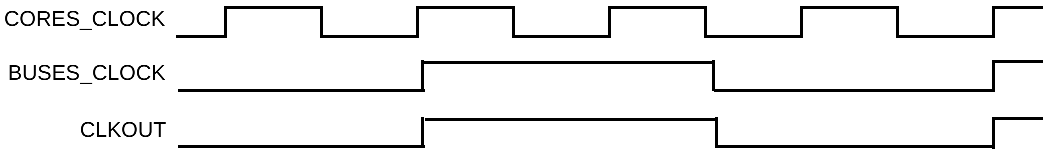


Figure 7-2. CORES_CLOCK, BUSES_CLOCK, and CLKOUT Example

7.2 Board-Level Clock Distribution

There are two board level clock distribution modes:

- Single-Master
- Multi-Master

7.2.1 Single Master Mode Board-Level Clock Distribution

There are three ways to implement clock distribution in Single-Master mode:

- CLKOUT with zero-delay buffers
- CLKOUT with no buffers
- CLKIN mode. This is the recommended clock scheme for high-frequency synchronous memory interface (SDRAM).

In the Single-Master CLKOUT with zero-delay buffers method, each MSC8113 device has dedicated slave devices on the board (SDRAMs, for example). With this method, the CLKOUT of each MSC8113 device connects through a zero-delay buffer to the clock input pin of its dedicated slave devices on the board. **Figure 7-3** illustrates a system in which each of three MSC8113 devices connects to a dedicated SDRAM memory device on the board through the zero-delay buffers.

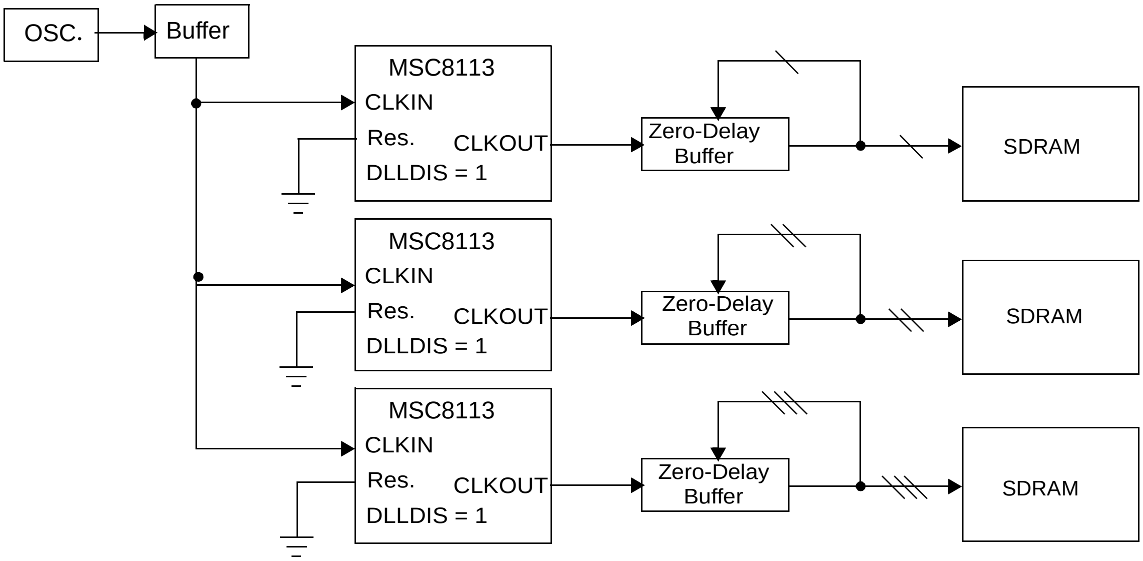


Figure 7-3. MSC8113 Clock Distribution In Single-Master Mode Using CLKOUT With Zero-Delay Buffers

You must maintain the following guidelines for this mode:

- Clocks marked with the same number of parallel lines must use an equivalent buffer and route on the board.
- Maximum load on CLKOUT must not exceed 10 pF.

- Each zero-delay buffer must have a peak-to-peak phase jitter of less than 0.3 ns.

In the Single-Master CLKOUT synchronization mode with no buffers, each MSC8113 device has dedicated slave devices on the board (SDRAMs, for example). In this mode, the CLKOUT of each MSC8113 device connects directly to the clock input pin of its dedicated slave devices. **Figure 7-4** illustrates a system in which each of three MSC8113 devices connects to a dedicated SDRAM memory device on the board. The maximum load on CLKOUT must not exceed 10 pF.

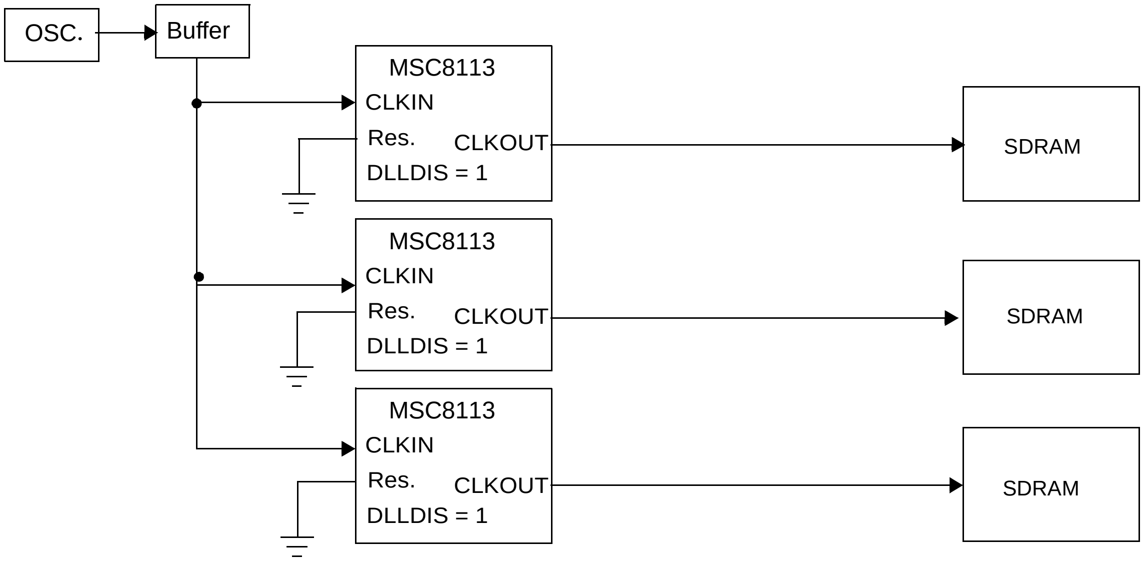


Figure 7-4. MSC8113 Clock Distribution And Synchronization In Single-Master Mode Using CLKOUT With No Zero-Delay Buffers

In the Single-Master CLKIN method, CLKIN is provided by one or more on-board oscillators and connects via the on-board balanced clock tree to the CLKIN input of each MSC8113 device and clock input of the dedicated slave devices. You must choose one of the clock configuration modes for which the BUSES_CLOCK:CLKIN ratio is 1:1 (modes 0, 7, 15, 19, 21, 23, or 28–31 in **Table 7-1**). **Figure 7-5** illustrates a system in which three MSC8113 devices and three dedicated SDRAM memory devices connect on the board.

Note: Clocks marked with the same number of parallel lines should have an equivalent buffer and route on the board.

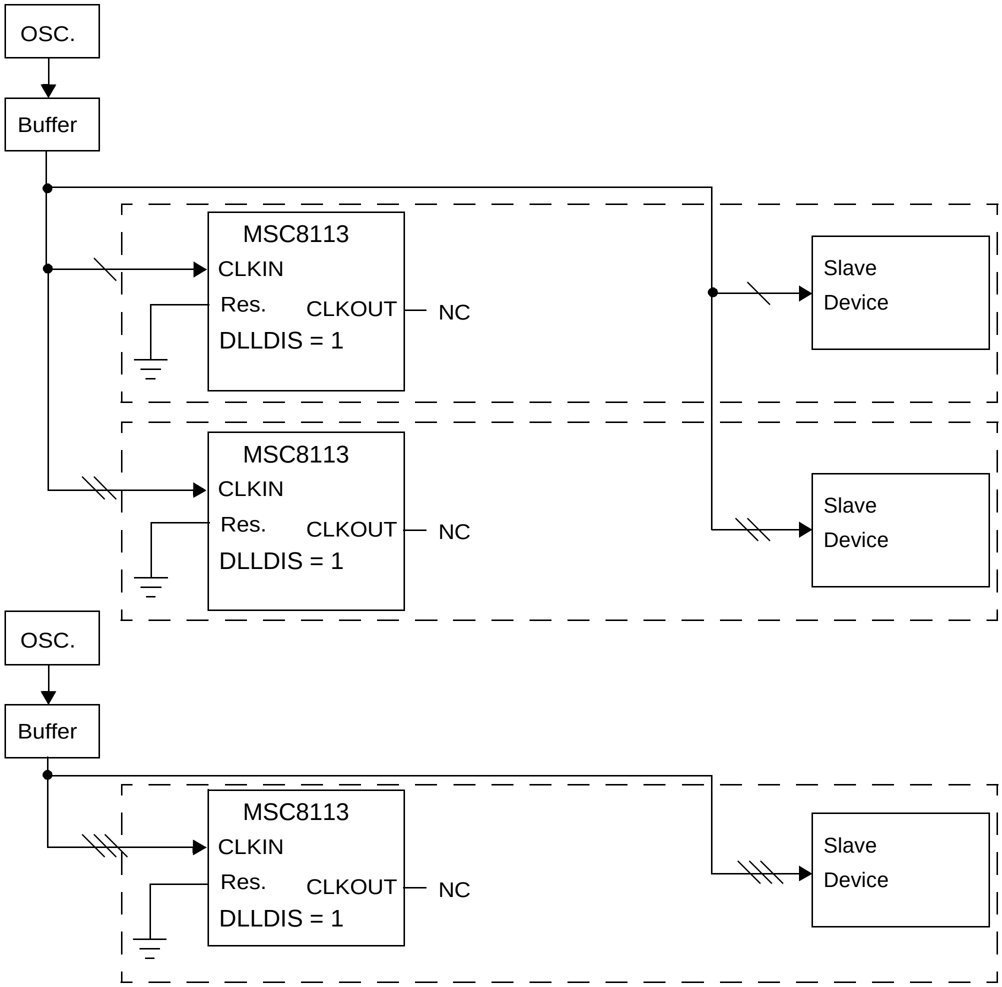


Figure 7-5. MSC8113 Clock Distribution in Single-Master Mode Using CLKIN

7.2.2 Multi-Master Mode Board-Level Clock Distribution

There are two methods for clock distribution on a circuit board in Multi-Master mode:

- CLKOUT (only recommended for designs migrated from the MSC8102 device)
- CLKIN

In the Multi-Master CLKOUT distribution method, the MSC8113 devices share slave devices on the board (SDRAMs for example). The CLKOUT of the master MSC8113 device connects through a zero-delay buffer, to the clock input pins of the shared slave devices on the board. **Figure 7-6** illustrates a system in which three MSC8113 devices connect to the one shared SDRAM memory device on the board.

Note: Clocks marked with the same number of parallel lines should use an equivalent buffer and route on the board.

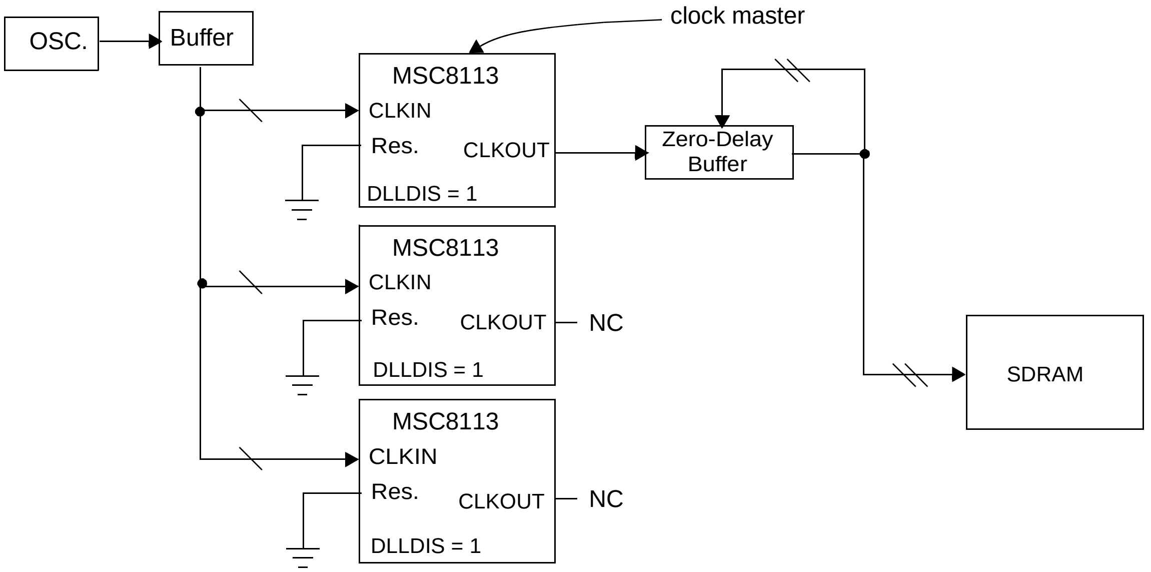


Figure 7-6. MSC8113 Clock Distribution Using CLKOUT in Multi Master Mode

In the Multi-Master CLKIN distribution method, CLKIN is clocked by an on-board oscillator and connects to the CLKIN input port of all the MSC8113 devices on the board and all the shared slave device clock input ports as shown in **Figure 7-7**. You must chose one of the clock configuration modes for which the BUSES_CLOCK:CLKIN ratio is 1:1 (modes 0, 7, 15, 19, 21, 23, or 28–31 in **Table 7-1**). **Figure 7-7** illustrates a system in which three MSC8113 devices and also three shared SDRAM memory devices connect on the board.

Note: Clocks marked with the same number of parallel lines should use an equivalent buffer and route on the board.

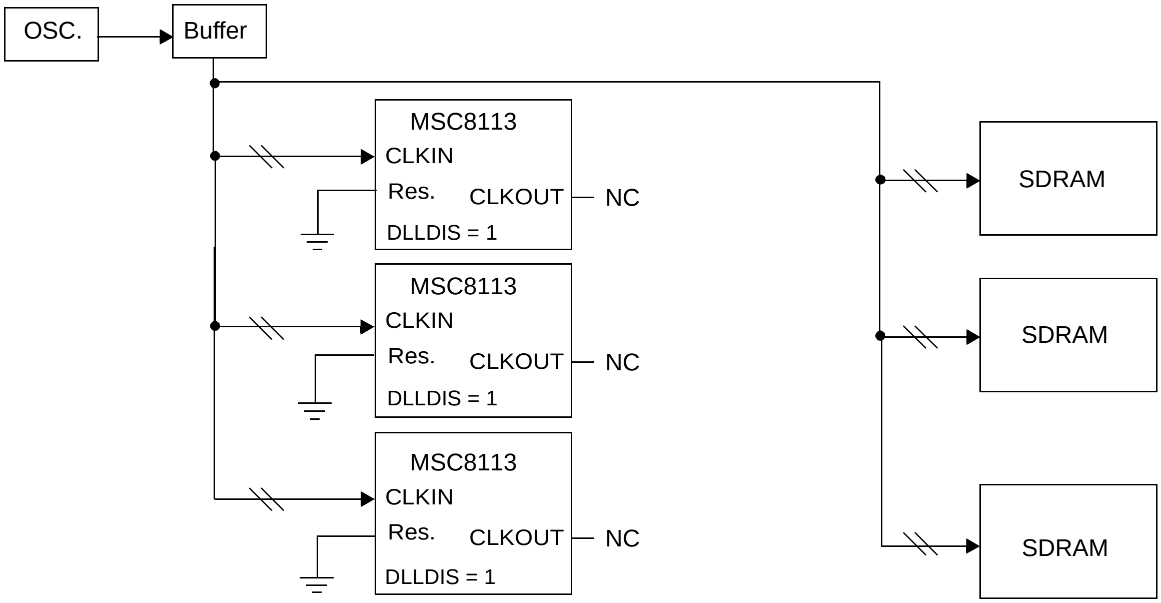


Figure 7-7. MSC8113 Clock Distribution in Multi-Master Mode Using CLKIN Example

7.3 Clock Configuration

MODCK[1–2] and the MODCK[3–5] bits of the Hard Reset Configuration Word (HRCW), discussed in **Section 5.6.1**, map the MSC8113 clocks to one of the valid 27 configuration mode options. Each option determines the CLKIN, BUSES_CLOCK, and CORES_CLOCK frequency ratios. The MODCK inputs define the SPLL input clock division factor, feedback clock division factor and output clock division factor. In addition, the MODCK inputs define the BUSES_CLOCK division factor. MODCK[1–2] are sampled at the deassertion of the power-on reset signal ($\overline{\text{PORESET}}$). The other three mode bits MODCK[3–5] are initialized during the reset configuration sequence. The clock configuration changes only after $\overline{\text{PORESET}}$ is asserted. You can select a configuration to provide the required frequencies for an existing clock or define the clock setting to achieve the performance required.

The following five factors can be configured (see **Section 7.4**, *Clocks Programming Model*, on page 7-10):

- SPLL input clock division factor (PLLRDF)
- SPLL feedback clock division factor (PLLFDF)
- SPLL output clock division factor (PLLODF)
- SPLL loop filter tuning factor (PLLTP)
- BUS post-division factor (BUSDF)

Table 7-1 lists the possible configuration mode options.The following formulas explicitly calculate the BUSES_CLOCK and CORES_CLOCK frequencies:

$$F_{REF} = \frac{F_{CLKIN}}{PLLRDF}$$

$$F_{VCO} = 2 \times \frac{F_{CLKIN}}{PLLRDF} \times PLLFDF \times PLLODF \times BUSDF$$

$$F_{CORE} = \frac{1}{2} \times \frac{F_{VCO}}{PLLODF} = \frac{F_{CLKIN}}{PLLRDF} \times PLLFDF \times BUSDF$$

$$F_{BUS} = \frac{F_{CORE}}{BUSDF} = \frac{F_{CLKIN}}{PLLRDF} \times PLLFDF$$

$$F_{CLKOUT} = F_{BUS} = \frac{F_{CLKIN}}{PLLRDF} \times PLLFDF$$

Table 7-1. Clock Configuration Modes

Mode	MODCK[3–5]–MODCK[1–2]	PLLRDF	PLLFDF	PLLODF	PLLTP	BUSDF	BUS/CLKIN Ratio	Core/Bus Ratio
0	000-00	2	2	2	5	3	1x	3x
1	000-01	1	2	2	5	3	2x	3x
2	000-10	1	2	2	7	4	2x	4x
3	000-11	1	4	1	7	4	4x	4x
4	001-00	1	3	2	8	3	3x	3x
5	001-01	1	4	2	11	3	4x	3x
6	001-10	1	3	2	11	4	3x	4x
7	001-11	2	2	1	3	4	1x	4x
8	010-00	1	2	1	3	4	2x	4x
9	010-01	1	4	1	7	4	4x	4x
10	010-10	1	2	2	5	3	2x	3x
11	010-11	2	3	2	11	4	1.5x	4x
12	011-00	Reserved						
13	011-01	1	2	2	7	4	2x	4x
14	011-10	1	2	2	9	5	2x	5x
15	011-11	2	2	1	4	5	1x	5x
16	100-00	1	2	1	4	5	2x	5x
17	100-01	1	3	1	7	5	3x	5x
18	100-10	1	2	2	11	6	2x	6x
19	100-11	2	2	1	5	6	1x	6x
20	101-00	1	2	1	5	6	2x	6x
21	101-01	1	1	1	3	8	1x	8x
22	101-10	1	2	1	7	8	2x	8x
23	101-11	1	1	1	4	10	1x	10x
24-27	110-xx	Reserved						
28	111-00	2	2	2	5	3	1x	3x
29	111-01	1	1	2	3	4	1x	4x
30	111-10	1	1	2	4	5	1x	5x
31	111-11	1	1	2	5	6	1x	6x

Table 7-2 shows the frequency range of clock frequencies achieved by selecting the clock configurations listed in **Table 7-1**. The values in **Table 7-2** are for illustration only. It is important to ensure that the selected configuration setting does not exceed the following maximum and minimum frequencies permitted on each section of the device:

- F_{REF} frequency range: 20–133 MHz.
- F_{VCO} frequency range: 800–2000 MHz.

Table 7-2. Clock Configuration Frequency Examples

Configuration Mode	Bus Clock / CLKIN	Core/Bus Clock	CLKIN	Core	System Bus
0	1x	3x	66.67 MHz	200 MHz	66.67 MHz
			133.33 MHz	400 MHz	133.33 MHz
1	2x	3x	33.33 MHz	200 MHz	66.67 MHz
			66.67 MHz	400 MHz	133.33 MHz
2	2x	4x	25 MHz	200 MHz	50 MHz
			50 MHz	400 MHz	100 MHz
3	4x	4x	25 MHz	400 MHz	100 MHz
4	3x	3x	22.22 MHz	200 MHz	66.67 MHz
			44.44 MHz	400 MHz	133.33 MHz
5	4x	3x	20 MHz	240 MHz	80 MHz
			33.33 MHz	400 MHz	133.33 MHz
6	3x	4x	20 MHz	240 MHz	60 MHz
			33.33 MHz	400 MHz	100 MHz
7	1x	4x	100 MHz	400 MHz	100 MHz
8	2x	4x	50 MHz	400 MHz	100 MHz
9	4x	4x	25 MHz	400 MHz	100 MHz
10	2x	3x	33.33 MHz	200 MHz	66.67 MHz
			66.67 MHz	400 MHz	133.33 MHz
11	1.5x	4x	40 MHz	240 MHz	60 MHz
			66.67 MHz	400 MHz	100 MHz
12	Reserved				
13	2x	4x	25 MHz	200 MHz	50 MHz
			50 MHz	400 MHz	100 MHz
14	2x	5x	20 MHz	200 MHz	40 MHz
			40 MHz	400 MHz	80 MHz
15	1x	5x	80 MHz	400 MHz	80 MHz
			100 MHz	500 MHz	100 MHz
16	2x	5x	40 MHz	400 MHz	80 MHz
17	3x	5x	26.67 MHz	400 MHz	80 MHz
18	2x	6x	20 MHz	240 MHz	40 MHz
			33.33 MHz	400 MHz	66.67 MHz
19	1x	6x	66.67 MHz	400 MHz	66.67 MHz
			83.33 MHz	500 MHz	83.33 MHz
20	2x	6x	33.33 MHz	400 MHz	66.67 MHz
21	1x	8x	50 MHz	400 MHz	50 MHz
22	2x	8x	25 MHz	400 MHz	50 MHz
23	1x	10x	40 MHz	400 MHz	40 MHz
24-27	Reserved				
28	1x	3x	66.67 MHz	200 MHz	66.67 MHz
			133.33 MHz	400 MHz	133.33 MHz
29	1x	4x	50 MHz	200 MHz	50 MHz
			100 MHz	400 MHz	100 MHz
30	1x	5x	40 MHz	200 MHz	40 MHz
			80 MHz	400 MHz	80 MHz
31	1x	6x	33.33 MHz	200 MHz	33.33 MHz
			66.67 MHz	400 MHz	66.67 MHz

7.4 Clocks Programming Model

SCMSR System Clocks Mode Status Register **0x10C88**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								PLLTP				—			
Type	R															
Reset	0															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	—	PLLRDF		PLLFDF					PLLODF	DLLDIS	—	—	BUSDF			
Type	R															
Reset	0															

SCMSR is updated during power-on reset (POR) and provides the mode control signals to the PLL, DLL, and clock logic. This register reflects the currently defined configuration settings. For details on the available setting options, see **Table 7-1**.

Table 7-3. SCMR Bit Descriptions

Name	Defaults		Description	Settings
	POR	Hard Reset		
— 0–7	—	0	Reserved	
PLLTP 8-11	Configuration Signal	Unaffected	**SPLL Loop Bandwidth Tuning Field**	0011 Tuning Factor = 3 0010 Tuning Factor = 4 0101 Tuning Factor = 5 0111 Tuning Factor = 7 1000 Tuning Factor = 8 1001 Tuning Factor = 9 1011 Tuning Factor = 11 All other combinations are not used.
— 12–16	—	0	Reserved	
PLLRDF 17–18	Configuration Signal	Unaffected	**SPLL Input Clock Division Factor**	00 SPLL RDF = 1 01 SPLL RDF = 2 All other combinations are not used.
PLLFDF 19–23	Configuration Signal	Unaffected	**SPLL Feedback Clock Division Factor**	00000 SPLL FDF = 1 00001 SPLL FDF = 2 00010 SPLL FDF = 3 00011 SPLL FDF = 4 All other combinations are not used.
PLLODF 24	Configuration Signal	Unaffected	**SPLL Output Clock Division Factor**	0 SPLL PODF = 1 1 SPLL PODF = 2
DLLDIS 25	Configuration Signal	0	**DLL Disable** **Note:** DLL operation is not supported. Always write a 1 to this bit.	0 DLL enabled. 1 DLL disabled.

Table 7-3. SCMR Bit Descriptions (Continued)

Name	Defaults		Description	Settings
	POR	Hard Reset		
— 26–27	—	0	Reserved	
BUSDF 28–31	Configuration Signal	Unaffected	**60x Bus Division Factor**	0010 Bus DF = 3 0011 Bus DF = 4 0100 Bus DF = 5 0101 Bus DF = 6 0111 Bus DF = 8 1001 Bus DF = 10 All other combinations are not used.

Memory Map 8

The memory map of the MSC8113 system is composed of the following address spaces:

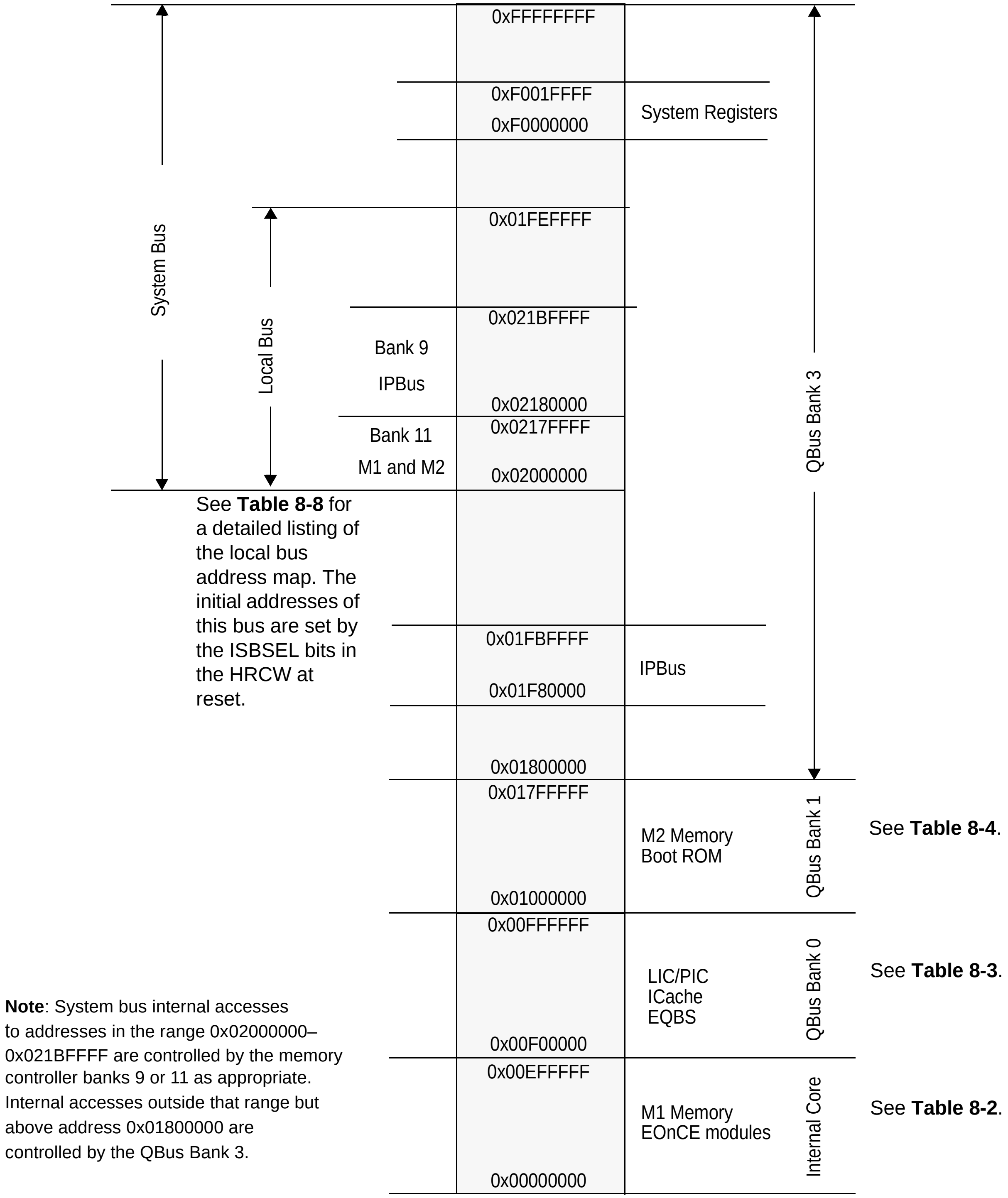
- *SC140 core internal address space*. Each SC140 core can access its M1 memory and EOnCE registers. (see **Table 8-2**). A boot master accesses the SC140 core 0 internal address space through the I^{2}C, TDM, or UART interface.
- *QBus address space*. Each SC140 core accesses its PIC, its LIC, its EQBS, and the MQBus or SQBus address space using its QBus banks, as described in **Chapter 9**, *Extended Core* (see **Table 8-3**). A boot master accesses the SC140 core 0 QBus space through the I^{2}C, TDM, or UART interface.
- *MQBus address space*. All SC140 cores share this space, which enables access to the M2 memory and to the boot ROM, as described in **Chapter 10**, *MQBus and M2 Memory* (see **Table 8-4**).
- *SQBus address space*. Each SC140 core can access this address space, which maps the IPBus address space and 60x-compatible address space, as described in **Chapter 11**, *SQBus* (see **Table 8-5**)
- *IPBus address space*. Each SC140 core or an external host on the system bus or DSI can access this space as well as boot from an I^{2}C, TDM, or UART device. It maps the TDMs, timers, UART, DSI, hardware semaphores, Ethernet, and GIC control registers (see **Table 8-6**).
- *System bus address space*. All three SC140 cores and an external host can access this address space. It contains devices that are located externally on the system bus as well as the system registers.
- *System Registers address space*. The system registers are mapped in a contiguous 128 KB block of memory. The block base address is programmed as described in **Section 4.2** (see **Table 8-9**).
- *Local bus address space*. This space contains devices in internal memories and peripherals connected to the local bus. The shared M2 memory and the three M1 memories are accessed through memory controller bank 11. The IPBus address space is mapped to bank 9. Refer to **Section 12.8**, *Memory Controller Programming Model* (see **Table 8-7** and **Table 8-8**).
- *Pseudo Command address space*. This space is accessible to the following boot masters: I^{2}C, TDM, and UART.

Table 8-1 summarizes which address space is viewed by each master type:

Table 8-1. Address Spaces

Master Device	SC140 core Internal	QBUS	MQBUS	SQBUS	IPBus	System Bus	System Registers	Local Bus	Pseudo Command
SC140 Core	+	+	+	+	+	+	+	+	
Host on System Bus					+	+	+	+	
Host on DSI					+	+	+	+	
TDM								+	
DMA						+		+	
Ethernet						+		+	
TDM boot	+	+	+	+	+	+	+	+	+
UART boot	+	+	+	+	+	+	+	+	+
I^{2}C Boot	+	+	+	+	+	+	+	+	+
Note: The UART boot, TDM boot, UART boot, and I^{2}C boot view of the SC140 core internal space and the QBus is the same as viewed by core 0.									

Figure 8-1 shows the SC140 view of the memory map immediately after boot for the case in which the IMMR[ISBSEL] field equals 0. This figure is an example only; the ISBSEL field value is programmable.



See **Table 8-8** for a detailed listing of the local bus address map. The initial addresses of this bus are set by the ISBSEL bits in the HRCW at reset.

See **Table 8-4**.

See **Table 8-3**.

See **Table 8-2**.

Note: System bus internal accesses to addresses in the range 0x02000000–0x021BFFFF are controlled by the memory controller banks 9 or 11 as appropriate. Internal accesses outside that range but above address 0x01800000 are controlled by the QBus Bank 3.

Figure 8-1. SC140 Core View Memory Map Example

Figure 8-2 shows the memory map as viewed by a host on the system bus when the IMMR[ISBSEL] field equals 0. This figure is an example only; the ISBSEL field value is programmable.

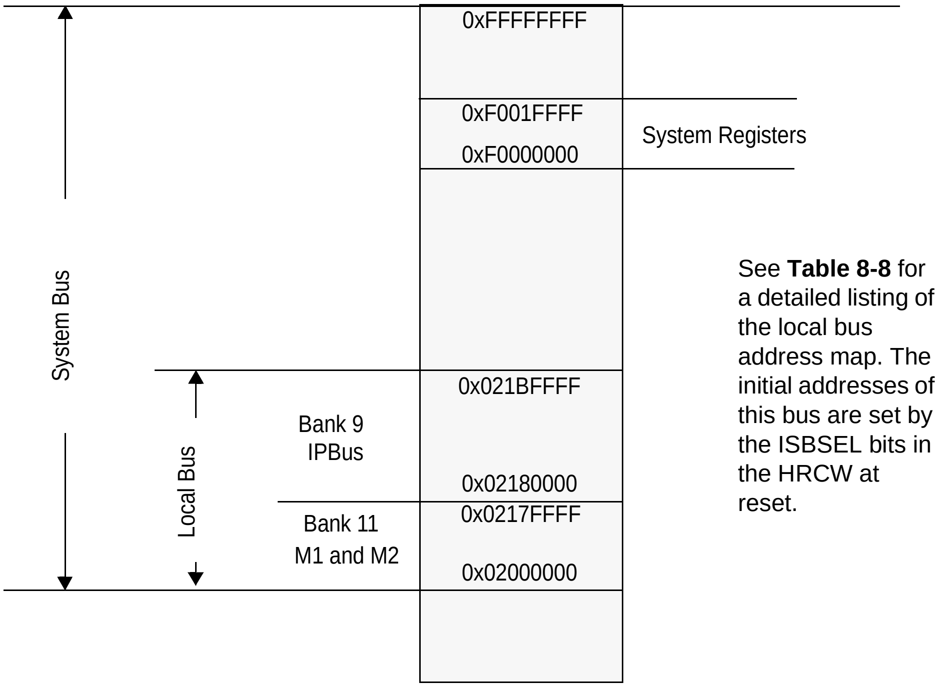

0xFFFFFFFF
0xF001FFFF
0xF0000000
System Registers
System Bus
0x021BFFFF
Bank 9
IPBus
0x02180000
Local Bus
0x0217FFFF
Bank 11
M1 and M2
0x02000000
See **Table 8-8** for a detailed listing of the local bus address map. The initial addresses of this bus are set by the ISBSEL bits in the HRCW at reset.

Figure 8-2. Host on the System Bus Memory Map View Example

Figure 8-3 shows the view of the memory map from a host accessing through the DSI.

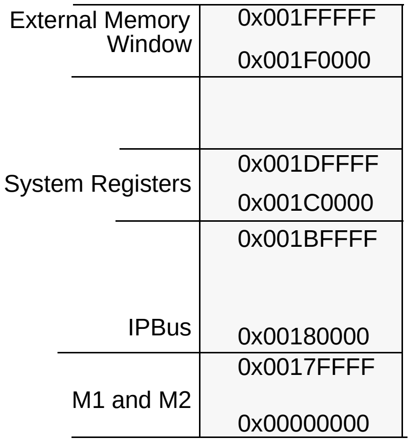


Figure 8-3. Host Accessing through the DSI Memory Map

Figure 8-4 shows the TDM view immediately after boot when the IMMR[ISBSEL] field equals 0. This figure is an example only; the ISBSEL field value is programmable.

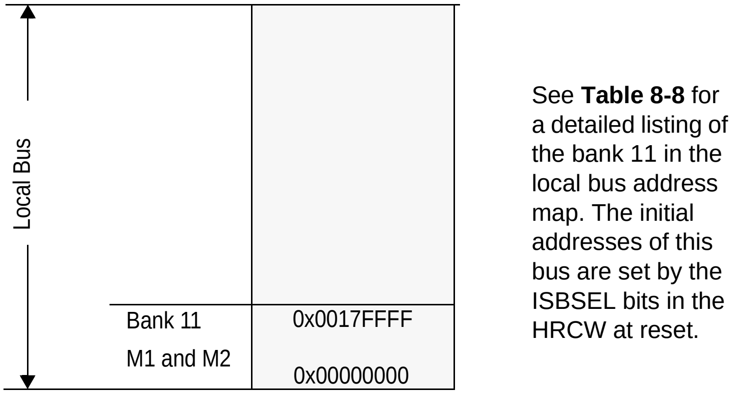

Local Bus
Bank 11
M1 and M2
0x0017FFFF
0x00000000
See **Table 8-8** for a detailed listing of the bank 11 in the local bus address map. The initial addresses of this bus are set by the ISBSEL bits in the HRCW at reset.

Figure 8-4. TDM View Memory Map

Figure 8-5 shows the memory map as viewed by the DMA and the Ethernet controllers.

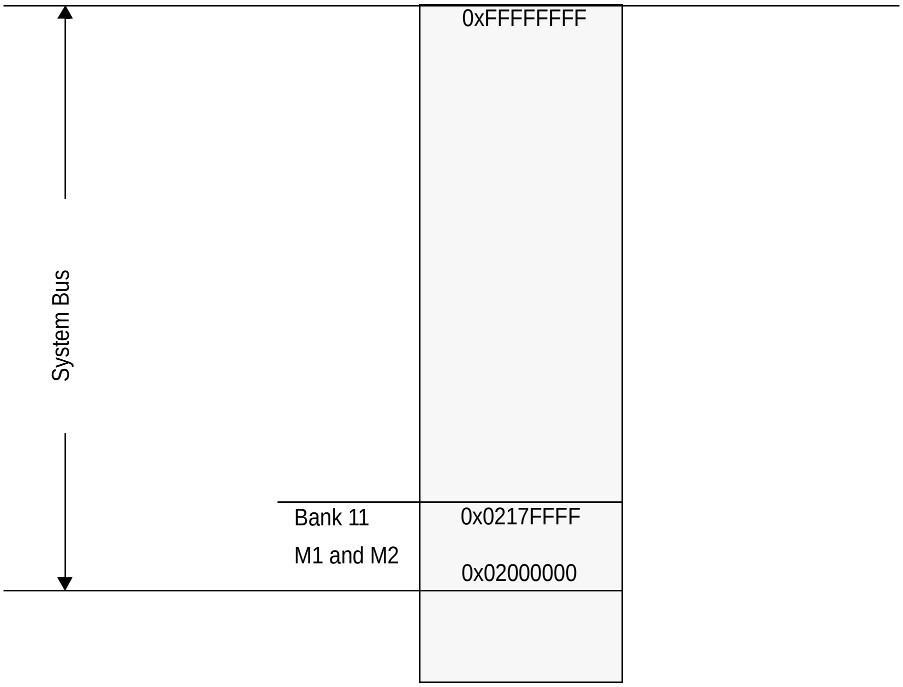


Figure 8-5. DMA and Ethernet Controller Memory Map

Figure 8-6 shows the memory map as viewed by the a boot master when booting through the TDM or UART interface. The boot view spans 16 MB, but is reserved except for the first 2 MB and the addresses that can be accessed by banks 9 and 11.

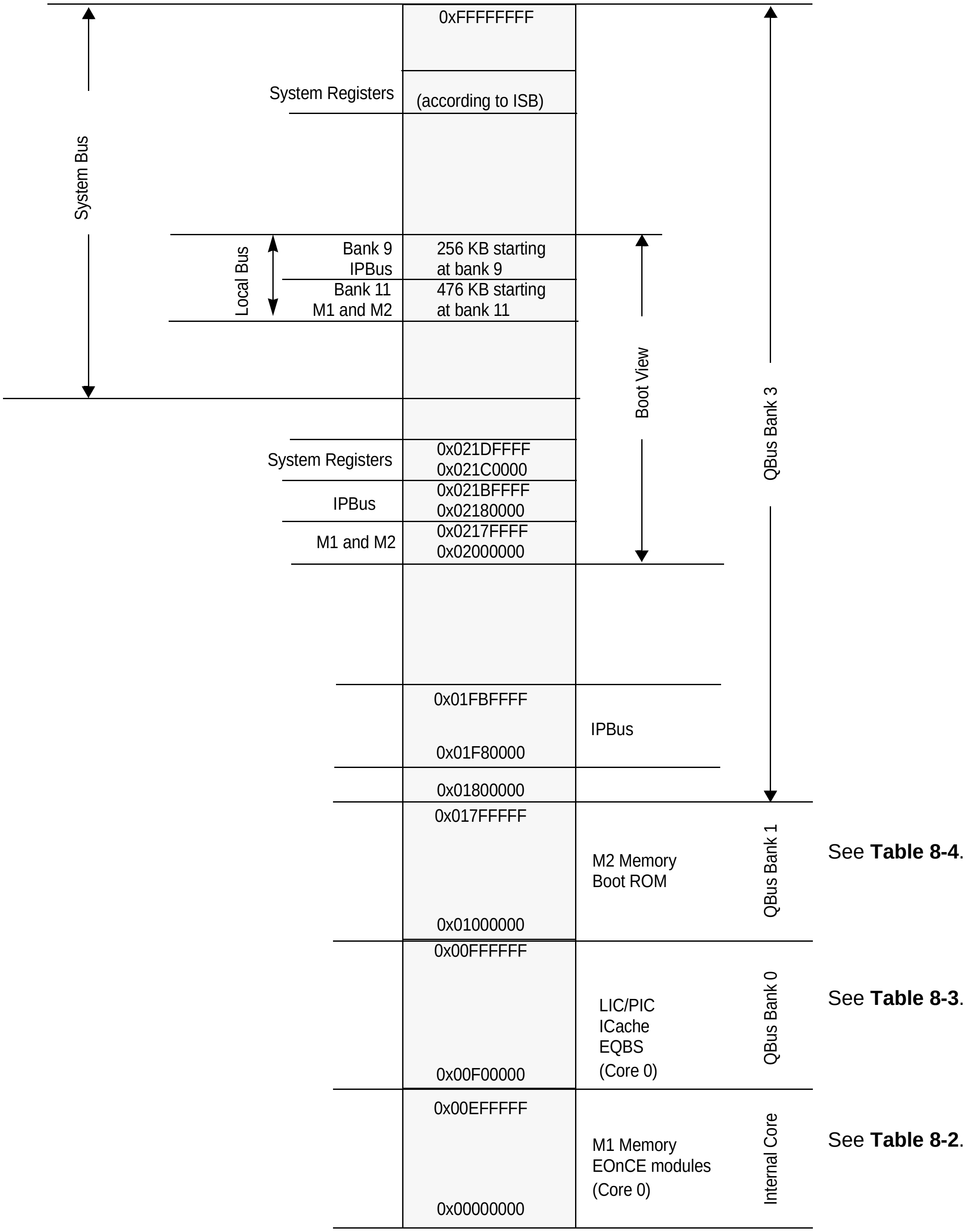


Figure 8-6. UART and TDM Boot Master Memory Map View

Figure 8-6 shows the memory map as viewed by the a boot master when booting through the I^{2}C interface. The boot view spans 16 MB, but is reserved except for the first 2 MB and the addresses that can be accessed by banks 9 and 11.

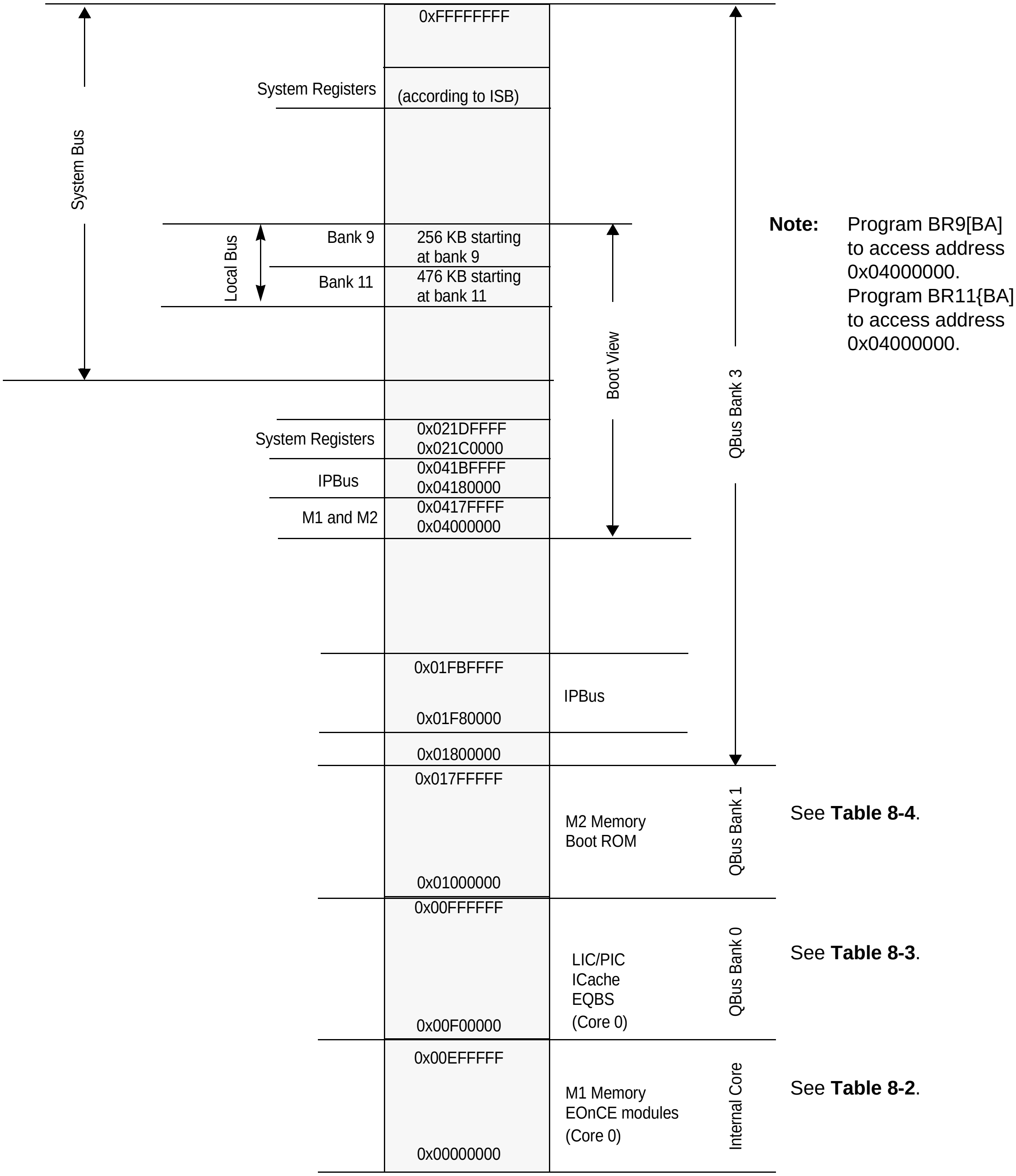


Figure 8-7. I^{2}C Boot Master Memory Map View

8.1 SC140 Core Internal Address Space

Each SC140 core accesses its EOnCE registers and M1 memory, as shown in **Table 8-2**.

Table 8-2. SC140 Core Internal Memory Map (0x00000000–0x00EFFFFF)

Address	Abbreviation	Name	Size in Bytes
00000000–00037FFF	M1MEM	M1 Memory for each core **Note:** Use the address range 00037FC0–00037FFF for data only. Storing instruction code in this range may cause the pipeline to fetch data beyond the physical range of the M1 memory and cause a system lockup.	224 K
00038000–00EFFDFF	Reserved		
00EFFE00	ESR	EOnCE Status Register	4
00EFFE04	EMCR	EOnCE Monitor and Control Register	4
00EFFE08	ERCV	EOnCE Receive Register (LSBs)	4
00EFFE0C		EOnCE Receive Register (MSBs)	4
00EFFE10	ETRSMT	EOnCE Transmit Register (LSBs)	4
00EFFE14		EOnCE Transmit Register (MSBs)	4
00EFFE18	EE_CTRL	EE Signals Control Register	2
00EFFE1C	PC_EXCP	Exception PC Register	4
00EFFE20	PC_NEXT	PC of next execution set	4
00EFFE24	PC_LAST	PC of last execution set	4
00EFFE28	PC_DETECT	PC Breakpoint Detection Register	4
00EFFE2C–00EFFE39	Reserved		
00EFFE40	EDCA0_CTRL	EDCA0 Control Register	2
00EFFE44	EDCA1_CTRL	EDCA1 Control Register	2
00EFFE48	EDCA2_CTRL	EDCA2 Control Register	2
00EFFE4C	EDCA3_CTRL	EDCA3 Control Register	2
00EFFE50	EDCA4_CTRL	EDCA4 Control Register	2
00EFFE54	EDCA5_CTRL	EDCA5 Control Register	2
00EFFE58–00EFFE5F	Reserved		
00EFFE60	EDCA0_REFA	EDCA0 reference value A	4
00EFFE64	EDCA1_REFA	EDCA1 reference value A	4
00EFFE68	EDCA2_REFA	EDCA2 reference value A	4
00EFFE6C	EDCA3_REFA	EDCA3 reference value A	4
00EFFE70	EDCA4_REFA	EDCA4 reference value A	4
00EFFE74	EDCA5_REFA	EDCA5 reference value A	4
00EFFE78–00EFFE7F	Reserved		
00EFFE80	EDCA0_REFB	EDCA0 reference value B	4
00EFFE84	EDCA1_REFB	EDCA1 reference value B	4
00EFFE88	EDCA2_REFB	EDCA2 reference value B	4

Table 8-2. SC140 Core Internal Memory Map (0x00000000–0x00EFFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
00EFFE8C	EDCA3_REFB	EDCA3 reference value B	4
00EFFE90	EDCA4_REFB	EDCA4 reference value B	4
00EFFE94	EDCA5_REFB	EDCA5 reference value B	4
00EFFE98–00EFFEBF	Reserved		
00EFFEC0	EDCA0_MASK	EDCA0 Mask Register	4
00EFFEC4	EDCA1_MASK	EDCA1 Mask Register	4
00EFFEC8	EDCA2_MASK	EDCA2 Mask Register	4
00EFFECC	EDCA3_MASK	EDCA3 Mask Register	4
00EFFED0	EDCA4_MASK	EDCA4 Mask Register	4
00EFFED4	EDCA5_MASK	EDCA5 Mask Register	4
00EFFED8–00EFFEDF	Reserved		
00EFFEE0	EDCD_CTRL	EDCD Control Register	2
00EFFEE4	EDCD_REF	EDCD Reference Register	4
00EFFEE8	EDCD_MASK	EDCD Mask Register	4
00EFFEEC–00EFFEFF	Reserved		
00EFFF00	ECNT_CTRL	EOnCE Counter Control Register	2
00EFFF04	ECNT_VAL	EOnCE Counter Value	4
00EFFF08	ECNT_EXT	EOnCE Extension Counter Value	4
00EFFF0C–00EFFF1F	Reserved		
00EFFF20	ESEL_CTRL	EOnCE Selector Control Register	1
00EFFF24	ESEL_DM	EOnCE Selector DM Mask	2
00EFFF28	ESEL_DI	EOnCE Selector DI Mask	2
00EFFF2C–00EFFF2F	Reserved		
00EFFF30	ESEL_ETB	EOnCE Selector Enable TB Mask	2
00EFFF34	ESEL_DTB	EOnCE Selector Disable TB Mask	2
00EFFF38–00EFFF3F	Reserved		
00EFFF40	TB_CTRL	Trace Buffer Control Register	1
00EFFF44	TB_RD	Trace Buffer Read Pointer	2
00EFFF48	TB_WR	Trace Buffer Write Pointer	2
00EFFF4C	TB_BUFF	Trace Buffer	4
00EFFF50–00EFFFF7	Reserved		
00EFFFF8	CORE_CMD	Core Command Register	6
00EFFFFC	NOREG	No Register Selected	—

8.2 QBus Address Space

The QBus address space encompasses the registers on QBus Bank 0–3 and is accessible only to its local SC140 core. QBus bank 0 includes the PIC, the LIC, and the EQBS registers. QBus bank 1 includes the MQBus interface through which the SC140 core accesses the M2 memory and the boot ROM. QBus bank 2 is reserved and QBus bank 3 enables access to the SQBus through which the SC140 core accesses the IP address space and the 60x-compatible address space. The reset value of the DSP peripherals base address register, QBUSBR0, is 0x00F0. A register address includes the base address and the offset for that register. For example, if the base address of QBUSBR0 is 0x00F0, an access to the ELIRA (whose offset is 0x9C00) is mapped to 0x00F09C00 (a concatenation of 0x00F0 and 0x9C00). **Table 8-3** lists the registers residing in QBus Bank0 using the default base address after reset.

Table 8-3. QBus Bank 0 Memory Map (0x00F00000–0x00FFFFFF)

Address	Abbreviation	Name	Size in Bytes
00F00000–00F07FFF		Write Buffer access locations	32 K
00F08000–00F09BFF	Reserved		
00F09C00	ELIRA	Edge/Level-Triggered Interrupt Register A	2
00F09C08	ELIRB	Edge/Level-Triggered Interrupt Register B	2
00F09C10	ELIRC	Edge/Level-Triggered Interrupt Register C	2
00F09C18	ELIRD	Edge/Level-Triggered Interrupt Register D	2
00F09C20	ELIRE	Edge/Level-Triggered Interrupt Register E	2
00F09C28	ELIRF	Edge/Level-Triggered Interrupt Register F	2
00F09C30	IPRA	Interrupt Pending Register A	2
00F09C38	IPRB	Interrupt Pending Register B	2
00F09C40–00F0ABFF	Reserved		
00F0AC00	LICAICR0	LIC Group A Interrupt Configuration Register 0	2
00F0AC08	LICAICR1	LIC Group A Interrupt Configuration Register 1	2
00F0AC10	LICAICR2	LIC Group A Interrupt Configuration Register 2	2
00F0AC18	LICAICR3	LIC Group A Interrupt Configuration Register 3	2
00F0AC20	LICAIER	LIC Group A Interrupt Enable Register	2
00F0AC28	LICAISR	LIC Group A Interrupt Status Register	2
00F0AC30	LICAIESR	LIC Group A Interrupt Error Status Register	2
00F0AC40	LICBICR0	LIC Group B Interrupt Configuration Register 0	2
00F0AC48	LICBICR1	LIC Group B Interrupt Configuration Register 1	2
00F0AC50	LICBICR2	LIC Group B Interrupt Configuration Register 2	2
00F0AC58	LICBICR3	LIC Group B Interrupt Configuration Register 3	2
00F0AC60	LICBIER	LIC Group B Interrupt Enable Register	2
00F0AC68	LICBISR	LIC Group B Interrupt Status Register	2
00F0AC70	LICBIESR	LIC Group B Interrupt Error Status Register	2
00F0AC72–00F0FBFF	Reserved		
00F0FC00	ICCR	ICache Control Register	2
00F0FC02	ICCMR	ICache Command Register	2
00F0FC04–00F0FC0F	Reserved		
00F0FC10	LRUSR	LRU Status Register	2

Table 8-3. QBus Bank 0 Memory Map (0x00F00000–0x00FFFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
00F0FC12	TASR	Tag Array Status Register	2
00F0FC14	VBASR	Valid Bit Array Status Register	2
00F0FC16–00F0FEFF	Reserved		
00F0FF00	QBUSMR0	QBus Mask for Bank 0	2
00F0FF02	QBUSBR0	QBus Base for Bank 0	2
00F0FF04	QBUSMR1	QBus Mask for Bank 1	2
00F0FF06	QBUSBR1	QBus Base for Bank 1	2
00F0FF08	QBUSMR2	QBus Mask for Bank 2	2
00F0FF0A	QBUSBR2	QBus Base for Bank 2	2
00F0FF0C–00F0FF1F	Reserved		
00F0FF20	EQBSBR	EQBS Bank Register	2
00F0FF22–00F0FF2F	Reserved		
00F0FF30	ICACR	Instruction Cacheable Area Control Register	2
00F0FF32	ICABR	Instruction Cacheable Area Base Register	2
00F0FF34–00F0FF3F	Reserved		
00F0FF60	IFUR	Instruction FU Configuration Register	2
00F0FF62–00F0FF7F	Reserved		
00F0FF80	WBFR	WB Flush Register	2
00F0FF82	WBCR	WB Control Register	2
00F0FF84–00F0FF9F	Reserved		
00F0FFA0	DBR0	Data Bank 0	4
00F0FFA4	DBR1	Data Bank 1	4
00F0FFA8	DBR2	Data Bank 2	4
00F0FFAC	DBR3	Data Bank 3	4
00F0FFB0–00F0FFEF	Reserved		
00F0FFF0	CIDR	Core ID Register	2
00F0FFF2	VR	Version Register	2
00F0FFF4	FLBACRA	FlyBy Address Control Register A	4
00F0FFF8	FLBACRB	FlyBy Address Control Register B	4
00F0FFFC–00FFFFFF	Reserved		

8.3 MQBus Address Space

Each SC140 core accesses the shared M2 memory and the boot ROM through the MQBus, which is mapped on bank 1 of the QBus. The Base Address Register (QBUSBR1) has a reset value of

0x0100. The base address can be reconfigured after reset. **Table 8-4** lists the M2 memory and boot ROM locations after reset.

Table 8-4. QBus Bank 1 (MQBus) Memory Map (0x01000000–0x017FFFFF)

Address	Acronym	Name	Size in KB
01000000–01076FFF	M2MEM	M2 Memory	476
01077000–01077FFF	BOOTROM	MSC8113 Boot ROM	4
01078000–017FFFFF	Reserved		

8.4 SQBus Address Space

Each SC140 core accesses the IPBus address space or the 60x-compatible address space, which are mapped on the SQBus on bank 3 of the QBus. Accesses not directed to banks 0–1 are directed to bank 3. Bank 2 is reserved and not used.

Table 8-5. QBus Bank 3 Memory Map (0x01800000–0xFFFFFFFF)

Address	Descriptor	Name	Size
01800000–01F7FFFF	Reserved		
01F80000–01FBFFFF	IPBus	IPBus Address Space	256 KB
01FC0000–01FEFFFF	Reserved		
01FF0000–FFFFFFFF	Local/System Bus	60x-compatible Bus Space (Local and System Buses)	3.968 GB

8.5 IPBus Address Space

Each of the three SC140 cores as well as an external host on either the system bus or the DSI bus can access the IPBus address space, which maps to the control registers of the TDMs, timers, UART, DSI, Ethernet Controller, RMII, HS, GPIO, and GIC. **Table 8-6** lists all the IPBus registers using the QBus Bank 3 default addressing.

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01F80000–01F807FF		TDM0 Receive Local Memory	2 K
01F80800–01F80FFF	Reserved		
01F81000–01F813FC	TDM0 RCPR[0–255]	TDM0 Receive Channel Parameters Register 0–255	4 each
01F81400–01F817FF	Reserved		
01F81800–01F81FFF		TDM0 Transmit Local Memory	2 K
01F82000–01F827FF	Reserved		
01F82800–01F82BFC	TDM0 TCPR[0–255]	TDM0 Transmit Channel Parameters Register 0–255	4 each
01F82C00–01F83F1F	Reserved		

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01F83F20	TDM0TSR	TDM0 Transmit Status Register	4
01F83F28	TDM0RSR	TDM0 Receive Status Register	4
01F83F30	TDM0ASR	TDM0 Adaptation Status Register	4
01F83F38	TDM0TER	TDM0 Transmit Event Register	4
01F83F40	TDM0RER	TDM0 Receive Event Register	4
01F83F48	TDM0TNB	TDM0 Transmit Number of Buffers	4
01F83F50	TDM0RNB	TDM0 Receive Number of Buffers	4
01F83F58	TDM0TDBDR	TDM0 Transmit Data Buffer Displacement Register	4
01F83F60	TDM0RDBDR	TDM0 Receive Data Buffer Displacement Register	4
01F83F68	TDM0ASDR	TDM0 Adaptation Sync Distance Register	4
01F83F70	TDM0TIER	TDM0 Transmit Interrupt Enable Register	4
01F83F78	TDM0RIER	TDM0 Receive Interrupt Enable Register	4
01F83F80	TDM0TDBST	TDM0 Transmit Data Buffer Second Threshold	4
01F83F88	TDM0RDBST	TDM0 Receive Data Buffer Second Threshold	4
01F83F90	TDM0TDBFT	TDM0 Transmit Data Buffer First Threshold	4
01F83F98	TDM0RDBFT	TDM0 Receive Data Buffer First Threshold	4
01F83FA0	TDM0TCR	TDM0 Transmit Control Register	4
01F83FA8	TDM0RCR	TDM0 Receive Control Register	4
01F83FB0	TDM0ACR	TDM0 Adaptation Control Register	4
01F83FB8	TDM0TGBA	TDM0 Transmit Global Base Address	4
01F83FC0	TDM0RGBA	TDM0 Receive Global Base Address	4
01F83FC8	TDM0TDBS	TDM0 Transmit Data Buffer Size	4
01F83FD0	TDM0RDBS	TDM0 Receive Data Buffer Size	4
01F83FD8	TDM0TFP	TDM0 Transmit Frame Parameters	4
01F83FE0	TDM0RFP	TDM0 Receive Frame Parameters	4
01F83FE8	TDM0TIR	TDM0 Transmit Interface Register	4
01F83FF0	TDM0RIR	TDM0 Receive Interface Register	4
01F83FF8	TDM0GIR	TDM0 General Interface Register	4
01F84000–01F847FF		TDM1 Receive Local Memory	2 K
01F84800–01F84FFF	Reserved		
01F85000–01F853FC	TDM1 RCPR[0–255]	TDM1 Receive Channel Parameters Register 0–255	4 each
01F85400–01F857FF	Reserved		
01F85800–01F85FFF		TDM1 Transmit Local Memory	2 K
01F86000–01F867FF	Reserved		
01F86800–01F86BFC	TDM1 TCPR[0–255]	TDM1 Transmit Channel Parameters Register 0–255	4 each
01F86C00–01F87F1F	Reserved		
01F87F20	TDM1TSR	TDM1 Transmit Status Register	4
01F87F28	TDM1RSR	TDM1 Receive Status Register	4
01F87F30	TDM1ASR	TDM1 Adaptation Status Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01F87F38	TDM1TER	TDM1 Transmit Event Register	4
01F87F40	TDM1RER	TDM1 Receive Event Register	4
01F87F48	TDM1TNB	TDM1 Transmit Number of Buffers	4
01F87F50	TDM1RNB	TDM1 Receive Number of Buffers	4
01F87F58	TDM1TDBDR	TDM1 Transmit Data Buffer Displacement Register	4
01F87F60	TDM1RDBDR	TDM1 Receive Data Buffer Displacement Register	4
01F87F68	TDM1ASDR	TDM1 Adaptation Sync Distance Register	4
01F87F70	TDM1TIER	TDM1 Transmit Interrupt Enable Register	4
01F87F78	TDM1RIER	TDM1 Receive Interrupt Enable Register	4
01F87F80	TDM1TDBST	TDM1 Transmit Data Buffer Second Threshold	4
01F87F88	TDM1RDBST	TDM1 Receive Data Buffer Second Threshold	4
01F87F90	TDM1TDBFT	TDM1 Transmit Data Buffer First Threshold	4
01F87F98	TDM1RDBFT	TDM1 Receive Data Buffer First Threshold	4
01F87FA0	TDM1TCR	TDM1 Transmit Control Register	4
01F87FA8	TDM1RCR	TDM1 Receive Control Register	4
01F87FB0	TDM1ACR	TDM1 Adaptation Control Register	4
01F87FB8	TDM1TGBA	TDM1 Transmit Global Base Address	4
01F87FC0	TDM1RGBA	TDM1 Receive Global Base Address	4
01F87FC8	TDM1TDBS	TDM1 Transmit Data Buffer Size	4
01F87FD0	TDM1RDBS	TDM1 Receive Data Buffer Size	4
01F87FD8	TDM1TFP	TDM1 Transmit Frame Parameters	4
01F87FE0	TDM1RFP	TDM1 Receive Frame Parameters	4
01F87FE8	TDM1TIR	TDM1 Transmit Interface Register	4
01F87FF0	TDM1RIR	TDM1 Receive Interface Register	4
01F87FF8	TDM1GIR	TDM1 General Interface Register	4
01F88000–01F887FF		TDM2 Receive Local Memory	2 K
01F88800–01F88FFF	Reserved		
01F89000–01F893FC	TDM2 RCPR[0–255]	TDM2 Receive Channel Parameters Register 0–255	4 each
01F89400–01F897FF	Reserved		
01F89800–01F89FFF		TDM2 Transmit Local Memory	2 K
01F8A000–01F8A7FF	Reserved		
01F8A800–01F8ABFC	TDM2 TCPR[0–255]	TDM2 Transmit Channel Parameters Register 0–255	4 each
01F8AC00–01F8BF1F	Reserved		
01F8BF20	TDM2TSR	TDM2 Transmit Status Register	4
01F8BF28	TDM2RSR	TDM2 Receive Status Register	4
01F8BF30	TDM2ASR	TDM2 Adaptation Status Register	4
01F8BF38	TDM2TER	TDM2 Transmit Event Register	4
01F8BF40	TDM2RER	TDM2 Receive Event Register	4
01F8BF48	TDM2TNB	TDM2 Transmit Number of Buffers	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01F8BF50	TDM2RNB	TDM2 Receive Number of Buffers	4
01F8BF58	TDM2TDBDR	TDM2 Transmit Data Buffer Displacement Register	4
01F8BF60	TDM2RDBDR	TDM2 Receive Data Buffer Displacement Register	4
01F8BF68	TDM2ASDR	TDM2 Adaptation Sync Distance Register	4
01F8BF70	TDM2TIER	TDM2 Transmit Interrupt Enable Register	4
01F8BF78	TDM2RIER	TDM2 Receive Interrupt Enable Register	4
01F8BF80	TDM2TDBST	TDM2 Transmit Data Buffer Second Threshold	4
01F8BF88	TDM2RDBST	TDM2 Receive Data Buffer Second Threshold	4
01F8BF90	TDM2TDBFT	TDM2 Transmit Data Buffer First Threshold	4
01F8BF98	TDM2RDBFT	TDM2 Receive Data Buffer First Threshold	4
01F8BFA0	TDM2TCR	TDM2 Transmit Control Register	4
01F8BFA8	TDM2RCR	TDM2 Receive Control Register	4
01F8BFB0	TDM2ACR	TDM2 Adaptation Control Register	4
01F8BFB8	TDM2TGBA	TDM2 Transmit Global Base Address	4
01F8BFC0	TDM2RGBA	TDM2 Receive Global Base Address	4
01F8BFC8	TDM2TDBS	TDM2 Transmit Data Buffer Size	4
01F8BFD0	TDM2RDBS	TDM2 Receive Data Buffer Size	4
01F8BFD8	TDM2TFP	TDM2 Transmit Frame Parameters	4
01F8BFE0	TDM2RFP	TDM2 Receive Frame Parameters	4
01F8BFE8	TDM2TIR	TDM2 Transmit Interface Register	4
01F8BFF0	TDM2RIR	TDM2 Receive Interface Register	4
01F8BFF8	TDM2GIR	TDM2 General Interface Register	4
01F8C000–01F8C7FF		TDM3 Receive Local Memory	2 K
01F8C800–01F8CFFF	Reserved		
01F8D000–01F8D3FC	TDM3 RCPR[0–255]	TDM3 Receive Channel Parameters Register 0–255	4 each
01F8D400–01F8D7FF	Reserved		
01F8D800–01F8DFFF		TDM3 Transmit Local Memory	2 K
01F8E000–01F8E7FF	Reserved		
01F8E800–01F8EBFC	TDM3 TCPR[0–255]	TDM3 Transmit Channel Parameters Register 0–255	4 each
01F8EC00–01F8FF1F	Reserved		
01F8FF20	TDM3TSR	TDM3 Transmit Status Register	4
01F8FF28	TDM3RSR	TDM3 Receive Status Register	4
01F8FF30	TDM3ASR	TDM3 Adaptation Status Register	4
01F8FF38	TDM3TER	TDM3 Transmit Event Register	4
01F8FF40	TDM3RER	TDM3 Receive Event Register	4
01F8FF48	TDM3TNB	TDM3 Transmit Number of Buffers	4
01F8FF50	TDM3RNB	TDM3 Receive Number of Buffers	4
01F8FF58	TDM3TDBDR	TDM3 Transmit Data Buffer Displacement Register	4
01F8FF60	TDM3RDBDR	TDM3 Receive Data Buffer Displacement Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01F8FF68	TDM3ASDR	TDM3 Adaptation Sync Distance Register	4
01F8FF70	TDM3TIER	TDM3 Transmit Interrupt Enable Register	4
01F8FF78	TDM3RIER	TDM3 Receive Interrupt Enable Register	4
01F8FF80	TDM3TDBST	TDM3 Transmit Data Buffer Second Threshold	4
01F8FF88	TDM3RDBST	TDM3 Receive Data Buffer Second Threshold	4
01F8FF90	TDM3TDBFT	TDM3 Transmit Data Buffer First Threshold	4
01F8FF98	TDM3RDBFT	TDM3 Receive Data Buffer First Threshold	4
01F8FFA0	TDM3TCR	TDM3 Transmit Control Register	4
01F8FFA8	TDM3RCR	TDM3 Receive Control Register	4
01F8FFB0	TDM3ACR	TDM3 Adaptation Control Register	4
01F8FFB8	TDM3TGBA	TDM3 Transmit Global Base Address	4
01F8FFC0	TDM3RGBA	TDM3 Receive Global Base Address	4
01F8FFC8	TDM3TDBS	TDM3 Transmit Data Buffer Size	4
01F8FFD0	TDM3RDBS	TDM3 Receive Data Buffer Size	4
01F8FFD8	TDM3TFP	TDM3 Transmit Frame Parameters	4
01F8FFE0	TDM3RFP	TDM3 Receive Frame Parameters	4
01F8FFE8	TDM3TIR	TDM3 Transmit Interface Register	4
01F8FFF0	TDM3RIR	TDM3 Receive Interface Register	4
01F8FFF8	TDM3GIR	TDM3 General Interface Register	4
01F90000–01FB800F	Reserved		
01FB8010	IEVENT	Interrupt Event Register	4
01FB8014	IMASK	Interrupt Mask Register	4
01FB8018–01FB801F	Reserved		
01FB8020	ECNTRL	Ethernet Control Register	4
01FB8024	MINFLR	Minimum Frame Length Register	4
01FB8028	PTV	Pause Time Value Register	4
01FB802C	DMACTRL	DMA Control Register	4
01FB8034–01FB8037	Reserved		
01FB8038	DMAMR	DMA Maintenance Register	4
01FB803C–01FB8047	Reserved		
01FB8048	FRXSTATR	FIFO Receive Status Register	4
01FB804C	FRXCTRLR	FIFO Receive Control Register	4
01FB8050	FRXALAR	FIFO Receive Alarm Register	4
01FB8054	FRXSHR	FIFO Receive Alarm Shutoff Register	4
01FB8058	FRXPAR	FIFO Receive Panic Register	4
01FB805C	FRXPSR	FIFO Receive Panic Shutoff Register	4
01FB8078	FTXSTATR	FIFO Transmit Status Register	4
01FB807C–-1FB808B	Reserved		
01FB808C	FTXTHR	FIFO Transmit Threshold Register	4
01FB8094	FTXSPR	FIFO Transmit Space Available Register	4
01FB8098	FTXSR	FIFO Transmit Starve Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB809C	FTXSSR	FIFO Transmit Starve Shutoff Register	4
01FB80A0–01FB80FF	Reserved		
01FB8100	TCTRL	Transmit Control Register	4
01FB8104	TSTAT	Transmit Status Register	4
01FB8108–01FB810B	Reserved		
01FB810C	TBDLEN	TxBD Data Length	4
01FB8110–01FB8123	Reserved		
01FB8124	CTBPTR	Current TxBD Pointer	4
01FB8128–01FB8183	Reserved		
01FB8184	TBPTR	TxBD Pointer	4
01FB8188–01FB8203	Reserved		
01FB8204	TBASE	Transmit Descriptor Base Address	4
01FB8208–01FB82AF	Reserved		
01FB82B0	OSTBD	Out-of-sequence TxBD Register	4
01FB82B4	OSTBDP	Out-of-sequence Tx Data Buffer Pointer Register	4
01FB82B8	OS32TBDP	Out-of-sequence 32 Bytes Tx Data Buffer Pointer Register	4
01FB82C0	OS32IPTR	Out-of-sequence 32 Bytes TxBD Insert Pointer Register	4
01FB82C4	OS32TBDR	Out-of-sequence 32 Bytes TxBD Reserved Register	4
01FB82C8	OS32IIL	Out-of-sequence 32 Bytes TxBD Insert Index/length Register	4
01FB82CC–01FB82FF	Reserved		
01FB8300	RCTRL	Receive Control Register	4
01FB8304	RSTAT	Receive Status Register	4
01FB8308–01FB830B	Reserved		
01FB830C	RBDLEN	RxBD Data Length	4
01FB8310–01FB8323	Reserved		
01FB8324	CRBPTRL	Current RxBD Pointer	4
01FB8328–01FB833F	Reserved		
01FB8340	MRBLR0R1	Maximum Receive Buffer Length R0R1 Register	4
01FB8344	MRBLR2R3	Maximum Receive Buffer Length R2R3 Register	4
01FB8348–01FB8383	Reserved		
01FB8384	RBPTR0	RxBD Pointer 0	4
01FB8388–01FB838B	Reserved		
01FB838C	RBPTR1	RxBD Pointer 1	4
01FB8390–01FB8393	Reserved		
01FB8394	RBPTR2	RxBD Pointer 2	4
01FB8398–01FB839B	Reserved		
01FB839C	RBPTR3	RxBD Pointer 3	4
01FB83A0–01FB8403	Reserved		
01FB8404	RBASE0	Receive Descriptor Base Address 0	4
01FB8408–01FB840B	Reserved		
01FB840C	RBASE1	Receive Descriptor Base Address 1	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB8410–01FB8413	Reserved		
01FB8414	RBASE2	Receive Descriptor Base Address 2	4
01FB8418–01FB841B	Reserved		
01FB841C	RBASE3	Receive Descriptor Base Address 3	4
01FB8420–01FB84FF	Reserved		
01FB8500	MACCFG1R	MAC Configuration 1 Register	4
01FB8504	MACCFG2R	MAC Configuration 2 Register	4
01FB8508	IPGIFGIR	Inter Packet Gap/Inter Frame Gap Register	4
01FB850C	HAFDUPR	Half-Duplex Register	4
01FB8510	MAXFRMR	Maximum Frame Register	4
01FB8514–01FB851F	Reserved		
01FB8520	MIIMCFGR	MII Management Configuration Register	4
01FB8524	MIIMCOMR	MII Management Command Register	4
01FB8528	MIIMADDR	MII Management Address Register	4
01FB852C	MIIMCONR	MII Management Control Register	4
01FB8530	MIIMSTATR	MII Management Status Register	4
01FB8534	MIIMINDR	MII Management Indicator Register	4
01FB8538–01FB853B	Reserved		
01FB853C	IFSTATR	Interface Status Register	4
01FB8540	MACSTADDR1R	MAC Station Address Part 1 Register	4
01FB8544	MACSTADDR2R	MAC Station Address Part 2 Register	4
01FB8548–01FB867F	Reserved		
01FB8680	TR64	Transmit And Receive 64-byte Frame Counter	4
01FB8684	TR127	Transmit and Receive 65- to 127-byte Frame Counter	4
01FB8688	TR255	Transmit and Receive 128- to 255-byte Frame Counter	4
01FB868C	TR511	Transmit and Receive 256- to 511-byte Frame Counter	4
01FB8690	TR1K	Transmit and Receive 512- to 1023-byte Frame Counter	4
01FB8694	TRMAX	Transmit and Receive 1024- to 1518-byte Frame Counter	4
01FB8698	TRMGV	Transmit and Receive 1519- to 1522-byte Good VLAN Frame Count	4
01FB869C	RBYT	Receive Byte Counter	4
01FB86A0	RPKT	Receive Packet Counter	4
01FB86A4	RFCS	Receive FCS Error Counter	4
01FB86A8	RMCA	Receive Multicast Packet Counter	4
01FB86AC	RBCA	Receive Broadcast Packet Counter	4
01FB86B0	RXCF	Receive Control Frame Packet Counter	4
01FB86B4	RXPF	Receive PAUSE Frame Packet Counter	4
01FB86B8	RXUO	Receive Unknown OP Code Counter	4
01FB86BC	RALN	Receive Alignment Error Counter	4
01FB86C0	RFLR	Receive Frame Length Error Counter	4
01FB86C4	RCDE	Receive Code Error Counter	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB86C8	RCSE	Receive Carrier Sense Error Counter	4
01FB86CC	RUND	Receive Undersize Packet Counter	4
01FB86D0	ROVR	Receive Oversize Packet Counter	4
01FB86D4	RFRG	Receive Fragments Counter	4
01FB86D8	RJBR	Receive Jabber Counter	4
01FB86DC	RDRP	Receive Drop	4
01FB86E0	TBYT	Transmit Byte Counter	4
01FB86E4	TPKT	Transmit Packet Counter	4
01FB86E8	TMCA	Transmit Multicast Packet Counter	4
01FB86EC	TBCA	Transmit Broadcast Packet Counter	4
01FB86F0	TXPF	Transmit PAUSE control frame counter	4
01FB86F4	TDFR	Transmit Deferral Packet Counter	4
01FB86F8	TEDF	Transmit Excessive Deferral Packet Counter	4
01FB86FC	TSCL	Transmit Single Collision Packet Counter	4
01FB8700	TMCL	Transmit Multiple Collision Packet Counter	4
01FB8704	TLCL	Transmit Late Collision Packet Counter	4
01FB8708	TXCL	Transmit Excessive Collision Packet Counter	4
01FB870C	TNCL	Transmit Total Collision Counter	4
01FB8714–01FB8717	Reserved		
01FB8718	TJBR	Transmit Jabber Frame Counter	4
01FB871c	TFCS	Transmit FCS Error Counter	4
01FB8720	TXCF	Transmit Control Frame Counter	4
01FB8724	TOVR	Transmit Oversize Frame Counter	4
01FB8728	TUND	Transmit Undersize Frame Counter	4
01FB872c	TFRG	Transmit Fragments Frame Counter	4
01FB8730	CAR1	Carry Register One	4
01FB8734	CAR2	Carry Register Two	4
01FB8738	CAM1	Carry Register One Mask	4
01FB873C	CAM2	Carry Register Two Mask	4
01FB8740–01FB87FF	Reserved		
01FB8800	IADDR0	Individual Address Register 0	4
01FB8804	IADDR1	Individual Address Register 1	4
01FB8808	IADDR2	Individual Address Register 2	4
01FB880C	IADDR3	Individual Address Register 3	4
01FB8810	IADDR4	Individual Address Register 4	4
01FB8814	IADDR5	Individual Address Register 5	4
01FB8818	IADDR6	Individual Address Register 6	4
01FB881C	IADDR7	Individual Address Register 7	4
01FB8820–01FB887F	Reserved		
01FB8880	GADDR0	Group Address Register 0	4
01FB8884	GADDR1	Group Address Register 1	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB8888	GADDR2	Group Address Register 2	4
01FB888C	GADDR3	Group Address Register 3	4
01FB8890	GADDR4	Group Address Register 4	4
01FB8894	GADDR5	Group Address Register 5	4
01FB8898	GADDR6	Group Address Register 6	4
01FB889C	GADDR7	Group Address Register 7	4
01FB88A0–01FB88FF	Reserved		
01FB8900	PMD0	Pattern Match Data 0	4
01FB8904–01FB8907	Reserved		
01FB8908	PMASK0	Pattern Mask 0 Register	4
01FB890C–01FB890F	Reserved		
01FB8910	PCNTRL0	Pattern Control 0 Register	4
01FB8914–01FB8917	Reserved		
01FB8918	PATTRB0	Pattern Attributes 0 Register	4
01FB8920	PMD1	Pattern Match Data 1	4
01FB8924–01FB8927	Reserved		
01FB8928	PMASK1	Pattern Mask 1 Register	4
01FB892C–01FB892F	Reserved		
01FB8930	PCNTRL1	Pattern Control 1 Register	4
01FB8934–01FB8937	Reserved		
01FB8938	PATTRB1	Pattern Attributes 1 Register	4
01FB8940	PMD2	Pattern Match Data 2	4
01FB8944–01FB8947	Reserved		
01FB8948	PMASK2	Pattern Mask 2 Register	4
01FB894C–01FB894F	Reserved		
01FB8950	PCNTRL2	Pattern Control 2 Register	4
01FB8954–01FB8957	Reserved		
01FB8958	PATTRB2	Pattern Attributes 2 Register	4
01FB8960	PMD3	Pattern Match Data 3	4
01FB8964–01FB8967	Reserved		
01FB8968	PMASK3	Pattern Mask 3 Register	4
01FB896C–01FB896F	Reserved		
01FB8970	PCNTRL3	Pattern Control 3 Register	4
01FB8974–01FB8977	Reserved		
01FB8978	PATTRB3	Pattern Attributes 3 Register	4
01FB8980	PMD4	Pattern Match Data 4	4
01FB8984–01FB8987	Reserved		
01FB8988	PMASK4	Pattern Mask 4 Register	4
01FB898C–01FB898F	Reserved		
01FB8990	PCNTRL4	Pattern Control 4 Register	4
01FB8994–01FB8997	Reserved		

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB8998	PATTRB4	Pattern Attributes 4 Register	4
01FB89A0	PMD5	Pattern Match Data 5	4
01FB89A4–01FB89A7	Reserved		
01FB89A8	PMASK5	Pattern Mask 5 Register	4
01FB89AC–01FB89AF	Reserved		
01FB89B0	PCNTRL5	Pattern Control 5 Register	4
01FB89B4–01FB89B7	Reserved		
01FB89B8	PATTRB5	Pattern Attributes 5 Register	4
01FB89C0	PMD6	Pattern Match Data 6	4
01FB89C4–01FB89C7	Reserved		
01FB89C8	PMASK6	Pattern Mask 6 Register	4
01FB89CC–01FB89CF	Reserved		
01FB89D0	PCNTRL6	Pattern Control 6 Register	4
01FB89D4–01FB89D7	Reserved		
01FB89D8	PATTRB6	Pattern Attributes 6 Register	4
01FB89E0	PMD7	Pattern Match Data 7	4
01FB89E4–01FB89E7	Reserved		
01FB89E8	PMASK7	Pattern Mask 7 Register	4
01FB89EC–01FB89EF	Reserved		
01FB89F0	PCNTRL7	Pattern Control 7 Register	4
01FB89F4–01FB89F7	Reserved		
01FB89F8	PATTRB7	Pattern Attributes 7 Register	4
01FB8A00	PMD8	Pattern Match Data 8	4
01FB8A04–01FB8A07	Reserved		
01FB8A08	PMASK8	Pattern Mask 8 Register	4
01FB8A0C–01FB8A0F	Reserved		
01FB8A10	PCNTRL8	Pattern Control 8 Register	4
01FB8A14–01FB8A17	Reserved		
01FB8A18	PATTRB8	Pattern Attributes 8 Register	4
01FB8A20	PMD9	Pattern Match Data 9	4
01FB8A24–01FBA27	Reserved		
01FB8A28	PMASK9	Pattern Mask 9 Register	4
01FB8A2C–01FBA2F	Reserved		
01FB8A30	PCNTRL9	Pattern Control 9 Register	4
01FB8A34–01FB8A37	Reserved		
01FB8A38	PATTRB9	Pattern Attributes 9 Register	4
01FB8A40	PMD10	Pattern Match Data 10	4
01FB8A44–01FB8A47	Reserved		
01FB8A48	PMASK10	Pattern Mask 10 Register	4
01FB8A4C–01FB8A4F	Reserved		
01FB8A50	PCNTRL10	Pattern Control 10 Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB8A54–01FB8A57	Reserved		
01FB8A58	PATTRB10	Pattern Attributes 10 Register	
01FB8A60	PMD11	Pattern Match Data 11	4
01FB8A64–01FB8A67	Reserved		
01FB8A68	PMASK11	Pattern Mask 11 Register	4
01FB8A6C–01FB8A6F	Reserved		
01FB8A70	PCNTRL11	Pattern Control 11 Register	4
01FB8A74–01FB8A77	Reserved		
01FB8A78	PATTRB11	Pattern Attributes 11 Register	4
01FB8A80	PMD12	Pattern Match Data 12	4
01FB8A84–01FB8A87	Reserved		
01FB8A88	PMASK12	Pattern Mask 12 Register	4
01FB8A8C–01FB8A8F	Reserved		
01FB8A90	PCNTRL12	Pattern Control 12 Register	4
01FB8A94–01FB8A97	Reserved		
01FB8A98	PATTRB12	Pattern Attributes 12 Register	4
01FB8AA0	PMD13	Pattern Match Data 13	4
01FB8AA4–01FB8AA7	Reserved		
01FB8AA8	PMASK13	Pattern Mask 13 Register	4
01FB8AAC–01FB8AAF	Reserved		
01FB8AB0	PCNTRL13	Pattern Control 13 Register	4
01FB8AB4–01FB8AB7	Reserved		
01FB8AB8	PATTRB13	Pattern Attributes 13 Register	4
01FB8AC0	PMD14	Pattern Match Data 14	4
01FB8AC4–01FB8AC7	Reserved		
01FB8AC8	PMASK14	Pattern Mask 14 Register	4
01FB8ACC–01FB8ACF	Reserved		
01FB8AD0	PCNTRL14	Pattern Control 14 Register	4
01FB8AD4–01FB8AD7	Reserved		
01FB8AD8	PATTRB14	Pattern Attributes 14 Register	4
01FB8AE0	PMD15	Pattern Match Data 15	4
01FB8AE4–01FB8AE7	Reserved		
01FB8AE8	PMASK15	Pattern Mask 15 Register	4
01FB8AEC–01FB8AEF	Reserved		
01FB8AF0	PCNTRL15	Pattern Control 15 Register	4
01FB8AF4–01FB8AF7	Reserved		
01FB8AF8	PATTRB15	Pattern Attributes 15 Register	4
01FB8B00–01FB8BF7	Reserved		
01FB8BF8	DATTR	Default Attribute Register	4
01FB8C00–01F8FFFF	Reserved		
01FB9000	MIIGSK_CFGR	MIIGSK Configuration Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FB9004	MIIGSK_GPR	MIIGSK General-Purpose Register	4
01FB9008	MIIGSK_ENR	MIIGSK Enable Register	4
01FB900C	MIIGSK_SMII_ SYNCDIR	MIIGSK SMII SYNC Direction Register	4
01FB9010	MIIGSK_TIFBR	MIIGSK Transmit Inter-Frame Bits Register	4
01FB9014	MIIGSK_RIFBR	MIIGSK Receive Inter-Frame Bits Register	4
01FB9018	MIIGSK_ERIFBR	MIIGSK Expected Receive Inter-Frame Bits Register	4
01FB901C	MIIGSK_IEVENT	MIIGSK SMII Interrupt Event Register	4
01FB9020	MIIGSK_IMASK	MIIGSK SMII Interrupt Mask Register	4
01FB9024– 01FBAFFF	Reserved		
01FBB000	SCR	Stop Control Register	4
01FBB008	SASR	Stop Acknowledge Status Register	4
01FBC000	VIGR	Virtual Interrupt Generation Register	4
01FBC008	VISR	Virtual Interrupt Status Register	4
01FBC010	VNMIGR	Virtual NMI Generation Register	4
01FBC018	GICR	GIC Interrupt Configuration Register	4
01FBC020	GEIER	GIC External Interrupt Enable Register	4
01FBC028	GCIER	GIC Core Interrupt Enable Register	4
01FBC030	GISR	GIC Interrupt Status Register	4
01FBC038–01FBC0FF	Reserved		
01FBC100	HSMPR0	Hardware Semaphore Register 0	4
01FBC108	HSMPR1	Hardware Semaphore Register 1	4
01FBC110	HSMPR2	Hardware Semaphore Register 2	4
01FBC118	HSMPR3	Hardware Semaphore Register 3	4
01FBC120	HSMPR4	Hardware Semaphore Register 4	4
01FBC128	HSMPR5	Hardware Semaphore Register 5	4
01FBC130	HSMPR6	Hardware Semaphore Register 6	4
01FBC138	HSMPR7	Hardware Semaphore Register 7	4
01FBC140–01FBC1FF	Reserved		
01FBC200	PODR	Pin Open-Drain Register	4
01FBC208	PDAT	Pin Data Register	4
01FBC210	PDIR	Pin Direction Register	4
01FBC218	PAR	Pin Assignment Register	4
01FBC220	PSOR	Pin Special Options Register	4
01FBC228–01FBCFFF	Reserved		
01FBD000	SCIBR	SCI Baud Rate Register	4
01FBD008	SCICR	SCI Control Register	4
01FBD010	SCISR	SCI Status Register	4
01FBD018	SCIDR	SCI Data Register	4
01FBD020–01FBD027	Reserved		
01FBD028	SCIDDR	SCI Data Direction Register	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FBD030–01FBDFFF	Reserved		
01FBE000	DCR	DSI Control Register	4
01FBE008	DSWBAR	DSI Sliding Window Base Address Register	4
01FBE010	DIBAR9	DSI Internal Base Address Register Bank 9	4
01FBE018	Reserved		
01FBE020	DIBAR11	DSI Internal Base Address Register Bank 11	4
01FBE028	DIAMR9	DSI Internal Address Mask Register Bank 9	4
01FBE030	Reserved		
01FBE038	DIAMR11	DSI Internal Address Mask Register Bank 11	4
01FBE040	DCIR	DSI Chip ID Register	4
01FBE048	DDR	DSI Disable Register	4
01FBE050–01FBE058	Reserved		
01FBE060	DEXTBAR	DSI External Sliding Window Base Address Register	4
01FBE068–01FBE7FF	Reserved		
01FBE800	DSR	DSI Status Register	4
01FBE808	DER	DSI Error Register	4
01FBE810–01FBEFFF	Reserved		
01FBF000	TCFRA0	Timer Configuration Register of Timer A0	4
01FBF008	TCFRA1	Timer Configuration Register of Timer A1	4
01FBF010	TCFRA2	Timer Configuration Register of Timer A2	4
01FBF018	TCFRA3	Timer Configuration Register of Timer A3	4
01FBF020	TCFRA4	Timer Configuration Register of Timer A4	4
01FBF028	TCFRA5	Timer Configuration Register of Timer A5	4
01FBF030	TCFRA6	Timer Configuration Register of Timer A6	4
01FBF038	TCFRA7	Timer Configuration Register of Timer A7	4
01FBF040	TCFRA8	Timer Configuration Register of Timer A8	4
01FBF048	TCFRA9	Timer Configuration Register of Timer A9	4
01FBF050	TCFRA10	Timer Configuration Register of Timer A10	4
01FBF058	TCFRA11	Timer Configuration Register of Timer A11	4
01FBF060	TCFRA12	Timer Configuration Register of Timer A12	4
01FBF068	TCFRA13	Timer Configuration Register of Timer A13	4
01FBF070	TCFRA14	Timer Configuration Register of Timer A14	4
01FBF078	TCFRA15	Timer Configuration Register of Timer A15	4
01FBF080	TCMPA0	Timer Compare Register of Timer A0	4
01FBF088	TCMPA1	Timer Compare Register of Timer A1	4
01FBF090	TCMPA2	Timer Compare Register of Timer A2	4
01FBF098	TCMPA3	Timer Compare Register of Timer A3	4
01FBF0A0	TCMPA4	Timer Compare Register of Timer A4	4
01FBF0A8	TCMPA5	Timer Compare Register of Timer A5	4
01FBF0B0	TCMPA6	Timer Compare Register of Timer A6	4
01FBF0B8	TCMPA7	Timer Compare Register of Timer A7	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FBF0C0	TCMPA8	Timer Compare Register of Timer A8	4
01FBF0C8	TCMPA9	Timer Compare Register of Timer A9	4
01FBF0D0	TCMPA10	Timer Compare Register of Timer A10	4
01FBF0D8	TCMPA11	Timer Compare Register of Timer A11	4
01FBF0E0	TCMPA12	Timer Compare Register of Timer A12	4
01FBF0E8	TCMPA13	Timer Compare Register of Timer A13	4
01FBF0F0	TCMPA14	Timer Compare Register of Timer A14	4
01FBF0F8	TCMPA15	Timer Compare Register of Timer A15	4
01FBF100	TCRA0	Timer Control Register of Timer A0	4
01FBF108	TCRA1	Timer Control Register of Timer A1	4
01FBF110	TCRA2	Timer Control Register of Timer A2	4
01FBF118	TCRA3	Timer Control Register of Timer A3	4
01FBF120	TCRA4	Timer Control Register of Timer A4	4
01FBF128	TCRA5	Timer Control Register of Timer A5	4
01FBF130	TCRA6	Timer Control Register of Timer A6	4
01FBF138	TCRA7	Timer Control Register of Timer A7	4
01FBF140	TCRA8	Timer Control Register of Timer A8	4
01FBF148	TCRA9	Timer Control Register of Timer A9	4
01FBF150	TCRA10	Timer Control Register of Timer A10	4
01FBF158	TCRA11	Timer Control Register of Timer A11	4
01FBF160	TCRA12	Timer Control Register of Timer A12	4
01FBF168	TCRA13	Timer Control Register of Timer A13	4
01FBF170	TCRA14	Timer Control Register of Timer A14	4
01FBF178	TCRA15	Timer Control Register of Timer A15	4
01FBF180	TCNRA0	Timer Count Register of Timer A0	4
01FBF188	TCNRA1	Timer Count Register of Timer A1	4
01FBF190	TCNRA2	Timer Count Register of Timer A2	4
01FBF198	TCNRA3	Timer Count Register of Timer A3	4
01FBF1A0	TCNRA4	Timer Count Register of Timer A4	4
01FBF1A8	TCNRA5	Timer Count Register of Timer A5	4
01FBF1B0	TCNRA6	Timer Count Register of Timer A6	4
01FBF1B8	TCNRA7	Timer Count Register of Timer A7	4
01FBF1C0	TCNRA8	Timer Count Register of Timer A8	4
01FBF1C8	TCNRA9	Timer Count Register of Timer A9	4
01FBF1D0	TCNRA10	Timer Count Register of Timer A10	4
01FBF1D8	TCNRA11	Timer Count Register of Timer A11	4
01FBF1E0	TCNRA12	Timer Count Register of Timer A12	4
01FBF1E8	TCNRA13	Timer Count Register of Timer A13	4
01FBF1F0	TCNRA14	Timer Count Register of Timer A14	4
01FBF1F8	TCNRA15	Timer Count Register of Timer A15	4
01FBF200–01FBF37F	Reserved		

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FBF380	TGCRA	Timer General Configuration Register of Timers Module A	4
01FBF388	TERA	Timer Event Register of Timers Module A	4
01FBF390	TIERA	Timer Interrupt Enable Register of Timers Module A	4
01FBF398	TSRA	Timer Status Register of Timers Module A	4
01FBF3A0–01FBF3FF	Reserved		
01FBF400	TCFRB0	Timer Configuration Register of Timer B0	4
01FBF408	TCFRB1	Timer Configuration Register of Timer B1	4
01FBF410	TCFRB2	Timer Configuration Register of Timer B2	4
01FBF418	TCFRB3	Timer Configuration Register of Timer B3	4
01FBF420	TCFRB4	Timer Configuration Register of Timer B4	4
01FBF428	TCFRB5	Timer Configuration Register of Timer B5	4
01FBF430	TCFRB6	Timer Configuration Register of Timer B6	4
01FBF438	TCFRB7	Timer Configuration Register of Timer B7	4
01FBF440	TCFRB8	Timer Configuration Register of Timer B8	4
01FBF448	TCFRB9	Timer Configuration Register of Timer B9	4
01FBF450	TCFRB10	Timer Configuration Register of Timer B10	4
01FBF458	TCFRB11	Timer Configuration Register of Timer B11	4
01FBF460	TCFRB12	Timer Configuration Register of Timer B12	4
01FBF468	TCFRB13	Timer Configuration Register of Timer B13	4
01FBF470	TCFRB14	Timer Configuration Register of Timer B14	4
01FBF478	TCFRB15	Timer Configuration Register of Timer B15	4
01FBF480	TCMPB0	Timer Compare Register of Timer B0	4
01FBF488	TCMPB1	Timer Compare Register of Timer B1	4
01FBF490	TCMPB2	Timer Compare Register of Timer B2	4
01FBF498	TCMPB3	Timer Compare Register of Timer B3	4
01FBF4A0	TCMPB4	Timer Compare Register of Timer B4	4
01FBF4A8	TCMPB5	Timer Compare Register of Timer B5	4
01FBF40	TCMPB6	Timer Compare Register of Timer B6	4
01FBF48	TCMPB7	Timer Compare Register of Timer B7	4
01FBF4C0	TCMPB8	Timer Compare Register of Timer B8	4
01FBF4C8	TCMPB9	Timer Compare Register of Timer B9	4
01FBF4D0	TCMPB10	Timer Compare Register of Timer B10	4
01FBF4D8	TCMPB11	Timer Compare Register of Timer B11	4
01FBF4E0	TCMPB12	Timer Compare Register of Timer B12	4
01FBF4E8	TCMPB13	Timer Compare Register of Timer B13	4
01FBF4F0	TCMPB14	Timer Compare Register of Timer B14	4
01FBF4F8	TCMPB15	Timer Compare Register of Timer B15	4
01FBF500	TCRB0	Timer Control Register of Timer B0	4
01FBF508	TCRB1	Timer Control Register of Timer B1	4
01FBF510	TCRB2	Timer Control Register of Timer B2	4
01FBF518	TCRB3	Timer Control Register of Timer B3	4

Table 8-6. IPBus Memory Map QBus Bank 3 Addresses (0x01F80000–0x01FBFFFF)

Address	Acronym	Name	Size in Bytes
01FBF520	TCRB4	Timer Control Register of Timer B4	4
01FBF528	TCRB5	Timer Control Register of Timer B5	4
01FBF530	TCRB6	Timer Control Register of Timer B6	4
01FBF538	TCRB7	Timer Control Register of Timer B7	4
01FBF540	TCRB8	Timer Control Register of Timer B8	4
01FBF548	TCRB9	Timer Control Register of Timer B9	4
01FBF550	TCRB10	Timer Control Register of Timer B10	4
01FBF558	TCRB11	Timer Control Register of Timer B11	4
01FBF560	TCRB12	Timer Control Register of Timer B12	4
01FBF568	TCRB13	Timer Control Register of Timer B13	4
01FBF570	TCRB14	Timer Control Register of Timer B14	4
01FBF578	TCRB15	Timer Control Register of Timer B15	4
01FBF580	TCNRB0	Timer Count Register of Timer B0	4
01FBF588	TCNRB1	Timer Count Register of Timer B1	4
01FBF590	TCNRB2	Timer Count Register of Timer B2	4
01FBF598	TCNRB3	Timer Count Register of Timer B3	4
01FBF5A0	TCNRB4	Timer Count Register of Timer B4	4
01FBF5A8	TCNRB5	Timer Count Register of Timer B5	4
01FBF5B0	TCNRB6	Timer Count Register of Timer B6	4
01FBF5B8	TCNRB7	Timer Count Register of Timer B7	4
01FBF5C0	TCNRB8	Timer Count Register of Timer B8	4
01FBF5C8	TCNRB9	Timer Count Register of Timer B9	4
01FBF5D0	TCNRB10	Timer Count Register of Timer B10	4
01FBF5D8	TCNRB11	Timer Count Register of Timer B11	4
01FBF5E0	TCNRB12	Timer Count Register of Timer B12	4
01FBF5E8	TCNRB13	Timer Count Register of Timer B13	4
01FBF5F0	TCNRB14	Timer Count Register of Timer B14	4
01FBF5F8	TCNRB15	Timer Count Register of Timer B15	4
01FBF600–01FBF77F	Reserved		
01FBF780	TGCRB	Timer General Configuration Register of Timers Module B	4
01FBF788	TERB	Timer Event Register of Timers Module B	4
01FBF790	TIERB	Timer Interrupt Enable Register of Timers Module B	4
01FBF798	TSRB	Timer Status Register of Timers Module B	4
01FBF7A0–01FBFFFF	Reserved		

8.6 Local Bus Address Space

The local bus address space comprises internal devices. The memory controller identifies banks 9–11 for these devices and generates chip selects and other signals for accessing these banks. The following resources reside on the local bus:

- Three M1 memories
- M2 memory
- IPBus peripherals (TDM interfaces, GIC, HSRs, GPIO, UART/SCI, DSI, Ethernet, and timers)

The SC140 configures the local bus address space during the boot procedure. The boot procedure includes the whole memory controller configuration, including the base register, the mask register, and the UPM (see **Section 12.7**, *Internal SRAM and IPBus Peripherals Support*). You can modify the base address of the resources in the local address space by changing BR9 and BR11.

Table 8-7 describes the address spaces allocated to banks 9 and 11 as a function of the ISBSEL in the Hard Reset Configuration Word (HRCW). The EMR[ISBSEL] field contains the value of the ISBSEL; for details, see **Section 2.2.3**, *Program Control Unit Programming Model,* on page 2-12.

Table 8-7. Banks 9 and 11 Address Space

ISBSEL	Bank 9 Base Address	Bank 11 Base Address
0	02180000–021BFFFF	02000000–0217FFFF
1	02380000–023BFFFF	02200000–0237FFFF
2	02580000–025BFFFF	02400000–0257FFFF
3	02780000–027BFFFF	02600000–0277FFFF
6	02D80000–02DBFFFF	02C00000–02D7FFFF
7	02F80000–02FBFFFF	02E00000–02F7FFFF

Table 8-8 lists the local bus registers with addresses based on the ISB selection at reset.

Table 8-8. Local Bus Banks 9, 11 Memory Map

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
02000000	02200000	02400000	02600000	02C00000	02E00000	Bank 11 Base Address Starting Address (UPMC)		
02000000– 02076FFF	02200000– 02276FFF	02400000– 02476FFF	02600000– 02676FFF	02C00000– 02C76FFF	02E00000– 02E76FFF	M2MEM	M2 Memory	476 K
02077000– 0207FFFF	02277000– 0227FFFF	02477000– 0247FFFF	02677000– 0267FFFF	02C77000– 02C7FFFF	02E77000– 02E7FFFF	Reserved		
02080000– 020B7FFF	02280000– 022B7FFF	02480000– 0247FFF	02680000– 026B7FFF	02C80000– 02CB7FFF	02E80000– 02EB7FFF	M1MEM0	M1 Memory Core 0	224 K
020B8000– 020BBFFF	022B8000– 022BBFFF	0248000– 024BFFF	026B8000– 026BBFFF	02CB8000– 02CBBFFF	02EB8000– 02EBBFFF	ICC0	ICache Core 0	16 K
020BC000– 020BFFFF	022BC000– 022BFFFF	024C000– 024FFFF	026BC000– 026BFFFF	02CBC000– 02CBFFFF	02EBC000– 02EBFFFF	Reserved		

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
020C0000–020F7FFF	022C0000–022F7FFF	024C0000–024F7FFF	026C0000–026F7FFF	02CC0000–02CF7FFF	02EC0000–02EF7FFF	M1MEM1	M1 Memory Core 1	224 K
020F8000–020FBFFF	022F8000–022FBFFF	024F8000–024FBFFF	026F8000–026FBFFF	02CF8000–02CFBFFF	02EF8000–02EFBFFF	ICC1	ICache Core 1	16 K
020FC000–020FFFFF	022FC000–022FFFFF	024FC000–024FFFFF	026FC000–026FFFFF	02CFC000–02CFFFFF	02EFC000–02EFFFFF	Reserved		
02100000–02137FFF	02300000–02337FFF	02500000–02537FFF	02700000–02737FFF	02D00000–02D37FFF	02F00000–02F37FFF	M1MEM2	M1 Memory Core 2	224 K
02138000–0213BFFF	02338000–0233BFFF	02538000–0253BFFF	02738000–0273BFFF	02D38000–02D3BFFF	02F38000–02F3BFFF	ICC2	ICache Core 2	16 K
0213C000–0213FFFF	0233C000–0233FFFF	0253C000–0253FFFF	0273C000–0273FFFF	02D3C000–02D3FFFF	02F3C000–02F3FFFF	Reserved		
0214C000–0217FFFF	0234C000–0237FFFF	0254C000–0257FFFF	0274C000–0277FFFF	02D4C000–02D7FFFF	02F4C000–02F7FFFF	Reserved		
02180000	02380000	02580000	02780000	02D80000	02F80000	Bank 9 Base Address Starting Address (GPCM-Local)		
02180000–021807FF	02380000–023807FF	02580000–025807FF	02780000–027807FF	02D80000–02D807FF	02F80000–02F807FF		TDM0 Receive Local Memory	2 K
02180800–02180FFF	02380800–02380FFF	02580800–02580FFF	02780800–02780FFF	02D80800–02D80FFF	02F80800–02F80FFF	Reserved		
02181000–021813FC	02381000–023813FC	02581000–025813FC	02781000–027813FC	02D81000–02D813FC	02F81000–02F813FC	TDM0 RCPR [0–255]	TDM0 Receive Channel Parameters Register 0–255	4 each
02181400–021817FF	02381400–023817FF	02581400–025817FF	02781400–027817FF	02D81400–02D817FF	02F81400–02F817FF	Reserved		
02181800–02181FFF	02381800–02381FFF	02581800–02581FFF	02781800–02781FFF	02D81800–02D81FFF	02F81800–02F81FFF		TDM0 Transmit Local Memory	2 K
02182000–021827FF	02382000–023827FF	02582000–025827FF	02782000–027827FF	02D82000–02D827FF	02F82000–02F827FF	Reserved		
02182800–02182BFC	02382800–02382BFC	02582800–02582BFC	02782800–02782BFC	02D82800–02D82BFC	02F82800–02F82BFC	TDM0 TCPR [0–255]	TDM0 Transmit Channel Parameters Register 0–255	4
02182C00–02183F1F	02382C00–02383F1F	02582C00–02583F1F	02782C00–02783F1F	02D82C00–02D83F1F	02F82C00–02F83F1F	Reserved		
02183F20	02383F20	02583F20	02783F20	02D83F20	02F83F20	TDM0TSR	TDM0 Transmit Status Register	4
02183F28	02383F28	02583F28	02783F28	02D83F28	02F83F28	TDM0RSR	TDM0 Receive Status Register	4
02183F30	02383F30	02583F30	02783F30	02D83F30	02F83F30	TDM0ASR	TDM0 Adaptation Status Register	4
02183F38	02383F38	02583F38	02783F38	02D83F38	02F83F38	TDM0TER	TDM0 Transmit Event Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
02183F40	02383F40	02583F40	02783F40	02D83F40	02F83F40	TDM0RER	TDM0 Receive Event Register	4
02183F48	02383F48	02583F48	02783F48	02D83F48	02F83F48	TDM0TNB	TDM0 Transmit Number of Buffers	4
02183F50	02383F50	02583F50	02783F50	02D83F50	02F83F50	TDM0RNB	TDM0 Receive Number of Buffers	4
02183F58	02383F58	02583F58	02783F58	02D83F58	02F83F58	TDM0TDBDR	TDM0 Transmit Data Buffer Displacement Register	4
02183F60	02383F60	02583F60	02783F60	02D83F60	02F83F60	TDM0RDBDR	TDM0 Receive Data Buffer Displacement Register	4
02183F68	02383F68	02583F68	02783F68	02D83F68	02F83F68	TDM0ASDR	TDM0 Adaptation Sync Distance Register	4
02183F70	02383F70	02583F70	02783F70	02D83F70	02F83F70	TDM0TIER	TDM0 Transmit Interrupt Enable Register	4
02183F78	02383F78	02583F78	02783F78	02D83F78	02F83F78	TDM0RIER	TDM0 Receive Interrupt Enable Register	4
02183F80	02383F80	02583F80	02783F80	02D83F80	02F83F80	TDM0TDBST	TDM0 Transmit Data Buffer Second Threshold	4
02183F88	02383F88	02583F88	02783F88	02D83F88	02F83F88	TDM0RDBST	TDM0 Receive Data Buffer Second Threshold	4
02183F90	02383F90	02583F90	02783F90	02D83F90	02F83F90	TDM0TDBFT	TDM0 Transmit Data Buffer First Threshold	4
02183F98	02383F98	02583F98	02783F98	02D83F98	02F83F98	TDM0RDBFT	TDM0 Receive Data Buffer First Threshold	4
02183FA0	02383FA0	02583FA0	02783FA0	02D83FA0	02F83FA0	TDM0TCR	TDM0 Transmit Control Register	4
02183FA8	02383FA8	02583FA8	02783FA8	02D83FA8	02F83FA8	TDM0RCR	TDM0 Receive Control Register	4
02183FB0	02383FB0	02583FB0	02783FB0	02D83FB0	02F83FB0	TDM0ACR	TDM0 Adaptation Control Register	4
02183FB8	02383FB8	02583FB8	02783FB8	02D83FB8	02F83FB8	TDM0TGBA	TDM0 Transmit Global Base Address	4
02183FC0	02383FC0	02583FC0	02783FC0	02D83FC0	02F83FC0	TDM0RGBA	TDM0 Receive Global Base Address	4
02183FC8	02383FC8	02583FC8	02783FC8	02D83FC8	02F83FC8	TDM0TDBS	TDM0 Transmit Data Buffer Size	4
02183FD0	02383FD0	02583FD0	02783FD0	02D83FD0	02F83FD0	TDM0RDBS	TDM0 Receive Data Buffer Size	4
02183FD8	02383FD8	02583FD8	02783FD8	02D83FD8	02F83FD8	TDM0TFP	TDM0 Transmit Frame Parameters	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
02183FE0	02383FE0	02583FE0	02783FE0	02D83FE0	02F83FE0	TDM0RFP	TDM0 Receive Frame Parameters	4
02183FE8	02383FE8	02583FE8	02783FE8	02D83FE8	02F83FE8	TDM0TIR	TDM0 Transmit Interface Register	4
02183FF0	02383FF0	02583FF0	02783FF0	02D83FF0	02F83FF0	TDM0RIR	TDM0 Receive Interface Register	4
02183FF8	02383FF8	02583FF8	02783FF8	02D83FF8	02F83FF8	TDM0GIR	TDM0 General Interface Register	4
02184000–021847FF	02384000–023847FF	02584000–025847FF	02784000–027847FF	02D84000–02D847FF	02F84000–02F847FF		TDM1 Receive Local Memory	2 K
02184800–02184FFF	02384800–02384FFF	02584800–02584FFF	02784800–02784FFF	02D84800–02D84FFF	02F84800–02F84FFF	Reserved		
02185000–021853FC	02385000–023853FC	02585000–025853FC	02785000–027853FC	02D85000–02D853FC	02F85000–02F853FC	TDM1 RCPR [0–255]	TDM1 Receive Channel Parameters Register 0–255	4 each
02185400–021857FF	02385400–023857FF	02585400–025857FF	02785400–027857FF	02D85400–02D857FF	02F85400–02F857FF	Reserved		
02185800–02185FFF	02385800–02385FFF	02585800–02585FFF	02785800–02785FFF	02D85800–02D85FFF	02F85800–02F85FFF		TDM1 Transmit Local Memory	2 K
02186000–021867FF	02386000–023867FF	02586000–025867FF	02786000–027867FF	02D86000–02D867FF	02F86000–02F867FF	Reserved		
02186800–02186BFC	02386800–02386BFC	02586800–02586BFC	02786800–02786BFC	02D86800–02D86BFC	02F86800–02F86BFC	TDM1 TCPR [0–255]	TDM1 Transmit Channel Parameters Register 0–255	4 each
02186C00–02187F1F	02386C00–02387F1F	02586C00–02587F1F	02786C00–02787F1F	02D86C00–02D87F1F	02F86C00–02F87F1F	Reserved		
02187F20	02387F20	02587F20	02787F20	02D87F20	02F87F20	TDM1TSR	TDM1 Transmit Status Register	4
02187F28	02387F28	02587F28	02787F28	02D87F28	02F87F28	TDM1RSR	TDM1 Receive Status Register	4
02187F30	02387F30	02587F30	02787F30	02D87F30	02F87F30	TDM1ASR	TDM1 Adaptation Status Register	4
02187F38	02387F38	02587F38	02787F38	02D87F38	02F87F38	TDM1TER	TDM1 Transmit Event Register	4
02187F40	02387F40	02587F40	02787F40	02D87F40	02F87F40	TDM1RER	TDM1 Receive Event Register	4
02187F48	02387F48	02587F48	02787F48	02D87F48	02F87F48	TDM1TNB	TDM1 Transmit Number of Buffers	4
02187F50	02387F50	02587F50	02787F50	02D87F50	02F87F50	TDM1RNB	TDM1 Receive Number of Buffers	4
02187F58	02387F58	02587F58	02787F58	02D87F58	02F87F58	TDM1TDBDR	TDM1 Transmit Data Buffer Displacement Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
02187F60	02387F60	02587F60	02787F60	02D87F60	02F87F60	TDM1RDBDR	TDM1 Receive Data Buffer Displacement Register	4
02187F68	02387F68	02587F68	02787F68	02D87F68	02F87F68	TDM1ASDR	TDM1 Adaptation Sync Distance Register	4
02187F70	02387F70	02587F70	02787F70	02D87F70	02F87F70	TDM1TIER	TDM1 Transmit Interrupt Enable Register	4
02187F78	02387F78	02587F78	02787F78	02D87F78	02F87F78	TDM1RIER	TDM1 Receive Interrupt Enable Register	4
02187F80	02387F80	02587F80	02787F80	02D87F80	02F87F80	TDM1TDBST	TDM1 Transmit Data Buffer Second Threshold	4
02187F88	02387F88	02587F88	02787F88	02D87F88	02F87F88	TDM1RDBST	TDM1 Receive Data Buffer Second Threshold	4
02187F90	02387F90	02587F90	02787F90	02D87F90	02F87F90	TDM1TDBFT	TDM1 Transmit Data Buffer First Threshold	4
02187F98	02387F98	02587F98	02787F98	02D87F98	02F87F98	TDM1RDBFT	TDM1 Receive Data Buffer First Threshold	4
02187FA0	02387FA0	02587FA0	02787FA0	02D87FA0	02F87FA0	TDM1TCR	TDM1 Transmit Control Register	4
02187FA8	02387FA8	02587FA8	02787FA8	02D87FA8	02F87FA8	TDM1RCR	TDM1 Receive Control Register	4
02187FB0	02387FB0	02587FB0	02787FB0	02D87FB0	02F87FB0	TDM1ACR	TDM1 Adaptation Control Register	4
02187FB8	02387FB8	02587FB8	02787FB8	02D87FB8	02F87FB8	TDM1TGBA	TDM1 Transmit Global Base Address	4
02187FC0	02387FC0	02587FC0	02787FC0	02D87FC0	02F87FC0	TDM1RGBA	TDM1 Receive Global Base Address	4
02187FC8	02387FC8	02587FC8	02787FC8	02D87FC8	02F87FC8	TDM1TDBS	TDM1 Transmit Data Buffer Size	4
02187FD0	02387FD0	02587FD0	02787FD0	02D87FD0	02F87FD0	TDM1RDBS	TDM1 Receive Data Buffer Size	4
02187FD8	02387FD8	02587FD8	02787FD8	02D87FD8	02F87FD8	TDM1TFP	TDM1 Transmit Frame Parameters	4
02187FE0	02387FE0	02587FE0	02787FE0	02D87FE0	02F87FE0	TDM1RFP	TDM1 Receive Frame Parameters	4
02187FE8	02387FE8	02587FE8	02787FE8	02D87FE8	02F87FE8	TDM1TIR	TDM1 Transmit Interface Register	4
02187FF0	02387FF0	02587FF0	02787FF0	02D87FF0	02F87FF0	TDM1RIR	TDM1 Receive Interface Register	4
02187FF8	02387FF8	02587FF8	02787FF8	02D87FF8	02F87FF8	TDM1GIR	TDM1 General Interface Register	4
02188000–021887FF	02388000–023887FF	02588000–025887FF	02788000–027887FF	02D88000–02D887FF	02F88000–02F887FF		TDM2 Receive Local Memory	2 K

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
02188800–02188FFF	02388800–02388FFF	02588800–02588FFF	02788800–02788FFF	02D88800–02D88FFF	02F88800–02F88FFF	Reserved		
02189000–021893FC	02389000–023893FC	02589000–025893FC	02789000–027893FC	02D89000–02D893FC	02F89000–02F893FC	TDM2 RCPR [0–255]	TDM2 Receive Channel Parameters Register 0–255	4 each
02189400–021897FF	02389400–023897FF	02589400–025897FF	02789400–027897FF	02D89400–02D897FF	02F89400–02F897FF	Reserved		
02189800–02189FFF	02389800–02389FFF	02589800–02589FFF	02789800–02789FFF	02D89800–02D89FFF	02F89800–02F89FFF		TDM2 Transmit Local Memory	2 K
0218A000–0218A7FF	0238A000–0238A7FF	0258A000–0258A7FF	0278A000–0278A7FF	02D8A000–02D8A7FF	02F8A000–02F8A7FF	Reserved		
0218A800–0218ABFC	0238A800–0238ABFC	0258A800–0258ABFC	0278A800–0278ABFC	02D8A800–02D8ABFC	02F8A800–02F8ABFC	TDM2 TCPR [0–255]	TDM2 Transmit Channel Parameters Register 0–255	4 each
0218AC00–0218BF1F	0238AC00–0238BF1F	0258AC00–0258BF1F	0278AC00–0278BF1F	02D8AC00–02D8BF1F	02F8AC00–02F8BF1F	Reserved		
0218BF20	0238BF20	0258BF20	0278BF20	02D8BF20	02F8BF20	TDM2TSR	TDM2 Transmit Status Register	4
0218BF28	0238BF28	0258BF28	0278BF28	02D8BF28	02F8BF28	TDM2RSR	TDM2 Receive Status Register	4
0218BF30	0238BF30	0258BF30	0278BF30	02D8BF30	02F8BF30	TDM2ASR	TDM2 Adaptation Status Register	4
0218BF38	0238BF38	0258BF38	0278BF38	02D8BF38	02F8BF38	TDM2TER	TDM2 Transmit Event Register	4
0218BF40	0238BF40	0258BF40	0278BF40	02D8BF40	02F8BF40	TDM2RER	TDM2 Receive Event Register	4
0218BF48	0238BF48	0258BF48	0278BF48	02D8BF48	02F8BF48	TDM2TNB	TDM2 Transmit Number of Buffers	4
0218BF50	0238BF50	0258BF50	0278BF50	02D8BF50	02F8BF50	TDM2RNB	TDM2 Receive Number of Buffers	4
0218BF58	0238BF58	0258BF58	0278BF58	02D8BF58	02F8BF58	TDM2TDBDR	TDM2 Transmit Data Buffer Displacement Register	4
0218BF60	0238BF60	0258BF60	0278BF60	02D8BF60	02F8BF60	TDM2RDBDR	TDM2 Receive Data Buffer Displacement Register	4
0218BF68	0238BF68	0258BF68	0278BF68	02D8BF68	02F8BF68	TDM2ASDR	TDM2 Adaptation Sync Distance Register	4
0218BF70	0238BF70	0258BF70	0278BF70	02D8BF70	02F8BF70	TDM2TIER	TDM2 Transmit Interrupt Enable Register	4
0218BF78	0238BF78	0258BF78	0278BF78	02D8BF78	02F8BF78	TDM2RIER	TDM2 Receive Interrupt Enable Register	4
0218BF80	0238BF80	0258BF80	0278BF80	02D8BF80	02F8BF80	TDM2TDBST	TDM2 Transmit Data Buffer Second Threshold	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
0218BF88	0238BF88	0258BF88	0278BF88	02D8BF88	02F8BF88	TDM2RDBST	TDM2 Receive Data Buffer Second Threshold	4
0218BF90	0238BF90	0258BF90	0278BF90	02D8BF90	02F8BF90	TDM2TDBFT	TDM2 Transmit Data Buffer First Threshold	4
0218BF98	0238BF98	0258BF98	0278BF98	02D8BF98	02F8BF98	TDM2RDBFT	TDM2 Receive Data Buffer First Threshold	4
0218BFA0	0238BFA0	0258BFA0	0278BFA0	02D8BFA0	02F8BFA0	TDM2TCR	TDM2 Transmit Control Register	4
0218BFA8	0238BFA8	0258BFA8	0278BFA8	02D8BFA8	02F8BFA8	TDM2RCR	TDM2 Receive Control Register	4
0218BFB0	0238BFB0	0258BFB0	0278BFB0	02D8BFB0	02F8BFB0	TDM2ACR	TDM2 Adaptation Control Register	4
0218BFB8	0238BFB8	0258BFB8	0278BFB8	02D8BFB8	02F8BFB8	TDM2TGBA	TDM2 Transmit Global Base Address	4
0218BFC0	0238BFC0	0258BFC0	0278BFC0	02D8BFC0	02F8BFC0	TDM2RGBA	TDM2 Receive Global Base Address	4
0218BFC8	0238BFC8	0258BFC8	0278BFC8	02D8BFC8	02F8BFC8	TDM2TDBS	TDM2 Transmit Data Buffer Size	4
0218BFD0	0238BFD0	0258BFD0	0278BFD0	02D8BFD0	02F8BFD0	TDM2RDBS	TDM2 Receive Data Buffer Size	4
0218BFD8	0238BFD8	0258BFD8	0278BFD8	02D8BFD8	02F8BFD8	TDM2TFP	TDM2 Transmit Frame Parameters	4
0218BFE0	0238BFE0	0258BFE0	0278BFE0	02D8BFE0	02F8BFE0	TDM2RFP	TDM2 Receive Frame Parameters	4
0218BFE8	0238BFE8	0258BFE8	0278BFE8	02D8BFE8	02F8BFE8	TDM2TIR	TDM2 Transmit Interface Register	4
0218BFF0	0238BFF0	0258BFF0	0278BFF0	02D8BFF0	02F8BFF0	TDM2RIR	TDM2 Receive Interface Register	4
0218BFF8	0238BFF8	0258BFF8	0278BFF8	02D8BFF8	02F8BFF8	TDM2GIR	TDM2 General Interface Register	4
0218C000–0218C7FF	0238C000–0238C7FF	0258C000–0258C7FF	0278C000–0278C7FF	02D8C000–02D8C7FF	02F8C000–02F8C7FF		TDM3 Receive Local Memory	2 K
0218C800–0218CFFF	0238C800–0238CFFF	0258C800–0258CFFF	0278C800–0278CFFF	02D8C800–02D8CFFF	02F8C800–02F8CFFF	Reserved		
0218D000–0218D3FC	0238D000–0238D3FC	0258D000–0258D3FC	0278D000–0278D3FC	02D8D000–02D8D3FC	02F8D000–02F8D3FC	TDM3 RCPR {0–255]	TDM3 Receive Channel Parameters Register 0–255	4 each
0218D400–0218D7FF	0238D400–0238D7FF	0258D400–0258D7FF	0278D400–0278D7FF	02D8D400–02D8D7FF	02F8D400–02F8D7FF	Reserved		
0218D800–0218DFFF	0238D800–0238DFFF	0258D800–0258DFFF	0278D800–0278DFFF	02D8D800–02D8DFFF	02F8D800–02F8DFFF		TDM3 Transmit Local Memory	2 K
0218E000–0218E7FF	0238E000–0238E7FF	0258E000–0258E7FF	0278E000–0278E7FF	02D8E000–02D8E7FF	02F8E000–02F8E7FF	Reserved		

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
0218E800–0218EBFC	0238E800–0238EBFC	0258E800–0258EBFC	0278E800–0278EBFC	02D8E800–02D8EBFC	02F8E800–02F8EBFC	TDM3 TCPR [0–255]	TDM3 Transmit Channel Parameters Register 0–255	4 each
0218EC00–0218FF1F	0238EC00–0238FF1F	0258EC00–0258FF1F	0278EC00–0278FF1F	02D8EC00–02D8FF1F	02F8EC00–02F8FF1F	Reserved		
0218FF20	0238FF20	0258FF20	0278FF20	02D8FF20	02F8FF20	TDM3TSR	TDM3 Transmit Status Register	4
0218FF28	0238FF28	0258FF28	0278FF28	02D8FF28	02F8FF28	TDM3RSR	TDM3 Receive Status Register	4
0218FF30	0238FF30	0258FF30	0278FF30	02D8FF30	02F8FF30	TDM3ASR	TDM3 Adaptation Status Register	4
0218FF38	0238FF38	0258FF38	0278FF38	02D8FF38	02F8FF38	TDM3TER	TDM3 Transmit Event Register	4
0218FF40	0238FF40	0258FF40	0278FF40	02D8FF40	02F8FF40	TDM3RER	TDM3 Receive Event Register	4
0218FF48	0238FF48	0258FF48	0278FF48	02D8FF48	02F8FF48	TDM3TNB	TDM3 Transmit Number of Buffers	4
0218FF50	0238FF50	0258FF50	0278FF50	02D8FF50	02F8FF50	TDM3RNB	TDM3 Receive Number of Buffers	4
0218FF58	0238FF58	0258FF58	0278FF58	02D8FF58	02F8FF58	TDM3TDBDR	TDM3 Transmit Data Buffer Displacement Register	4
0218FF60	0238FF60	0258FF60	0278FF60	02D8FF60	02F8FF60	TDM3RDBDR	TDM3 Receive Data Buffer Displacement Register	4
0218FF68	0238FF68	0258FF68	0278FF68	02D8FF68	02F8FF68	TDM3ASDR	TDM3 Adaptation Sync Distance Register	4
0218FF70	0238FF70	0258FF70	0278FF70	02D8FF70	02F8FF70	TDM3TIER	TDM3 Transmit Interrupt Enable Register	4
0218FF78	0238FF78	0258FF78	0278FF78	02D8FF78	02F8FF78	TDM3RIER	TDM3 Receive Interrupt Enable Register	4
0218FF80	0238FF80	0258FF80	0278FF80	02D8FF80	02F8FF80	TDM3TDBST	TDM3 Transmit Data Buffer Second Threshold	4
0218FF88	0238FF88	0258FF88	0278FF88	02D8FF88	02F8FF88	TDM3RDBST	TDM3 Receive Data Buffer Second Threshold	4
0218FF90	0238FF90	0258FF90	0278FF90	02D8FF90	02F8FF90	TDM3TDBFT	TDM3 Transmit Data Buffer First Threshold	4
0218FF98	0238FF98	0258FF98	0278FF98	02D8FF98	02F8FF98	TDM3RDBFT	TDM3 Receive Data Buffer First Threshold	4
0218FFA0	0238FFA0	0258FFA0	0278FFA0	02D8FFA0	02F8FFA0	TDM3TCR	TDM3 Transmit Control Register	4
0218FFA8	0238FFA8	0258FFA8	0278FFA8	02D8FFA8	02F8FFA8	TDM3RCR	TDM3 Receive Control Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
0218FFB0	0238FFB0	0258FFB0	0278FFB0	02D8FFB0	02F8FFB0	TDM3ACR	TDM3 Adaptation Control Register	4
0218FFB8	0238FFB8	0258FFB8	0278FFB8	02D8FFB8	02F8FFB8	TDM3TGBA	TDM3 Transmit Global Base Address	4
0218FFC0	0238FFC0	0258FFC0	0278FFC0	02D8FFC0	02F8FFC0	TDM3RGBA	TDM3 Receive Global Base Address	4
0218FFC8	0238FFC8	0258FFC8	0278FFC8	02D8FFC8	02F8FFC8	TDM3TDBS	TDM3 Transmit Data Buffer Size	4
0218FFD0	0238FFD0	0258FFD0	0278FFD0	02D8FFD0	02F8FFD0	TDM3RDBS	TDM3 Receive Data Buffer Size	4
0218FFD8	0238FFD8	0258FFD8	0278FFD8	02D8FFD8	02F8FFD8	TDM3TFP	TDM3 Transmit Frame Parameters	4
0218FFE0	0238FFE0	0258FFE0	0278FFE0	02D8FFE0	02F8FFE0	TDM3RFP	TDM3 Receive Frame Parameters	4
0218FFE8	0238FFE8	0258FFE8	0278FFE8	02D8FFE8	02F8FFE8	TDM3TIR	TDM3 Transmit Interface Register	4
0218FFF0	0238FFF0	0258FFF0	0278FFF0	02D8FFF0	02F8FFF0	TDM3RIR	TDM3 Receive Interface Register	4
0218FFF8	0238FFF8	0258FFF8	0278FFF8	02D8FFF8	02F8FFF8	TDM3GIR	TDM3 General Interface Register	4
02190000–021B800F	02390000–023B800F	02590000–025B800F	02790000–027B800F	02D90000–02DB800F	02F90000–02FB800F	Reserved		
021B8010	023B8010	025B8010	027B8010	02DB8010	02FB8010	IEVENT	Interrupt Event Register	4
021B8014	023B8014	025B8014	027B8014	02DB8014	02FB8014	IMASK	Interrupt Mask Register	4
021B8018	023B8018	025B8018	027B8018	02DB8018	02FB8018	Reserved		
021B8020	023B8020	025B8020	027B8020	02DB8020	02FB8020	ECNTRL	Ethernet Control Register	4
021B8024	023B8024	025B8024	027B8024	02DB8024	02FB8024	MINFLR	Minimum Frame Length Register	4
021B8028	023B8028	025B8028	027B8028	02DB8028	02FB8028	PTV	Pause Time Value Register	4
021B802C	023B802C	025B802C	027B802C	02DB802C	02FB802C	DMACTRL	DMA Control Register	4
021B8034	023B8034	025B8034	027B8034	02DB8034	02FB8034	Reserved		
021B8038	023B8038	025B8038	027B8038	02DB8038	02FB8038	DMAMR	DMA Maintenance Register	4
021B803C	023B803C	025B803C	027B803C	02DB803C	02FB803C	Reserved		
021B8048	023B8048	025B8048	027B8048	02DB8048	02FB8048	FRXSTATR	FIFO Receive Status Register	4
021B804C	023B804C	025B804C	027B804C	02DB804C	02FB804C	FRXCTRLR	FIFO Receive Control Register	4
021B8050	023B8050	025B8050	027B8050	02DB8050	02FB8050	FRXALAR	FIFO Receive Alarm Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B8054	023B8054	025B8054	027B8054	02DB8054	02FB8054	FRXSHR	FIFO Receive Alarm Shutoff Register	4
021B8058	023B8058	025B8058	027B8058	02DB8058	02FB8058	FRXPAR	FIFO Receive Panic Register	4
021B805C	023B805C	025B805C	027B805C	02DB805C	02FB805C	FRXPSR	FIFO Receive Shutoff Register	4
021B8078	023B8078	025B8078	027B8078	02DB8078	02FB8078	FTXSTATR	FIFO Transmit Status Register	4
021B807C	023B807C	025B807C	027B807C	02DB807C	02FB807C	Reserved		
021B808C	023B808C	025B808C	027B808C	02DB808C	02FB808C	FTXTHR	FIFO Transmit Threshold Register	4
021B8094	023B8094	025B8094	027B8094	02DB8094	02FB8094	FTXSPR	FIFO Transmit Space Available Register	4
021B8098	023B8098	025B8098	027B8098	02DB8098	02FB8098	FTXSR	FIFO Transmit Starve Register	4
021B809C	023B809C	025B809C	027B809C	02DB809C	02FB809C	FTXSSR	FIFO Transmit Starve Shutoff Register	4
021B80A0–021B80FC	023B80A0–023B80FC	025B80A0–025B80FC	027B80A0–027B80FC	02DB80A0–02DB80FC	02FB80A0–02FB80FC	Reserved		
021B8100	023B8100	025B8100	027B8100	02DB8100	02FB8100	TCTRL	Transmit Control Register	4
021B8104	023B8104	025B8104	027B8104	02DB8104	02FB8104	TSTAT	Transmit Status Register	4
021B8108	023B8108	025B8108	027B8108	02DB8108	02FB8108	Reserved		
021B810c	023B810c	025B810c	027B810c	02DB810c	02FB810c	TBDLEN	TxBD Data Length	4
021B8110–021B811C	023B8110–023B811C	025B8110–025B811C	027B8110–027B811C	02DB8110–02DB811C	02FB8110–02FB811C	Reserved		
021B8124	023B8124	025B8124	027B8124	02DB8124	02FB8124	CTBPTR	Current TxBD Pointer	4
021B8128–021B8183	023B8128–023B8183	025B8128–025B8183	027B8128–027B8183	02DB8128–02DB8183	02FB8128–02FB8183	Reserved		
021B8184	023B8184	025B8184	027B8184	02DB8184	02FB8184	TBPTR	TxBD Pointer	4
021B8188–021B8203	023B8188–023B8203	025B8188–025B8203	027B8188–027B8203	02DB8188–02DB8203	02FB8188–02FB8203	Reserved		
021B8204	023B8204	025B8204	027B8204	02DB8204	02FB8204	TBASE	Transmit Descriptor Base Address	4
021B8208–021B82AF	023B8208–023B82AF	025B8208–025B82AF	027B8208–027B82AF	02DB8208–02DB82AF	02FB8208–02FB82AF	Reserved		
021B82B0	023B82B0	025B82B0	027B82B0	02DB82B0	02FB82B0	OSTBD	Out-of-sequence TxBD Register	4
021B82B4	023B82B4	025B82B4	027B82B4	02DB82B4	02FB82B4	OSTBDP	Out-of-sequence Tx Data Buffer Pointer Register	4
021B82B8	023B82B8	025B82B8	027B82B8	02DB82B8	02FB82B8	OS32TBDP	Out-of-sequence 32 Bytes Tx Data Buffer Pointer Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B82C0	023B82C0	025B82C0	027B82C0	02DB82C0	02FB82C0	OS32IPTR	Out-of-sequence 32 Bytes TxBD Insert Pointer Register	4
021B82C4	023B82C4	025B82C4	027B82C4	02DB82C4	02FB82C4	OS32TBDR	Out-of-sequence 32 Bytes TxBD Reserved Register	4
021B82C8	023B82C8	025B82C8	027B82C8	02DB82C8	02FB82C8	OS32IIL	Out-of-sequence 32 Bytes TxBD Insert Index/Length Register	4
021B82CC–021B82FF	023B82CC–023B82FF	025B82CC–025B82FF	027B82CC–027B82FF	02DB82CC–02DB82FF	02FB82CC–02FB82FF	Reserved		
021B8300	023B8300	025B8300	027B8300	02DB8300	02FB8300	RCTRL	Receive Control Register	4
021B8304	023B8304	025B8304	027B8304	02DB8304	02FB8304	RSTAT	Receive Status Register	4
021B8308	023B8308	025B8308	027B8308	02DB8308	02FB8308	Reserved		
021B830C	023B830C	025B830C	027B830C	02DB830C	02FB830C	RBDLEN	RxBD Data Length	4
021B8310–021B8323	023B8310–023B8323	025B8310–025B8323	027B8310–027B8323	02DB8310–02DB8323	02FB8310–02FB8323	Reserved		
021B8324	023B8324	025B8324	027B8324	02DB8324	02FB8324	CRBPTRL	Current RxBD Pointer	4
021B8328–021B833F	023B8328–023B833F	025B8328–025B833F	027B8328–027B833F	02DB8328–02DB833F	02FB8328–02FB833F	Reserved		
021B8340	023B8340	025B8340	027B8340	02DB8340	02FB8340	MRBLR0R1	Maximum Receive Buffer Length R0R1 Register	4
021B8344	023B8344	025B8344	027B8344	02DB8344	02FB8344	MRBLR2R3	Maximum Receive Buffer Length R2R3 Register	4
021B8348–021B8383	023B8348–023B8383	025B8348–025B8383	027B8348–027B8383	02DB8348–02DB8383	02FB8348–02FB8383	Reserved		
021B8384	023B8384	025B8384	027B8384	02DB8384	02FB8384	RBPTR0	RxBD Pointer 0	4
021B8388–021B838B	023B8388–023B838B	025B8388–025B838B	027B8388–027B838B	02DB8388–02DB838B	02FB8388–02FB838B	Reserved		
021B838C	023B838C	025B838C	027B838C	02DB838C	02FB838C	RBPTR1	RxBD Pointer 1	4
021B8390–021B8393	023B8390–023B8393	025B8390–025B8393	027B8390–027B8393	02DB8390–02DB8393	02FB8390–02FB8393	Reserved		
021B8394	023B8394	025B8394	027B8394	02DB8394	02FB8394	RBPTR2	RxBD Pointer 2	4
021B8398–021B839B	023B8398–023B839B	025B8398–025B839B	027B8398–027B839B	02DB8398–02DB839B	02FB8398–02FB839B	Reserved		
021B839C	023B839C	025B839C	027B839C	02DB839C	02FB839C	RBPTR3	RxBD Pointer 3	4
021B83A0–021B8403	023B83A0–023B8403	025B83A0–025B8403	027B83A0–027B8403	02DB83A0–02DB8403	02FB83A0–02FB8403	Reserved		
021B8404	023B8404	025B8404	027B8404	02DB8404	02FB8404	RBASE0	Receive Descriptor Base Address 0	4
021B8408–021B840B	023B8408–023B840B	025B8408–025B840B	027B8408–027B840B	02DB8408–02DB840B	02FB8408–02FB840B	Reserved		

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B840C	023B840C	025B840C	027B840C	02DB840C	02FB840C	RBASE1	Receive Descriptor Base Address 1	4
021B8410–021B8413	023B8410–023B8413	025B8410–025B8413	027B8410–027B8413	02DB8410–02DB8413	02FB8410–02FB8413	Reserved		
021B8414	023B8414	025B8414	027B8414	02DB8414	02FB8414	RBASE2	Receive Descriptor Base Address 2	4
021B8418–021B841B	023B8418–023B841B	025B84148 – 025B841B	027B8418–027B841B	02DB8418–02DB841B	02FB8418–02FB841B	Reserved		
021B841C	023B841C	025B841C	027B841C	02DB841C	02FB841C	RBASE3	Receive Descriptor Base Address 3	4
021B8420–021B84FF	023B8420–023B84FF	025B8420–025B84FF	027B8420–027B84FF	02DB8420–02DB84FF	02FB8420–02FB84FF	Reserved		
021B8500	023B8500	025B8500	027B8500	02DB8500	02FB8500	MACCFG1R	MAC Configuration 1 Register	4
021B8504	023B8504	025B8504	027B8504	02DB8504	02FB8504	MACCFG2R	MAC Configuration 2 Register	4
021B8508	023B8508	025B8508	027B8508	02DB8508	02FB8508	IPGIFGIR	Inter Packet Gap/Inter Frame Gap Register	4
021B850C	023B850C	025B850C	027B850C	02DB850C	02FB850C	HAFDUPR	Half-Duplex Register	4
021B8510	023B8510	025B8510	027B8510	02DB8510	02FB8510	MAXFRMR	Maximum Frame Register	4
021B8514–021B851F	023B8514–023B851F	025B8514–025B851F	027B8514–027B851F	02DB8514–02DB851F	02FB8514–02FB851F	Reserved		
021B8520	023B8520	025B8520	027B8520	02DB8520	02FB8520	MIIMCFGR	MII Management Configuration Register	4
021B8524	023B8524	025B8524	027B8524	02DB8524	02FB8524	MIIMCOMR	MII Management Command Register	4
021B8528	023B8528	025B8528	027B8528	02DB8528	02FB8528	MIIMADDR	MII Management Address Register	4
021B852C	023B852C	025B852C	027B852C	02DB852C	02FB852C	MIIMCONR	MII Management Control Register	4
021B8530	023B8530	025B8530	027B8530	02DB8530	02FB8530	MIIMSTATR	MII Management Status Register	4
021B8534	023B8534	025B8534	027B8534	02DB8534	02FB8534	MIIMINDR	MII Management Indicator Register	4
021B8538–021B853B	023B8538–023B853B	025B8538–025B853B	027B8538–027B853B	02DB8538–02DB853B	02FB8538–02FB853B	Reserved		
021B853C	023B853C	025B853C	027B853C	02DB853C	02FB853C	IFSTATR	Interface Status Register	4
021B8540	023B8540	025B8540	027B8540	02DB8540	02FB8540	MACSTADDR 1R	MAC Station Address Part 1 Register	4
021B8544	023B8544	025B8544	027B8544	02DB8544	02FB8544	MACSTADDR 2R	MAC Station Address Part 2 Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B8548–021B867F	023B8548–023B867F	025B8548–025B867F	027B8548–027B867F	02DB8548–02DB867F	02FB8548–02FB867F	Reserved		
021B8680	023B8680	025B8680	027B8680	02DB8680	02FB8680	TR64	Transmit And Receive 64-byte Frame Counter	4
021B8684	023B8684	025B8684	027B8684	02DB8684	02FB8684	TR127	Transmit And Receive 65- To 127-byte Frame Counter	4
021B8688	023B8688	025B8688	027B8688	02DB8688	02FB8688	TR255	Transmit And Receive 128- To 255-byte Frame Counter	4
021B868C	023B868C	025B868C	027B868C	02DB868C	02FB868C	TR511	Transmit And Receive 256- To 511-byte Frame Counter	4
021B8690	023B8690	025B8690	027B8690	02DB8690	02FB8690	TR1K	Transmit And Receive 512- To 1023-byte Frame Counter	4
021B8694	023B8694	025B8694	027B8694	02DB8694	02FB8694	TRMAX	Transmit And Receive 1024- To 1518-byte Frame Counter	4
021B8698	023B8698	025B8698	027B8698	02DB8698	02FB8698	TRMGV	Transmit And Receive 1519- To 1522-byte Good VLAN Frame Count	4
021B869C	023B869C	025B869C	027B869C	02DB869C	02FB869C	RBYT	Receive Byte Counter	4
021B86A0	023B86A0	025B86A0	027B86A0	02DB86A0	02FB86A0	RPKT	Receive Packet Counter	4
021B86A4	023B86A4	025B86A4	027B86A4	02DB86A4	02FB86A4	RFCS	Receive FCS Error Counter	4
021B86A8	023B86A8	025B86A8	027B86A8	02DB86A8	02FB86A8	RMCA	Receive Multicast Packet Counter	4
021B86AC	023B86AC	025B86AC	027B86AC	02DB86AC	02FB86AC	RBCA	Receive Broadcast Packet Counter	4
021B86B0	023B86B0	025B86B0	027B86B0	02DB86B0	02FB86B0	RXCF	Receive Control Frame Packet Counter	4
021B86B4	023B86B4	025B86B4	027B86B4	02DB86B4	02FB86B4	RXPF	Receive PAUSE Frame Packet Counter	4
021B86B8	023B86B8	025B86B8	027B86B8	02DB86B8	02FB86B8	RXUO	Receive Unknown OP Code Counter	4
021B86BC	023B86BC	025B86BC	027B86BC	02DB86BC	02FB86BC	RALN	Receive Alignment Error Counter	4
021B86C0	023B86C0	025B86C0	027B86C0	02DB86C0	02FB86C0	RFLR	Receive Frame Length Error Counter	4
021B86C4	023B86C4	025B86C4	027B86C4	02DB86C4	02FB86C4	RCDE	Receive Code Error Counter	4
021B86C8	023B86C8	025B86C8	027B86C8	02DB86C8	02FB86C8	RCSE	Receive Carrier Sense Error Counter	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B86CC	023B86CC	025B86CC	027B86CC	02DB86CC	02FB86CC	RUND	Receive Undersize Packet Counter	4
021B86D0	023B86D0	025B86D0	027B86D0	02DB86D0	02FB86D0	ROVR	Receive Oversize Packet Counter	4
021B86D4	023B86D4	025B86D4	027B86D4	02DB86D4	02FB86D4	RFRG	Receive Fragments Counter	4
021B86D8	023B86D8	025B86D8	027B86D8	02DB86D8	02FB86D8	RJBR	Receive Jabber Counter	4
021B86DC	023B86DC	025B86DC	027B86DC	02DB86DC	02FB86DC	RDRP	Receive Drop	4
021B86E0	023B86E0	025B86E0	027B86E0	02DB86E0	02FB86E0	TBYT	Transmit Byte Counter	4
021B86E4	023B86E4	025B86E4	027B86E4	02DB86E4	02FB86E4	TPKT	Transmit Packet Counter	4
021B86E8	023B86E8	025B86E8	027B86E8	02DB86E8	02FB86E8	TMCA	Transmit Multicast Packet Counter	4
021B86EC	023B86EC	025B86EC	027B86EC	02DB86EC	02FB86EC	TBCA	Transmit Broadcast Packet Counter	4
021B86F0	023B86F0	025B86F0	027B86F0	02DB86F0	02FB86F0	TXPF	Transmit PAUSE Control Frame Counter	4
021B86F4	023B86F4	025B86F4	027B86F4	02DB86F4	02FB86F4	TDFR	Transmit Deferral Packet Counter	4
021B86F8	023B86F8	025B86F8	027B86F8	02DB86F8	02FB86F8	TEDF	Transmit Excessive Deferral Packet Counter	4
021B86FC	023B86FC	025B86FC	027B86FC	02DB86FC	02FB86FC	TSCL	Transmit Single Collision Packet Counter	4
021B8700	023B8700	025B8700	027B8700	02DB8700	02FB8700	TMCL	Transmit Multiple Collision Packet Counter	4
021B8704	023B8704	025B8704	027B8704	02DB8704	02FB8704	TLCL	Transmit Late Collision Packet Counter	4
021B8708	023B8708	025B8708	027B8708	02DB8708	02FB8708	TXCL	Transmit Excessive Collision Packet Counter	4
021B870C	023B870C	025B870C	027B870C	02DB870C	02FB870C	TNCL	Transmit Total Collision Counter	4
021B8714	023B8714	025B8714	027B8714	02DB8714	02FB8714	reserved		
021B8718	023B8718	025B8718	027B8718	02DB8718	02FB8718	TJBR	Transmit Jabber Frame Counter	4
021B871C	023B871C	025B871C	027B871C	02DB871C	02FB871C	TFCS	Transmit FCS Error Counter	4
021B8720	023B8720	025B8720	027B8720	02DB8720	02FB8720	TXCF	Transmit Control Frame Counter	4
021B8724	023B8724	025B8724	027B8724	02DB8724	02FB8724	TOVR	Transmit Oversize Frame Counter	4
021B8728	023B8728	025B8728	027B8728	02DB8728	02FB8728	TUND	Transmit Undersize Frame Counter	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B872C	023B872C	025B872C	027B872C	02DB872C	02FB872C	TFRG	Transmit Fragments Frame Counter	4
021B8730	023B8730	025B8730	027B8730	02DB8730	02FB8730	CAR1	Carry Register One	4
021B8734	023B8734	025B8734	027B8734	02DB8734	02FB8734	CAR2	Carry Register Two	4
021B8738	023B8738	025B8738	027B8738	02DB8738	02FB8738	CAM1	Carry Register One Mask	4
021B873C	023B873C	025B873C	027B873C	02DB873C	02FB873C	CAM2	Carry Register Two Mask	4
021B8740–021B87FF	023B8740–023B87FF	025B8740–025B87FF	027B8740–027B87FF	02DB8740–02DB87FF	02FB8740–02FB87FF	Reserved		
021B8800	023B8800	025B8800	027B8800	02DB8800	02FB8800	IADDR0	Individual Address Register 0	4
021B8804	023B8804	025B8804	027B8804	02DB8804	02FB8804	IADDR0	Individual Address Register 1	4
021B8808	023B8808	025B8808	027B8808	02DB8808	02FB8808	IADDR2	Individual Address Register 2	4
021B880C	023B880C	025B880C	027B880C	02DB880C	02FB880C	IADDR3	Individual Address Register 3	4
021B8810	023B8810	025B8810	027B8810	02DB8810	02FB8810	IADDR4	Individual Address Register 4	4
021B8814	023B8814	025B8814	027B8814	02DB8814	02FB8814	IADDR5	Individual Address Register 5	4
021B8818	023B8818	025B8818	027B8818	02DB8818	02FB8818	IADDR6	Individual Address Register 6	4
021B881C	023B881C	025B881C	027B881C	02DB881C	02FB881C	IADDR7	Individual Address Register 7	4
021B8820–021B887F	023B8820–023B887F	025B8820–025B887F	027B8820–027B887F	02DB8820–02DB887F	02FB8820–02FB887F	Reserved		
021B8880	023B8880	025B8880	027B8880	02DB8880	02FB8880	GADDR0	Group Address Register 0	4
021B8884	023B8884	025B8884	027B8884	02DB8884	02FB8884	GADDR1	Group Address Register 1	4
021B8888	023B8888	025B8888	027B8888	02DB8888	02FB8888	GADDR2	Group Address Register 2	4
021B888C	023B888C	025B888C	027B888C	02DB888C	02FB888C	GADDR3	Group Address Register 3	4
021B8890	023B8890	025B8890	027B8890	02DB8890	02FB8890	GADDR4	Group Address Register 4	4
021B8894	023B8894	025B8894	027B8894	02DB8894	02FB8894	GADDR5	Group Address Register 5	4
021B8898	023B8898	025B8898	027B8898	02DB8898	02FB8898	GADDR6	Group Address Register 6	4
021B889C	023B889C	025B889C	027B889C	02DB889C	02FB889C	GADDR7	Group Address Register 7	4
021B88A0–021B88FF	023B88A0–023B88FF	025B88A0–025B88FF	027B88A0–027B88FF	02DB88A0–02DB88FF	02FB88A0–02FB88FF	Reserved		
021B8900	023B8900	025B8900	027B8900	02DB8900	02FB8900	PMD0	Pattern Match Data 0	4
021B8904	023B8904	025B8904	027B8904	02DB8904	02FB8904	Reserved		
021B8908	023B8908	025B8908	027B8908	02DB8908	02FB8908	PMASK0	Pattern Mask 0 Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B890C	023B890C	025B890C	027B890C	02DB890C	02FB890C	Reserved		
021B8910	023B8910	025B8910	027B8910	02DB8910	02FB8910	PCNTRL0	Pattern Control 0 Register	4
021B8914	023B8914	025B8914	027B8914	02DB8914	02FB8914	Reserved		
021B8918	023B8918	025B8918	027B8918	02DB8918	02FB8918	PATTRB0	Pattern Attributes 0 Register	4
021B8920	023B8920	025B8920	027B8920	02DB8920	02FB8920	PMD1	Pattern Match Data 1	4
021B8924	023B8924	025B8924	027B8924	02DB8924	02FB8924	Reserved		
021B8928	023B8928	025B8928	027B8928	02DB8928	02FB8928	PMASK1	Pattern Mask 1 Register	4
021B892C	023B892C	025B892C	027B892C	02DB892C	02FB892C	Reserved		
021B8930	023B8930	025B8930	027B8930	02DB8930	02FB8930	PCNTRL1	Pattern Control 1 Register	4
021B8934	023B8934	025B8934	027B8934	02DB8934	02FB8934	Reserved		
021B8938	023B8938	025B8938	027B8938	02DB8938	02FB8938	PATTRB1	Pattern Attributes 1 Register	4
021B8940	023B8940	025B8940	027B8940	02DB8940	02FB8940	PMD2	Pattern Match Data 2	4
021B8944	023B8944	025B8944	027B8944	02DB8944	02FB8944	Reserved		
021B8948	023B8948	025B8948	027B8948	02DB8948	02FB8948	PMASK2	Pattern Mask 2 Register	4
021B894C	023B894C	025B894C	027B894C	02DB894C	02FB894C	Reserved		
021B8950	023B8950	025B8950	027B8950	02DB8950	02FB8950	PCNTRL2	Pattern Control 2 Register	4
021B8954	023B8954	025B8954	027B8954	02DB8954	02FB8954	Reserved		
021B8958	023B8958	025B8958	027B8958	02DB8958	02FB8958	PATTRB2	Pattern Attributes 2 Register	4
021B8960	023B8960	025B8960	027B8960	02DB8960	02FB8960	PMD3	Pattern Match Data 3	4
021B8964	023B8964	025B8964	027B8964	02DB8964	02FB8964	Reserved		
021B8968	023B8968	025B8968	027B8968	02DB8968	02FB8968	PMASK3	Pattern Mask 3 Register	4
021B896C	023B896C	025B896C	027B896C	02DB896C	02FB896C	Reserved		
021B8970	023B8970	025B8970	027B8970	02DB8970	02FB8970	PCNTRL3	Pattern Control 3 Register	4
021B8974	023B8974	025B8974	027B8974	02DB8974	02FB8974	Reserved		
021B8978	023B8978	025B8978	027B8978	02DB8978	02FB8978	PATTRB3	Pattern Attributes 3 Register	4
021B8980	023B8980	025B8980	027B8980	02DB8980	02FB8980	PMD4	Pattern Match Data 4	4
021B8984	023B8984	025B8984	027B8984	02DB8984	02FB8984	Reserved		
021B8988	023B8988	025B8988	027B8988	02DB8988	02FB8988	PMASK4	Pattern Mask 4 Register	4
021B898C	023B898C	025B898C	027B898C	02DB898C	02FB898C	Reserved		
021B8990	023B8990	025B8990	027B8990	02DB8990	02FB8990	PCNTRL4	Pattern Control 4 Register	4
021B8994	023B8994	025B8994	027B8994	02DB8994	02FB8994	Reserved		
021B8998	023B8998	025B8998	027B8998	02DB8998	02FB8998	PATTRB4	Pattern Attributes 4 Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B89A0	023B89A0	025B89A0	027B89A0	02DB89A0	02FB89A0	PMD5	Pattern Match Data 5	4
021B89A4	023B89A4	025B89A4	027B89A4	02DB89A4	02FB89A4	Reserved		
021B89A8	023B89A8	025B89A8	027B89A8	02DB89A8	02FB89A8	PMASK5	Pattern Mask 5 Register	4
021B89AC	023B89AC	025B89AC	027B89AC	02DB89AC	02FB89AC	Reserved		
021B89B0	023B89B0	025B89B0	027B89B0	02DB89B0	02FB89B0	PCNTRL5	Pattern Control 5 Register	4
021B89B4	023B89B4	025B89B4	027B89B4	02DB89B4	02FB89B4	Reserved		
021B89B8	023B89B8	025B89B8	027B89B8	02DB89B8	02FB89B8	PATTRB5	Pattern Attributes 5 Register	4
021B89C0	023B89C0	025B89C0	027B89C0	02DB89C0	02FB89C0	PMD6	Pattern Match Data 6	4
021B89C4	023B89C4	025B89C4	027B89C4	02DB89C4	02FB89C4	Reserved		
021B89C8	023B89C8	025B89C8	027B89C8	02DB89C8	02FB89C8	PMASK6	Pattern Mask 6 Register	4
021B89CC	023B89CC	025B89CC	027B89CC	02DB89CC	02FB89CC	Reserved		
021B89D0	023B89D0	025B89D0	027B89D0	02DB89D0	02FB89D0	PCNTRL6	Pattern Control 6 Register	4
021B89D4	023B89D4	025B89D4	027B89D4	02DB89D4	02FB89D4	Reserved		
021B89D8	023B89D8	025B89D8	027B89D8	02DB89D8	02FB89D8	PATTRB6	Pattern Attributes 6 Register	4
021B89E0	023B89E0	025B89E0	027B89E0	02DB89E0	02FB89E0	PMD7	Pattern Match Data 7	4
021B89E4	023B89E4	025B89E4	027B89E4	02DB89E4	02FB89E4	Reserved		
021B89E8	023B89E8	025B89E8	027B89E8	02DB89E8	02FB89E8	PMASK7	Pattern Mask 7 Register	4
021B89EC	023B89EC	025B89EC	027B89EC	02DB89EC	02FB89EC	Reserved		
021B89F0	023B89F0	025B89F0	027B89F0	02DB89F0	02FB89F0	PCNTRL7	Pattern Control 7 Register	4
021B89F4	023B89F4	025B89F4	027B89F4	02DB89F4	02FB89F4	Reserved		
021B89F8	023B89F8	025B89F8	027B89F8	02DB89F8	02FB89F8	PATTRB7	Pattern Attributes 7 Register	4
021B8A00	023B8A00	025B8A00	027B8A00	02DB8A00	02FB8A00	PMD8	Pattern Match Data 8	4
021B8A04	023B8A04	025B8A04	027B8A04	02DB8A04	02FB8A04	Reserved		
021B8A08	023B8A08	025B8A08	027B8A08	02DB8A08	02FB8A08	PMASK8	Pattern Mask 8 Register	4
021B8A0C	023B8A0C	025B8A0C	027B8A0C	02DB8A0C	02FB8A0C	Reserved		
021B8A10	023B8A10	025B8A10	027B8A10	02DB8A10	02FB8A10	PCNTRL8	Pattern Control 8 Register	4
021B8A14	023B8A14	025B8A14	027B8A14	02DB8A14	02FB8A14	Reserved		
021B8A18	023B8A18	025B8A18	027B8A18	02DB8A18	02FB8A18	PATTRB8	Pattern Attributes 8 Register	4
021B8A20	023B8A20	025B8A20	027B8A20	02DB8A20	02FB8A20	PMD9	Pattern Match Data 9	4
021B8A24	023B8A24	025B8A24	027B8A24	02DB8A24	02FB8A24	Reserved		
021B8A28	023B8A28	025B8A28	027B8A28	02DB8A28	02FB8A28	PMASK9	Pattern Mask 9 Register	4
021B8A2C	023B8A2C	025B8A2C	027B8A2C	02DB8A2C	02FB8A2C	Reserved		
021B8A30	023B8A30	025B8A30	027B8A30	02DB8A30	02FB8A30	PCNTRL9	Pattern Control 9 Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B8A34	023B8A34	025B8A34	027B8A34	02DB8A34	02FB8A34	Reserved		
021B8A38	023B8A38	025B8A38	027B8A38	02DB8A38	02FB8A38	PATTRB9	Pattern Attributes 9 Register	4
021B8A40	023B8A40	025B8A40	027B8A40	02DB8A40	02FB8A40	PMD10	Pattern Match Data 10	4
021B8A44	023B8A44	025B8A44	027B8A44	02DB8A44	02FB8A44	Reserved		
021B8A48	023B8A48	025B8A48	027B8A48	02DB8A48	02FB8A48	PMASK10	Pattern Mask 10 Register	4
021B8A4C	023B8A4C	025B8A4C	027B8A4C	02DB8A4C	02FB8A4C	Reserved		
021B8A50	023B8A50	025B8A50	027B8A50	02DB8A50	02FB8A50	PCNTRL10	Pattern Control 10 Register	4
021B8A54	023B8A54	025B8A54	027B8A54	02DB8A54	02FB8A54	Reserved		
021B8A58	023B8A58	025B8A58	027B8A58	02DB8A58	02FB8A58	PATTRB10	Pattern Attributes 10 Register	
021B8A60	023B8A60	025B8A60	027B8A60	02DB8A60	02FB8A60	PMD11	Pattern Match Data 11	4
021B8A64	023B8A64	025B8A64	027B8A64	02DB8A64	02FB8A64	Reserved		
021B8A68	023B8A68	025B8A68	027B8A68	02DB8A68	02FB8A68	PMASK11	Pattern Mask 11 Register	4
021B8A6C	023B8A6C	025B8A6C	027B8A6C	02DB8A6C	02FB8A6C	Reserved		
021B8A70	023B8A70	025B8A70	027B8A70	02DB8A70	02FB8A70	PCNTRL11	Pattern Control 11 Register	4
021B8A74	023B8A74	025B8A74	027B8A74	02DB8A74	02FB8A74	Reserved		
021B8A78	023B8A78	025B8A78	027B8A78	02DB8A78	02FB8A78	PATTRB11	Pattern Attributes 11 Register	4
021B8A80	023B8A80	025B8A80	027B8A80	02DB8A80	02FB8A80	PMD12	Pattern Match Data 12	4
021B8A84	023B8A84	025B8A84	027B8A84	02DB8A84	02FB8A84	Reserved		
021B8A88	023B8A88	025B8A88	027B8A88	02DB8A88	02FB8A88	PMASK12	Pattern Mask 12 Register	4
021B8A8C	023B8A8C	025B8A8C	027B8A8C	02DB8A8C	02FB8A8C	Reserved		
021B8A90	023B8A90	025B8A90	027B8A90	02DB8A90	02FB8A90	PCNTRL12	Pattern Control 12 Register	4
021B8A94	023B8A94	025B8A94	027B8A94	02DB8A94	02FB8A94	Reserved		
021B8A98	023B8A98	025B8A98	027B8A98	02DB8A98	02FB8A98	PATTRB12	Pattern Attributes 12 Register	4
021B8AA0	023B8AA0	025B8AA0	027B8AA0	02DB8AA0	02FB8AA0	PMD13	Pattern Match Data 13	4
021B8AA4	023B8AA4	025B8AA4	027B8AA4	02DB8AA4	02FB8AA4	Reserved		
021B8AA8	023B8AA8	025B8AA8	027B8AA8	02DB8AA8	02FB8AA8	PMASK13	Pattern Mask 13 Register	4
021B8AAC	023B8AAC	025B8AAC	027B8AAC	02DB8AAC	02FB8AAC	Reserved		
021B8AB0	023B8AB0	025B8AB0	027B8AB0	02DB8AB0	02FB8AB0	PCNTRL13	Pattern Control 13 Register	4
021B8AB4	023B8AB4	025B8AB4	027B8AB4	02DB8AB4	02FB8AB4	Reserved		

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B8AB8	023B8AB8	025B8AB8	027B8AB8	02DB8AB8	02FB8AB8	PATTRB13	Pattern Attributes 13 Register	4
021B8AC0	023B8AC0	025B8AC0	027B8AC0	02DB8AC0	02FB8AC0	PMD14	Pattern Match Data 14	4
021B8AC4	023B8AC4	025B8AC4	027B8AC4	02DB8AC4	02FB8AC4	Reserved		
021B8AC8	023B8AC8	025B8AC8	027B8AC8	02DB8AC8	02FB8AC8	PMASK14	Pattern Mask 14 Register	4
021B8ACC	023B8ACC	025B8ACC	027B8ACC	02DB8ACC	02FB8ACC	Reserved		
021B8AD0	023B8AD0	025B8AD0	027B8AD0	02DB8AD0	02FB8AD0	PCNTRL14	Pattern Control 14 Register	4
021B8AD4	023B8AD4	025B8AD4	027B8AD4	02DB8AD4	02FB8AD4	Reserved		
021B8AD8	023B8AD8	025B8AD8	027B8AD8	02DB8AD8	02FB8AD8	PATTRB14	Pattern Attributes 14 Register	4
021B8AE0	023B8AE0	025B8AE0	027B8AE0	02DB8AE0	02FB8AE0	PMD15	Pattern Match Data 15	4
021B8AE4	023B8AE4	025B8AE4	027B8AE4	02DB8AE4	02FB8AE4	Reserved		
021B8AE8	023B8AE8	025B8AE8	027B8AE8	02DB8AE8	02FB8AE8	PMASK15	Pattern Mask 15 Register	4
021B8AEC	023B8AEC	025B8AEC	027B8AEC	02DB8AEC	02FB8AEC	Reserved		
021B8AF0	023B8AF0	025B8AF0	027B8AF0	02DB8AF0	02FB8AF0	PCNTRL15	Pattern Control 15 Register	4
021B8AF4	023B8AF4	025B8AF4	027B8AF4	02DB8AF4	02FB8AF4	Reserved		
021B8AF8	023B8AF8	025B8AF8	027B8AF8	02DB8AF8	02FB8AF8	PATTRB15	Pattern Attributes 15 Register	4
021B8B00–021B8BF4	023B8B00–023B8BF4	025B8B00–025B8BF4	027B8B00–027B8BF4	02DB8B00–02DB8BF4	02FB8B00–02FB8BF4	Reserved		
021B8BF8	023B8BF8	025B8BF8	027B8BF8	02DB8BF8	02FB8BF8	DATTR	Default Attribute Register	4
021B8C00–021B8FFF	023B8C00–023B8FFF	025B8C00–025B8FFF	027B8C00–027B8FFF	02DB8C00–02DB8FFF	02FB8C00–02FB8FFF	Reserved		
021B9000	023B9000	025B9000	027B9000	02DB9000	02FB9000	MIIGSK_ CFGR	MIIGSK Configuration Register	4
021B9004	023B9004	025B9004	027B9004	02DB9004	02FB9004	MIIGSK_GPR	MIIGSK General-Purpose Register	4
021B9008	023B9008	025B9008	027B9008	02DB9008	02FB9008	MIIGSK_ENR	MIIGSK Enable Register	4
021B900C	023B900C	025B900C	027B900C	02DB900C	02FB900C	MIIGSK_SMII _SYNCDIR	MIIGSK SMII SYNC Direction Register	4
021B9010	023B9010	025B9010	027B9010	02DB9010	02FB9010	MIIGSK_ TIFBR	MIIGSK Transmit Inter-Frame Bits Register	4
021B9014	023B9014	025B9014	027B9014	02DB9014	02FB9014	MIIGSK_ RIFBR	MIIGSK Receive Inter-Frame Bits Register	4
021B9018	023B9018	025B9018	027B9018	02DB9018	02FB9018	MIIGSK_ ERIFBR	MIIGSK Expected Receive Inter-Frame Bits Register	4
021B901C	023B901C	025B901C	027B901C	02DB901C	02FB901C	MIIGSK_ IEVENT	MIIGSK SMII Interrupt Event Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021B9020	023B9020	025B9020	027B9020	02DB9020	02FB9020	MIIGSK_ IMASK	MIIGSK SMII Interrupt Mask Register	4
021B9024–021BAFFF	023B9100–023BAFFF	025B9100–025BAFFF	027B9100–027BAFFF	02DB9100–02DBAFFF	02FB9100–02FBAFFF	Reserved		
021BB000	023BB000	025BB000	027BB000	02DBB000	02FBB000	SCR	Stop Control Register	4
021BB008	023BB008	025BB008	027BB008	02DBB008	02FBB008	SASR	Stop Acknowledge Status Register	4
021BC000	023BC000	025BC000	027BC000	02DBC000	02FBC000	VIGR	Virtual Interrupt Generation Register	4
021BC008	023BC008	025BC008	027BC008	02DBC008	02FBC008	VISR	Virtual Interrupt Status Register	4
021BC010	023BC010	025BC010	027BC010	02DBC010	02FBC010	VNMIGR	Virtual NMI Generation Register	4
021BC018	023BC018	025BC018	027BC018	02DBC018	02FBC018	GICR	GIC Interrupt Configuration Register	4
021BC020	023BC020	025BC020	027BC020	02DBC020	02FBC020	GEIER	GIC External Interrupt Enable Register	4
021BC028	023BC028	025BC028	027BC028	02DBC028	02FBC028	GCIER	GIC Core Interrupt Enable Register	4
021BC030	023BC030	025BC030	027BC030	02DBC030	02FBC030	GISR	GIC Interrupt Status Register	4
021BC038–021BC0FF	023BC038–023BC0FF	025BC038–025BC0FF	027BC038–027BC0FF	02DBC038–02DBC0FF	02FBC038–02FBC0FF	Reserved		
021BC100	023BC100	025BC100	027BC100	02DBC100	02FBC100	HSMPR0	Hardware Semaphore Register 0	4
021BC108	023BC108	025BC108	027BC108	02DBC108	02FBC108	HSMPR1	Hardware Semaphore Register 1	4
021BC110	023BC110	025BC110	027BC110	02DBC110	02FBC110	HSMPR2	Hardware Semaphore Register 2	4
021BC118	023BC118	025BC118	027BC118	02DBC118	02FBC118	HSMPR3	Hardware Semaphore Register 3	4
021BC120	023BC120	025BC120	027BC120	02DBC120	02FBC120	HSMPR4	Hardware Semaphore Register 4	4
021BC128	023BC128	025BC128	027BC128	02DBC128	02FBC128	HSMPR5	Hardware Semaphore Register 5	4
021BC130	023BC130	025BC130	027BC130	02DBC130	02FBC130	HSMPR6	Hardware Semaphore Register 6	4
021BC138	023BC138	025BC138	027BC138	02DBC138	02FBC138	HSMPR7	Hardware Semaphore Register 7	4
021BC140–021BC1FF	023BC140–023BC1FF	025BC140–025BC1FF	027BC140–027BC1FF	02DBC140–02DBC1FF	02FBC140–02FBC1FF	Reserved		
021BC200	023BC200	025BC200	027BC200	02DBC200	02FBC200	PODR	Pin Open-Drain Register	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BC208	023BC208	025BC208	027BC208	02DBC208	02FBC208	PDAT	Pin Data Register	4
021BC210	023BC210	025BC210	027BC210	02DBC210	02FBC210	PDIR	Pin Direction Register	4
021BC218	023BC218	025BC218	027BC218	02DBC218	02FBC218	PAR	Pin Assignment Register	4
021BC220	023BC220	025BC220	027BC220	02DBC220	02FBC220	PSOR	Pin Special Options Register	4
021BC228–021BCFFF	023BC228–023BCFFF	025BC228–025BCFFF	027BC228–027BCFFF	02DBC228–02DBCFFF	02FBC228–02FBCFFF	Reserved		
021BD000	023BD000	025BD000	027BD000	02DBD000	02FBD000	SCIBR	SCI Baud Rate Register	4
021BD008	023BD008	025BD008	027BD008	02DBD008	02FBD008	SCICR	SCI Control Register	4
021BD010	023BD010	025BD010	027BD010	02DBD010	02FBD010	SCISR	SCI Status Register	4
021BD018	023BD018	025BD018	027BD018	02DBD018	02FBD018	SCIDR	SCI Data Register	4
021BD020–021BD027	023BD020–023BD027	025BD020–025BD027	027BD020–027BD027	02DBD020–02DBD027	02FBD020–02FBD027	Reserved		
021BD028	023BD028	025BD028	027BD028	02DBD028	02FBD028	SCIDDR	SCI Data Direction Register	4
021BD030–021BDFFF	023BD030–023BDFFF	025BD030–025BDFFF	027BD030–027BDFFF	02DBD030–02DBDFFF	02FBD030–02FBDFFF	Reserved		
021BE000	023BE000	025BE000	027BE000	02DBE000	02FBE000	DCR	DSI Control Register	4
021BE008	023BE008	025BE008	027BE008	02DBE008	02FBE008	DSWBAR	DSI Sliding Window Base Address Register	4
021BE010	023BE010	025BE010	027BE010	02DBE010	02FBE010	DIBAR9	DSI Internal Base Address Register Bank 9	4
021BE018–021BE01F	023BE018–023BE01F	025BE018–025BE01F	027BE018–027BE01F	02DBE018–02DBE01F	02FBE018–02FBE01F	Reserved		
021BE020	023BE020	025BE020	027BE020	02DBE020	02FBE020	DIBAR11	DSI Internal Base Address Register Bank 11	4
021BE028	023BE028	025BE028	027BE028	02DBE028	02FBE028	DIAMR9	DSI Internal Address Mask Register Bank 9	4
021BE030–021BE037	023BE030–023BE037	025BE030–025BE037	027BE030–027BE037	02DBE030–02DBE037	02FBE030–02FBE037	Reserved		
021BE038	023BE038	025BE038	027BE038	02DBE038	02FBE038	DIAMR11	DSI Internal Address Mask Register Bank 11	4
021BE040	023BE040	025BE040	027BE040	02DBE040	02FBE040	DCIR	DSI Chip ID Register	4
021BE048	023BE048	025BE048	027BE048	02DBE048	02FBE048	DDR	DSI Disable Register	4
021BE050–021BE05F	023BE050–023BE05F	025BE050–025BE05F	027BE050–027BE05F	02DBE050–02DBE05F	02FBE050–02FBE05F	Reserved		
021BE060	023BE060	025BE060	027BE060	02DBE060	02FBE060	DEXTBAR	DSI External Sliding Window Base Address Register	4
021BE068–021BE7FF	023BE068–023BE7FF	025BE068–025BE7FF	027BE068–027BE7FF	02DBE068–02DBE7FF	02FBE068–02FBE7FF	Reserved		

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BE800	023BE800	025BE800	027BE800	02DBE800	02FBE800	DSR	DSI Status Register	4
021BE808	023BE808	025BE808	027BE808	02DBE808	02FBE808	DER	DSI Error Register	4
021BE810–021BEFFF	023BE810–023BEFFF	025BE810–025BEFFF	027BE810–027BEFFF	02DBE810–02DBEFFF	02FBE810–02FBEFFF	Reserved		
021BF000	023BF000	025BF000	027BF000	02DBF000	02FBF000	TCFRA0	Timer Configuration Register of Timer A0	4
021BF008	023BF008	025BF008	027BF008	02DBF008	02FBF008	TCFRA1	Timer Configuration Register of Timer A1	4
021BF010	023BF010	025BF010	027BF010	02DBF010	02FBF010	TCFRA2	Timer Configuration Register of Timer A2	4
021BF018	023BF018	025BF018	027BF018	02DBF018	02FBF018	TCFRA3	Timer Configuration Register of Timer A3	4
021BF020	023BF020	025BF020	027BF020	02DBF020	02FBF020	TCFRA4	Timer Configuration Register of Timer A4	4
021BF028	023BF028	025BF028	027BF028	02DBF028	02FBF028	TCFRA5	Timer Configuration Register of Timer A5	4
021BF030	023BF030	025BF030	027BF030	02DBF030	02FBF030	TCFRA6	Timer Configuration Register of Timer A6	4
021BF038	023BF038	025BF038	027BF038	02DBF038	02FBF038	TCFRA7	Timer Configuration Register of Timer A7	4
021BF040	023BF040	025BF040	027BF040	02DBF040	02FBF040	TCFRA8	Timer Configuration Register of Timer A8	4
021BF048	023BF048	025BF048	027BF048	02DBF048	02FBF048	TCFRA9	Timer Configuration Register of Timer A9	4
021BF050	023BF050	025BF050	027BF050	02DBF050	02FBF050	TCFRA10	Timer Configuration Register of Timer A10	4
021BF058	023BF058	025BF058	027BF058	02DBF058	02FBF058	TCFRA11	Timer Configuration Register of Timer A11	4
021BF060	023BF060	025BF060	027BF060	02DBF060	02FBF060	TCFRA12	Timer Configuration Register of Timer A12	4
021BF068	023BF068	025BF068	027BF068	02DBF068	02FBF068	TCFRA13	Timer Configuration Register of Timer A13	4
021BF070	023BF070	025BF070	027BF070	02DBF070	02FBF070	TCFRA14	Timer Configuration Register of Timer A14	4
021BF078	023BF078	025BF078	027BF078	02DBF078	02FBF078	TCFRA15	Timer Configuration Register of Timer A15	4
021BF080	023BF080	025BF080	027BF080	02DBF080	02FBF080	TCMPA0	Timer Compare Register of Timer A0	4
021BF088	023BF088	025BF088	027BF088	02DBF088	02FBF088	TCMPA1	Timer Compare Register of Timer A1	4
021BF090	023BF090	025BF090	027BF090	02DBF090	02FBF090	TCMPA2	Timer Compare Register of Timer A2	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BF098	023BF098	025BF098	027BF098	02DBF098	02FBF098	TCMPA3	Timer Compare Register of Timer A3	4
021BF0A0	023BF0A0	025BF0A0	027BF0A0	02DBF0A0	02FBF0A0	TCMPA4	Timer Compare Register of Timer A4	4
021BF0A8	023BF0A8	025BF0A8	027BF0A8	02DBF0A8	02FBF0A8	TCMPA5	Timer Compare Register of Timer A5	4
021BF0B0	023BF0B0	025BF0B0	027BF0B0	02DBF0B0	02FBF0B0	TCMPA6	Timer Compare Register of Timer A6	4
021BF0B8	023BF0B8	025BF0B8	027BF0B8	02DBF0B8	02FBF0B8	TCMPA7	Timer Compare Register of Timer A7	4
021BF0C0	023BF0C0	025BF0C0	027BF0C0	02DBF0C0	02FBF0C0	TCMPA8	Timer Compare Register of Timer A8	4
021BF0C8	023BF0C8	025BF0C8	027BF0C8	02DBF0C8	02FBF0C8	TCMPA9	Timer Compare Register of Timer A9	4
021BF0D0	023BF0D0	025BF0D0	027BF0D0	02DBF0D0	02FBF0D0	TCMPA10	Timer Compare Register of Timer A10	4
021BF0D8	023BF0D8	025BF0D8	027BF0D8	02DBF0D8	02FBF0D8	TCMPA11	Timer Compare Register of Timer A11	4
021BF0E0	023BF0E0	025BF0E0	027BF0E0	02DBF0E0	02FBF0E0	TCMPA12	Timer Compare Register of Timer A12	4
021BF0E8	023BF0E8	025BF0E8	027BF0E8	02DBF0E8	02FBF0E8	TCMPA13	Timer Compare Register of Timer A13	4
021BF0F0	023BF0F0	025BF0F0	027BF0F0	02DBF0F0	02FBF0F0	TCMPA14	Timer Compare Register of Timer A14	4
021BF0F8	023BF0F8	025BF0F8	027BF0F8	02DBF0F8	02FBF0F8	TCMPA15	Timer Compare Register of Timer A15	4
021BF100	023BF100	025BF100	027BF100	02DBF100	02FBF100	TCRA0	Timer Control Register of Timer A0	4
021BF108	023BF108	025BF108	027BF108	02DBF108	02FBF108	TCRA1	Timer Control Register of Timer A1	4
021BF110	023BF110	025BF110	027BF110	02DBF110	02FBF110	TCRA2	Timer Control Register of Timer A2	4
021BF118	023BF118	025BF118	027BF118	02DBF118	02FBF118	TCRA3	Timer Control Register of Timer A3	4
021BF120	023BF120	025BF120	027BF120	02DBF120	02FBF120	TCRA4	Timer Control Register of Timer A4	4
021BF128	023BF128	025BF128	027BF128	02DBF128	02FBF128	TCRA5	Timer Control Register of Timer A5	4
021BF130	023BF130	025BF130	027BF130	02DBF130	02FBF130	TCRA6	Timer Control Register of Timer A6	4
021BF138	023BF138	025BF138	027BF138	02DBF138	02FBF138	TCRA7	Timer Control Register of Timer A7	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BF140	023BF140	025BF140	027BF140	02DBF140	02FBF140	TCRA8	Timer Control Register of Timer A8	4
021BF148	023BF148	025BF148	027BF148	02DBF148	02FBF148	TCRA9	Timer Control Register of Timer A9	4
021BF150	023BF150	025BF150	027BF150	02DBF150	02FBF150	TCRA10	Timer Control Register of Timer A10	4
021BF158	023BF158	025BF158	027BF158	02DBF158	02FBF158	TCRA11	Timer Control Register of Timer A11	4
021BF160	023BF160	025BF160	027BF160	02DBF160	02FBF160	TCRA12	Timer Control Register of Timer A12	4
021BF168	023BF168	025BF168	027BF168	02DBF168	02FBF168	TCRA13	Timer Control Register of Timer A13	4
021BF170	023BF170	025BF170	027BF170	02DBF170	02FBF170	TCRA14	Timer Control Register of Timer A14	4
021BF178	023BF178	025BF178	027BF178	02DBF178	02FBF178	TCRA15	Timer Control Register of Timer A15	4
021BF180	023BF180	025BF180	027BF180	02DBF180	02FBF180	TCNRA0	Timer Count Register of Timer A0	4
021BF188	023BF188	025BF188	027BF188	02DBF188	02FBF188	TCNRA1	Timer Count Register of Timer A1	4
021BF190	023BF190	025BF190	027BF190	02DBF190	02FBF190	TCNRA2	Timer Count Register of Timer A2	4
021BF198	023BF198	025BF198	027BF198	02DBF198	02FBF198	TCNRA3	Timer Count Register of Timer A3	4
021BF1A0	023BF1A0	025BF1A0	027BF1A0	02DBF1A0	02FBF1A0	TCNRA4	Timer Count Register of Timer A4	4
021BF1A8	023BF1A8	025BF1A8	027BF1A8	02DBF1A8	02FBF1A8	TCNRA5	Timer Count Register of Timer A5	4
021BF1B0	023BF1B0	025BF1B0	027BF1B0	02DBF1B0	02FBF1B0	TCNRA6	Timer Count Register of Timer A6	4
021BF1B8	023BF1B8	025BF1B8	027BF1B8	02DBF1B8	02FBF1B8	TCNRA7	Timer Count Register of Timer A7	4
021BF1C0	023BF1C0	025BF1C0	027BF1C0	02DBF1C0	02FBF1C0	TCNRA8	Timer Count Register of Timer A8	4
021BF1C8	023BF1C8	025BF1C8	027BF1C8	02DBF1C8	02FBF1C8	TCNRA9	Timer Count Register of Timer A9	4
021BF1D0	023BF1D0	025BF1D0	027BF1D0	02DBF1D0	02FBF1D0	TCNRA10	Timer Count Register of Timer A10	4
021BF1D8	023BF1D8	025BF1D8	027BF1D8	02DBF1D8	02FBF1D8	TCNRA11	Timer Count Register of Timer A11	4
021BF1E0	023BF1E0	025BF1E0	027BF1E0	02DBF1E0	02FBF1E0	TCNRA12	Timer Count Register of Timer A12	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BF1E8	023BF1E8	025BF1E8	027BF1E8	02DBF1E8	02FBF1E8	TCNRA13	Timer Count Register of Timer A13	4
021BF1F0	023BF1F0	025BF1F0	027BF1F0	02DBF1F0	02FBF1F0	TCNRA14	Timer Count Register of Timer A14	4
021BF1F8	023BF1F8	025BF1F8	027BF1F8	02DBF1F8	02FBF1F8	TCNRA15	Timer Count Register of Timer A15	4
021BF200–021BF37F	023BF200–023BF37F	025BF200–025BF37F	027BF200–027BF37F	02DBF200–02DBF37F	02FBF200–02FBF37F	Reserved		
021BF380	023BF380	025BF380	027BF380	02DBF380	02FBF380	TGCRA	Timer General Configuration Register of Timers Module A	4
021BF388	023BF388	025BF388	027BF388	02DBF388	02FBF388	TERA	Timer Event Register of Timers Module A	4
021BF390	023BF390	025BF390	027BF390	02DBF390	02FBF390	TIERA	Timer Interrupt Enable Register of Timers Module A	4
021BF398	023BF398	025BF398	027BF398	02DBF398	02FBF398	TSRA	Timer Status Register of Timers Module A	4
021BF3A0–021BF3FF	023BF3A0–023BF3FF	025BF3A0–025BF3FF	027BF3A0–027BF3FF	02DBF3A0–02DBF3FF	02FBF3A0–02FBF3FF	Reserved		
021BF400	023BF400	025BF400	027BF400	02DBF400	02FBF400	TCFRB0	Timer Configuration Register of Timer B0	4
021BF408	023BF408	025BF408	027BF408	02DBF408	02FBF408	TCFRB1	Timer Configuration Register of Timer B1	4
021BF410	023BF410	025BF410	027BF410	02DBF410	02FBF410	TCFRB2	Timer Configuration Register of Timer B2	4
021BF418	023BF418	025BF418	027BF418	02DBF418	02FBF418	TCFRB3	Timer Configuration Register of Timer B3	4
021BF420	023BF420	025BF420	027BF420	02DBF420	02FBF420	TCFRB4	Timer Configuration Register of Timer B4	4
021BF428	023BF428	025BF428	027BF428	02DBF428	02FBF428	TCFRB5	Timer Configuration Register of Timer B5	4
021BF430	023BF430	025BF430	027BF430	02DBF430	02FBF430	TCFRB6	Timer Configuration Register of Timer B6	4
021BF438	023BF438	025BF438	027BF438	02DBF438	02FBF438	TCFRB7	Timer Configuration Register of Timer B7	4
021BF440	023BF440	025BF440	027BF440	02DBF440	02FBF440	TCFRB8	Timer Configuration Register of Timer B8	4
021BF448	023BF448	025BF448	027BF448	02DBF448	02FBF448	TCFRB9	Timer Configuration Register of Timer B9	4
021BF450	023BF450	025BF450	027BF450	02DBF450	02FBF450	TCFRB10	Timer Configuration Register of Timer B10	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	**001**	**010**	**011**	**110**	**111**			
021BF458	023BF458	025BF458	027BF458	02DBF458	02FBF458	TCFRB11	Timer Configuration Register of Timer B11	4
021BF460	023BF460	025BF460	027BF460	02DBF460	02FBF460	TCFRB12	Timer Configuration Register of Timer B12	4
021BF468	023BF468	025BF468	027BF468	02DBF468	02FBF468	TCFRB13	Timer Configuration Register of Timer B13	4
021BF470	023BF470	025BF470	027BF470	02DBF470	02FBF470	TCFRB14	Timer Configuration Register of Timer B14	4
021BF478	023BF478	025BF478	027BF478	02DBF478	02FBF478	TCFRB15	Timer Configuration Register of Timer B15	4
021BF480	023BF480	025BF480	027BF480	02DBF480	02FBF480	TCMPB0	Timer Compare Register of Timer B0	4
021BF488	023BF488	025BF488	027BF488	02DBF488	02FBF488	TCMPB1	Timer Compare Register of Timer B1	4
021BF490	023BF490	025BF490	027BF490	02DBF490	02FBF490	TCMPB2	Timer Compare Register of Timer B2	4
021BF498	023BF498	025BF498	027BF498	02DBF498	02FBF498	TCMPB3	Timer Compare Register of Timer B3	4
021BF4A0	023BF4A0	025BF4A0	027BF4A0	02DBF4A0	02FBF4A0	TCMPB4	Timer Compare Register of Timer B4	4
021BF4A8	023BF4A8	025BF4A8	027BF4A8	02DBF4A8	02FBF4A8	TCMPB5	Timer Compare Register of Timer B5	4
021BF40	023BF40	025BF40	027BF40	02DBF40	02FBF40	TCMPB6	Timer Compare Register of Timer B6	4
021BF48	023BF48	025BF48	027BF48	02DBF48	02FBF48	TCMPB7	Timer Compare Register of Timer B7	4
021BF4C0	023BF4C0	025BF4C0	027BF4C0	02DBF4C0	02FBF4C0	TCMPB8	Timer Compare Register of Timer B8	4
021BF4C8	023BF4C8	025BF4C8	027BF4C8	02DBF4C8	02FBF4C8	TCMPB9	Timer Compare Register of Timer B9	4
021BF4D0	023BF4D0	025BF4D0	027BF4D0	02DBF4D0	02FBF4D0	TCMPB10	Timer Compare Register of Timer B10	4
021BF4D8	023BF4D8	025BF4D8	027BF4D8	02DBF4D8	02FBF4D8	TCMPB11	Timer Compare Register of Timer B11	4
021BF4E0	023BF4E0	025BF4E0	027BF4E0	02DBF4E0	02FBF4E0	TCMPB12	Timer Compare Register of Timer B12	4
021BF4E8	023BF4E8	025BF4E8	027BF4E8	02DBF4E8	02FBF4E8	TCMPB13	Timer Compare Register of Timer B13	4
021BF4F0	023BF4F0	025BF4F0	027BF4F0	02DBF4F0	02FBF4F0	TCMPB14	Timer Compare Register of Timer B14	4
021BF4F8	023BF4F8	025BF4F8	027BF4F8	02DBF4F8	02FBF4F8	TCMPB15	Timer Compare Register of Timer B15	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BF500	023BF500	025BF500	027BF500	02DBF500	02FBF500	TCRB0	Timer Control Register of Timer B0	4
021BF508	023BF508	025BF508	027BF508	02DBF508	02FBF508	TCRB1	Timer Control Register of Timer B1	4
021BF510	023BF510	025BF510	027BF510	02DBF510	02FBF510	TCRB2	Timer Control Register of Timer B2	4
021BF518	023BF518	025BF518	027BF518	02DBF518	02FBF518	TCRB3	Timer Control Register of Timer B3	4
021BF520	023BF520	025BF520	027BF520	02DBF520	02FBF520	TCRB4	Timer Control Register of Timer B4	4
021BF528	023BF528	025BF528	027BF528	02DBF528	02FBF528	TCRB5	Timer Control Register of Timer B5	4
021BF530	023BF530	025BF530	027BF530	02DBF530	02FBF530	TCRB6	Timer Control Register of Timer B6	4
021BF538	023BF538	025BF538	027BF538	02DBF538	02FBF538	TCRB7	Timer Control Register of Timer B7	4
021BF540	023BF540	025BF540	027BF540	02DBF540	02FBF540	TCRB8	Timer Control Register of Timer B8	4
021BF548	023BF548	025BF548	027BF548	02DBF548	02FBF548	TCRB9	Timer Control Register of Timer B9	4
021BF550	023BF550	025BF550	027BF550	02DBF550	02FBF550	TCRB10	Timer Control Register of Timer B10	4
021BF558	023BF558	025BF558	027BF558	02DBF558	02FBF558	TCRB11	Timer Control Register of Timer B11	4
021BF560	023BF560	025BF560	027BF560	02DBF560	02FBF560	TCRB12	Timer Control Register of Timer B12	4
021BF568	023BF568	025BF568	027BF568	02DBF568	02FBF568	TCRB13	Timer Control Register of Timer B13	4
021BF570	023BF570	025BF570	027BF570	02DBF570	02FBF570	TCRB14	Timer Control Register of Timer B14	4
021BF578	023BF578	025BF578	027BF578	02DBF578	02FBF578	TCRB15	Timer Control Register of Timer B15	4
021BF580	023BF580	025BF580	027BF580	02DBF580	02FBF580	TCNRB0	Timer Count Register of Timer B0	4
021BF588	023BF588	025BF588	027BF588	02DBF588	02FBF588	TCNRB1	Timer Count Register of Timer B1	4
021BF590	023BF590	025BF590	027BF590	02DBF590	02FBF590	TCNRB2	Timer Count Register of Timer B2	4
021BF598	023BF598	025BF598	027BF598	02DBF598	02FBF598	TCNRB3	Timer Count Register of Timer B3	4
021BF5A0	023BF5A0	025BF5A0	027BF5A0	02DBF5A0	02FBF5A0	TCNRB4	Timer Count Register of Timer B4	4

Table 8-8. Local Bus Banks 9, 11 Memory Map (Continued)

Address for ISB =						Acronym	Name	Size in Bytes
000	001	010	011	110	111			
021BF5A8	023BF5A8	025BF5A8	027BF5A8	02DBF5A8	02FBF5A8	TCNRB5	Timer Count Register of Timer B5	4
021BF5B0	023BF5B0	025BF5B0	027BF5B0	02DBF5B0	02FBF5B0	TCNRB6	Timer Count Register of Timer B6	4
021BF5B8	023BF5B8	025BF5B8	027BF5B8	02DBF5B8	02FBF5B8	TCNRB7	Timer Count Register of Timer B7	4
021BF5C0	023BF5C0	025BF5C0	027BF5C0	02DBF5C0	02FBF5C0	TCNRB8	Timer Count Register of Timer B8	4
021BF5C8	023BF5C8	025BF5C8	027BF5C8	02DBF5C8	02FBF5C8	TCNRB9	Timer Count Register of Timer B9	4
021BF5D0	023BF5D0	025BF5D0	027BF5D0	02DBF5D0	02FBF5D0	TCNRB10	Timer Count Register of Timer B10	4
021BF5D8	023BF5D8	025BF5D8	027BF5D8	02DBF5D8	02FBF5D8	TCNRB11	Timer Count Register of Timer B11	4
021BF5E0	023BF5E0	025BF5E0	027BF5E0	02DBF5E0	02FBF5E0	TCNRB12	Timer Count Register of Timer B12	4
021BF5E8	023BF5E8	025BF5E8	027BF5E8	02DBF5E8	02FBF5E8	TCNRB13	Timer Count Register of Timer B13	4
021BF5F0	023BF5F0	025BF5F0	027BF5F0	02DBF5F0	02FBF5F0	TCNRB14	Timer Count Register of Timer B14	4
021BF5F8	023BF5F8	025BF5F8	027BF5F8	02DBF5F8	02FBF5F8	TCNRB15	Timer Count Register of Timer B15	4
021BF600–021BF77F	023BF600–023BF77F	025BF600–025BF77F	027BF600–027BF77F	02DBF600–02DBF77F	02FBF600–02FBF77F	Reserved		
021BF780	023BF780	025BF780	027BF780	02DBF780	02FBF780	TGCRB	Timer General Configuration Register of Timers Module B	4
021BF788	023BF788	025BF788	027BF788	02DBF788	02FBF788	TERB	Timer Event Register of Timers Module B	4
021BF790	023BF790	025BF790	027BF790	02DBF790	02FBF790	TIERB	Timer Interrupt Enable Register of Timers Module B	4
021BF798	023BF798	025BF798	027BF798	02DBF798	02FBF798	TSRB	Timer Status Register of Timers Module B	4
021BF7A0–021BFFFF	023BF7A0–023BFFFF	025BF7A0–025BFFFF	027BF7A0–027BFFFF	02DBF7A0–02DBFFFF	02FBF7A0–02FBFFFF	Reserved		

8.7 System Bus Address Space

The system bus address space includes devices residing on the on-device or off-device system bus, as follows:

- The MSC8113 system registers are located on the system bus. These registers are mapped within a contiguous block of 128 KB of memory. The base address for this block is programmed as shown in **Section 4.2**, *SIU Programming Model*. The SC140 cores and external hosts access these registers through either the system bus or the DSI.
- Other devices, such as memories and external peripherals, may reside on the external system bus. Both the SC140 cores and hosts access these external devices on the system bus. The internal memory controller monitors access to these devices.

Table 8-9 lists the internal devices that reside on the system bus.

Table 8-9. System Registers Memory Map (QBus Bank 3)

Address for ISB =						Acronym	Name	Size
000	001	010	011	110	111			
F0000000–F000FFFF	F0F00000–F0F0FFFF	FF000000–FF00FFFF	FFF00000–FFF0FFFF	0F000000–0F00FFFF	0FF00000–0FF0FFFF	Reserved		
General SIU								
F0010000	F0F10000	FF010000	FFF10000	0F010000	0FF10000	SIUMCR	SIU Module Configuration Register	4
F0010004	F0F10004	FF010004	FFF10004	0F010004	0FF10004	SYPCR	System Protection Control Register	4
F0010008–F001000D	F0F10008–F0F1000D	FF010008–FF01000D	FFF10008–FFF1000D	0F010008–0F01000D	0FF10008–0FF1000D	Reserved		
F001000E	F0F1000E	FF01000E	FFF1000E	0F01000E	0FF1000E	SWSR	Software Service Register	2
F0010010–F0010023	F0F10010–F0F10023	FF010010–FF010023	FFF10010–FFF10023	0F010010–0F010023	0FF10010–0FF10023	Reserved		
F0010024	F0F10024	FF010024	FFF10024	0F010024	0FF10024	BCR	Bus Configuration Register	4
F0010028	F0F10028	FF010028	FFF10028	0F010028	0FF10028	PPC_ACR	System Bus Arbiter Configuration Register	1 B
F001002C	F0F1002C	FF01002C	FFF1002C	0F01002C	0FF1002C	PPC_ALRH	System Bus Arbitration Level Register (bus masters 0–7)	4
F0010030	F0F10030	FF010030	FFF10030	0F010030	0FF10030	PPC_ALRL	System Bus Arbitration Level Register (bus masters 8–15)	4
F0010034	F0F10034	FF010034	FFF10034	0F010034	0FF10034	LCL_ACR	Local Arbiter Configuration Register	1 B
F0010038	F0F10038	FF010038	FFF10038	0F010038	0FF10038	LCL_ALRH	Local Arbitration Level Register (bus masters 0–7)	4
F001003C	F0F1003C	FF01003C	FFF1003C	0F01003C	0FF1003C	LCL_ALRL	Local Arbitration Level Register (bus masters 8–15)	4
F0010040	F0F10040	FF010040	FFF10040	0F010040	0FF10040	TESCR1	System Bus Transfer Error Status Control Register 1	4

Table 8-9. System Registers Memory Map (QBus Bank 3) (Continued)

Address for ISB =						Acronym	Name	Size
000	001	010	011	110	111			
F0010044	F0F10044	FF010044	FFF10044	0F010044	0FF10044	TESCR2	System Bus Transfer Error Status Control Register 2	4
F0010048	F0F10048	FF010048	FFF10048	0F010048	0FF10048	L_TESCR1	Local Bus Transfer Error Status Control Register 1	4
F001004C–F0010057	F0F1004C–F0F10057	FF01004C–FF010057	FFF1004C–FFF10057	0F01004C–0F010057	0FF1004C–0FF10057	Reserved		
F0010058	F0F10058	FF010058	FFF10058	0F010058	0FF10058	LGTDTEA	Local Bus GTD (Global TDM DMA) Transfer Error Address	4
F001005C	F0F1005C	FF01005C	FFF1005C	0F01005C	0FF1005C	LGTDTEM	Local Bus GTD (Global TDM DMA) Transfer Error TDMNUM_TR	1 B
F001005D–F001005F	F0F1005D–F0F1005F	FF01005D–FF01005F	FFF1005D–FFF1005F	0F01005D–0F01005F	0FF1005D–0FF1005F	Reserved		
F0010060	F0F10060	FF010060	FFF10060	0F010060	0FF10060	PDMTEA	System Bus DMA Transfer Error Address	4
F0010064	F0F10064	FF010064	FFF10064	0F010064	0FF10064	PDMTER	System Bus DMA Transfer Error RQNUM	1 B
F0010065–F0010067	F0F10065–F0F10067	FF010065–FF010067	FFF10065–FFF10067	0F010065–0F010067	0FF10065–0FF10067	Reserved		
F0010068	F0F10068	FF010068	FFF10068	0F010068	0FF10068	LDMTEA	Local Bus DMA Transfer Error Address	4
F001006C	F0F1006C	FF01006C	FFF1006C	0F01006C	0FF1006C	LDMTER	Local Bus DMA Transfer Error RQNUM	1 B
F001006D–F00100FF	F0F1006D–F0F100FF	FF01006D–FF0100FF	FFF1006D–FFF100FF	0F01006D–0F0100FF	0FF1006D–0FF100FF	Reserved		
Memory Controller								
F0010100	F0F10100	FF010100	FFF10100	0F010100	0FF10100	BR0	Base Register Bank0	4
F0010104	F0F10104	FF010104	FFF10104	0F010104	0FF10104	OR0	Option Register Bank0	4
F0010108	F0F10108	FF010108	FFF10108	0F010108	0FF10108	BR1	Base Register Bank1	4
F001010C	F0F1010C	FF01010C	FFF1010C	0F01010C	0FF1010C	OR1	Option Register Bank1	4
F0010110	F0F10110	FF010110	FFF10110	0F010110	0FF10110	BR2	Base Register Bank2	4
F0010114	F0F10114	FF010114	FFF10114	0F010114	0FF10114	OR2	Option Register Bank2	4
F0010118	F0F10118	FF010118	FFF10118	0F010118	0FF10118	BR3	Base Register Bank3	4
F001011C	F0F1011C	FF01011C	FFF1011C	0F01011C	0FF1011C	OR3	Option Register Bank3	4
F0010120	F0F10120	FF010120	FFF10120	0F010120	0FF10120	BR4	Base Register Bank4	4
F0010124	F0F10124	FF010124	FFF10124	0F010124	0FF10124	OR4	Option Register Bank4	4
F0010128	F0F10128	FF010128	FFF10128	0F010128	0FF10128	BR5	Base Register Bank5	4
F001012C	F0F1012C	FF01012C	FFF1012C	0F01012C	0FF1012C	OR5	Option Register Bank5	4
F0010130	F0F10130	FF010130	FFF10130	0F010130	0FF10130	BR6	Base Register Bank6	4
F0010134	F0F10134	FF010134	FFF10134	0F010134	0FF10134	OR6	Option Register Bank6	4

Table 8-9. System Registers Memory Map (QBus Bank 3) (Continued)

Address for ISB =						Acronym	Name	Size
000	001	010	011	110	111			
F0010138	F0F10138	FF010138	FFF10138	0F010138	0FF10138	BR7	Base Register Bank7	4
F001013C	F0F1013C	FF01013C	FFF1013C	0F01013C	0FF1013C	OR7	Option Register Bank7	4
F0010140	F0F10140	FF010140	FFF10140	0F010140	0FF10140	Reserved		
F0010144	F0F10144	FF010144	FFF10144	0F010144	0FF10144	Reserved		
F0010148	F0F10148	FF010148	FFF10148	0F010148	0FF10148	BR9	Base Register Bank9	4
F001014C	F0F1014C	FF01014C	FFF1014C	0F01014C	0FF1014C	OR9	Option Register Bank9	4
F0010150	F0F10150	FF010150	FFF10150	0F010150	0FF10150	BR10	Base Register Bank10	4
F0010154	F0F10154	FF010154	FFF10154	0F010154	0FF10154	OR10	Option Register Bank10	4
F0010158	F0F10158	FF010158	FFF10158	0F010158	0FF10158	BR11	Base Register Bank11	4
F001015C	F0F1015C	FF01015C	FFF1015C	0F01015C	0FF1015C	OR11	Option Register Bank11	4
F0010160–F0010167	F0F10160–F0F10167	FF010160–FF010167	FFF10160–FFF10167	0F010160–0F010167	0FF10160–0FF10167	Reserved		
F0010168	F0F10168	FF010168	FFF10168	0F010168	0FF10168	MAR	Memory Address Register	4
F001016C–F001016F	F0F1016C–F0F1016F	FF01016C–FF01016F	FFF1016C–FFF1016F	0F01016C–0F01016F	0FF1016C–0FF1016F	Reserved		
F0010170	F0F10170	FF010170	FFF10170	0F010170	0FF10170	MAMR	Machine A Mode Register	4
F0010174	F0F10174	FF010174	FFF10174	0F010174	0FF10174	MBMR	Machine B Mode Register	4
F0010178	F0F10178	FF010178	FFF10178	0F010178	0FF10178	MCMR	Machine C Mode Register	4
F001017C–F0010183	F0F1017C–F0F10183	FF01017C–FF010183	FFF1017C–FFF10183	0F01017C–0F010183	0FF1017C–0FF10183	Reserved		
F0010184	F0F10184	FF010184	FFF10184	0F010184	0FF10184	MPTPR	Memory Refresh Timer Prescaler	2
F0010188	F0F10188	FF010188	FFF10188	0F010188	0FF10188	MDR	Memory Data Register	4
F001018C–F001018F	F0F1018C–F0F1018F	FF01018C–FF01018F	FFF1018C–FFF1018F	0F01018C–0F01018F	0FF1018C–0FF1018F	Reserved		
F0010190	F0F10190	FF010190	FFF10190	0F010190	0FF10190	PSDMR	System Bus SDRAM Mode Register	4
F0010194	F0F10194	FF010194	FFF10194	0F010194	0FF10194	LSDMR	Local Bus SDRAM Mode Register	4
F0010198	F0F10198	FF010198	FFF10198	0F010198	0FF10198	PURT	System Bus-Assigned UPM Refresh Timer	1 B
F001019C	F0F1019C	FF01019C	FFF1019C	0F01019C	0FF1019C	PSRT	System Bus-Assigned SDRAM Refresh Timer	1 B
F00101A0	F0F101A0	FF0101A0	FFF101A0	0F0101A0	0FF101A0	LURT	Local Bus-Assigned UPM Refresh Timer	1 B
F00101A4	F0F101A4	FF0101A4	FFF101A4	0F0101A4	0FF101A4	LSRT	Local Bus-Assigned SDRAM Refresh Timer	1 B
F00101A8	F0F101A8	FF0101A8	FFF101A8	0F0101A8	0FF101A8	IMMR	Internal Memory Map Register	4

Table 8-9. System Registers Memory Map (QBus Bank 3) (Continued)

Address for ISB =						Acronym	Name	Size
000	001	010	011	110	111			
F00101AC–F00101FF	F0F101AC–F0F101FF	FF0101AC–FF0101FF	FFF101AC – FFF101FF	0F0101AC–0F0101FF	0FF101AC–0FF101FF	Reserved		
System Integration Timers								
F0010200–F001021F	F0F10200–F0F1021F	FF010200–FF01021F	FFF10200–FFF1021F	0F010200–0F01021F	0FF10200–0FF1021F	Reserved		
F0010220	F0F10220	FF010220	FFF10220	0F010220	0FF10220	TMCNTSC	Time Counter Status and Control Register	2
F0010224	F0F10224	FF010224	FFF10224	0F010224	0FF10224	TMCNT	Time Counter Register	4
F0010228–F001022B	F0F10228–F0F1022B	FF010228–FF01022B	FFF10228–FFF1022B	0F010228–0F01022B	0FF10228–0FF1022B	Reserved		
F001022C	F0F1022C	FF01022C	FFF1022C	0F01022C	0FF1022C	TMCNTAL	Time Counter Alarm Register	4
F0010230–F001023F	F0F10230–F0F1023F	FF010230–FF01023F	FFF10230–FFF1023F	0F010230–0F01023F	0FF10230–0FF1023F	Reserved		
F0010240	F0F10240	FF010240	FFF10240	0F010240	0FF10240	PISCR	Periodic Interrupt Status and Control Register	2
F0010244	F0F10244	FF010244	FFF10244	0F010244	0FF10244	PITC	Periodic Interrupt Count Register	4
F0010248	F0F10248	FF010248	FFF10248	0F010248	0FF10248	PITR	Periodic Interrupt Timer Register	4
F001024C–F001029F	F0F1024C–F0F1029F	FF01024C–FF01029F	FFF1024C–FFF1029F	0F01024C–0F01029F	0FF1024C–0FF1029F	Reserved		
F00102A0–F00106FF	F0F102A0–F0F106FF	FF0102A0–FF0106FF	FFF102A0–FFF106FF	0F0102A0–0F0106FF	0FF102A0–0FF106FF	Reserved		
DMA Channels 0– 15								
F0010700	F0F10700	FF010700	FFF10700	0F010700	0FF10700	DCHCR0	DMA Channel 0 Configuration Register	4
F0010704	F0F10704	FF010704	FFF10704	0F010704	0FF10704	DCHCR1	DMA Channel 1 Configuration Register	4
F0010708	F0F10708	FF010708	FFF10708	0F010708	0FF10708	DCHCR2	DMA Channel 2 Configuration Register	4
F001070C	F0F1070C	FF01070C	FFF1070C	0F01070C	0FF1070C	DCHCR3	DMA Channel 3 Configuration Register	4
F0010710	F0F10710	FF010710	FFF10710	0F010710	0FF10710	DCHCR4	DMA Channel 4 Configuration Register	4
F0010714	F0F10714	FF010714	FFF10714	0F010714	0FF10714	DCHCR5	DMA Channel 5 Configuration Register	4
F0010718	F0F10718	FF010718	FFF10718	0F010718	0FF10718	DCHCR6	DMA Channel 6 Configuration Register	4
F001071C	F0F1071C	FF01071C	FFF1071C	0F01071C	0FF1071C	DCHCR7	DMA Channel 7 Configuration Register	4
F0010720	F0F10720	FF010720	FFF10720	0F010720	0FF10720	DCHCR8	DMA Channel 8 Configuration Register	4

Table 8-9. System Registers Memory Map (QBus Bank 3) (Continued)

Address for ISB =						Acronym	Name	Size
000	001	010	011	110	111			
F0010724	F0F10724	FF010724	FFF10724	0F010724	0FF10724	DCHCR9	DMA Channel 9 Configuration Register	4
F0010728	F0F10728	FF010728	FFF10728	0F010728	0FF10728	DCHCR10	DMA Channel 10 Configuration Register	4
F001072C	F0F1072C	FF01072C	FFF1072C	0F01072C	0FF1072C	DCHCR11	DMA Channel 11 Configuration Register	4
F0010730	F0F10730	FF010730	FFF10730	0F010730	0FF10730	DCHCR12	DMA Channel 12 Configuration Register	4
F0010734	F0F10734	FF010734	FFF10734	0F010734	0FF10734	DCHCR13	DMA Channel 13 Configuration Register	4
F0010738	F0F10738	FF010738	FFF10738	0F010738	0FF10738	DCHCR14	DMA Channel 14 Configuration Register	4
F001073C	F0F1073C	FF01073C	FFF1073C	0F01073C	0FF1073C	DCHCR15	DMA Channel 15 Configuration Register	4
F0010740–F001077F	F0F10740–F0F1077F	FF010740–FF01077F	FFF10740–FFF1077F	0F010740–0F01077F	0FF10740–0FF1077F	Reserved		
F0010780	F0F10780	FF010780	FFF10780	0F010780	0FF10780	DIMR	DMA Internal Mask Register	4
F0010784	F0F10784	FF010784	FFF10784	0F010784	0FF10784	DSTR	DMA Status Register	4
F0010788	F0F10788	FF010788	FFF10788	0F010788	0FF10788	DTEAR	DMA TEA Status Register	1 B
F0010789–F001078B	F0F10789–F0F1078B	FF010789–FF01078B	FFF10789–FFF1078B	0F010789–0F01078B	0FF10789–0FF1078B	Reserved		
F001078C	F0F1078C	FF01078C	FFF1078C	0F01078C	0FF1078C	DPCR	DMA Pin Configuration Register	1 B
F0010790	F0F10790	FF010790	FFF10790	0F010790	0FF10790	DEMR	DMA External Mask Register	4
F0010794–F00107FF	F0F10794–F0F107FF	FF010794–FF0107FF	FFF10794–FFF107FF	0F010794–0F0107FF	0FF10794–0FF107FF	Reserved		
F0010800–F0010BFF	F0F10800–F0F10BFF	FF010800–FF010BFF	FFF10800–FFF10BFF	0F010800–0F010BFF	0FF10800–0FF10BFF	DCPRAM	DMA Channel Parameter RAM	128 B
F0010C00–F0010C80	F0F10C00–F0F10C80	FF010C00–FF010C80	FFF10C00–FFF10C80	0F010C00–0F010C80	0FF10C00–0FF10C80	Reserved		
Clocks and Reset								
F0010C88	F0F10C88	FF010C88	FFF10C88	0F010C88	0FF10C88	SCMSR	System Clock Mode Register	4
F0010C90	F0F10C90	FF010C90	FFF10C90	0F010C90	0FF10C90	RSR	Reset Status Register	4
F0010C94–F001FFFF	F0F10C94–F0F1FFFF	FF010C94–FF01FFFF	FFF10C94–FFF1FFFF	0F010C94–0F01FFFF	0FF10C94–0FF1FFFF	Reserved		

8.8 DSI Address Map

Table 8-10 is a detailed listing of the DSI address map.

Table 8-10. DSI Address Map (0x000000–0x1FFFFF)

Address	Abbreviation	Name	Size in Bytes
000000–076FFF	M2MEM	M2 Memory	476 K
077000–077FFF	BOOTROM	MSC8113 Boot ROM	4 K
078000–07FFFF	Reserved		
080000–0B7FFF	M1MEM0	M1 Memory Core 0	224 K
0B8000–0BFFFF	Reserved		
0C0000–0F7FFF	M1MEM1	M1 Memory Core 1	224 K
0F8000–0FFFFF	Reserved		
100000–137FFF	M1MEM2	M1 Memory Core 2	224 K
138000–17FFFF	Reserved		
180000–1807FF		TDM0 Receive Local Memory	2 K
180800–180FFF	Reserved		
181000–1813FC	TDM0 RCPR[0–255]	TDM0 Receive Channel Parameters Register 0–255	4 each
181400–1817FF	Reserved		
181800–181FFF		TDM0 Transmit Local Memory	2 K
182000–1827FF	Reserved		
182800–182BFC	TDM0 TCPR[0–255]	TDM0 Transmit Channel Parameters Register 0–255	4 each
182C00–183F1F	Reserved		
183F20	TDM0TSR	TDM0 Transmit Status Register	4
183F28	TDM0RSR	TDM0 Receive Status Register	4
183F30	TDM0ASR	TDM0 Adaptation Status Register	4
183F38	TDM0TER	TDM0 Transmit Event Register	4
183F40	TDM0RER	TDM0 Receive Event Register	4
183F48	TDM0TNB	TDM0 Transmit Number of Buffers	4
183F50	TDM0RNB	TDM0 Receive Number of Buffers	4
183F58	TDM0TDBDR	TDM0 Transmit Data Buffer Displacement Register	4
183F60	TDM0RDBDR	TDM0 Receive Data Buffer Displacement Register	4
183F68	TDM0ASDR	TDM0 Adaptation Sync Distance Register	4
183F70	TDM0TIER	TDM0 Transmit Interrupt Enable Register	4
183F78	TDM0RIER	TDM0 Receive Interrupt Enable Register	4
183F80	TDM0TDBST	TDM0 Transmit Data Buffer Second Threshold	4
183F88	TDM0RDBST	TDM0 Receive Data Buffer Second Threshold	4
183F90	TDM0TDBFT	TDM0 Transmit Data Buffer First Threshold	4
183F98	TDM0RDBFT	TDM0 Receive Data Buffer First Threshold	4
183FA0	TDM0TCR	TDM0 Transmit Control Register	4
183FA8	TDM0RCR	TDM0 Receive Control Register	4
183FB0	TDM0ACR	TDM0 Adaptation Control Register	4
183FB8	TDM0TGBA	TDM0 Transmit Global Base Address	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
183FC0	TDM0RGBA	TDM0 Receive Global Base Address	4
183FC8	TDM0TDBS	TDM0 Transmit Data Buffer Size	4
183FD0	TDM0RDBS	TDM0 Receive Data Buffer Size	4
183FD8	TDM0TFP	TDM0 Transmit Frame Parameters	4
183FE0	TDM0RFP	TDM0 Receive Frame Parameters	4
183FE8	TDM0TIR	TDM0 Transmit Interface Register	4
183FF0	TDM0RIR	TDM0 Receive Interface Register	4
183FF8	TDM0GIR	TDM0 General Interface Register	4
184000–1847FF		TDM1 Receive Local Memory	2 KB
184800–184FFF	Reserved		
185000–1853FC	TDM1 RCPR[0–255]	TDM1 Receive Channel Parameters Register 0–255	4 each
185400–1857FF	Reserved		
185800–185FFF		TDM1 Transmit Local Memory	2 KB
186000–1867FF	Reserved		
186800–186BFC	TDM1 TCPR[0–255]	TDM1 Transmit Channel Parameters Register 0–255	4 each
186C00–187F1F	Reserved		
187F20	TDM1TSR	TDM1 Transmit Status Register	4
187F28	TDM1RSR	TDM1 Receive Status Register	4
187F30	TDM1ASR	TDM1 Adaptation Status Register	4
187F38	TDM1TER	TDM1 Transmit Event Register	4
187F40	TDM1RER	TDM1 Receive Event Register	4
187F48	TDM1TNB	TDM1 Transmit Number of Buffers	4
187F50	TDM1RNB	TDM1 Receive Number of Buffers	4
187F58	TDM1TDBDR	TDM1 Transmit Data Buffer Displacement Register	4
187F60	TDM1RDBDR	TDM1 Receive Data Buffer Displacement Register	4
187F68	TDM1ASDR	TDM1 Adaptation Sync Distance Register	4
187F70	TDM1TIER	TDM1 Transmit Interrupt Enable Register	4
187F78	TDM1RIER	TDM1 Receive Interrupt Enable Register	4
187F80	TDM1TDBST	TDM1 Transmit Data Buffer Second Threshold	4
187F88	TDM1RDBST	TDM1 Receive Data Buffer Second Threshold	4
187F90	TDM1TDBFT	TDM1 Transmit Data Buffer First Threshold	4
187F98	TDM1RDBFT	TDM1 Receive Data Buffer First Threshold	4
187FA0	TDM1TCR	TDM1 Transmit Control Register	4
187FA8	TDM1RCR	TDM1 Receive Control Register	4
187FB0	TDM1ACR	TDM1 Adaptation Control Register	4
187FB8	TDM1TGBA	TDM1 Transmit Global Base Address	4
187FC0	TDM1RGBA	TDM1 Receive Global Base Address	4
187FC8	TDM1TDBS	TDM1 Transmit Data Buffer Size	4
187FD0	TDM1RDBS	TDM1 Receive Data Buffer Size	4
187FD8	TDM1TFP	TDM1 Transmit Frame Parameters	4
187FE0	TDM1RFP	TDM1 Receive Frame Parameters	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
187FE8	TDM1TIR	TDM1 Transmit Interface Register	4
187FF0	TDM1RIR	TDM1 Receive Interface Register	4
187FF8	TDM1GIR	TDM1 General Interface Register	4
188000–1887FF		TDM2 Receive Local Memory	2 K
188800–188FFF	Reserved		
189000–1893FC	TDM2 RCPR[0–255]	TDM2 Receive Channel Parameters Register 0–255	4 each
189400–1897FF	Reserved		
189800–189FFF		TDM2 Transmit Local Memory	2 K
18A000–18A7FF	Reserved		
18A800–18ABFC	TDM2 TCPR[0–255]	TDM2 Transmit Channel Parameters Register 0–255	4 each
18AC00–18BF1F	Reserved		
18BF20	TDM2TSR	TDM2 Transmit Status Register	4
18BF28	TDM2RSR	TDM2 Receive Status Register	4
18BF30	TDM2ASR	TDM2 Adaptation Status Register	4
18BF38	TDM2TER	TDM2 Transmit Event Register	4
18BF40	TDM2RER	TDM2 Receive Event Register	4
18BF48	TDM2TNB	TDM2 Transmit Number of Buffers	4
18BF50	TDM2RNB	TDM2 Receive Number of Buffers	4
18BF58	TDM2TDBDR	TDM2 Transmit Data Buffer Displacement Register	4
18BF60	TDM2RDBDR	TDM2 Receive Data Buffer Displacement Register	4
18BF68	TDM1ASDR	TDM2 Adaptation Sync Distance Register	4
18BF70	TDM2TIER	TDM2 Transmit Interrupt Enable Register	4
18BF78	TDM2RIER	TDM2 Receive Interrupt Enable Register	4
18BF80	TDM2TDBST	TDM2 Transmit Data Buffer Second Threshold	4
18BF88	TDM2RDBST	TDM2 Receive Data Buffer Second Threshold	4
18BF90	TDM2TDBFT	TDM2 Transmit Data Buffer First Threshold	4
18BF98	TDM2RDBFT	TDM2 Receive Data Buffer First Threshold	4
18BFA0	TDM2TCR	TDM2 Transmit Control Register	4
18BFA8	TDM2RCR	TDM2 Receive Control Register	4
18BFB0	TDM2ACR	TDM2 Adaptation Control Register	4
18BFB8	TDM2TGBA	TDM2 Transmit Global Base Address	4
18BFC0	TDM2RGBA	TDM2 Receive Global Base Address	4
18BFC8	TDM2TDBS	TDM2 Transmit Data Buffer Size	4
18BFD0	TDM2RDBS	TDM2 Receive Data Buffer Size	4
18BFD8	TDM2TFP	TDM2 Transmit Frame Parameters	4
18BFE0	TDM2RFP	TDM2 Receive Frame Parameters	4
18BFE8	TDM2TIR	TDM2 Transmit Interface Register	4
18BFF0	TDM2RIR	TDM2 Receive Interface Register	4
18BFF8	TDM2GIR	TDM2 General Interface Register	4
18C000–18C7FF		TDM3 Receive Local Memory	2 K
18C800–18CFFF	Reserved		

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
18D000–18D3FC	TDM3 RCPR[0–255]	TDM3 Receive Channel Parameters Register 0–255	4 each
18D400–18D7FF	Reserved		
18D800–18DFFF		TDM3 Transmit Local Memory	2 K
18E000–18E7FF	Reserved		
18E800–18EBFC	TDM3 TCPR[0–255]	TDM3 Transmit Channel Parameters Register 0–255	4 each
18EC00–18FF1F	Reserved		
18FF20	TDM3TSR	TDM3 Transmit Status Register	4
18FF28	TDM3RSR	TDM3 Receive Status Register	4
18FF30	TDM3ASR	TDM3 Adaptation Status Register	4
18FF38	TDM3TER	TDM3 Transmit Event Register	4
18FF40	TDM3RER	TDM3 Receive Event Register	4
18FF48	TDM3TNB	TDM3 Transmit Number of Buffers	4
18FF50	TDM3RNB	TDM3 Receive Number of Buffers	4
18FF58	TDM3TDBDR	TDM3 Transmit Data Buffer Displacement Register	4
18FF60	TDM3RDBDR	TDM3 Receive Data Buffer Displacement Register	4
18FF68	TDM3ASDR	TDM3 Adaptation Sync Distance Register	4
18FF70	TDM3TIER	TDM3 Transmit Interrupt Enable Register	4
18FF78	TDM3RIER	TDM3 Receive Interrupt Enable Register	4
18FF80	TDM3TDBST	TDM3 Transmit Data Buffer Second Threshold	4
18FF88	TDM3RDBST	TDM3 Receive Data Buffer Second Threshold	4
18FF90	TDM3TDBFT	TDM3 Transmit Data Buffer First Threshold	4
18FF98	TDM3RDBFT	TDM3 Receive Data Buffer First Threshold	4
18FFA0	TDM3TCR	TDM3 Transmit Control Register	4
18FFA8	TDM3RCR	TDM3 Receive Control Register	4
18FFB0	TDM3ACR	TDM3 Adaptation Control Register	4
18FFB8	TDM3TGBA	TDM3 Transmit Global Base Address	4
18FFC0	TDM3RGBA	TDM3 Receive Global Base Address	4
18FFC8	TDM3TDBS	TDM3 Transmit Data Buffer Size	4
18FFD0	TDM3RDBS	TDM3 Receive Data Buffer Size	4
18FFD8	TDM3TFP	TDM3 Transmit Frame Parameters	4
18FFE0	TDM3RFP	TDM3 Receive Frame Parameters	4
18FFE8	TDM3TIR	TDM3 Transmit Interface Register	4
18FFF0	TDM3RIR	TDM3 Receive Interface Register	4
18FFF8	TDM3GIR	TDM3 General Interface Register	4
190000–1B800F	Reserved		
1B8010	IEVENT	Interrupt Event Register	4
1B8014	IMASK	Interrupt Mask Register	4
1B8020	ECNTRL	Ethernet Control Register	4
1B8024	MINFLR	Minimum Frame Length Register	4
1B8028	PTV	Pause Time Value Register	4
1B802C	DMACTRL	DMA Control Register	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8034–1B8037	Reserved		
1B8038	DMAMR	DMA Maintenance Register	4
1B803C–1B8047	Reserved		
1B8048	FRXSTATR	FIFO Receive Status Register	4
1B804C	FRXCTRLR	FIFO Receive Control Register	4
1B8050	FRXALAR	FIFO Receive Alarm Register	4
1B8054	FRXSHR	FIFO Receive Alarm Shutoff Register	4
1B8058	FRXPAR	FIFO Receive Panic Register	4
1B805C	FRXPSR	FIFO Receive Shutoff Register	4
1B8078	FTXSTATR	FIFO Transmit Status Register	4
1B807C–1B808B	Reserved		
1B808C	FTXTHR	FIFO Transmit Threshold Register	4
1B8094	FTXSPR	FIFO Transmit Space Available Register	4
1B8098	FTXSR	FIFO Transmit Starve Register	4
1B809C	FTXSSR	FIFO transmit Starve Shutoff register	4
1B80A0–1B80FF	Reserved		
1B8100	TCTRL	Transmit Control Register	4
1B8104	TSTAT	Transmit Status Register	4
1B8108–1B810B	Reserved		
1B810C	TBDLEN	TxBD Data Length	4
1B8110–1B8123	Reserved		
1B8124	CTBPTR	Current TxBD Pointer	4
1B8128–1B8183	Reserved		
1B8184	TBPTR	TxBD Pointer	4
1B8188–1B8203	Reserved		
1B8204	TBASE	Transmit Descriptor Base Address	4
1B8208–1B82AF	Reserved		
1B82B0	OSTBD	Out-of-sequence TxBD Register	4
1B82B4	OSTBDP	Out-of-sequence Tx Data Buffer Pointer Register	4
1B82B8	OS32TBDP	Out-of-sequence 32 Bytes Tx Data Buffer Pointer Register	4
1B82C0	OS32IPTR	Out-of-sequence 32 Bytes TxBD Insert Pointer Register	4
1B82C4	OS32TBDR	Out-of-sequence 32 Bytes TxBD Reserved Register	4
1B82C8	OS32IIL	Out-of-sequence 32 Bytes TxBD Insert Index/Length Register	4
1B82CC–1B82FF	Reserved		
1B8300	RCTRL	Receive Control Register	4
1B8304	RSTAT	Receive Status Register	4
1B8308–1B830B	Reserved		
1B830C	RBDLEN	RxBD Data Length	4
1B8310–1B8323	Reserved		
1B8324	CRBPTRL	Current RxBD pointer	4
1B8328–1B833F	Reserved		

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8340	MRBLR0R1	Maximum Receive Buffer Length R0R1 Register	4
1B8344	MRBLR2R3	Maximum Receive Buffer Length R2R3 Register	4
1B8348–1B8383	Reserved		
1B8384	RBPTR0	RxBD Pointer 0	4
1B8388–1B838B	Reserved		
1B838C	RBPTR1	RxBD Pointer 1	4
1B8390–1B8393	Reserved		
1B8394	RBPTR2	RxBD Pointer 2	4
1B8398–1B839B	Reserved		
1B839C	RBPTR3	RxBD Pointer 3	4
1B83A0–1B8403	Reserved		
1B8404	RBASE0	Receive Descriptor Base Address 0	4
1B8408–1B840B	Reserved		
1B840C	RBASE1	Receive Descriptor Base Address 1	4
1B8410–1B8413	Reserved		
1B8414	RBASE2	Receive Descriptor Base Address 2	4
1B8418–1B841B	Reserved		
1B841C	RBASE3	Receive Descriptor Base Address 3	4
1B8420–1B84FF	Reserved		
1B8500	MACCFG1R	MAC Configuration 1 Register	4
1B8504	MACCFG2R	MAC Configuration 2 Register	4
1B8508	IPGIFGIR	Inter Packet Gap/Inter Frame Gap Register	4
1B850C	HAFDUPR	Half-Duplex Register	4
1B8510	MAXFRMR	Maximum Frame Register	4
1B8514–1B851F	Reserved		
1B8520	MIIMCFGR	MII Management Configuration Register	4
1B8524	MIIMCOMR	MII Management Command Register	4
1B8528	MIIMADDR	MII Management Address Register	4
1B852C	MIIMCONR	MII Management Control Register	4
1B8530	MIIMSTATR	MII Management Status Register	4
1B8534	MIIMINDR	MII Management Indicator Register	4
1B8538–1B853B	Reserved		
1B853C	IFSTATR	Interface Status Register	4
1B8540	MACSTADDR1R	Station Address Part 1 Register	4
1B8544	MACSTADDR2R	Station Address Part 2 Register	4
1B8548–1B867F	Reserved		
1B8680	TR64	Transmit and Receive 64-byte Frame Counter	4
1B8684	TR127	Transmit and Receive 65- To 127-byte Frame Counter	4
1B8688	TR255	Transmit and Receive 128- To 255-byte Frame Counter	4
1B868C	TR511	Transmit and Receive 256- To 511-byte Frame Counter	4
1B8690	TR1K	Transmit and Receive 512- To 1023-byte Frame Counter	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8694	TRMAX	Transmit and Receive 1024- To 1518-byte Frame Counter	4
1B8698	TRMGV	Transmit and Receive 1519- to 1522-byte Good VLAN Frame Count	4
1B869C	RBYT	Receive Byte Counter	4
1B86A0	RPKT	Receive Packet Counter	4
1B86A4	RFCS	Receive FCS Error Counter	4
1B86A8	RMCA	Receive Multicast Packet Counter	4
1B86AC	RBCA	Receive Broadcast Packet Counter	4
1B86B0	RXCF	Receive Control Frame Packet Counter	4
1B86B4	RXPF	Receive PAUSE Frame Packet Counter	4
1B86B8	RXUO	Receive Unknown OP code counter	4
1B86BC	RALN	Receive Alignment Error Counter	4
1B86C0	RFLR	Receive Frame Length Error Counter	4
1B86C4	RCDE	Receive Code Error Counter	4
1B86C8	RCSE	Receive Carrier Sense Error Counter	4
1B86CC	RUND	Receive Undersize Packet Counter	4
1B86D0	ROVR	Receive Oversize Packet Counter	4
1B86D4	RFRG	Receive Fragments Counter	4
1B86D8	RJBR	Receive Jabber Counter	4
1B86DC	RDRP	Receive Drop	4
1B86E0	TBYT	Transmit Byte Counter	4
1B86E4	TPKT	Transmit Packet Counter	4
1B86E8	TMCA	Transmit Multicast Packet Counter	4
1B86EC	TBCA	Transmit Broadcast Packet Counter	4
1B86F0	TXPF	Transmit Pause Control Frame Counter	4
1B86F4	TDFR	Transmit Deferral Packet Counter	4
1B86F8	TEDF	Transmit Excessive Deferral Packet Counter	4
1B86FC	TSCL	Transmit Single Collision Packet Counter	4
1B8700	TMCL	Transmit Multiple Collision Packet Counter	4
1B8704	TLCL	Transmit Late Collision Packet Counter	4
1B8708	TXCL	Transmit Excessive Collision Packet Counter	4
1B870C	TNCL	Transmit Total Collision Counter	4
1B8714–1B8717	Reserved		
1B8718	TJBR	Transmit Jabber Frame Counter	4
1B871c	TFCS	Transmit FCS Error Counter	4
1B8720	TXCF	Transmit Control Frame Counter	4
1B8724	TOVR	Transmit Oversize Frame Counter	4
1B8728	TUND	Transmit Undersize Frame Counter	4
1B872C	TFRG	Transmit Fragments Frame Counter	4
1B8730	CAR1	Carry Register One	4
1B8734	CAR2	Carry Register Two	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8738	CAM1	Carry Register One Mask	4
1B873C	CAM2	Carry Register Two Mask	4
1B8740–1B87FF	Reserved		
1B8800	IADDR0	Individual Address Register 0	4
1B8804	IADDR1	Individual Address Register 1	4
1B8808	IADDR2	Individual Address Register 2	4
1B880C	IADDR3	Individual Address Register 3	4
1B8810	IADDR4	Individual Address Register 4	4
1B8814	IADDR5	Individual Address Register 5	4
1B8818	IADDR6	Individual Address Register 6	4
1B881C	IADDR7	Individual Address Register 7	4
1B8820–1B887F	Reserved		
1B8880	GADDR0	Group Address Register 0	4
1B8884	GADDR1	Group Address Register 1	4
1B8888	GADDR2	Group Address Register 2	4
1B888C	GADDR3	Group Address Register 3	4
1B8890	GADDR4	Group Address Register 4	4
1B8894	GADDR5	Group Address Register 5	4
1B8898	GADDR6	Group Address Register 6	4
1B889C	GADDR7	Group Address Register 7	4
1B88A0–1B88FF	Reserved		
1B8900	PMD0	Pattern Match Data 0	4
1B8904–1B8907	Reserved		
1B8908	PMASK0	Pattern Mask 0 Register	4
1B890C–1B890F	Reserved		
1B8910	PCNTRL0	Pattern Control 0 Register	4
1B8914–1B8917	Reserved		
1B8918	PATTRB0	Pattern Attributes 0 Register	4
1B8920	PMD1	Pattern Match Data 1	4
1B8924–1B8927	Reserved		
1B8928	PMASK1	Pattern Mask 1 Register	4
1B892C–1B892F	Reserved		
1B8930	PCNTRL1	Pattern Control 1 Register	4
1B8934–1B8937	Reserved		
1B8938	PATTRB1	Pattern Attributes 1 Register	4
1B8940	PMD2	Pattern Match Data 2	4
1B8944–1B8947	Reserved		
1B8948	PMASK2	Pattern Mask 2 Register	4
1B894C–1B894F	Reserved		
1B8950	PCNTRL2	Pattern Control 2 Register	4
1B8954–1B8957	Reserved		

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8958	PATTRB2	Pattern Attributes 2 Register	4
1B8960	PMD3	Pattern Match Data 3	4
1B8964–1B8967	Reserved		
1B8968	PMASK3	Pattern Mask 3 Register	4
1B896C–1B896F	Reserved		
1B8970	PCNTRL3	Pattern Control 3 Register	4
1B8974–1B8977	Reserved		
1B8978	PATTRB3	Pattern Attributes 3 Register	4
1B8980	PMD4	Pattern Match Data 4	4
1B8984–1B8987	Reserved		
1B8988	PMASK4	Pattern Mask 4 Register	4
1B898C–01F898F	Reserved		
1B8990	PCNTRL4	Pattern Control 4 Register	4
1B8994–1B8997	Reserved		
1B8998	PATTRB4	Pattern Attributes 4 Register	4
1B89A0	PMD5	Pattern Match Data 5	4
1B89A4–1B89A7	Reserved		
1B89A8	PMASK5	Pattern Mask 5 Register	4
1B89AC–1B89AF	Reserved		
1B89B0	PCNTRL5	Pattern Control 5 Register	4
1B89B4–1B89B7	Reserved		
1B89B8	PATTRB5	Pattern Attributes 5 Register	4
1B89C0	PMD6	Pattern Match Data 6	4
1B89C4–1B89C7	Reserved		
1B89C8	PMASK6	Pattern Mask 6 Register	4
1B89CC–1B89CF	Reserved		
1B89D0	PCNTRL6	Pattern Control 6 Register	4
1B89D4–1B89D7	Reserved		
1B89D8	PATTRB6	Pattern Attributes 6 Register	4
1B89E0	PMD7	Pattern Match Data 7	4
1B89E4–1B89E7	Reserved		
1B89E8	PMASK7	Pattern Mask 7 Register	4
1B89EC–1B89EF	Reserved		
1B89F0	PCNTRL7	Pattern Control 7 Register	4
1B89F4–1B89F7	Reserved		
1B89F8	PATTRB7	Pattern Attributes 7 Register	4
1B8A00	PMD8	Pattern Match Data 8	4
1B8A04–1B8A07	Reserved		
1B8A08	PMASK8	Pattern Mask 8 Register	4
1B8A0C–1B8A0F	Reserved		
1B8A10	PCNTRL8	Pattern Control 8 Register	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8A14–1B8A17	Reserved		
1B8A18	PATTRB8	Pattern Attributes 8 Register	4
1B8A20	PMD9	Pattern Match Data 9	4
1B8A24–1B8A27	Reserved		
1B8A28	PMASK9	Pattern Mask 9 Register	4
1B8A2C–1B8A2F	Reserved		
1B8A30	PCNTRL9	Pattern Control 9 Register	4
1B8A34–1B8A37	Reserved		
1B8A38	PATTRB9	Pattern Attributes 9 Register	4
1B8A40	PMD10	Pattern Match Data 10	4
1B8A44–1B8A47	Reserved		
1B8A48	PMASK10	Pattern Mask 10 Register	4
1B8A4C–1B8A4F	Reserved		
1B8A50	PCNTRL10	Pattern Control 10 Register	4
1B8A54–1B8A57		Reserved	4
1B8A58	PATTRB10	Pattern Attributes 10 Register	
1B8A60	PMD11	Pattern Match Data 11	4
1B8A64–1B8A67	Reserved		
1B8A68	PMASK11	Pattern Mask 11 Register	4
1B8A6C–1B8A6F	Reserved		
1B8A70	PCNTRL11	Pattern Control 11 Register	4
1B8A74–1B8A77	Reserved		
1B8A78	PATTRB11	Pattern Attributes 11 Register	4
1B8A80	PMD12	Pattern Match Data 12	4
1B8A84–1B8A87	Reserved		
1B8A88	PMASK12	Pattern Mask 12 Register	4
1B8A8C–1B8A8F	Reserved		
1B8A90	PCNTRL12	Pattern Control 12 Register	4
1B8A94–1B8A97		Reserved	4
1B8A98	PATTRB12	Pattern Attributes 12 Register	4
1B8AA0	PMD13	Pattern Match Data 13	4
1B8AA4–1B8AA7	Reserved		
1B8AA8	PMASK13	Pattern Mask 13 Register	4
1B8AAC–1B8AAF	Reserved		
1B8AB0	PCNTRL13	Pattern Control 13 Register	4
1B8AB4–1B8AB7	Reserved		
1B8AB8	PATTRB13	Pattern Attributes 13 Register	4
1B8AC0	PMD14	Pattern Match Data 14	4
1B8AC4–1B8AC7	Reserved		
1B8AC8	PMASK14	Pattern Mask 14 Register	4
1B8ACC–1B8ACF	Reserved		

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1B8AD0	PCNTRL14	Pattern Control 14 Register	4
1B8AD4–1B8AD7		Reserved	4
1B8AD8	PATTRB14	Pattern Attributes 14 Register	4
1B8AE0	PMD15	Pattern Match Data 15	4
1B8AE4–1B8AE7	Reserved		
1B8AE8	PMASK15	Pattern Mask 15 Register	4
1B8AEC–1B8AEF	Reserved		
1B8AF0	PCNTRL15	Pattern Control 15 Register	4
1B8AF4–1B8AF7	Reserved		
1B8AF8	PATTRB15	Pattern Attributes 15 Register	4
1B8B00–1B8BF7	Reserved		
1B8BF8	DATTR	Default Attribute Register	4
1B8C00–1B8FFF	Reserved		
1B9000	MIIGSK_CFGR	MIIGSK Configuration Register	4
1B9004	MIIGSK_GPR	MIIGSK General-Purpose Register	4
1B9008	MIIGSK_ENR	MIIGSK Enable Register	4
1B900C	MIIGSK_SMII_SYNCDIR	MIIGSK SMII SYNC Direction Register	4
1B9010	MIIGSK_TIFBR	MIIGSK Transmit Inter-Frame Bits Register	4
1B9014	MIIGSK_RIFBR	MIIGSK Receive Inter-Frame Bits Register	4
1B9018	MIIGSK_ERIFBR	MIIGSK Expected Receive Inter-Frame Bits Register	4
1B901C	MIIGSK_IEVENT	MIIGSK SMII Interrupt Event Register	4
1B9020	MIIGSK_IMASK	MIIGSK SMII Interrupt Mask Register	4
1B9024–1BAFFF	Reserved		
1BB000	SCR	Stop Control Register	4
1BB008	SASR	Stop Acknowledge Status Register	4
1BC000	VIGR	Virtual Interrupt Generation Register	4
1BC008	VISR	Virtual Interrupt Status Register	4
1BC010	VNMIGR	Virtual NMI Generation Register	4
1BC018	GICR	GIC Interrupt Configuration Register	4
1BC020	GEIER	GIC External Interrupt Enable Register	4
1BC028	GCIER	GIC Core Interrupt Enable Register	4
1BC030	GISR	GIC Interrupt Status Register	4
1BC038–1BC0FF	Reserved		
1BC100	HSMPR0	Hardware Semaphore Register 0	4
1BC108	HSMPR1	Hardware Semaphore Register 1	4
1BC110	HSMPR2	Hardware Semaphore Register 2	4
1BC118	HSMPR3	Hardware Semaphore Register 3	4
1BC120	HSMPR4	Hardware Semaphore Register 4	4
1BC128	HSMPR5	Hardware Semaphore Register 5	4
1BC130	HSMPR6	Hardware Semaphore Register 6	4
1BC138	HSMPR7	Hardware Semaphore Register 7	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1BC140–1BC1FF	Reserved		
1BC200	PODR	Pin Open-Drain Register	4
1BC208	PDAT	Pin Data Register	4
1BC210	PDIR	Pin Direction Register	4
1BC218	PAR	Pin Assignment Register	4
1BC220	PSOR	Pin Special Options Register	4
1BC228–1BCFFF	Reserved		
1BD000	SCIBR	SCI Baud Rate Register	4
1BD008	SCICR	SCI Control Register	4
1BD010	SCISR	SCI Status Register	4
1BD018	SCIDR	SCI Data Register	4
1BD020–1BD027	Reserved		
1BD028	SCIDDR	SCI Data Direction Register	4
1BD030–1BDFFF	Reserved		
1BE000	DCR	DSI Control Register	4
1BE008	DSWBAR	DSI Sliding Window Base Address Register	4
1BE010	DIBAR9	DSI Internal Base Address Register Bank 9	4
1BE018–1BE01F	Reserved		
1BE020	DIBAR11	DSI Internal Base Address Register Bank 11	4
1BE028	DIAMR9	DSI Internal Address Mask Register Bank 9	4
1BE030–1BE037	Reserved		
1BE038	DIAMR11	DSI Internal Address Mask Register Bank 11	4
1BE040	DCIR	DSI Chip ID Register	4
1BE048	DDR	DSI Disable Register	4
1BE050	HRCW	Hard Reset Configuration Word	4
1BE058–1BE7FF	Reserved		
1BE050–1BE058	Reserved		
1BE060	DEXTBAR	DSI External Sliding Window Base Address Register	4
1BE068–1BE7FF	Reserved		
1BE800	DSR	DSI Status Register	4
1BE808	DER	DSI Error Register	4
1BE810–1BEFFF	Reserved		
1BF000	TCFRA0	Timer Configuration Register of Timer A0	4
1BF008	TCFRA1	Timer Configuration Register of Timer A1	4
1BF010	TCFRA2	Timer Configuration Register of Timer A2	4
1BF018	TCFRA3	Timer Configuration Register of Timer A3	4
1BF020	TCFRA4	Timer Configuration Register of Timer A4	4
1BF028	TCFRA5	Timer Configuration Register of Timer A5	4
1BF030	TCFRA6	Timer Configuration Register of Timer A6	4
1BF038	TCFRA7	Timer Configuration Register of Timer A7	4
1BF040	TCFRA8	Timer Configuration Register of Timer A8	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1BF048	TCFRA9	Timer Configuration Register of Timer A9	4
1BF050	TCFRA10	Timer Configuration Register of Timer A10	4
1BF058	TCFRA11	Timer Configuration Register of Timer A11	4
1BF060	TCFRA12	Timer Configuration Register of Timer A12	4
1BF068	TCFRA13	Timer Configuration Register of Timer A13	4
1BF070	TCFRA14	Timer Configuration Register of Timer A14	4
1BF078	TCFRA15	Timer Configuration Register of Timer A15	4
1BF080	TCMPA0	Timer Compare Register of Timer A0	4
1BF088	TCMPA1	Timer Compare Register of Timer A1	4
1BF090	TCMPA2	Timer Compare Register of Timer A2	4
1BF098	TCMPA3	Timer Compare Register of Timer A3	4
1BF0A0	TCMPA4	Timer Compare Register of Timer A4	4
1BF0A8	TCMPA5	Timer Compare Register of Timer A5	4
1BF0B0	TCMPA6	Timer Compare Register of Timer A6	4
1BF0B8	TCMPA7	Timer Compare Register of Timer A7	4
1BF0C0	TCMPA8	Timer Compare Register of Timer A8	4
1BF0C8	TCMPA9	Timer Compare Register of Timer A9	4
1BF0D0	TCMPA10	Timer Compare Register of Timer A10	4
1BF0D8	TCMPA11	Timer Compare Register of Timer A11	4
1BF0E0	TCMPA12	Timer Compare Register of Timer A12	4
1BF0E8	TCMPA13	Timer Compare Register of Timer A13	4
1BF0F0	TCMPA14	Timer Compare Register of Timer A14	4
1BF0F8	TCMPA15	Timer Compare Register of Timer A15	4
1BF100	TCRA0	Timer Control Register of Timer A0	4
1BF108	TCRA1	Timer Control Register of Timer A1	4
1BF110	TCRA2	Timer Control Register of Timer A2	4
1BF118	TCRA3	Timer Control Register of Timer A3	4
1BF120	TCRA4	Timer Control Register of Timer A4	4
1BF128	TCRA5	Timer Control Register of Timer A5	4
1BF130	TCRA6	Timer Control Register of Timer A6	4
1BF138	TCRA7	Timer Control Register of Timer A7	4
1BF140	TCRA8	Timer Control Register of Timer A8	4
1BF148	TCRA9	Timer Control Register of Timer A9	4
1BF150	TCRA10	Timer Control Register of Timer A10	4
1BF158	TCRA11	Timer Control Register of Timer A11	4
1BF160	TCRA12	Timer Control Register of Timer A12	4
1BF168	TCRA13	Timer Control Register of Timer A13	4
1BF170	TCRA14	Timer Control Register of Timer A14	4
1BF178	TCRA15	Timer Control Register of Timer A15	4
1BF180	TCNRA0	Timer Count Register of Timer A0	4
1BF188	TCNRA1	Timer Count Register of Timer A1	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1BF190	TCNRA2	Timer Count Register of Timer A2	4
1BF198	TCNRA3	Timer Count Register of Timer A3	4
1BF1A0	TCNRA4	Timer Count Register of Timer A4	4
1BF1A8	TCNRA5	Timer Count Register of Timer A5	4
1BF1B0	TCNRA6	Timer Count Register of Timer A6	4
1BF1B8	TCNRA7	Timer Count Register of Timer A7	4
1BF1C0	TCNRA8	Timer Count Register of Timer A8	4
1BF1C8	TCNRA9	Timer Count Register of Timer A9	4
1BF1D0	TCNRA10	Timer Count Register of Timer A10	4
1BF1D8	TCNRA11	Timer Count Register of Timer A11	4
1BF1E0	TCNRA12	Timer Count Register of Timer A12	4
1BF1E8	TCNRA13	Timer Count Register of Timer A13	4
1BF1F0	TCNRA14	Timer Count Register of Timer A14	4
1BF1F8	TCNRA15	Timer Count Register of Timer A15	4
1BF200–1BF37F	Reserved		
1BF380	TGCRA	Timer General Configuration Register of Timers Module A	4
1BF388	TERA	Timer Event Register of Timers Module A	4
1BF390	TIERA	Timer Interrupt Enable Register of Timers Module A	4
1BF398	TSRA	Timer Status Register of Timers Module A	4
1BF3A0–1BF3FF	Reserved		
1BF400	TCFRB0	Timer Configuration Register of Timer B0	4
1BF408	TCFRB1	Timer Configuration Register of Timer B1	4
1BF410	TCFRB2	Timer Configuration Register of Timer B2	4
1BF418	TCFRB3	Timer Configuration Register of Timer B3	4
1BF420	TCFRB4	Timer Configuration Register of Timer B4	4
1BF428	TCFRB5	Timer Configuration Register of Timer B5	4
1BF430	TCFRB6	Timer Configuration Register of Timer B6	4
1BF438	TCFRB7	Timer Configuration Register of Timer B7	4
1BF440	TCFRB8	Timer Configuration Register of Timer B8	4
1BF448	TCFRB9	Timer Configuration Register of Timer B9	4
1BF450	TCFRB10	Timer Configuration Register of Timer B10	4
1BF458	TCFRB11	Timer Configuration Register of Timer B11	4
1BF460	TCFRB12	Timer Configuration Register of Timer B12	4
1BF468	TCFRB13	Timer Configuration Register of Timer B13	4
1BF470	TCFRB14	Timer Configuration Register of Timer B14	4
1BF478	TCFRB15	Timer Configuration Register of Timer B15	4
1BF480	TCMPB0	Timer Compare Register of Timer B0	4
1BF488	TCMPB1	Timer Compare Register of Timer B1	4
1BF490	TCMPB2	Timer Compare Register of Timer B2	4
1BF498	TCMPB3	Timer Compare Register of Timer B3	4
1BF4A0	TCMPB4	Timer Compare Register of Timer B4	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1BF4A8	TCMPB5	Timer Compare Register of Timer B5	4
1BF40	TCMPB6	Timer Compare Register of Timer B6	4
1BF48	TCMPB7	Timer Compare Register of Timer B7	4
1BF4C0	TCMPB8	Timer Compare Register of Timer B8	4
1BF4C8	TCMPB9	Timer Compare Register of Timer B9	4
1BF4D0	TCMPB10	Timer Compare Register of Timer B10	4
1BF4D8	TCMPB11	Timer Compare Register of Timer B11	4
1BF4E0	TCMPB12	Timer Compare Register of Timer B12	4
1BF4E8	TCMPB13	Timer Compare Register of Timer B13	4
1BF4F0	TCMPB14	Timer Compare Register of Timer B14	4
1BF4F8	TCMPB15	Timer Compare Register of Timer B15	4
1BF500	TCRB0	Timer Control Register of Timer B0	4
1BF508	TCRB1	Timer Control Register of Timer B1	4
1BF510	TCRB2	Timer Control Register of Timer B2	4
1BF518	TCRB3	Timer Control Register of Timer B3	4
1BF520	TCRB4	Timer Control Register of Timer B4	4
1BF528	TCRB5	Timer Control Register of Timer B5	4
1BF530	TCRB6	Timer Control Register of Timer B6	4
1BF538	TCRB7	Timer Control Register of Timer B7	4
1BF540	TCRB8	Timer Control Register of Timer B8	4
1BF548	TCRB9	Timer Control Register of Timer B9	4
1BF550	TCRB10	Timer Control Register of Timer B10	4
1BF558	TCRB11	Timer Control Register of Timer B11	4
1BF560	TCRB12	Timer Control Register of Timer B12	4
1BF568	TCRB13	Timer Control Register of Timer B13	4
1BF570	TCRB14	Timer Control Register of Timer B14	4
1BF578	TCRB15	Timer Control Register of Timer B15	4
1BF580	TCNRB0	Timer Count Register of Timer B0	4
1BF588	TCNRB1	Timer Count Register of Timer B1	4
1BF590	TCNRB2	Timer Count Register of Timer B2	4
1BF598	TCNRB3	Timer Count Register of Timer B3	4
1BF5A0	TCNRB4	Timer Count Register of Timer B4	4
1BF5A8	TCNRB5	Timer Count Register of Timer B5	4
1BF5B0	TCNRB6	Timer Count Register of Timer B6	4
1BF5B8	TCNRB7	Timer Count Register of Timer B7	4
1BF5C0	TCNRB8	Timer Count Register of Timer B8	4
1BF5C8	TCNRB9	Timer Count Register of Timer B9	4
1BF5D0	TCNRB10	Timer Count Register of Timer B10	4
1BF5D8	TCNRB11	Timer Count Register of Timer B11	4
1BF5E0	TCNRB12	Timer Count Register of Timer B12	4
1BF5E8	TCNRB13	Timer Count Register of Timer B13	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1BF5F0	TCNRB14	Timer Count Register of Timer B14	4
1BF5F8	TCNRB15	Timer Count Register of Timer B15	4
1BF600–1BF77F	Reserved		
1BF780	TGCRB	Timer General Configuration Register of Timers Module B	4
1BF788	TERB	Timer Event Register of Timers Module B	4
1BF790	TIERB	Timer Interrupt Enable Register of Timers Module B	4
1BF798	TSRB	Timer Status Register of Timers Module B	4
1BF7A0–1CFFFF	Reserved		
1D0000	SIUMCR	SIU Module Configuration Register	4
1D0004	SYPCR	System Protection Control Register	4
1D0008–1D000B	Reserved		
1D000C	SWSR	Software Service Register (occupies lowest two bytes)	2
1D0010–1D0023	Reserved		
1D0024	BCR	Bus Configuration Register	4
1D0028	PPC_ACR	System Bus Arbiter Configuration Register	1 B
1D0029–1D001B	Reserved		
1D002C	PPC_ALRH	System Bus Arbitration Level Register (bus masters 0–7)	4
1D0030	PPC_ALRL	System Bus Arbitration Level Register (bus masters 8–15)	4
1D0034	LCL_ACR	Local Arbiter Configuration Register	1 B
1D0035–1D0037	Reserved		
1D0038	LCL_ALRH	Local Arbitration Level Register (bus masters 0–7)	4
1D003C	LCL_ALRL	Local Arbitration Level Register (bus masters 8–15)	4
1D0040	TESCR1	System Bus Transfer Error Status Control Register 1	4
1D0044	TESCR2	System Bus Transfer Error Status Control Register 2	4
1D0048	L_TESCR1	Local Bus Transfer Error Status Control Register 1	4
1D004C–1D0057	Reserved		
1D0058	LGTDTEA	Local Bus GTD (Global TDM DMA) Transfer Error Address	4
1D005C	LGTDTEM	Local Bus GTD (Global TDM DMA) Transfer Error TDMNUM_TR	1 B
1D005D–1D005F	Reserved		
1D0060	PDMTEA	System Bus DMA Transfer Error Address	4
1D0064	PDMTER	System Bus DMA Transfer Error RQNUM	1 B
1D0065–1D0067	Reserved		
1D0068	LDMTEA	Local Bus DMA Transfer Error Address	4
1D006C	LDMTER	Local Bus DMA Transfer Error RQNUM	1 B
1D006D–1D00FF	Reserved		
1D0100	BR0	Base Register Bank0	4
1D0104	OR0	Option Register Bank0	4
1D0108	BR1	Base Register Bank1	4
1D010C	OR1	Option Register Bank1	4
1D0110	BR2	Base Register Bank2	4
1D0114	OR2	Option Register Bank2	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1D0118	BR3	Base Register Bank3	4
1D011C	OR3	Option Register Bank3	4
1D0120	BR4	Base Register Bank4	4
1D0124	OR4	Option Register Bank4	4
1D0128	BR5	Base Register Bank5	4
1D012C	OR5	Option Register Bank5	4
1D0130	BR6	Base Register Bank6	4
1D0134	OR6	Option Register Bank6	4
1D0138	BR7	Base Register Bank7	4
1D013C	OR7	Option Register Bank7	4
1D0140–1D0147	Reserved		
1D0148	BR9	Base Register Bank9	4
1D014C	OR9	Option Register Bank9	4
1D0150	BR10	Base Register Bank10	4
1D0154	OR10	Option Register Bank10	4
1D0158	BR11	Base Register Bank11	4
1D015C	OR11	Option Register Bank11	4
1D0160–1D0167	Reserved		
1D0168	MAR	Memory Address Register	4
1D016C–1D016F	Reserved		
1D0170	MAMR	Machine A Mode Register	4
1D0174	MBMR	Machine B Mode Register	4
1D0178	MCMR	Machine C Mode Register	4
1D017C–1D0183	Reserved		
1D0184	MPTPR	Memory Refresh Timer Prescaler	2
1D0186–1D0187	Reserved		
1D0188	MDR	Memory Data Register	4
1D018C–1D018F	Reserved		
1D0190	PSDMR	System Bus SDRAM Mode Register	4
1D0194	Reserved		
1D0198	PURT	System Bus-Assigned UPM Refresh Timer	1 B
1D0199–1D019B	Reserved		
1D019C	PSRT	System Bus-Assigned SDRAM Refresh Timer	1 B
1D019D–1D01A7	Reserved		
1D01A8	IMMR	Internal Memory Map Register	4
1D01AC–1D021F	Reserved		
1D0220	TMCNTSC	Time Counter Status and Control Register	2
1D0222–1D0223	Reserved		
1D0224	TMCNT	Time Counter Register	4
1D0228–1D022B	Reserved		
1D022C	TMCNTAL	Time Counter Alarm Register	4

Table 8-10. DSI Address Map (0x000000–0x1FFFFF) (Continued)

Address	Abbreviation	Name	Size in Bytes
1D0230–1D023F	Reserved		
1D0240	PISCR	Periodic Interrupt Status and Control Register	2
1D0242–1D0243	Reserved		
1D0244	PITC	Periodic Interrupt Count Register	4
1D0248	PITR	Periodic Interrupt Timer Register	4
1D024C–1D06FF	Reserved		
1D0700	DCHCR0	DMA Channel 0 Configuration Register	4
1D0704	DCHCR1	DMA Channel 1 Configuration Register	4
1D0708	DCHCR2	DMA Channel 2 Configuration Register	4
1D070C	DCHCR3	DMA Channel 3 Configuration Register	4
1D0710	DCHCR4	DMA Channel 4 Configuration Register	4
1D0714	DCHCR5	DMA Channel 5 Configuration Register	4
1D0718	DCHCR6	DMA Channel 6 Configuration Register	4
1D071C	DCHCR7	DMA Channel 7 Configuration Register	4
1D0720	DCHCR8	DMA Channel 8 Configuration Register	4
1D0724	DCHCR9	DMA Channel 9 Configuration Register	4
1D0728	DCHCR10	DMA Channel 10 Configuration Register	4
1D072C	DCHCR11	DMA Channel 11 Configuration Register	4
1D0730	DCHCR12	DMA Channel 12 Configuration Register	4
1D0734	DCHCR13	DMA Channel 13 Configuration Register	4
1D0738	DCHCR14	DMA Channel 14 Configuration Register	4
1D073C	DCHCR15	DMA Channel 15 Configuration Register	4
1D0740–1D077F	Reserved		
1D0780	DIMR	DMA Internal Mask Register	4
1D0784	DSTR	DMA Status Register	4
1D0788	DTEAR	DMA TEA Status Register	1
1D0789–1D078B	Reserved		
1D078C	DPCR	DMA Pin Configuration Register	1
1D078D–1D078F	Reserved		
1D0790	DEMR	DMA External Mask Register	4
1D0794–1D07FF	Reserved		
1D0800–1D0BFF	DCPRAM	DMA Channel Parameter RAM	1024
1D0C00–1D0C87	Reserved		
1D0C88	SCMSR	System Clock Mode Register	4
1D0C90	RSR	Reset Status Register	4
1D0C94–1EFFFF	Reserved		
1F0000–1FFFFF	External memory access offset		

8.9 Pseudo Command Address Space

The Pseudo Command Address Space can be viewed by the boot code when the boot is done through the I^{2}C, TDM, or UART interface. **Table 8-11** lists the registers in this address space.

Table 8-11. Pseudo Command Memory Map (0x01FC0000–0x01FC00FF)

Address	Acronym	Name	Size in Bytes
0x01FC0000	BPCR	Boot Pseudo Command Register	4
0x01FC0008	I2CHPR	I^{2}C High Period Register	4
0x01FC000C	I2CHLPR	I^{2}C Half-Low Period Register	4
0x01FC0004–0x01FC00FF	—	Reserved	252

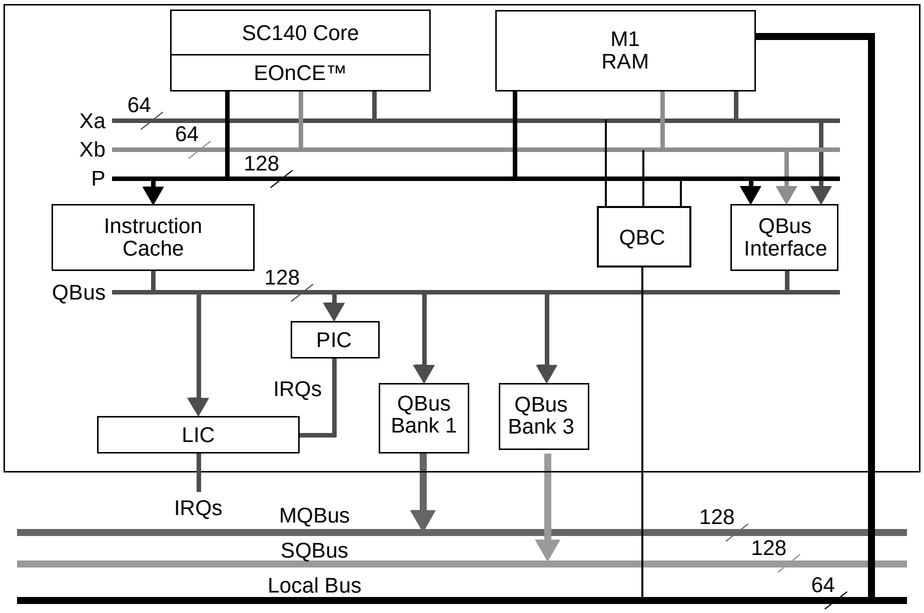
8.10 Notes

- Reading from an address that is not mapped or that is reserved may have the following effects:
  - — A time-out error. Refer to **Section 4.1.5**, *SIU and General Software Watchdog Timers*, on page 4-6.
  - — Read data not valid.
- Data stored in the internal memories is accessed by the extended core as big-endian.
- Bytes within registers can be accessed unless specified otherwise.

Extended Core 9

Each MSC8113 SC140 core is embedded in an extended core system that enhances the power of the SC140 core and provides a simple interface to each SC140 core. The extended core system includes:

- SC140 core
- M1 memory (224 KB)
- Instruction cache (16 KB, 16-way)
- QBus system
- Programmable interrupt controller (PIC)
- Local interrupt controller (LIC)
- Extended core power saving modes



Notes: 1. The arrows show the data transfer direction.
2. The QBus interface includes a bus switch, write buffer, fetch unit, and a control unit that defines four QBus banks. In addition, the QBC handles internal memory contentions.

Figure 9-1. Extended Core System

The remainder of this chapter describes each of these extended core components.

9.1 SC140 DSP Core

The innovative architecture of the SC140 DSP core addresses the key market needs of the next-generation DSP applications, mainly in the field of wireline and wireless infrastructure and subscriber communications. This flexible DSP core supports compute-intensive applications by providing high performance, low power, efficient compile, and high code density. The SC140 core efficiently deploys a novel variable-length execution set (VLES) execution model, maximizing parallelism by allowing multiple address generation and data arithmetic logic units to execute multiple operations in a single clock cycle. This section provides an overview of the key features and main modules of the SC140 core, as well as the programming model and instruction set list.

Note: The information in this chapter is based on Revision 3 of the *SC140 DSP Core Reference Manual*. To get updates or later revisions of this manual, visit the Freescale Web site shown on the back cover of this manual.

The 16-bit SC140 core packs four data arithmetic-logic execution units (ALUs), each consisting of a multiply-accumulate unit (MAC), a logic unit, and a bit field unit (BFU), which also serves as a barrel shifter. This number of MAC units yields the performance needed for essential DSP tasks such as finite impulse response (FIR) and infinite impulse response (IIR) filters and fast Fourier transforms (FFTs). In addition to the four data execution units, the core contains two address arithmetic units (AAUs), one bit manipulation unit (BMU) and one branch unit. Overall, the SC140 can issue and execute up to six instructions per clock—for example, four independent arithmetic instructions and two pointer-related instructions (such as moves or other operations on addresses). At a clock speed of 400 MHz, the SC140 can therefore execute 1600 true DSP MIPS—1600 million multiply-accumulate operations per second (MMACS), together with associated data movement functions and pointer updates.

The SC140 can sustain this high performance over time, owing to the flexibility of its data execution units. The four data execution units can operate simultaneously in any combination. For example, the core can execute four multiply-accumulate operations in a single clock, or one MAC, two arithmetic/logical operations and one bit field operation. All four data ALUs are identical. This permits great flexibility in the assignment and execution of instructions, increasing the likelihood that four execution units can be kept busy on any given cycle and enabling programs to take better advantage of the parallel architecture of the core.

Note: See **Chapter 2**, *SC140 Core Overview* for details on the SC140 core.

9.2 Extended Core Memory (M1)

A 224 KB SRAM memory unified for program and data is included within each extended core system. It is a zero wait state memory that operates at core frequency. The memory system includes the memory and the QBus control unit (QBC). The M1 memory divided into seven groups of 32 KB each. Each group contains eight 4 KB modules (see **Figure 9-2**). Each module is organized as an array of 128 rows, each 256 bit wide. Complex interleaving in the modules minimizes contentions.

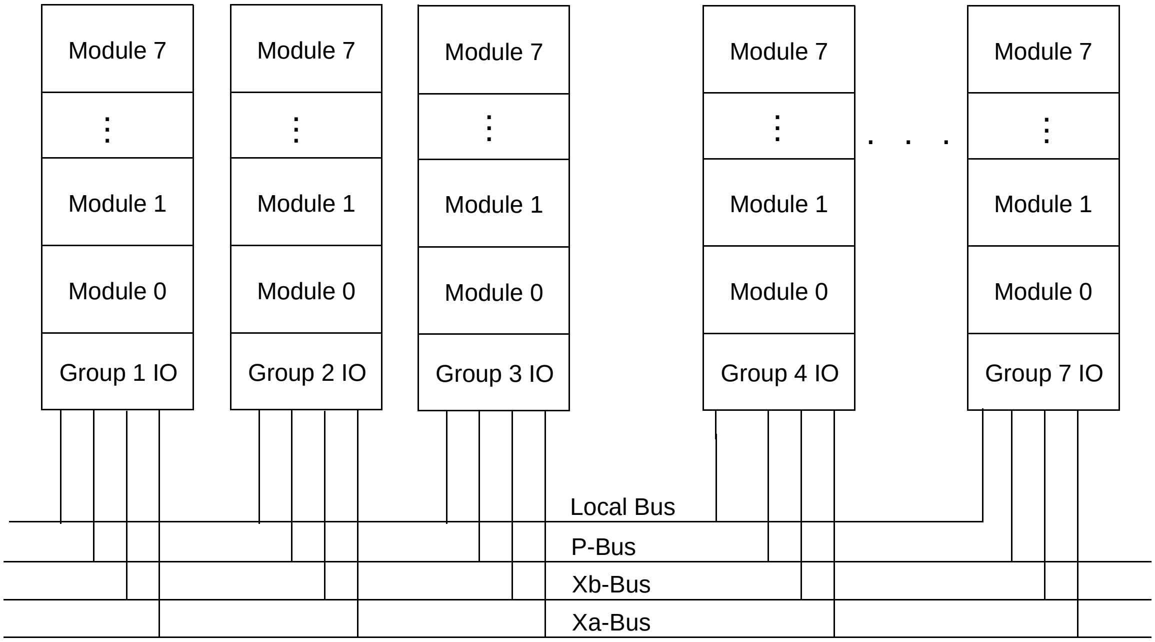


Figure 9-2. Logical Memory Organization

Each memory group has four ports, three of which enable access from the SC140 core buses (P-bus and the two data buses Xa and Xb), and one of which is the L port. The L port connects directly to the local bus. The four 24-bit address memory ports include:

- P (program) 128-bit data read.
- Xa (data) 64-bit read and write.
- Xb (data) 64-bit read and write.
- L (data) 64-bit read and write.

All data ports (Xa, Xb, and L) are byte addressable and can have accesses of different sizes—byte, word (16 bits), long word (32 bits), and quad word (64 bits). The P port is accessed in 128-bit resolution only. The priority of accesses between the buses is as follows:

- Local bus.
- Program fetch.
- Xa read.
- Xb read.
- Xa write.
- Xb write.

The bus control unit is responsible for the following tasks:

- Handling internal memory contentions.
- Freezing the address bus during contentions.
- Freezing the SC140 core during memory contentions.
- Handling bus access exceptions.
- Acknowledging atomic (reservation) instructions to the SC140 core.
- Protecting atomic instructions and internal memory.
- Disabling the SC140 core in ABIST or Disable Core mode.

9.2.1 Memory Organization

All M1 memory groups are connected to the SC140 core main buses (Xa bus, Xb bus, and P-bus). The L port of the memory is connected directly to the local bus.

9.2.1.1 Memory Groups

Each memory group contains an I/O group buffer and eight modules of 4 KB each. Each module is 256 bits wide and has 128 rows. Each group has four ports (Xa, Xb, P, L) connected to the I/O group buffer. From this buffer, two buses connect to the modules (see **Figure 9-3**). The eight modules of a memory group are interleaved so that the next row of an address is always at the next module (see **Figure 9-4**). This interleaving makes efficient use of the two data buses and minimizes memory contentions.

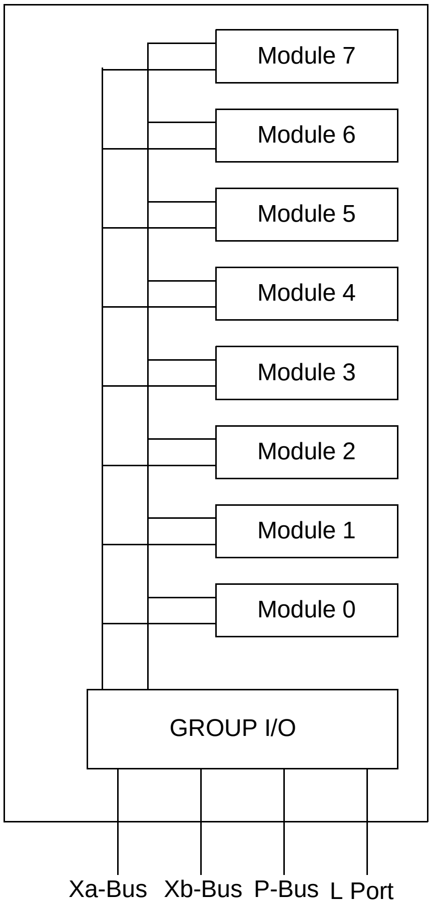


Figure 9-3. Memory Group

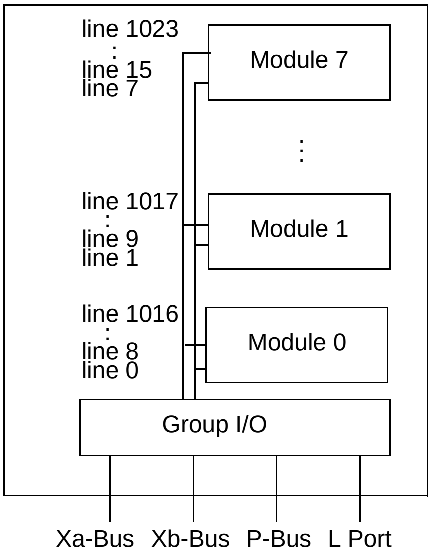


Figure 9-4. Memory Interleaving

9.2.1.2 Memory Contention and Priority

Each memory group has four I/O ports: L, P, Xa, and Xb, which are accessed by the local bus, internal program bus, and the two internal data buses, respectively. The two data buses, connected to ports Xa and Xb, can accesses the same memory group simultaneously. However, the local bus (L port) and the program bus (P port) cannot access the same 32 KB memory group simultaneously; nor can either bus access the same 32 KB memory group as a data bus. If such access is attempted in the same cycle, a contention causes a lost cycle in the SC140 core. Except for dual data bus access, each memory module serves only one bus in a cycle. The use of memory interleaving and the fact that there is no contention on the same line of a module minimize the probability of a contention when close data is addressed with the Xa and Xb buses. In summary:

- Group contention occurs between the X, P, and L ports. There is no contention between Xa and Xb to the same group.
- Module contention occurs between Xa and Xb. There is no contention on the same line.

9.2.2 Errors and Exceptions

The QBus control unit detects contentions and errors on the internal core buses and outputs exception signals to the interrupt controller. See **Chapter 17**, *Interrupt Processing*.

9.2.2.1 Errors

Errors generate interrupts using the $\overline{\text{NMI}}$ inputs to the interrupt controller, as follows:

- *Bus Error*. When an address on the internal bus does not match any physical address in the internal memory space, a bus error occurs and $\overline{\text{NMI4}}$ is generated internally. Such accesses include implicit accesses by pipelined program fetches. In rare cases, the bus error interrupt does not occur after an access to a non-valid address in internal memory. If this problem occurs during software development, place a **debug** instruction before the NMI4 handler and transfer control to the debugger to search for the root cause of the problem.
- *Misaligned program*. SC140 instructions are 16 bits (two bytes) and must be aligned. If the address on the program bus is not 16-bit aligned, a misaligned program error occurs and $\overline{\text{NMI3}}$ is generated internally.

9.2.2.2 Exceptions

Exceptions assert the interrupt request lines of the interrupt controller and can be masked. The contention exceptions are mainly used for debug and profiling and can be masked otherwise. The exceptions generate the following interrupts:

- *Misaligned data*. When the address on the data buses (Xa or Xb) is misaligned with the data size, a misaligned data exception occurs and $\overline{\text{IRQ13}}$ is generated internally.

- *X and P Contention*. When there is contention between one of the data buses (Xa or Xb) and the program bus (the program bus and one of the data buses address the same group in the same cycle), an X-P contention exception occurs and $\overline{\text{IRQ12}}$ is generated internally.
- *Local Bus Contention*. When there is an local bus contention—that is, the local bus and one of the other buses (Xa, Xb, or P) address the same group in the same cycle—a local bus contention exception occurs and $\overline{\text{IRQ11}}$ is generated internally.

9.3 Extended QBus System

The EQBS is the SC140 extended core interface to the MSC8113 system through the QBus. The module handles the SC140 core and the instruction cache requests, bringing the data on the QBus. As **Figure 9-5** shows, the EQBS consists of a bus switch, a write buffer, a fetch unit, a control unit, and the banks to handle the communication with the slaves and all EQBS registers. The QBus masters are fetch unit, write buffer, and bus switch. The CU is the arbiter for the QBus masters. The QBus Controller (QBC) is the unit handling internal memory contentions. It snoops the activity on the buses connected to the M1 memory and freezes the SC140 core and address bus activity if contention is detected. It creates the atomic instruction acknowledge to the SC140 core during the reservation process.

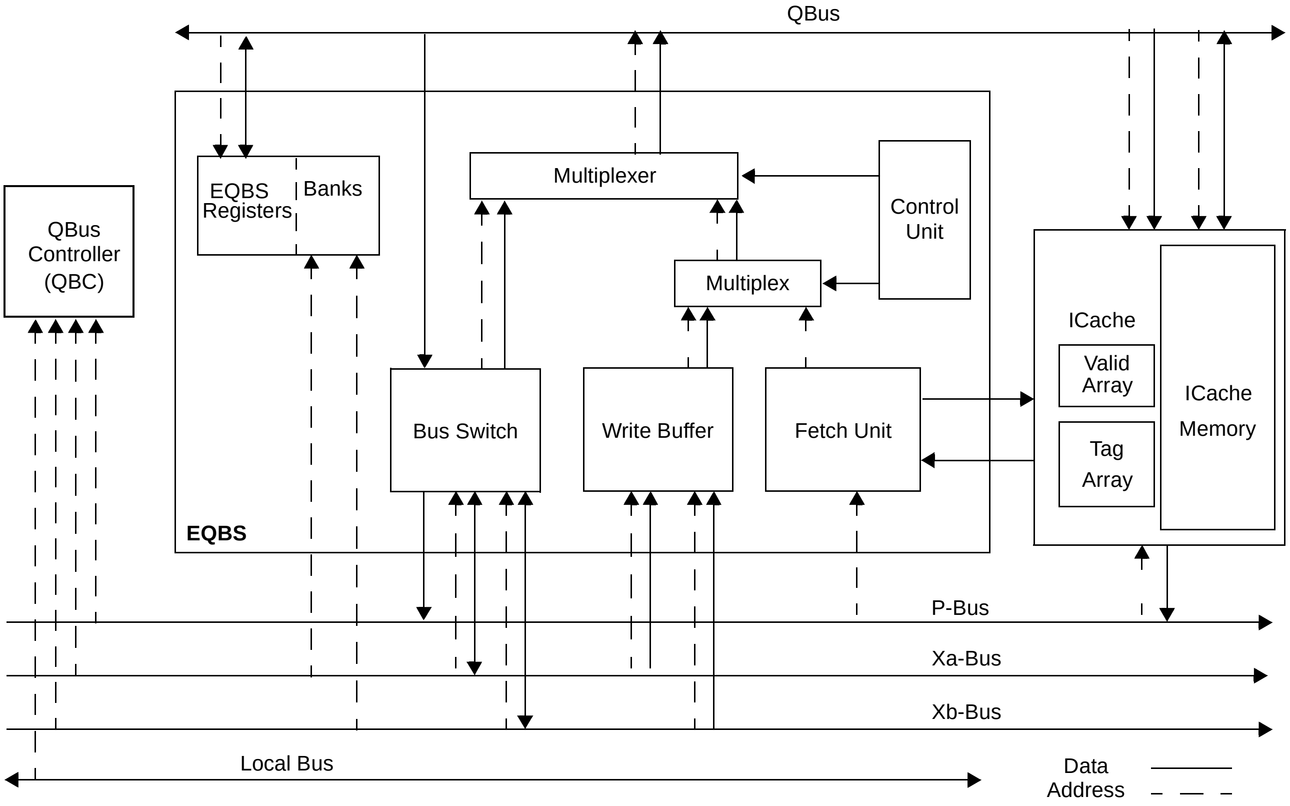


Figure 9-5. EQBS Block Diagram

9.3.1 Architecture

The EQBS enables the SC140 core to communicate with external devices efficiently. It handles the switching between the three core buses and QBus. SC140 core accesses that apply to memory space above the internal memory (QBus baseline = 0x00F00000) are transferred to the QBus through the EQBS. The EQBS also connects to the instruction cache and initiates requests for cache updates in order to improve the hit ratio. The EQBS operates at the same frequency as the SC140 core.

The bus switch handles all data read operations above the QBus baseline, write operations when the write buffer is disabled, and atomic (read-modify-write) operations.

The write buffer has a four-entry buffer that enables the SC140 core to write out to the external memory with no freeze. A write access above the QBus baseline goes to the buffer while the SC140 core continues execution. The write buffer operates like a FIFO, except for two cases:

- *Immediate accesses*. If a write access is for an immediate memory area, according to the data areas in the Banks, the access bypasses all other in-buffer commands. The write buffer halts the SC140 core in an immediate access.
- *Immediate access with no freeze*. This access is handled the same way as an immediate but with no freeze to the SC140 core. In the data areas registers the user can set a data area to immediate or immediate no freeze (see the *Data Area Registers* on **page 9-22**).

The buffer transfers its content to the destination without further SC140 core intervention. Exact timing of the transfer depends on the traffic on the QBus. In the following cases, the write buffer halts the SC140 core to protect data from running over:

- *Write buffer is full*. The write buffer is already full, and another write access is issued.
- *Immediate access*. An immediate write executes before all other writes in the write buffer queue and in order with the read access (the read access in the next cycle executes after the write immediate). To define a memory space as immediate, one should program the data area registers. However, the address range of the higher half of Bank 0 (from 0x00F08000–0x00F0FFFF) is always defined as a write immediate (this range includes the EQBS, ICache, PIC registers). This definition ensures in-order execution. For details on immediate write programming, see **Table 9-4 *Programming Data Area Base and Size,*** on page 9-15.
- *Flush of write buffer content*. The write buffer requests top priority by asserting the “flush” flag. A flush writes all contents of the write buffer to the QBus. It is initiated in four cases:
  - — *A read from an address within the write buffer*. To keep the logical constancy of commands, the write operation should execute before the read from the same address. Upon detecting a read from an address held in the write buffer, the write buffer flushes all its contents and only then execute the read.

— *A flush command in software*. To activate a software flush, one should issue a read from a pre-defined address. This address is set in hardware and is not programmable (see *Write Buffer Software Flush Register* on **page 9-21**).

— *A watchdog flush*. If the write buffer attempts to transfer an access for some time but does not get acknowledgment, a flush is initiated to give the write buffer the highest priority on the QBus. The watchdog expiration time is programmed in the *Write Buffer Control Register* (see **page 9-21**).

— The write buffer is turned off with the wb_off bit while the write buffer is not empty.

The write buffer does not handle accesses if there are atomic operations or if the *Write Buffer Off* flag is asserted (see **page 9-21**).

The write buffer can be turned off so that writes execute in order of appearance or to execute an atomic write command. See **Section 9.3.6**, *Reservation Process*.

The SC140 core uses the QBus for external accesses. It has a relatively simple protocol. The QBus features are as follows:

- Operates at SC140 core frequency.
- 32-bit address.
- 64-bit data write from the SC140 core.
- 128-bit data reads from the SC140 core.
- Pipeline between the address and data phases.
- Supports slaves with different response times.
- Holds four banks with different chip selects and 64 KB resolution. Three of the banks are configurable and the fourth is a default bank supporting all non-defined accesses.

9.3.1.1 Fetch Unit

The fetch unit handles all program accesses to external address space and conducts all ICache update activity. It accelerates SC140 core performance in accessing external memory by initiating cache updates of data needed with high probability in a sequential code (prefetch). Fetch unit operation includes two major phases: fetch and prefetch.

- *Fetch*. Triggered by a cache “miss.” Upon detecting a “miss,” the fetch unit initiates an access on the QBus to bring the data into the SC140 core.
- *Prefetch*. Includes cache updates with data of sequential addresses following the “miss.” It is triggered by the fetch and occurs in blocks for efficient use of the external memory and associated interfaces. The prefetch ends when it reaches the end of cache line or when a new “miss” access occurs. The prefetch phase greatly improves cache performance in a system running a program with sequential code because in the next access to the code area, the data is probably already in the cache. The prefetch occurs in two phases:

— *Phase I, prefetch of a block*. The prefetch occurs on the QBus following the first "miss." A block is defined as a programmable number of fetch sets (128 bits) that the fetch unit brings without interference with the "miss" and with a high priority. A block is the minimal unit of data in the first stage of fetch and prefetch.

— *Phase II, prefetch of data to end of cache line*. A cache line is 256 bytes long. The prefetch initiates cache updates from the address following the end of phase 1 until the end of the cache line. This phase of prefetching can be turned off to reduce fetch unit traffic on the QBus, making way for other devices on the QBus.

Figure 9-6 shows the fetch and prefetch and the relationship between the block and line. In this example, the block size is 4 (4 × 128 bits).

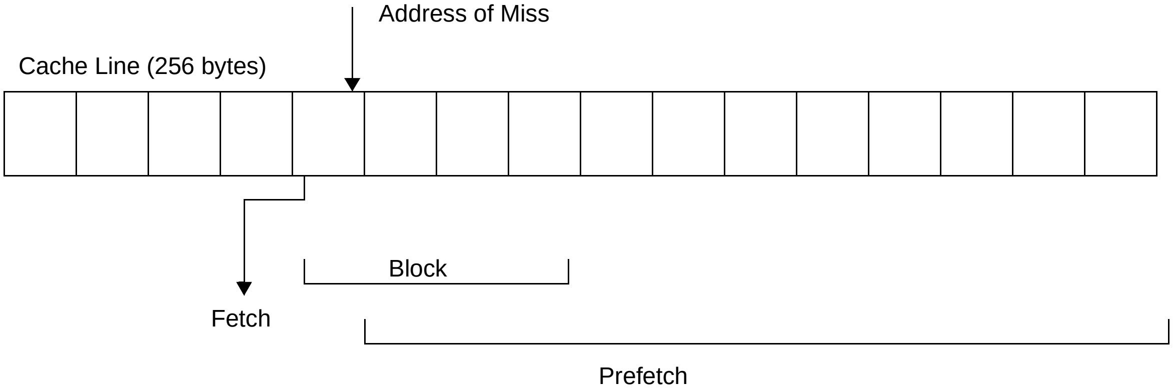


Figure 9-6. Fetch and Prefetch

The block size is programmable in the Banks (see EQBS programing model). The block size reflects the importance given to the fetch process on the QBus. A block is an inseparable unit on the QBus. Each "miss" causes a fetch + prefetch of a block (at a minimum), so setting a large block causes each "miss" to bring a lot of data into the cache. The SC140 core may not necessarily need this data, possibly delaying other operations (for example, a read) that occur during a block. However, it can also bring data that the SC140 core needs without being delayed by other operations.

The fetch unit controls the cache updates according to the cacheable memory area definition (see **Section 9.3.8**, *Instruction Cacheable Area*, on page 9-17). Access to a non-cacheable area does not result in cache activity. The cache serves a "hit" with 0 wait states, while the EQBS handles a "miss." The fetch unit identifies misses to an address included in the fetch unit prefetch that is being loaded. This identification process is called prefetch "hit." It saves an extra access on the system buses. This feature is effective when sequential data is transferred from an external memory. The fetch unit can stop an incomplete prefetch access. If a prefetch is executing and a new P access that is a "miss" is waiting on the SC140 core buses, the fetch unit stops the prefetch immediately and puts the new miss on the QBus.

The fetch unit transfers an attribute on the QBus to indicate whether a transaction is a block or prefetch. An upper arbiter that is a slave on the QBus uses this attribute to prioritize the access. During a block access (“miss and the following first step of prefetch) the priority is high. The priority is high also during a prefetch which is a prefetch “hit”.

9.3.2 QBus Execution Order

The QBus is the bus that connects the extended core to the MSC8113 system. All SC140 core accesses above the QBus base line and prefetch to the cache are transferred on the QBus. The EQBS prioritizes transfers on the QBus. The control unit sub-block in the EQBS determines the execution order on the QBus according to the following priorities in descending order:

- P-bus access (not prefetch)
- XA-bus read
- XB-bus read
- XA-bus write immediate or immediate with no freeze
- XB-bus write immediate or immediate with no freeze
- XA-bus write
- XB-bus write
- Prefetch

The addresses are serviced on the QBus according to their priority. However, for a write buffer flush, the write buffer gets the highest priority within accesses of the same core cycle.

9.3.3 QBus Banks

The bus has a single master (EQBS) and multiple slaves that are divided into four banks. There can be more than one slave on each bank, and the slaves are divided according to the address space. Each of the four QBus banks, Banks 0–3, has its own base address and size. Bank 0 has the highest priority. If an address matches more than one bank, it is directed to the bank with the highest priority. Each bank has a chip-select (CS) and some predefined dedicated signals for specific slaves. Each bank performs different functions, as follows:

- Bank 0 contains the addresses of the DSP peripherals, PIC, ICache, and the EQBS registers. The upper half of Bank 0 (which contains the ICache, EQBS, and PIC registers) is defined to execute all writes as immediate. This ensures that system register updates are not delayed in the write buffer.
- Bank 1 can be used to put any slave or interface on the QBus. In the MSC8113 it is used for M2 (shared) memory.
- Bank 2 is not used.
- Bank 3 (the default bank) contains the system interface to the system bus.

Bank 0 always operates in zero-wait state mode, and the other banks function in acknowledge mode. **Table 9-1** shows the configuration of the various banks.

Table 9-1. Bank Configuration

Bank Identification	Bank Contents	Operation Mode	Reset Value

Bank 0	DSP peripherals on the QBus, Icache registers and bank registers	Zero-wait state mode	base0 = 0x00F0
			mask0 = 0xFFFF

Bank 1	In the MSC8113 it is used for M2 (shared) memory	Acknowledge mode	base1 = 0x0100
			mask1 = 0xFF80
			Note: The mask1 value after reset is 0xFF00, but the boot program changes it to 0xFF80.

Bank 2	Reserved		base2 = 0x0000
			mask2 = 0xFFFF

Bank 3 (Default Bank)	The system bus. Any address that is not located in the other banks.	Acknowledge mode	No reset value

9.3.4 Bank Registers

Banks 0–2 each have two registers, a base address register and a mask register. The base address register is a 16-bit register that defines the 16 msb of the starting address of the bank in memory. The mask register determines the bank size. Each bit in the 16-bit mask register refers to the corresponding bit in the address. If the bit is 1, the corresponding bit in the address is compared to the base register value.

9.3.5 Bank Addressing

Because Bank 3 is the default bank, its address range and base address are not configured directly. Instead, its implied base address starts at the first location following the last address in Bank 1 and extends to the last location in addressable memory. All accesses to addresses not configured as part of Bank 0 or Bank 1 are directed to Bank 3. For Banks 0 and 1, the address range for each bank is defined as a combination of the base address register and the mask register. The functions of these two registers are interrelated in that the value entered into the base address register must be a multiple of the area size defined by the mask register. The mask register defines the size of the bank address range as defined in **Table 9-2**.

Table 9-2. Mask Register Value Definitions

Mask Value	Binary Mask Value	Defined Bank Size	Valid Base Register Value
0xFFFF	1111111111111111	64 KB	0xXXXX (any value)
0xFFFE	1111111111111110	128 KB	0xXXXE
0xFFFC	1111111111111100	256 KB	0xXXXC
0xFFF8	1111111111111000	512 KB	0xXXX8
0xFFF0	1111111111110000	1 MB	0xXXX0
0xFFE0	1111111111100000	2 MB	0xXXE0

Table 9-2. Mask Register Value Definitions

Mask Value	Binary Mask Value	Defined Bank Size	Valid Base Register Value
0xFFC0	1111111111000000	4 MB	0xXXC0
0xFF80	1111111110000000	8 MB	0xXX80
0xFF00	1111111100000000	16 MB	0xXX00
0xFE00	1111111000000000	32 MB	0xXE00
0xFC00	1111110000000000	64 MB	0xXC00
0xF800	1111100000000000	128 MB	0xX800
0xF000	1111000000000000	256 MB	0xX000
0xE000	1110000000000000	512 MB	0xE000
0xC000	1100000000000000	1 GB	0xC000
0x8000	1000000000000000	2 GB	0x8000
0x0000	0000000000000000	Do not use. This is below the QBus baseline.	
Note: The 16 lsb of the address can be any value from 0x0000–0xFFFF. Therefore, the minimum bank size is 64 KB. All other bank sizes (based on the mask value) are multiples of 64 KB.			

Figure 9-7 shows the location of a bank in memory.

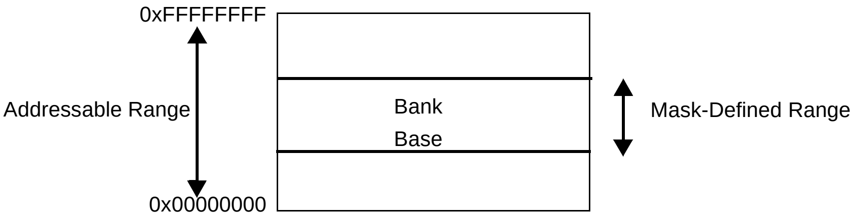


Figure 9-7. Bank Memory Location

At reset, the base address of Bank 0 is initialized to the QBus baseline (that is, the base address register is set to 0x00F0) and the size is set to 64 KB (that is, the mask value is set to 0xFFFF). The base address of Bank 1 is initialized to 0x0100 and the size is set at 1 MB. This is not the size of the memory space, but the limit of the address range. The values in the bank registers can be rewritten after reset. If new values are written to the bank registers, you must was at least two clock cycles before the new values are available for use. As the default bank, Bank 3 cannot be masked.

To check for a bank match, an AND operation is performed between the address on the QBus and the mask register. For Bank 0, the result is compared to base0 *and* mask0. For Bank 1, the result is compared to the appropriate base. If a match occurs, access to the bank is generated. If there is an overlap between banks, the match occurs in the bank with the highest priority. **Table 9-3** shows examples of bank address and mask register values.

Table 9-3. Example Bank Address and Mask Register Values

Base Register	Mask Register	Bank Size	Address Range for a Match
0x001F	0xFFFF	64 KB	0x001F0000–0x001FFFFF
0x001C	0xFFFC	256 KB	0x001C0000–0x001FFFFF

Table 9-3. Example Bank Address and Mask Register Values (Continued)

Base Register	Mask Register	Bank Size	Address Range for a Match
0x001F	0xFFFC	—	Null (no possible match), The base is not a multiple of the size.
Notes: 1. The QBus baseline in this example is 15 MB, (0xF00000)			

For a bank to operate correctly, its size as defined in the mask register must be a multiple of its base address. For example, if the bank size is 1 MB (the mask register has a value of 0xFFF0), the twenty least significant bits of the base address of the bank must be zeros.

Note: If QBUSBR0 (the QBus base register for Bank 0) is set to a value lower than the QBus baseline, the bank registers may become inaccessible. This situation can be corrected only by resetting the processor.

9.3.6 Reservation Process

The reservation (read-modify-write) operation in the SC140 occurs via the **bmtset** instruction. The **bmtset** instruction tests the destination, sets the true (T) bit if each bit that is 1 in the mask is also 1 in the destination, and sets every bit in the destination (register or memory) that has a value of 1 in the mask. This operation involves two cycles:

- A read cycle, during which an atomic signal is sent on the bus
- A write cycle

The success of the write operation is indicated in the atomic result signal sent by the slave. The QBus Control Unit (QBC) snooper snoops the local bus when a read with atomic signal is accepted (bit test) on the Xa or Xb buses to the SRAM location. The QBC tries to detect a write to a protected address before the SC140 core finishes the read-modify-write operation. Reservation occurs when a write to a protected address is detected. If the write operation fails, the T bit in the SC140 core is set. The resulting signal is optionally used in a lock mechanism. When the atomic signal is asserted, the slave locks the bus and the write is always successful. You can also use a read and reserve or a write and confirm mechanism. For details, refer to the *SC140 DSP Core Reference Manual*.

Note: If there is a write to the same word on the local bus between the read and the write, the QBC returns a fail in the cycle after the write. Otherwise, the QBC returns a success.

Reservation on the QBus, a read-modify-write operation, is supported by two signals:

- The master sends a signal to assert that the atomic transaction (read-modify-write) is valid on the bus.
- The slave asserts a result signal when the atomic operation succeeds. This result signal is duplicated for each bank.

9.3.7 Setting a Data Area

The data areas are determined in the Data Bank Registers 0–3 (DBR[0–3]). An area is determined with a base address and size of the area. The determined area indicates whether it is a global, immediate, or no freeze immediate memory area. Each of the four data area registers can be disabled or reversed. The area base should always be a multiple of the area size, with the exception of base=0, in which the size can be any value. The data area registers are programmed by setting the base register and size bit. **Table 9-4** summarizes the different cases of programming the data area registers. The original base column represents the 24 msb of the base address needed for the area definition. The 8 lsb are insignificant.

Table 9-4. Programming Data Area Base and Size

No.	Original Base[31-8]	Size	Base Bits	SIZE Bit
1	xxxxxxxxxxxxxxxxxxxxxxxx	256 B	xxxxxxxxxxxxxxxxxxxxxxxx	1
2	xxxxxxxxxxxxxxxxxxxxxxx0	512 B	xxxxxxxxxxxxxxxxxxxxxxx1	0
3	xxxxxxxxxxxxxxxxxxxxxx00	1 KB	xxxxxxxxxxxxxxxxxxxxxx10	0
4	xxxxxxxxxxxxxxxxxxxxx000	2 KB	xxxxxxxxxxxxxxxxxxxxx100	0
5	xxxxxxxxxxxxxxxxxxxx0000	4 KB	xxxxxxxxxxxxxxxxxxxx1000	0
6	xxxxxxxxxxxxxxxxxxx00000	8 KB	xxxxxxxxxxxxxxxxxxx10000	0
7	xxxxxxxxxxxxxxxxxx000000	16 KB	xxxxxxxxxxxxxxxxxx100000	0
8	xxxxxxxxxxxxxxxxx0000000	32 KB	xxxxxxxxxxxxxxxxx1000000	0
9	xxxxxxxxxxxxxxxx00000000	64 KB	xxxxxxxxxxxxxxxx10000000	0
10	xxxxxxxxxxxxxxx000000000	128 KB	xxxxxxxxxxxxxxx100000000	0
11	xxxxxxxxxxxxxx0000000000	256 KB	xxxxxxxxxxxxxx1000000000	0
12	xxxxxxxxxxxxx00000000000	512 KB	xxxxxxxxxxxxx10000000000	0
13	xxxxxxxxxxxx000000000000	1 MB	xxxxxxxxxxxx100000000000	0
14	xxxxxxxxxxx0000000000000	2 MB	xxxxxxxxxxx1000000000000	0
15	xxxxxxxxxx00000000000000	4 MB	xxxxxxxxxx10000000000000	0
16	xxxxxxxxx000000000000000	8 MB	xxxxxxxxx100000000000000	0
17	xxxxxxxx0000000000000000	16 MB	xxxxxxxx1000000000000000	0
18	xxxxxxx00000000000000000	32 MB	xxxxxxx10000000000000000	0
19	xxxxxx000000000000000000	64 MB	xxxxxx100000000000000000	0
20	xxxxx0000000000000000000	128 MB	xxxxx1000000000000000000	0
21	xxxx00000000000000000000	256 MB	xxxx10000000000000000000	0
22	xxx000000000000000000000	512 MB	xxx100000000000000000000	0
23	xx0000000000000000000000	1 GB	xx1000000000000000000000	0
24	x00000000000000000000000	2 GB	x10000000000000000000000	0
25	000000000000000000000000	4 GB	100000000000000000000000	0

For example, an area with base address = 1MB; size = 256 KB supports the basic condition that the base be an integer multiple of the size. The steps in defining this are as follows:

1. Write the base address in 32-bit representation. The 1 MB is written as 00000000000100000000000000000000.
2. According to the size (256 KB), choose line 11 in **Table 9-4**. The size_bit = 0.
3. Determine the base bits for the DBR. The 10 lowest bits are determined according to **Table 9-4** as 1000000000. The upper bits are determined according to the remaining bits ([31–18]) of the base address, which means 00000000000100.
4. This results in: base = 0x001200, or in binary form 000000000001001000000000. The 8 non-written lsb = 0x00.

After the area is sized, the global and IMM bits determine whether this area is global, immediate, or immediate with no freeze. The area can be global and immediate at the same time. The area can be reversed, so it does *not* have the value it was set to. For example, if the immediate bit is set and the reverse bit is set, this area is not immediate. A reverse area has a higher priority, so a reverse area can be set inside an area. **Figure 9-8** shows a reverse area inside an area in the memory space. Register 0 determines an area in memory, and register 1 determines a reverse area inside the register 0 area. If the immediate bit is set in both registers, the data area 0 without the data area 1 is the immediate area.

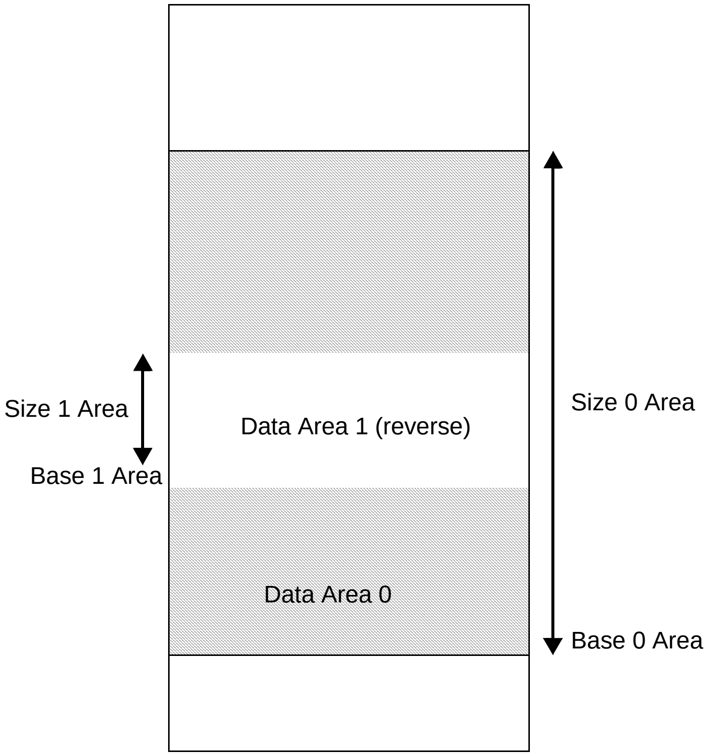


Figure 9-8. Reverse Area Inside an Area

9.3.8 Instruction Cacheable Area

Banks 1 and 3 can function as cacheable memory and must be programmed for use. The cacheable area is always higher than 16 MB and the QBus base address (the higher of the two) and can be programmed in the cacheable area register. In a conflict, the cacheable area is always higher than 16 MB and the QBus base line. The cacheable area is determined by a base address (the first address in the cache area) and the size. The area base should always be a multiple of the area size, with the exception of base = 0, in which the size can be any value. The cacheable area is programmed by setting the Instruction Cacheable Area Base Register (ICABR) and the size bit in the Instruction Cacheable Area Control Register (ICACR). **Table 9-5** summarizes the different cases of ICABR programming. The original base column represents the 16 msb of the base address needed for the area definition. The 16 lsb are insignificant.

Table 9-5. Cacheable Area Programming

No.	Original Base[31–16]	Size	Area Base Bits (ICABR)	ICACR[SIZE]
1	xxxxxxxxxxxxxxxx	64 KB	xxxxxxxxxxxxxxxx	1
2	xxxxxxxxxxxxxxx0	128 KB	xxxxxxxxxxxxxxx1	0
3	xxxxxxxxxxxxxx00	256 KB	xxxxxxxxxxxxxx10	0
4	xxxxxxxxxxxxx000	512 KB	xxxxxxxxxxxxx100	0
5	xxxxxxxxxxxx0000	1 MB	xxxxxxxxxxxx1000	0
6	xxxxxxxxxxx00000	2 MB	xxxxxxxxxxx10000	0
7	xxxxxxxxxx000000	4 MB	xxxxxxxxxx100000	0
8	xxxxxxxxx0000000	8 MB	xxxxxxxxx1000000	0
9	xxxxxxxx00000000	16 MB	xxxxxxxx10000000	0
10	xxxxxxx000000000	32 MB	xxxxxxx100000000	0
11	xxxxxx0000000000	64 MB	xxxxxx1000000000	0
12	xxxxx00000000000	128 MB	xxxxx10000000000	0
13	xxxx000000000000	256 MB	xxxx100000000000	0
14	xxx0000000000000	512 MB	xxx1000000000000	0
15	xx00000000000000	1 GB	xx10000000000000	0
16	x000000000000000	2 GB	x100000000000000	0
17	0000000000000000	4 GB	1000000000000000	0

For example, an area with base address = 32 MB; size = 256 KB supports the condition that the base be an integer multiple of the size. The steps in defining this area are as follows:

1. Write the base address in 32-bit representation. 32 MB is written as 00000010000000000000000000000000.
2. Based on the size (256 KB), choose line 3 in the table. The size_bit = 0.

3. Determine the base bits for the ICABR. The two lowest bits are 10. The upper bits are determined according to the remaining bits [31–18] of the base address, that is 00000010000000.

This results in: base = 0x0202, or in binary form, 0b0000001000000010. The 16 non-written lsb = 0x0000.

9.3.9 EQBS Programming Model

You can configure the EQBS in different ways to best fit the system architecture and the application. The EQBS registers are mainly of two types, those defining the bank and data areas and those defining the system mode of operation. These registers are mapped on the QBus. The address of an EQBS register is a concatenation of a base address and an offset. For the base address see **Section 8.2**, *QBus Address Space,* on page 8-9. For the complete list of registers and their addresses, refer to **Table 8-2**. All registers are memory-mapped.

To write/read to/from the registers, you must initiate a data access from the SC140 core that is of the same size as the register. For example, if the register size is 16 bits, the command is **move.w**. If the register size is 32 bits, the command is **move.l**. If a write is initiated to a register with a logical influence, the new value is valid according to the unit that is affected by the register:

- Changes in registers that affect the fetch unit (ICACR, ICABR, IFUR) are valid at the next fetch miss after the write to the register. Changing the registers during prefetch does not affect the prefetch. For example, turning the prefetch off (bit 4 in IFUR) does not stop the current prefetch but disables prefetching after the next fetch miss.
- Changes in registers that affect the write buffer are valid, as follows:
  - *WBCR[6–15] Watch Dog*. If the write buffer is empty, the new value is ready on the next core cycle after the write to the register. If the write buffer is not empty, the new value is valid at the next restart of the watch dog, that is, at flush or when the write buffer ends all accesses.
  - *WBCR[3] wb_off*. Valid at the next core cycle after the write to the register.
  - *WBFR software flush*. Valid at the next core cycle after the read from the register.
  - *DBRx memory areas*. Valid at the next core cycle after the write to the register.

The EQBS registers include:

- QBus Base Registers for Banks [0–2], **page 9-19**
- QBus Mask Registers for Banks 0–1 (QBUSMR[0–2]), **page 9-19**
- Instruction Cacheable Area Control Register (ICACR), **page 9-20**
- Instruction Cacheable Area Base Register (ICABR), **page 9-20**
- Instruction Fetch Unit Configuration Register (IFUR), **page 9-20**
- Write Buffer Software Flush Register (WBFR), **page 9-21**
- Write Buffer Control Register (WBCR), **page 9-21**

- Data Area Registers 0–3 (DBR[0–3], **page 9-22**
- Core ID Register (CIDR), **page 9-23**
- Version Register (VR), **page 9-23**
- FlyBy Address Control Register (FLBACR0), **page 9-23**

QBUSBR[0–2] QBus Base Register 0–2 (QBUSBR0) 0x00F0FF02 (QBUSBR1) 0x00F0FF06 (QBUSBR2) 0x00F0FF0A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	bank base[0–15]															
Type	R/W (QBUSBR0 is read only)															
Reset	base0 = 0x00F0 = QBus base line (read only) base1 = 0x0100 base2 = 0x0000															

Table 9-6. QBUSBR[0–2] Bit Descriptions

Name	Reset	Description	Restrictions

bank base
15–0
	base0 = 0x00F0
	base1 = 0x0100
	base2 = 0x0000
		Bank Base
		The base address of the bank.
			1. Do not parallel an external read or write with a write to the register.
			2. Do not access external read or write one cycle after writing to the register.

QBUSMR[0–2] QBus Mask Register 0–2 (QBUSMR0) 0x00F0FF00 (QBUSMR1) 0x00F0FF04 (QBUSMR2) 0x00F0FF08

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	bank maskl[0–15]]															
Type	R/W (QBUSMR0 is read only)															
Reset	mask0 = 0xFFFF (read only) mask1 = 0xFF00 (the boot program writes 0xFF80 to this register after reset) mask2 = 0xFFFF															

Table 9-7. QBUSMRx Bit Descriptions

Name	Reset	Description	Restrictions

bank mask
0–15
	mask0 = 0xFFFF
	mask1 = 0xFF80
	mask2 = 0xFFFF
		Bank Mask
		The mask value of the bank.
			1. Do not parallel an external read or write with a write to the register.
			2. Do not access an external read or write one cycle after writing to the register.

ICACR Instruction Cacheable Area Control Register 0x00F0FF30

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	—	EN	REV	SIZE	—	—	—	—	—	—	—	—
Type	R/W															
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Table 9-8. ICACR Bit Descriptions

Name	Reset	Description	Settings
— 0–4	0	Reserved. Write to zero for future compatibility.	
EN 5	1	**Enable Area Operation** Enables/disables the area operation.	0 Disabled. 1 Enabled.
REV 6	1	**Reverse Cacheable Area** Determines whether the cacheable area is inside or outside the area definition.	0 Cacheable area is inside the area definition. 1 Cacheable area is outside the area definition.
SIZE 7	0	**Size Indication** Sets the size to the 64 KB minimum or sets it to a different size.	0 Size is other than 64 KB. 1 Size is 64 KB.
— 8–15	0	Reserved. Write to zero for future compatibility.	

ICABR Instruction Cacheable Area Base Register 0x00F0FF32

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Area Base[15-0]															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 9-9. ICABR Bit Descriptions

Name	Reset	Description
Area Base 0–15	0x0080	**Area Base Address** The base address for the area defining the cacheable area. The range below the 16 MB address (0x00000000–0x00FFFFFF) is defined as not cacheable.

IFUR Instruction Fetch Unit Configuration Register 0x00F0FF60

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	—	—	—	—	—	—	—	PFOFF	—	SIZE		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-10. IFUR Bit Descriptions

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th><th>Settings</th></tr>
<tr><td>—
0–10</td><td>0</td><td colspan="2">Reserved. Write to zero for future compatibility.</td></tr>
<tr><td><b>PFOFF</b>
11</td><td>0</td><td><b>Prefetch</b>
Enables/disables Prefetch mode.</td><td>0 Prefetch mode enabled (default after boot).
1 Prefetch mode disabled (value after reset).</td></tr>
<tr><td>—
12</td><td>0</td><td colspan="2">Reserved. Write to zero for future compatibility.</td></tr>
<tr><td><b>SIZE</b>
13–15</td><td>000</td><td><b>Block Size</b>
Sets the block size.</td><td><table>
<tr><th>SIZE[2–0]</th><th>Block Size</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>010</td><td>4</td></tr>
<tr><td>011–111</td><td>Reserved</td></tr>
</table></td></tr>
</table>

WBFR Write Buffer Software Flush Register 0x00F0FF80

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

A read access to WBFR causes a software flush.

WBCR Write Buffer Control Register 0x00F0FF82

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	WBOFF	—	—	WD[9–0]									
Type	R/W															
Reset	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

Table 9-11. WBCR Bit Descriptions

Name	Reset	Description	Settings
— 0–2	0	Reserved. Write to zero for future compatibility.	
WBOFF 3	1	**Disable Write Buffer** Enables/disables the write buffer. When the write buffer is disabled, all write execute through the bus switch.	0 Enable write buffer operation. 1 Disable write buffer operation.
— 4–5	0	Reserved. Write to zero for future compatibility.	
WD 6–15	0x3ff	**Watchdog Count** Value for watch dog count.	

DBR[0–3] Data Area Register 0–3 **(DBR0) 0x00F0FFA0**
(DBR1) 0x00F0FFA4
(DBR2) 0x00F0FFA8
(DBR3) 0x00F0FFAC

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	BASE 31–16															
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**
	GBL	IMM		—	—	EN	RV	SIZE	BASE 15–8							
Type	R/W															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-12. DBRx Bit Descriptions

Name	Reset	Description	Settings
BASE 0–15	0	**Base Address 31–16** The area base address.	See **Section 9.3.7**, *Setting a Data Area,* on page 9-15.
GBL 16	1	**Global** Determines whether a memory area is non-global or global. This bit is usually used for data cache coherency.	0 Non-global. 1 Global.
IMM 17–18	00	**Immediate** Define the immediate access to the area, forcing in-order execution of writes. The write can be with or without a freeze to the SC140 core.	00 Regular write through write buffer 01 Write immediate 10 Write immediate with no freeze 11 Reserved
— 19–20	0	Reserved. Write to zero for future compatibility.	
EN 21	0	**Enable Operation** Enables/disables this area register operation.	0 Disable operation. 1 Enable operation.
RV 22	0	**Reverse Bit** Defines a non-immediate (or non-global) memory slice within an immediate (or global) area. If a memory area matches 2 areas, the area with RV bit = 1 dominates the definition of global and immediate behavior When the area definition is reversed and comes with flag IMM = 1, it indicates that the area is defined as non-immediate. If comes with GBL=1, indicates that the area is defined as non-global.	0 Normal area definition. 1 Reverse the area definition.
SIZE 23	0	**Size Indication** Indicates whether the size is 256 bytes or not.	0 Size is other than 256 bytes. 1 Size is 256 bytes.
BASE 24–31	0	**Base Address 15–8** The area base address.	See **Section 9.3.7**, *Setting a Data Area,* on page 9-15.

CIDR Core ID Register 0x00F0FFF0

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								CORE ID							
Type	R															
Reset	Different reset value for each SC140 core; see **Table 9-13**.															

Table 9-13. CIDR Bit Descriptions

Name	Reset	Description
— 0–7	0	Reserved. Write to zero for future compatibility.
COREID 8–15	See description	**Core Identification** Contains a unique value for each SC140 core, which enables different permissions for different cores. The reset values are: SC140 core 0: 0x00 SC140 core 1: 0x01 SC140 core 2: 0x02

VR Version Register 0x00F0FFF2

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PrVer								ECVer							
Type	R															
Reset	0x03								0x52							

Table 9-14. VR Bit Descriptions

Name	Reset	Description
PrVer 0–7	0x03	**Process Version** Contains a different number for each process version.
ECVer 8–15	0x52	**Extended Core Version** Contains a different number for each extended core version.

FLBACR0 FlyBy Address Control Register 0x00F0FFF4

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	FLBSA[20–5]															
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**
	FLBSA[4–0]					—										
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-15. FLBACR0 Bit Descriptions

Name	Reset	Description
FLBSA 0–20	0	**FlyBy Start Address** Contains bits 23–3 of the start address used during a flyby transfer. Refer to **Section 16.3.2**, *DMA Data Transfer Examples,* on page 16-33 for details.
— 21–31	0	Reserved. Write to zero for future compatibility.

9.4 Instruction Cache (ICache)

The ICache is located between the extended core bus (QBus) and the internal program address bus. The ICache includes a memory array that stores frequently used program instructions. When an instruction is not already stored in the ICache array, the ICache initiates a fetch from the external memory subsystem via the extended core bus system (EQBS). The SC140 core then halts for the number of clock cycles required to fetch the required instruction into the ICache array. The fetch unit in the EQBS accesses the external memory to fetch a stream of instructions, thus taking advantage of the capabilities of the external memory and associated interfaces. **Figure 9-9** shows the ICache in the extended SC140 core system.

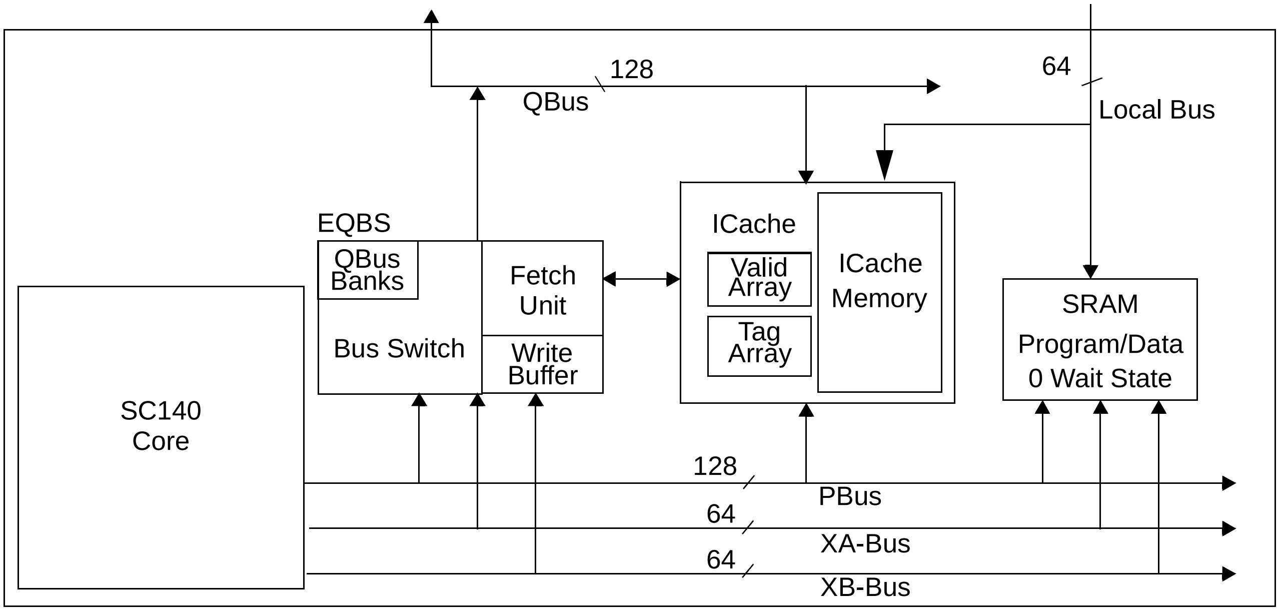


Figure 9-9. MSC8113 ICache System

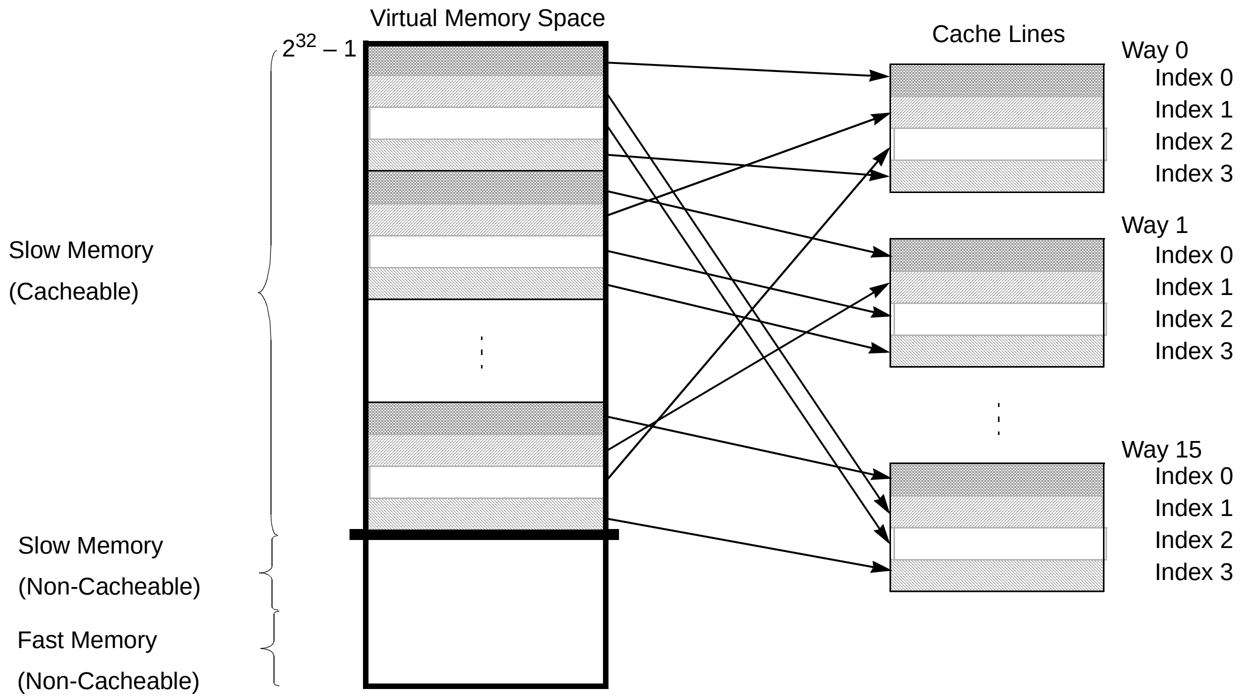
The MSC8113 ICache has the following features:

- 16 KB of memory
- 16-way associativity
- 4 indexes, so the ICache has a total of 64 lines.
- 16 fetch sets for each line. Each fetch set is 16 bytes.

- A total of 1024 valid bits with a total ICache program memory array of 16 KB (1024 × 16)
- Replacement based on least-recently used (LRU) algorithm
- Locking data in cache through flexible LRU boundaries, multi-task support
- Real-time debugging support with misses and hits counted through the EOnCE module
- Non-real-time debugging; enable to read full cache state, clear line command for breakpoint insertion
- ICache memory array is accessible in Debug Mode for read/write accesses through the local bus.
- Flexible cacheable area
- Prefetch Support
- Row and column redundancy in the cache memory, array check through an autonomous built-in self test (ABIST)
- External cache support by asserting the global signal ($\overline{\text{GBL}}$) when accessing predefined memory banks

Accessing instruction code from memory areas with high access latencies (for example, external memories, internal memories connected to slow buses, and so on) imposes timing penalties on the DSP core, causing performance degradation for the entire system. The ICache improves the system execution time by dynamically mapping relevant memory areas to a fast 16 KB memory. ICaches allow system designers to place a large amount of code into slower memories, yet achieve high performance as if the code were stored in a fast on-device zero-wait-state memory. Each of the three extended cores in the MSC8113 has its own ICache. Each of these ICache memories optimizes access to its instruction storage area by using a specialized indexing system. When the SC140 core requests instruction code, the first 22 bits of the requested address (A[31–10]) are used to identify a region in external memory, and form the TAG value for that region. The next two bits of the address (A[9–8]) identify a quadrant within the tag-defined memory region (that is, from 00, the lowest quarter to 11, the highest quarter), and specify the INDEX value for that region in the ICache. Each memory quadrant within the tag-defined memory region is further divided into 16 fetch sets of 16 bytes each by using the next four bits of the address (A[7–4])—this defines the POSITION of the set within the quadrant. Each fetch set is assigned an identifier bit (VALID) whose value is initialized as 0 and changed to 1 when that block has been written into the cache memory.

Each MSC8113 ICache has sixteen ways (way [0–15]), each with four INDEX values (Index [0–3]). The ways allow the ICache to define 64 tag-defined memory area quadrants, each defining 16 consecutive fetch sets in the memory space. Each quadrant is called a line in the cache. A line represents a fixed amount of consecutively located code. The MSC8113 ICache can contain up to 16 lines with the same index from different segments simultaneously (see **Figure 9-10**). Each line with the same index number is assigned a least-recently-used (LRU) value that specifies its LRU status level.



Note: The memory range shown in the figure is the maximum definable space. The minimum definable cacheable space is 64 KB. See Section 9.4 for details on configuring cacheable memory space.

Figure 9-10. Example of Code Position Distribution in ICache

A request for code that is already in the cache is termed a *cache hit*. When the required code is not present in the ICache memory array (termed a *cache miss*), the code is fetched from the slower memories to the SC140 core and simultaneously loaded into the ICache memory array. The performance degradation (DSP core timing penalty) resulting from the slower access is termed a *miss penalty*. Cache updates are initiated by the fetch unit in the EQBS when a cache miss occurs. The amount of data transferred by the fetch unit is configured in the IFUR, which defines the number of fetch sets to transfer in a block. In addition to the basic fetch operation (called phase 1), the prefetch unit (when enabled in the IFUR) fetches sets from the next consecutive addresses (phase 2) to the end of the cache line or until a new cache miss begins the next fetch sequence.

Note: Unlike a data cache, the instruction cache depends on code not changing during run time. If it does, the cache contents should be cleared (cache flush, no coherency support).

When a cache miss occurs, one of several events happens:

- If the upper 22 bits of the address match the tag and index in a cache line, but the VALID bit for the fetch set position is clear, code is transferred from memory to the SC140 core and written into the cache.

- If there is no tag match, but not all ways have been used for that INDEX number, code is transferred to the SC140 core and the cache line for the next available way and index number.
- If all sixteen ways with the INDEX number are used and there is no tag match, code is transferred to the SC140 core and written to the cache line for the INDEX number for which the
  LRU = 0.

Note: For all instruction fetches, the number of fetch sets written to the cache depends on the block setting configured in the IFUR. The VALID bit is set in the appropriate position for each fetch set retrieved. If prefetch is enabled, the remainder of the fetch units specified by the cache line are written into cache and the VALID bits set. In addition, the LRU value for the line is set to 0xF and the LRU values for all other lines with the same INDEX number, except for the line for which LRU = 0, are decremented by 1.

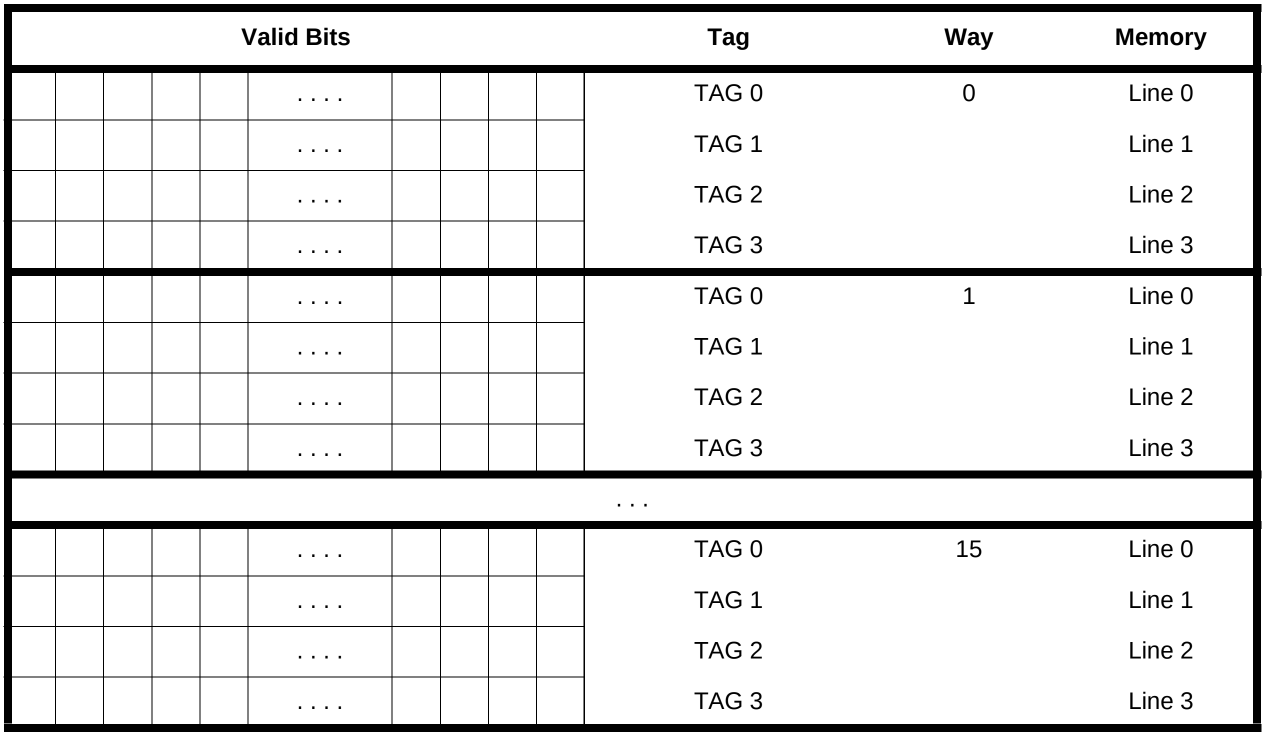
Replacement of the least-recently-used existing line with a new one is called thrashing. Frequent thrashing indicates cache ineffectiveness. Cache effectivity is based on locality. Programs have two locality attributes:

- *Temporal locality*. The likelihood that the SC140 core will often request a given address in memory. A high temporal locality can be caused by a large number of loops in the code.
- *Spatial locality*. The likelihood that, given a core program request to a certain address in memory, the SC140 core will also request the “close by” addresses. A high spatial locality means that the code has few change-of-flows.

Because the cacheable memory area and the block size are configurable, you may have to determine the optimal sizes and placement of code within the memory for your application. Configuration allows a trade-off between efficient use of the system bus and burdening the QBus with transactions that the SC140 core may not need.

9.4.1 ICache Attributes

The diagram in **Figure 9-11** gives a logical view of the MSC8113 ICache.



Valid Bits	Tag	Way	Memory
. . . .	TAG 0	0	Line 0
. . . .	TAG 1		Line 1
. . . .	TAG 2		Line 2
. . . .	TAG 3		Line 3
. . . .	TAG 0	1	Line 0
. . . .	TAG 1		Line 1
. . . .	TAG 2		Line 2
. . . .	TAG 3		Line 3
. . .			
. . . .	TAG 0	15	Line 0
. . . .	TAG 1		Line 1
. . . .	TAG 2		Line 2
. . . .	TAG 3		Line 3

Note: ICache size is 16 KB with 16-way cache, 4 lines per way, and 16 valid bits per line. Each valid bit represents one

Figure 9-11. MSC8113 ICache Logical Organization Diagram

The key attributes of the ICache are as follows:

- *Line*. The smallest division of cache memory for which there is a distinct tag (sometimes called a sector). The line size is an integer number of the processor's fetch sets. A fetch set is the number of bytes requested by the processor in a single request. A line must include consecutive program code.
- *Fetch block*. Number of bytes fetched every cache miss. A fetch block can be smaller than a line. See **Section 9.3**.
- *TAG*. Holds the upper 22 address bits for the corresponding line in the cache and is compared to the current access address.
- *Valid Bit*. Each fetch block in the cache has a bit indicating whether it is found or not found in the cache. This bit is called a valid bit.
- *Index and Way/Set*. As **Figure 9-10** shows, each "slow" memory segment is divided into a fixed number of lines. Each line is marked by a unique number called an index. A line with a particular index can be mapped to any of sixteen different places in the ICache memory array. The number of ways indicates a cache's degree of associativity.
- Memory address partitioning:

- — The tag field partitions the external memory into 64 KB segments
- — The index field partitions each tag-defined area into 16 KB segments
- — The position bits partition each index-defined area into 16-bit segments (fetch sets)

- ■ *Replacement Algorithm*. An algorithm that determines which line to replace when a miss occurs. The ICache uses the Least Recently Used (LRU) algorithm; that is, the line for a specific index number in a way that is marked as least-recently-used (LRU = 0) is replaced on a miss.

9.4.2 Debugging

The ICache debugging includes either run-time debug or Debug mode. The ICache enters Debug mode by setting a user-programmable bit in the ICache Control Register (ICCR). This Debug mode is not related to the SC140 Debug mode, so that the ICache can be in Debug mode while the SC140 core is in a normal running mode and *vice versa*. This schema is necessary because the SC140 core enters and exits Debug mode regardless of the extended core status, which may cause contentions between the ICache debug mechanisms and normal work mode. Entering Debug mode immediately stops the ICache update mechanism (load of new data by the fetch unit), regardless of the fetch unit status. In Debug mode, the ICache does not issue any hits (as it does in Lock mode) or perform thrashes. Debug mode is only for viewing the ICache status and breakpoint support.

The EOnCE module performs run-time debugging, which counts the hit and miss signals sent by the ICache. Each time the ICache answers an external access, a hit flag is raised for a counter in the EOnCE module. If a new meaningful cacheable area access is not found in the ICache, a miss flag goes up.

Note: To use the run-time debugging for the ICache, you must configure the EOnCE counter through the Event Counter Register (ECNT_CTRL). To enable the counter, write 0b1111 to the Event Counter Enable (ECNTEN) field in ECNT_CTRL. To count cache hits, write 0b1101 to the Events to Be Counted (ECNTWHAT) field in ECNT_CTRL. To count cache misses, write 0b1110 to the ECNTWHAT field. For details on EOnCE configuration, see the SC140 extended core header and the *SC140 DSP Core Reference Manual*.

In Debug mode, the SC140 core can read the ICache status (regular memory read). The ICache status consists of the contents of the tag array, valid bit array, and LRU registers. The status can give you a more in-depth view of ICache usage and bottlenecks (compared to the hit/miss count) when code performance is optimized. This information can also help devise LRU boundary allocation schemes for multi-task support.

Each main ICache resource (tag array, valid bit array, and LRU machine) has a memory-mapped status register that holds 16 bits of the contents of the resource to which it belongs. To read the ICache status, the SC140 core performs a read from a specific memory address. The contents of the register are sent to the SC140 core, and that the status register is reloaded from the next address of its resource for the next core read. For example, if the valid bit array status register currently holds the contents of the valid array bits of in-line position 10, of all lines indexed as 2 (a total of 16 bits, one for each way) and an SC140 core reads this register, the contents of the status register are sent to the SC140 core, and a reload occurs from the bits in position 11, index 2. In addition, a special ICache initialization command simultaneously initializes all status registers by reading the first data from each resource into the status register of that resource. **Table 9-16** shows an example of the flow needed to read the valid bit array contents (and the information read from each access).

Table 9-16. Read Valid Bit Array Status Example

SC140 Core	Debug Register
Status initialization command	Initial load (index0, position0)
One execution set delay (required)	
Read valid bit status 1: 16 bits, line: index 0, position 0	Reload (index0, position 1)
Read valid bit status 2: 16 bits, line: index 0, position 1	Reload (index0, position 2)
(More valid bit array status reads)	
Read valid bit status 16: 16 bits, line: index 0, position 15	Reload (index1, position0)
Read valid bit status 17: 16 bits, line: index 1, position 0	Reload (index1, position1)
(More valid bit array status reads)	
Read valid bit status 64: 16 bits, line: index 3, position 15	Reload (index0, position0)

Each state register is 16 bits long. The resources larger than 16 bits (tag is 22 bits and the LRU register is 64 bits) are read in more than one read (two reads per tag, first the lsbs and then the msbs (zero padded), four reads per LRU of a particular index, again lsb to msb in sequential order). The following tables describe a tag array and LRU machine reading sequences (accordingly).

Table 9-17. Tag Array Status Reading Sequence

SC140 Core	Debug Register
Status initialization command	Initial load (way0, index0, tag bits [15–0])
One execution set delay (required)	
Read tag array status 1: 16 bits, line: way 0, index0, tag bits [15–0]	Reload (way0, index0, tag bits [21–16] (padded))
Read tag array status 2: 16 bits, line: way 0, index0, tag bits [21–16] (padded)	Reload (way0, index1, tag bits [15–0])
Read tag array status 1: 16 bits, line: way 0, index1, tag bits [15–0]	Reload (way0, index1, tag bits [21–16] (padded))
(More tag array status reads)	

Table 9-17. Tag Array Status Reading Sequence (Continued)

SC140 Core	Debug Register
Read tag array status 8: 16 bits, line: way 0, index3, tag bits [21–16] (padded)	Reload (way1, index0, tag bits [15–0])
Read tag array status 9: 16 bits, line: way 1, index0, tag bits [15–0]	Reload (way1, index0, tag bits [21–16] (padded))
(More tag array status reads)	
Read tag array status 128: 16 bits, line: way 15, index3, tag bits [21–16] (padded)	Reload (way0,ind0,lsbs)

Table 9-18. LRU Machine Status Reading Sequence

SC140 Core	Debug Register
Status initialization command	Initial load (way0, index0, tag bits [15–0])
One execution set delay (required)	
Read LRU status 1: 16 bits, line: index0 register, bits [15–0]	Reload (index0 register, bits [31–16])
Read LRU status 2: 16 bits, line: index0 register, bits [31–16]	Reload (index0 register, bits [47–32])
Read LRU status 3: 16 bits, line: index0 register, bits [47–32]	Reload (index0 register, bits [63–48])
Read LRU status 4: 16 bits, line: index0 register, bits [63–48]	Reload (index1 register, bits [15–0])
Read LRU status 5: 16 bits, line: index1 register, bits [15–0]	Reload (index1 register, bits [31–16])
(More LRU register status reads)	
Read LRU status 16: 16 bits, line: index3 register, bits [63–48]	Reload (index0 register, bits [15–0])

Note: Tag status lines are read according to {way, index} addresses while valid bit array status lines are read in the order of {index, position}.

To enable breakpoint insertion in cached code, the ICache includes a clear line command to reset all valid bits for a specific line ({way,index}). This is not the same as a read line. The line to be cleared is obtained via the status reading mechanism. For detailed encodings and addresses, see **Section 9.4.4**, *ICache Programming Model*. Another breakpoint option is to access the ICache array through Debug mode. In this mode, the ICache memory array is accessible for read or write accesses through the local bus. See **Chapter 8**, *Memory Map* for more details.

Before a debug command can execute, the ICache must be programmed to Debug mode. If a debug command or a debug read arrives and the ICache is not in Debug mode, the command is discarded. An exception flag in the ICache is raised for a PIC to use, and the interrupt takes effect only in systems with the cache connected to the PIC.

9.4.3 Multi-Task Support

The ICache includes multi-tasking features. Allowing partial locks of the ICache, a multi-tasking operating system can return an old task while the ICache still holds the task’s most recently used code. Multi-task support includes both single-stack and multi-stack prioritized and preemptive real-time operating system (RTOS) models. Upon activation, the single-stack operating system (OS) model executes to completion, but the active task can be preempted by another task with a higher priority. In the multiple-stack OS model, a task can be activated and preempted at any time.

When multi-tasking is introduced, caches add non-determinism to the execution time of each task because of cache thrashing when a task switch occurs. Cache thrashing occurs when the cache contents are replaced by a preempting task and then the cache contents are reloaded to restore the state of the cache for the preempted task.

The user or OS can program the lower and upper boundaries of the LRU via the ICache Control Register (ICCR) based on the task priority scheme in use. The LRU mechanism is then functional only within the programmed boundaries, while the lines outside the boundaries are considered frozen. The boundary register (ICCR) must be written every time a task starts working. By programing only one register (which includes the lower and upper boundaries) the OS determines which tasks need fixed allocation and which tasks can work with a flexible boundaries mechanism. The boundary resolution is the size of one LRU priority level for all indexes, that is, one way.

Figure 9-12 describes the difference between flexible boundaries and fixed allocation. With flexible boundaries, the first task (task 1) can work with all the available cache space. When a new task arrives (task 2), it should change only the upper boundary, so the cache space of task 2 is smaller than the full cache space. When the first task (task 1) resumes operation, it should change the upper boundary back to the previous value so that the cache now consists of the least recently used parts of the task. This practice helps to ensure that there is no miss penalty if the task needs to use the code already in the cache. If the upper boundary is not changed at the transition from task 1 to task 2, some task 2 instructions overwrite task 1 instructions, with a resulting cache miss penalty when task 1 resumes. In the flexible boundary mechanism, each time a higher-priority task starts operation, a smaller cache space is available.

Note: In single-stack tasks, the nested task with the higher priority must end before the lower-priority task resumes operation.

In the fixed allocation mechanism, the OS reserves a cache space for each task. When a new task resumes operation, it works only with the cache space allocated for it, meaning that each task should change the lower and upper boundaries of the cache space.

The ICache has only one set of LRU boundaries (upper and lower) that are programmed in only one register, and each task should change these boundaries to enable all multi-task support:

- *Flexible boundaries*. Most suitable for the single-stack OS model.
- *Fixed allocation*. Most suitable for the multi-stack OS model.
- *Full cache shared for all tasks*. May be associated with extensive thrashing cost.

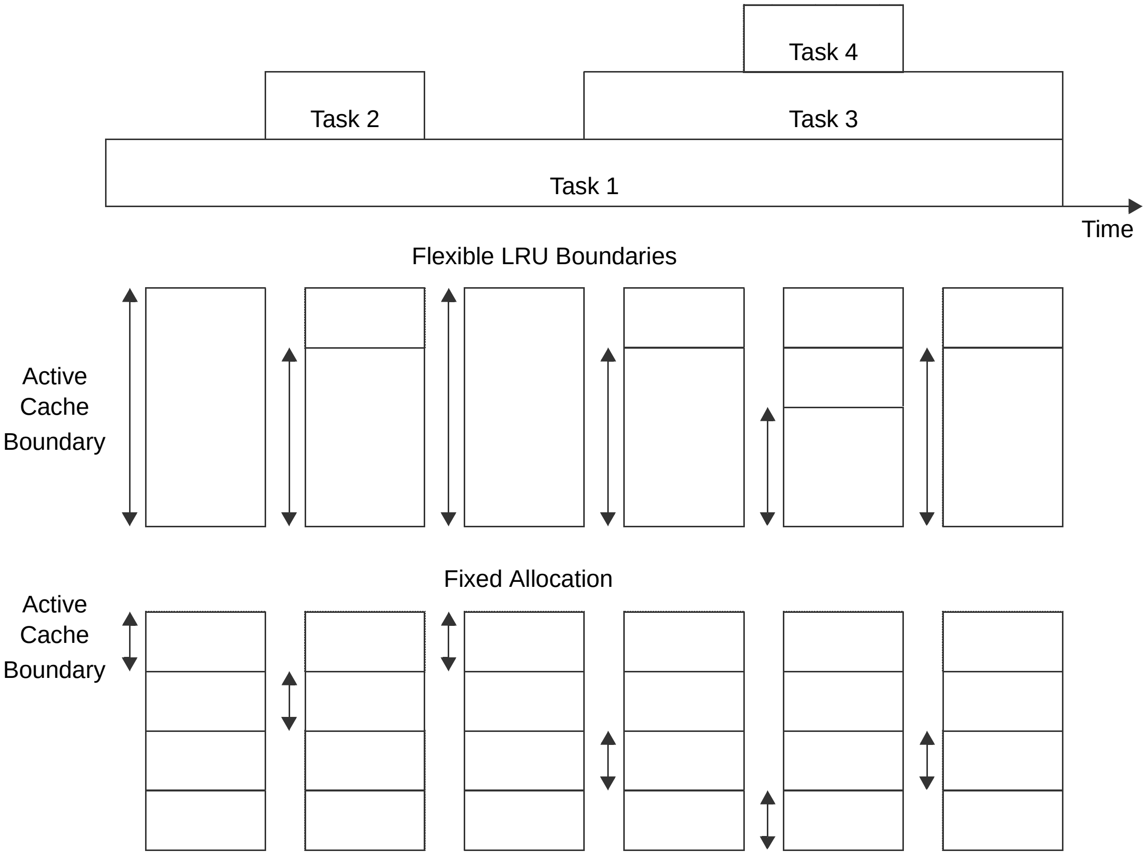


Figure 9-12. Cache Support in Run-Time Multi-tasking

9.4.4 ICache Programming Model

ICache programming refers to all memory accesses that can occur to the memory-mapped registers of the ICache. This section summarizes the different accesses, their functionality in the ICache, and restrictions. Notice that debug reads and commands are described in detail in **Section 9.4.2**, *Debugging*. The cache is programmed and read through the QBus. It acts as a zero-wait-state slave on QBus Bank 0, sharing it with other peripherals similarly connected (for example, the PIC). Bank 0 always has an *immediate* attribute, thus preventing ICache commands and mode changes from being randomly delayed by the write buffer and taking effect at unexpected times. Through the programming interface, you can set cache modes, send commands to the ICache, and read ICache registers.

9.4.4.1 Modes

Modes are set or reset by writing to a memory-mapped control register in the ICache (ICCR):

- *On/Off*. When the ICache is turned off, all caching-related machines and the commands and status mechanisms are off (clocks turned off). Only the control register periphery is still on.

Note: Turning the ICache off and then on only resets its state machines and does not erase its status (tag, valid, or LRU).

- *Debug mode*. Enables the cache non-real-time debug commands. All ICache updates are disabled in Debug mode (except the flush commands). The ICache does not enter Debug mode if it is set to off (mode bit is read as 0). For details, see **Section 9.4.2**, *Debugging*.
- *Lock mode*. Locks data in the ICache, with no updates permitted (thrashing/new valid bit setting). Hits are served in lock mode so all tag match LRU updates can take effect. All commands work in Lock mode, including flushes. The cache also enters lock mode if the upper boundary is set to be less than the lower boundary (when there is an attempt to read the register, the lock mode bit is on). The ICache does not enter Lock mode if it is off or set to Debug mode (mode bit is read as 0).

In summary, mode priority is as follows:

- Off
- Debug mode
- Lock mode

For a description of the ICCR bits, see **Table 9-19** on page 9-36.

9.4.4.2 Commands

All instructions are implemented by writes to a memory-mapped command register, the ICache Command Register (ICCMR). There are two types of ICache commands: run-time commands and Debug mode commands. The run-time commands are as follows:

- *Flush cache*. Reset all valid bit array and tag array.
- *Flush cache between boundaries*. Clear all valid bits and tags in the ways that are currently inside the LRU boundaries, partial flush.

The Debug mode commands (performed only in Debug mode) are as follows:

- *Clear line*. Clear all valid bits for a specific tag (line = {way[3–0],index[1–0]}, unlike lines for reads). For breakpoint insertion.
- *Initialize status registers*. Perform an initial load to the different cache status registers.

9.4.4.3 Reads

You can read the ICache state and mode information in the following four ICache registers:

- Read the tag array state (Debug mode only): Tag Array Status Register, **page 9-37**.
- Read the LRU State (Debug mode only): LRU Status Register (LRUSR), **page 9-37**.
- Read the valid bit array state (Debug mode only): Valid Bit Array Status Register (VBASR), **page 9-38**.
- Read the cache control register (cache mode and LRU boundaries): ICache Control Register (ICCR), **page 9-36**.

9.4.4.4 Restrictions

Following are the restrictions/issues on ICache programming:

- Control register newly written data can be read only in the second execution set following the write. There should be at least one execution set between the read and the write of the register so that the new data can be observed.
- Enabling a disabled cache in any way (either by setting the on bit, resetting the lock or debug mode bits, or returning the lower LRU boundary to be less or equal to the upper boundary) must be both preceded and followed by two no operation (**nop**) execution sets, as illustrated in the following code example.

  ```
  move.l #$0000f001,d1
  nop
  nop
  move.w d1,($<ICCR_ADDRESS>)
  nop
  nop
  ```

  In addition, any program that enables/disables the ICache must not be placed into the internal memory space accessible to the DMA controller.
- Paralleling a run-time command with a control register write has this effect: if a flush between boundaries is paralleled with a boundary change, the new boundaries are used. However, if any flush command is paralleled with a cache disable (cache off, debug mode, and so on), the flush is performed.
- Cache run-time commands cause SC140 core stall penalties.
- Cache run-time commands are performed in lock mode.
- If a flush command is paralleled with a flush between boundaries command, the full cache flush is performed, yet the timing penalty is of the flush between boundaries (the longer penalty of the two).
- Debug commands and read state registers are served in Debug mode only or they are discarded (an exception flag is raised). There must be at least one execution set between the time when the debug mode bit is turned on and the time when the first debug

command/debug is read. There must be at least one execution set between the last debug command and Debug mode exit.

- No Debug mode command can be paralleled with an ICCR write causing the ICache to exit Debug mode.
- Debug commands must be at least one execution set apart from any run-time command.
- The first read status command must be at least one execution set apart from an initialized debug command.

9.4.4.5 ICache Registers

The ICache registers include:

- ICache Control Register (ICCR), **page 9-36**
- ICache Command Register (ICCMR), **page 9-37**
- LRU Status Register (LRUSR), **page 9-37**
- Tag Array Status Register (TASR), **page 9-37**
- Valid Bit Array Status Register (VBASR), **page 9-38**

ICCR ICache Control Register **0x00F0FC00**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	UB				LB				—	—	—	—	DM	—	LM	ON
Type	R/W															
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1

Table 9-19. ICCR Bit Descriptions

Name	Reset	Description	Value
UB 0–3	1	**Upper Boundary Value** Selects the upper boundary (way number) for LRU consideration.	
LB 4–7	0	**Lower Boundary Value** Selects the lower boundary (way number) for LRU consideration. If LB > UB, the cache is locked. Values outside the range LB to UB are considered frozen.	
— 8–11	0	Reserved. Write to zero for future compatibility.	
DM 12	0	**Debug Mode** Enables/disables cache Debug mode.	0 Cache in normal mode. 1 Cache in debug mode.
— 13	0	Reserved. Write to zero for future compatibility.	
LM 14	0	**Cache Lock Mode** Enables/disables cache locking.	0 Cache not locked. 1 Cache locked.
ON 15	1	**On/Off Bit** Enables/disables the iCache.	0 Cache disabled. 1 Cache enabled.

ICCMR ICache Command Register 0x00F0FC02

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	C[3–0]				—	—	—	—	—	—	DA[5–0]					
Type	W															

Table 9-20. ICCMR Bit Descriptions

Name	Description	Value
C 0–3	**Commands Bits:**	0000Flush Cache. 0001 Flush Cache between boundaries. 1000 Initialize State Registers. 1001 Clear Line (Line to Clear in the DA bits). 1010– 1111reserved.
— 4–9	Reserved. Write to zero for future compatibility.	
DA 10–15	**Destination Address Field** Defines a line to clear.	00000Way 0, Index 0. ... 11111Way 15, Index 3.

LRUSR LRU Status Register 0x00F0FC10

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	LS[15–0]															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-21. LRUSR Bit Descriptions

Name	Reset	Description
LS 0–15	0x0000	**LRU Status Register Contents** An LRU status bit for each line that shares an index number. There is a register value stored for each index. The individual values are accessed by a sequential read. The first SC140 core read returns the value for Index = 0x0. A second read returns the value for Index = 0x1. A third read returns the value for Index = 0x2. A fourth read returns the value for Index = 0x3. For each bit, a 0 indicates that the line is not the LRU for the specified index and a 1 indicates that the line is the LRU for that index.

TASR Tag Array Status Register 0x00F0FC12

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	TS[15–0]															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-22. TASR Bit Descriptions

Name	Reset	Description
TS 0–15	0x0000	**Tag State Register** A TAG status bit for each line that shares an index number. A register value is stored for each index. The individual values are accessed by a sequential read. The first SC140 core read returns the value for Index = 0x0. A second read returns the value for Index = 0x1. A third read returns the value for Index = 0x2. A fourth read returns the value for Index = 0x3. For each bit, a 0 indicates that the TAG is not being used. A 1 indicates that TAG value exists for the cache line.

VBASR Valid Bit Array Status Register **0x00F0FC14**

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	VS[15–0]															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-23. VBASR Bit Descriptions

Name	Reset	Description
VS 0–15	0x0000	**Valid Bit Array Line Content** The array line content for each line and bit position. The individual values are accessed by a sequential read. The first SC140 core read returns the value for Index = 0x0, position 0. A second read returns the value for Index = 0x0, position 1. A third read returns the value for Index = 0x0, position 2, and so forth for each index up to position 15, and then for each index and position up to Index = 0x3, position 15. For each bit, a 0 indicates that memory location is not cached. A 1 indicates that the memory location is cached.

9.5 Programmable Interrupt Controller (PIC)

The MSC8113 PIC is a peripheral module that serves the $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ signals received from MSC8113 peripherals and GPIOs. The PIC is memory-mapped to the SC140 and is accessed via the SC140 QBus. The PIC includes 32 inputs for $\overline{\text{IRQ}}$ signals and $\overline{\text{NMI}}$ signals: eight asynchronous edge-triggered $\overline{\text{NMI}}$ inputs and the 24 asynchronous edge-triggered or level-triggered $\overline{\text{IRQ}}$ inputs. The PIC has an auto-vector interrupt generation that supports eight priority levels.

Note: For details, see **Chapter 17**, *Interrupt Processing*.

9.6 Local Interrupt Controller (LIC)

The LIC module complements the PIC. Its main function is interrupt concentration and localization in the SC140 core private peripheral address space to minimize the overhead of accessing the interrupt status registers at the origin and thus to maximize the performance of interrupt service routines. The LIC is optimally used in conjunction with peripherals that generate pulse interrupt requests (edge mode), but it also supports level operation mode, which is widely used in common peripherals. The LIC resides on the QBus together with the other SC140 core peripherals. It receives up to 64 interrupt sources and maps them to different PIC inputs. Interrupt priority between LIC sources is achieved by assigning a different priority level to each PIC interrupt originating in the LIC.

Note: For details, see **Chapter 17**, *Interrupt Processing*.

9.7 Extended Core Power Saving Modes

Each extended core can be put into either Stop or Wait mode.

9.7.1 Extended Core Wait Mode

An extended core enters Wait mode when it executes the **wait** instruction. In Wait mode, the SC140 core consumes minimal power because its clocks are frozen. The clocks of other modules inside the extended core do not stop, so the QBus, PIC, LIC, and MQBus and SQBus controllers are functional. The extended core exits Wait mode when there is an interrupt or a reset or when the MSC8113 device enters Debug mode by either a JTAG DEBUG_REQUEST command or assertion of EE0.

Note: When multiple cores are in Wait mode, issuing a simultaneous virtual interrupt to all these cores does not guarantee that the cores exit Wait mode on the same clock cycle.

To ensure that the SC140 core wakes up correctly from Wait mode, you must flush the write buffer before issuing the **wait** instruction. The following code shows a safe method for entering the Wait mode:

```
move.w $(wb_flush), d0
wait
```

9.7.2 Extended Core Stop Mode

An extended core enters Stop mode when it executes the **stop** instruction. In Stop mode, the SC140 core and all the extended core peripherals except for the MQBus and SQBus controllers consume minimal power because their clocks are frozen. The extended core exits this mode only when there is a reset.

MQBus and M2 Memory 10

The MQBus is a QBus protocol bus connecting the three extended cores to the M2 memory via their QBuses. The MQBus is highly optimized for a multicore environment of shared memory usage. The MQBus system is optimized to program read accesses. Together with the ICache and the prefetch mechanism, the MQBus ensures a low miss ratio for SC140 ICache accesses. The MQBus runs at the SC140 core frequency and allows data bus accesses of up to 128-bit reads and 64-bit writes. An efficient round-robin-based arbiter controls access of the SC140 cores to the MQBus. The arbiter also controls atomic operation accesses by the three SC140 cores to M2 memory. Through a parked grant mechanism, the arbitration winner holds the MQBus grant until another SC140 core initiates another request.

The 480 KB M2 memory contains 476 KB RAM and 4 KB ROM memory operating at the SC140 core frequency. M2 is a unified memory that stores both data and program code. The M2 memory can be accessed from either the MQBus or the Local bus ports. All SC140 cores can access the M2 memory through the MQBus. External hosts (on the DSI or system bus) as well as TDM and Ethernet can access the M2 memory through the local port. An intelligent arbitration algorithm between the SC140 cores efficiently uses the M2 shared memory resource.

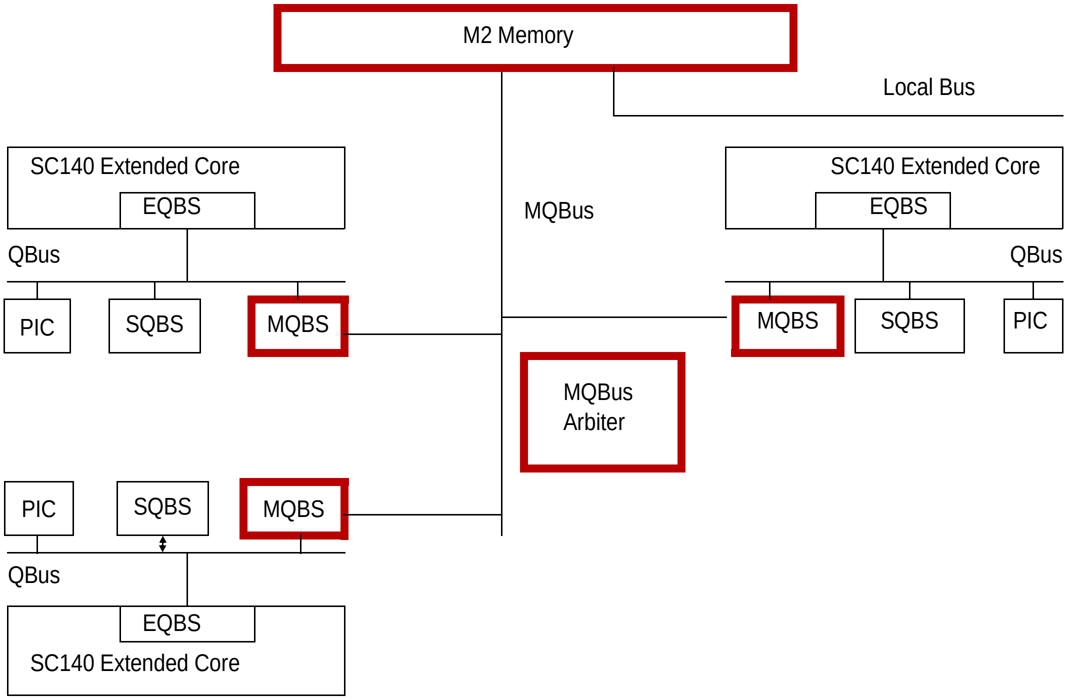


Figure 10-1. M2 Memory and MQBus System

10.1 MQBus Arbitration Model

The arbitration algorithm between the three SC140 cores controls the bus accesses so that the bus is efficiently used and there are zero gaps between accesses. The zero gaps are kept between accesses from the same SC140 core and also between accesses from different SC140 cores. The arbitration model is based on a round-robin algorithm. Each client requests a shared resource from an arbiter. One of the requesting clients—the one with the highest priority—is granted access and uses the shared resource. Through the parked grant mechanism, the arbitration winner holds the MQBus grant until another request is initiated. Therefore, the arbitration winner access path to the MQBus is shorter starting from the first consecutive access. After using the shared resource, this client is assigned the lowest priority. Then its priority increases each time another client is granted access to the shared resource.

The incoming request signals define three priority levels: high, middle, and low:

- High priority
  - Read accesses that are not prefetched
  - Immediate write accesses
- Middle priority
  - Non-immediate write accesses from the EQBS write buffer
- Low priority
  - Prefetch read accesses

During each clock cycle the access requests are handled as follows:

- If there are high-priority requests, the MQBus arbiter performs the round-robin algorithm between the high-priority requesting SC140 cores.
- If there are no high-priority requests but there are mid-priority requests, the MQBus arbiter performs the round-robin algorithm between the mid-priority requesting SC140 cores.
- If there are no high-priority or mid-priority requests but there are low-priority requests, the MQBus arbiter performs the round-robin algorithm between the low requesting SC140 cores.

Each SC140 core participating in the arbitration process may have a few queued requests in addition to the current request. These requests may have a higher priority than the current one. In these cases, the current priority is upgraded to the highest priority. For example, if the current request is for a prefetch read (low priority) and in addition the same SC140 core has another prefetch and a program miss (high priority) waiting for execution, the current request is upgraded to high priority so that the arbitration latency for the program miss access is reduced. In addition, the current priority is upgraded when the SC140 core is frozen as an outcome of an open access on the QBus.

Note: If the SC140 core is frozen waiting to perform an instruction during the middle of a cache line access, the prefetch access priority is not raised. If another core tries to access the same bus with a high priority during the bus access sampling, the bus arbitration will not grant access to the prefetch access. To avoid this unfair arbitration, the user should enable all caches and prefetch simultaneously when running code from external memory.

10.2 M2 Memory

M2 memory operates at the SC140 core frequency and is divided into 32 KB memory groups. The M2 memory system includes the M2 memory and the MQBus. The memory has two main ports:

- MQBus port, 128-bit data read, 64-bit data write.
- Local bus port, 64-bit read and write.

Each group is accessed through one of the main ports. A memory group is accessed from either the MQBus or the local bus but not from both concurrently. When both ports access the same memory group, the local bus port has a higher priority. When both ports access the same memory group, the MQBus access stalls.

The first M2 memory read access by an SC140 core without a parked grant requires seven wait states. If no other SC140 core requests the bus, the further consecutive accesses take six clock cycles. If an SC140 core has a parked grant, the first access requires six wait states. An SC140 core read from M2 memory requires at least six (or eight) wait states. Since the ICache hit ratio is high, these six (or eight) wait state accesses are rare. Because there are three SC140 cores in the MSC8113, the MQBus may be occupied and one SC140 core access may require more than six (or eight) wait states. However, an application that carefully considers memory allocation and wisely uses the ICache significantly reduces the miss ratio of all three SC140 cores, reducing the number of miss accesses to M2 memory.

10.3 Reservation Operation

The reservation atomic operation (**bmtest** instruction) is performed in two stages: a read of a certain address content and a write of the modified content back to the original address. Between these two accesses the atomic operation is defined as *open*. A write access of other SC140 cores or external hosts to the same address causes the atomic operation to fail, and the SC140 core atomic write operation to M2 memory is not performed.

The MQBus arbiter allows one such open atomic operation at a time. Other cores requesting an atomic operation are serviced only after the current atomic operation is closed.

SQBus 11

All SC140 cores can use the SQBus to fetch program code from an external memory on the 60x-compatible system bus. This feature is useful for applications needing more program memory than the internal memory. The bus runs at the SC140 core frequency and allows data bus accesses of up to 128-bit reads and 64 bit writes. The SQBus is also typically used to configure the MSC8113 device. Each SC140 core can access the IPBus through the SQBus for configuring modules such as the DSI, timers, Ethernet controller, and TDM. Accessing the system bus through the SQBus, an SC140 core can configure the DMA controller, the memory controller, and other modules. It can access the M1 memory of another extended core. Moreover, an SC140 core can access other devices on the system. For example, it can configure the DMA controller of yet another MSC8113 device on the system or directly access the M1 and M2 memories of that device.

The SQBus is a multi-master, multi-slave bus. The three SC140 cores are the masters of this bus. The slaves are the IP master for accesses to the IPBus and the system bus interface for accesses to the system bus. The IP master forwards accesses from the SQBus and from the local bus to the IPBus. When there are simultaneous requests from both the local bus and the SQBus, the SQBus receives higher priority and wins the arbitration. As **Figure 11-1** shows, the SQBus connects the three extended cores through their QBuses to the system bus and to the IPBus. The bus is highly optimized for sharing resources between multiple SC140 cores. The SQBus arbitration model is exactly the same as that for MQBus (see **Section 10.1**, *MQBus Arbitration Model*).

Access to addresses 0x01F80000–0x01FFFFFF are forwarded to the IP master. All other SQBus accesses are forwarded to the system bus interface. No pipeline is allowed when different slaves are accessed. If an access to slave A wins the arbitration and there is already an open access to slave B, then the access to slave A waits until the access to slave B completes before it executes.

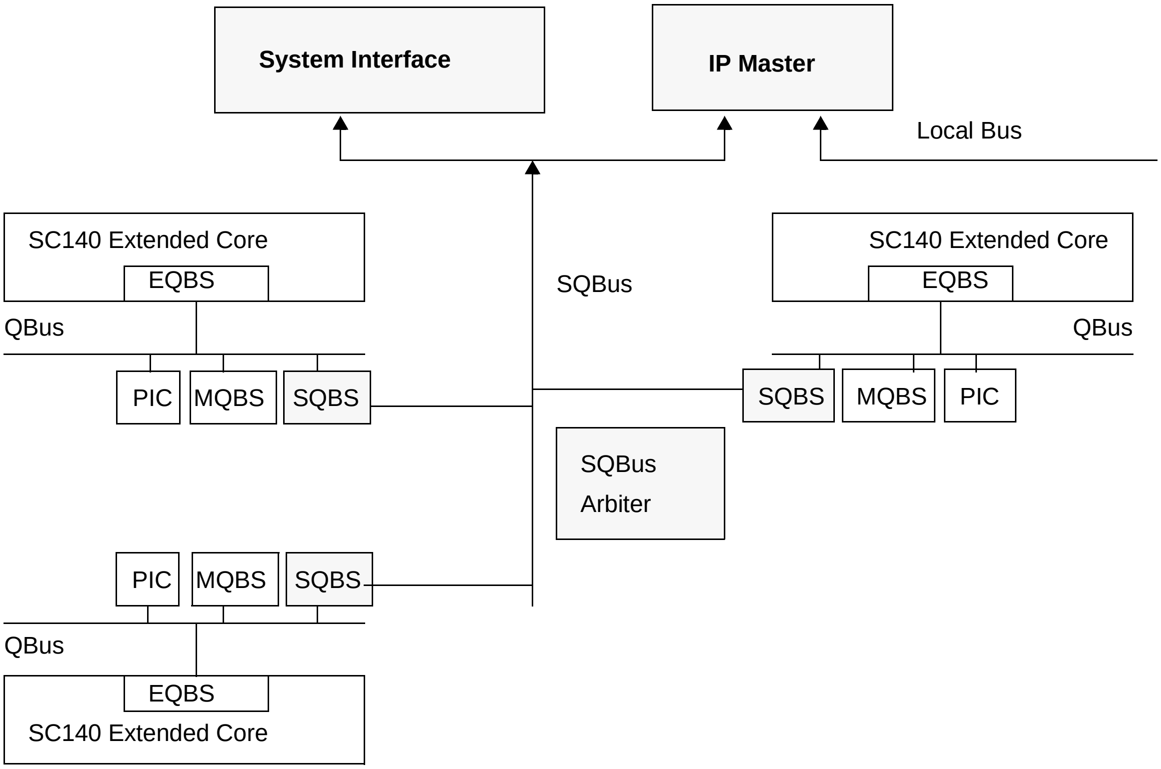


Figure 11-1. SQBus System

11.1 System Bus Interface

Each extended core accesses the system bus through the system bus interface. Access width is 8, 16, 32, 64, and 128 for reads and 8, 16, 32, and 64 for writes. The system bus interface performs 64-bit accesses. When the ETM bit in the Bus Configuration Register (BCR) is set, a 128-bit read access is performed as a burst of two 64-bit beats. Otherwise, two single 64-bit accesses are performed.

If the PRKM field of the System Bus Arbiter Configuration Register (PPC_ACR) equals 0b0101, this interface is considered as a parking master on the system bus and the access latency is improved.

Using the global bit described in **Section 9.3.9**, *EQBS Programming Model,* on page 9-18, the $\overline{\text{GBL}}$ signal on the system bus is asserted. This signal typically indicates that the MSC8113 device is writing to a cacheable area. The off-device data cache uses this signal to flag a corrupted entry.

11.2 Reservation (Atomic) Operation

The reservation atomic operation (**bmtest** instruction) is performed in two stages: a read of a certain address content and a write of the modified content back to the original address. Between these two accesses, the atomic operation is defined as open. A write access of another master to the same address space while an atomic operation is open causes the atomic operation to fail. An atomic operation closes when the atomic write operation executes. The SQBus system reservation operation is handled in two stages:

1. In the SQBus arbiter at the arbitration stage between the SC140 cores.
2. In the system bus by snooping the bus.

11.2.1 Reservation Operation in the SQBus Arbiter

The SQBus arbiter allows only one open atomic operation at a time. When an atomic operation is open, other SC140 cores requesting an atomic operation are serviced only after the atomic operation closes. When there is an open atomic operation, other SC140 cores can still perform read and write accesses. However, when another SC140 core performs a write operation to the address of the open atomic operation, the atomic operation fails.

11.2.2 Reservation Operation on the System Bus

A snooper on the system bus attempts to detect a non-atomic write to the address of an open atomic operation. When the system bus interface receives either a read or write with an atomic signal, a snooper starts to snoop the system bus. The snooper tries to detect a write on the bus to a protected address before the SQBus finishes the read modify write operation.

11.2.3 Conditions for Failure of the Reservation Operation

The reservation operation fails under the following conditions:

- A system bus write to one of the eight bytes that have the same most significant bits as the reserved address (ignoring the last three bits).
- A system bus burst write of 32 bytes or less (24 or 16 bytes) to one of the 32 bytes that have the same most significant bits as the reserved address (ignoring the last five bits).

Memory Controller 12

The MSC8113 memory controller serves two purposes:

- It supports a glueless interface to external memory and peripheral devices on the external system bus.
- It enables interfacing with the IPBus peripherals and internal memories through the internal local bus.

The memory controller controls up to eight external memory banks that are located on the external system bus and shared by a high-performance SDRAM machine, a General-Purpose Chip-Select Machine (GPCM), and three User-Programmable Machines (UPMs). It supports a glueless interface to synchronous DRAM (SDRAM), SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. Two additional memory banks control access to internal resources, using the internal local bus. This flexible memory controller allows the implementation of memory systems with very specific timing requirements:

- The SDRAM machine provides an interface to synchronous DRAMs using SDRAM pipelining, bank interleaving, and back-to-back read or write in page mode, to achieve the highest performance.
- The GPCM provides interfacing for simpler, lower-performance memory resources and memory-mapped devices. The GPCM has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals. The GPCM controls Bank 9 to access IPBus peripherals.
- The UPM supports address multiplexing of the external bus, refresh timers, and generation of programmable control signals for row address and column address strobes to allow for a glueless interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The refresh timers allow refresh cycles to be initiated. The UPM generates programmable timing patterns to the control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access request. Refresh timers periodically generate user-defined refresh cycles. There are three UPMs (A, B and C) in the memory controller. UPMs A and B can be assigned either to the system bus or to the local bus. UPM C is allocated for Bank 11, which is assigned to internal memories (M2 memory and three M1 memories).

Figure 12-1 shows the MSC8113 dual-bus architecture.

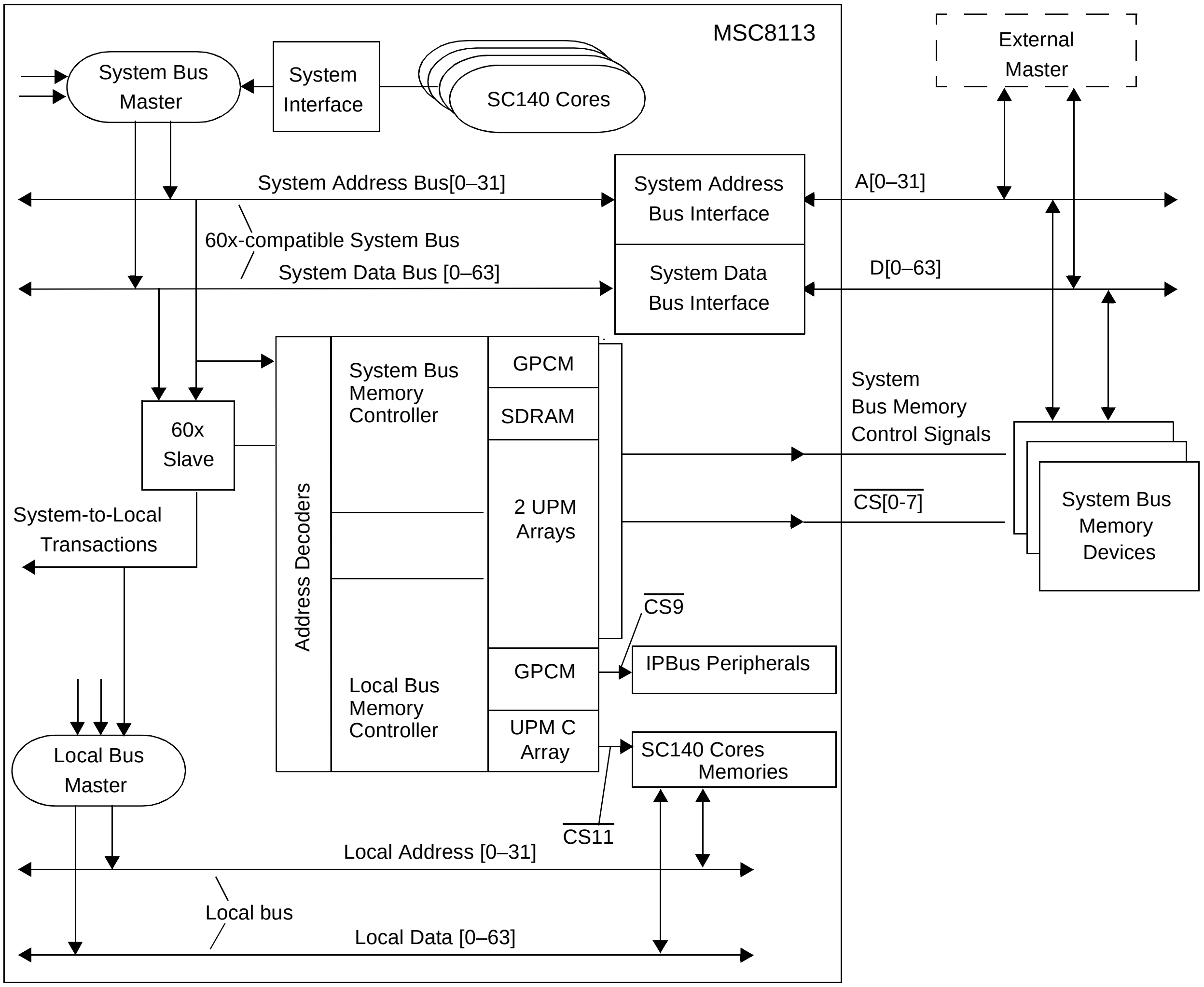


Figure 12-1. MSC8113 Dual-Bus Architecture

12.1 Basic Architecture

Each external bank can be assigned to any one of the memory controller machines (except UPMC and local bus GPCM) via BR*x*[MS] as shown in **Figure 12-2** (bank 0 and bank 2 are located on the external system bus). The BR*x*[MS] and M*x*MR[BSEL] bits (for UPMs) assign banks to the system bus or local bus as shown in **Figure 12-2**. Addresses are decoded by comparing BR*x*[BA] with a bit-wise AND of $\overline{\text{A[0–16]}}$ and OR*x*[AM]. If an address match occurs in multiple banks, the lowest numbered bank has priority. However, if a system bus access hits a bank allocated to the local bus, the access is transferred to the local bus. Local bus access hits to banks assigned to the system bus are ignored. When a memory address matches BR*x*[BA], the corresponding machine takes ownership of the external signals that control access and maintains control until the cycle ends. See **Section 12.8**, *Memory Controller Programming Model*, on page 12-95 for details on the BR*x*, OR*x*, and M*x*MR registers.

The following features are common to all machines:

- There is a 17-bit most-significant address decode on each memory bank.
- The block size of each memory bank varies between 32 KB (1 MB for SDRAM) and 4 GB (128 MB for SDRAM using bank-based interleaving).
- Normal parity can be generated and checked for any external memory bank.
- Read-modify-write parity can be generated and checked for any external memory bank with a 32-bit or 64-bit port size. Using RMW parity on a bank with a 32-bit port size requires the bus to be in strict 60x mode (BCR[ETM] = 0). See the discussion of the Bus Configuration Register (BCR) in **Section 4.2**, *SIU Programming Model*.
- Error checking and correction (ECC) can be generated for any external memory bank with a 64-bit port size.
- Each external memory bank can be selected for read-only or read/write operation.
- Each external memory bank can use data pipelining, which reduces the required data set-up time for synchronous devices.
- Each external memory bank can be controlled by an external memory controller or bus slave.

The memory controllers functionality minimizes the need for glue logic in MSC8113-based systems. In **Figure 12-3**, $\overline{\text{CS0}}$ is used with the 16-bit boot EPROM with BR0[MS] defaulting to select the GPCM. $\overline{\text{CS1}}$ is used as the $\overline{\text{RAS}}$ signal for 32-bit DRAM with BR1[MS] configured to select UPMA. $\overline{\text{PBS[0–3]}}$ are used as $\overline{\text{CAS}}$ signals on the DRAM. In the example, the boot EPROM and DRAM connect to the external system bus.

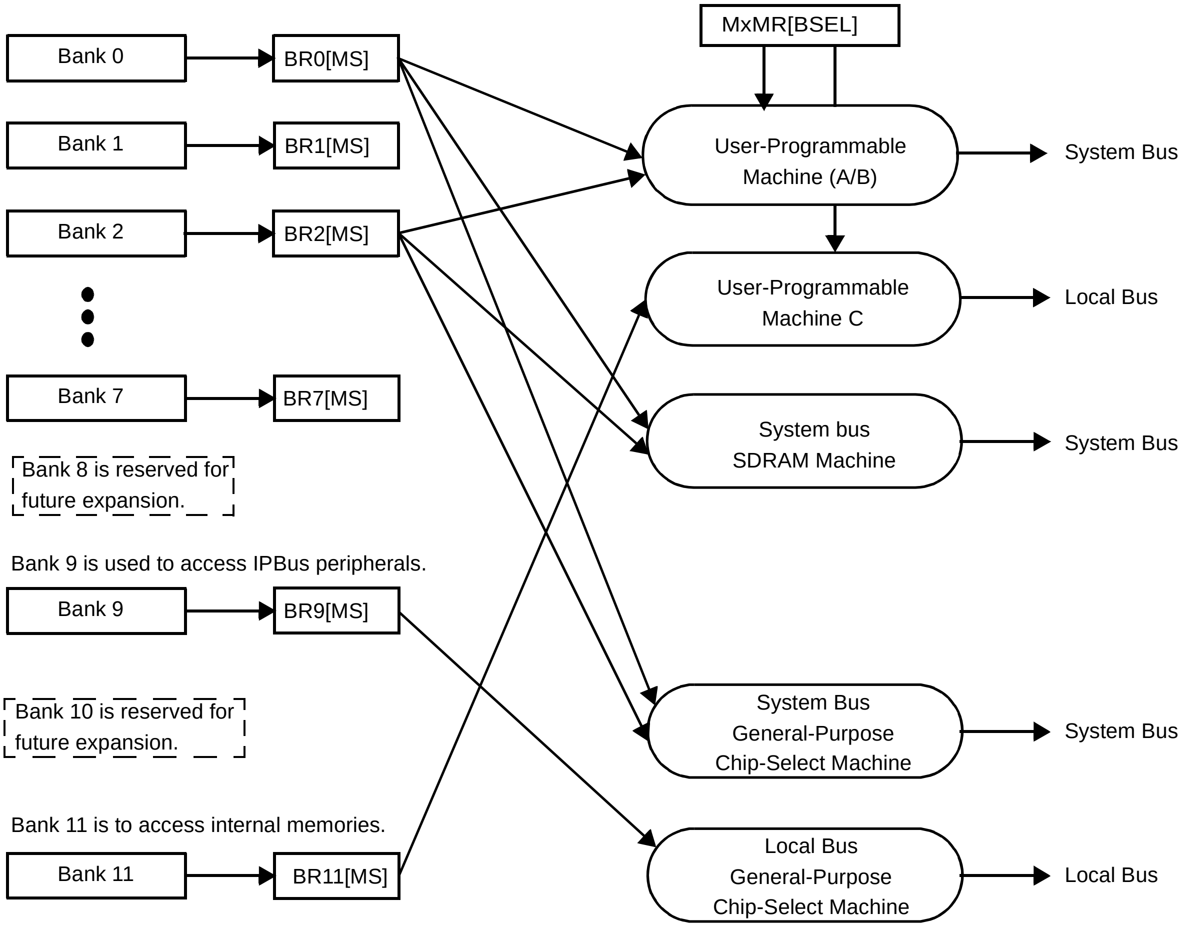

Bank 0
BR0[MS]
Bank 1
BR1[MS]
Bank 2
BR2[MS]
Bank 7
BR7[MS]
Bank 8 is reserved for future expansion.
Bank 9 is used to access IPBus peripherals.
Bank 9
BR9[MS]
Bank 10 is reserved for future expansion.
Bank 11 is to access internal memories.
Bank 11
BR11[MS]
MxMR[BSEL]
User-Programmable Machine (A/B)
User-Programmable Machine C
System bus SDRAM Machine
System Bus General-Purpose Chip-Select Machine
Local Bus General-Purpose Chip-Select Machine
System Bus
Local Bus
System Bus
System Bus
Local Bus

Figure 12-2. Memory Controller Machine Selection

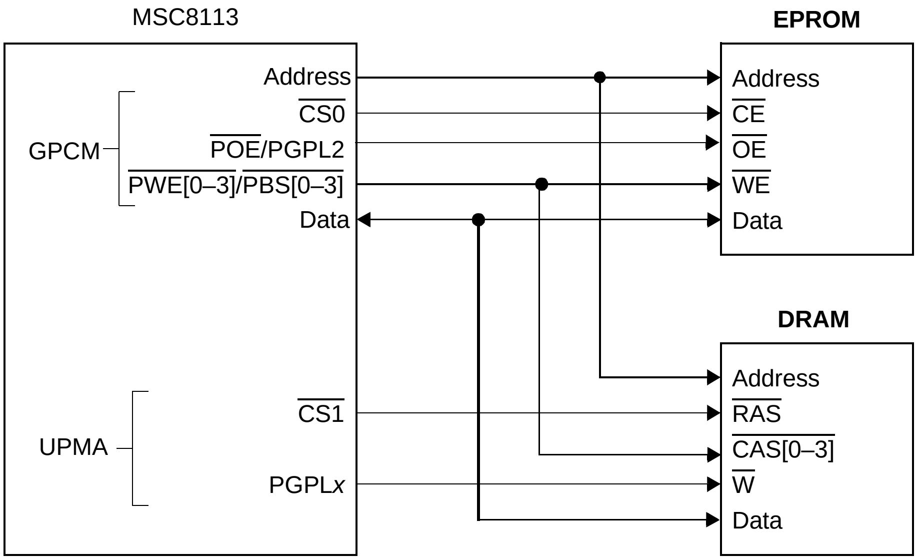


Figure 12-3. Simple System Configuration

Differences between the memory controller machines are as follows:

- *The SDRAM machine*. Provides a glueless interface to JEDEC-compliant SDRAM devices. It uses SDRAM pipelining, page mode, and bank interleaving, to deliver very high performance. To fine tune system performance, the SDRAM machine provides two types of page modes selectable per memory bank:

  — Page mode for consecutive back-to-back accesses (normal operation)
  — Page mode for intermittent accesses

  The SDRAM machine is available only on the system bus on $\overline{\text{CS[0–7]}}$; each external memory bank can be assigned to the SDRAM machine.
- *The GPCM*. Provides a glueless interface to EPROM, SRAM, Flash EPROM (FEPROM), and other peripherals. The GPCMs are available on both buses on $\overline{\text{CS[0–7]}}$ and $\overline{\text{CS9}}$; each external memory bank can be assigned to system bus GPCM. In the MSC8113 device, one internal bank uses the local bus GPCM to access the IPBus peripherals ($\overline{\text{CS9}}$). $\overline{\text{CS0}}$ is also the global (boot) chip-select for accessing the boot EPROM or Flash device. The chip-select allows 0 to 30 wait states.
- *The UPMs*. Provide a flexible interface to many types of memory devices. Each UPM controls the address multiplexing for accessing DRAM devices and the timings of $\overline{\text{BS[0–7]}}$ and GPL*x*. UPMs A and B can be assigned either to the system bus or to the local bus. Each external memory bank ($\overline{\text{CS[0–7]}}$) can be assigned to either UPMA or UPMB. In the MSC8113 device, one internal bank on the local bus uses the UPMC to access the internal memories ($\overline{\text{CS11}}$).

  Each UPM is a programmable RAM-based machine. It toggles the memory controller external signals as programmed in RAM when an internal or external master initiates any external read or write access. It also controls address multiplexing, address increment, and transfer acknowledge ($\overline{\text{TA}}$) assertion for each memory access. The UPM specifies a set of signal patterns for a user-specified number of clock cycles. The UPM RAM pattern run by the memory controller is selected according to the type of external access transacted. At every clock cycle, the logical value of the external signals specified in the RAM array is output on the corresponding UPM signals. **Figure 12-4** shows a basic configuration.

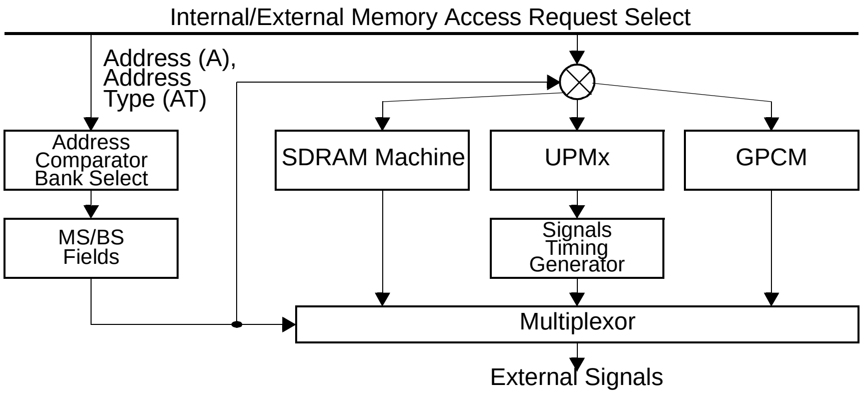

Internal/External Memory Access Request Select
Address (A), Address Type (AT)
Address Comparator Bank Select
MS/BS Fields
SDRAM Machine
UPMx
GPCM
Signals Timing Generator
Multiplexor
External Signals

Figure 12-4. Basic Memory Controller Operation

The system bus SDRAM Mode Register (PSDMR) defines the global parameters for the system bus SDRAM devices. Machine A/B/C mode registers (M*x*MR) define most of the global features for each UPM. GPCM parameters and some SDRAM and UPM parameters are defined in the Option Registers (OR*x*).

The memory controller, which is a 60x bus slave, supports two system configuration modes:

- *Non 60x-compatible mode (Single-Master mode)*. The only master on the system bus is the internal bus master.
- *60x-compatible mode (External Master mode)*. The internal master and external masters share the same system bus. Up to three external masters using the internal arbiter or more using an external arbiter.

Figure 12-5 shows an example of a typical system in which several devices can share the same system bus and memory controller.

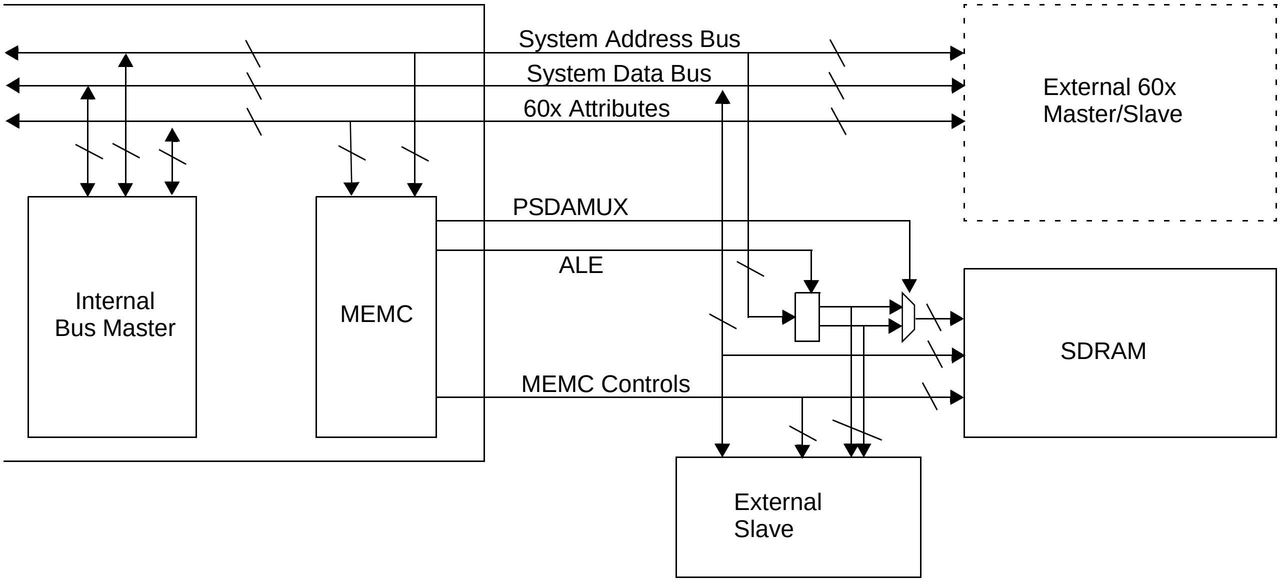


Figure 12-5. System Bus Sharing In Typical System

In Single-Master mode, since there is only one bus master, the address bus is fully driven using latches. Therefore, the memory controller can use the address for any manipulation that might needed for the current access (such as row and column addresses for SDRAM and/or address increment). **Figure 12-6** shows a memory controller access in the non 60x-compatible mode and demonstrates the validity period of each group of signals involved in the access, as well as the relationships between the various groups.

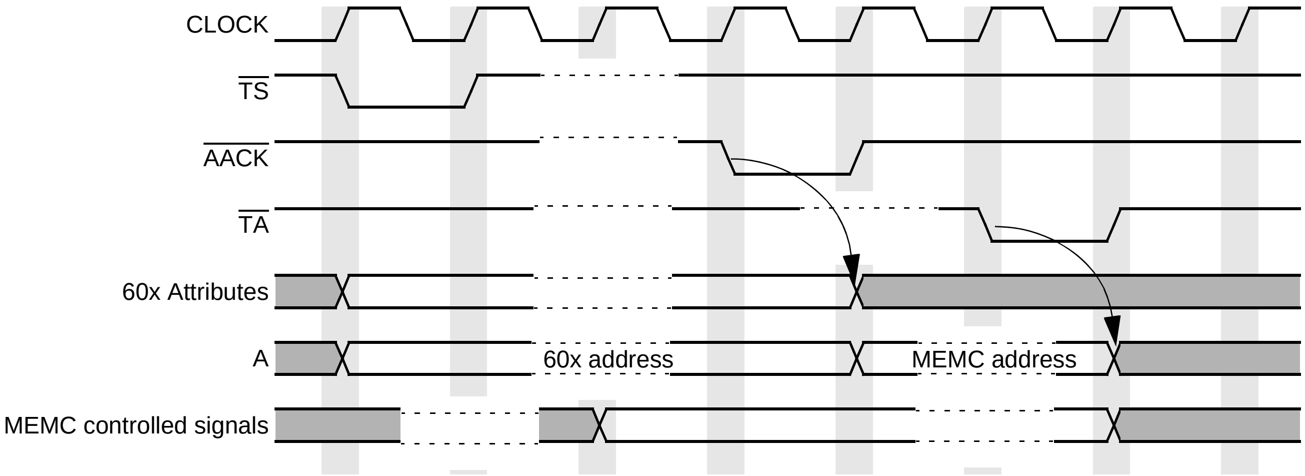
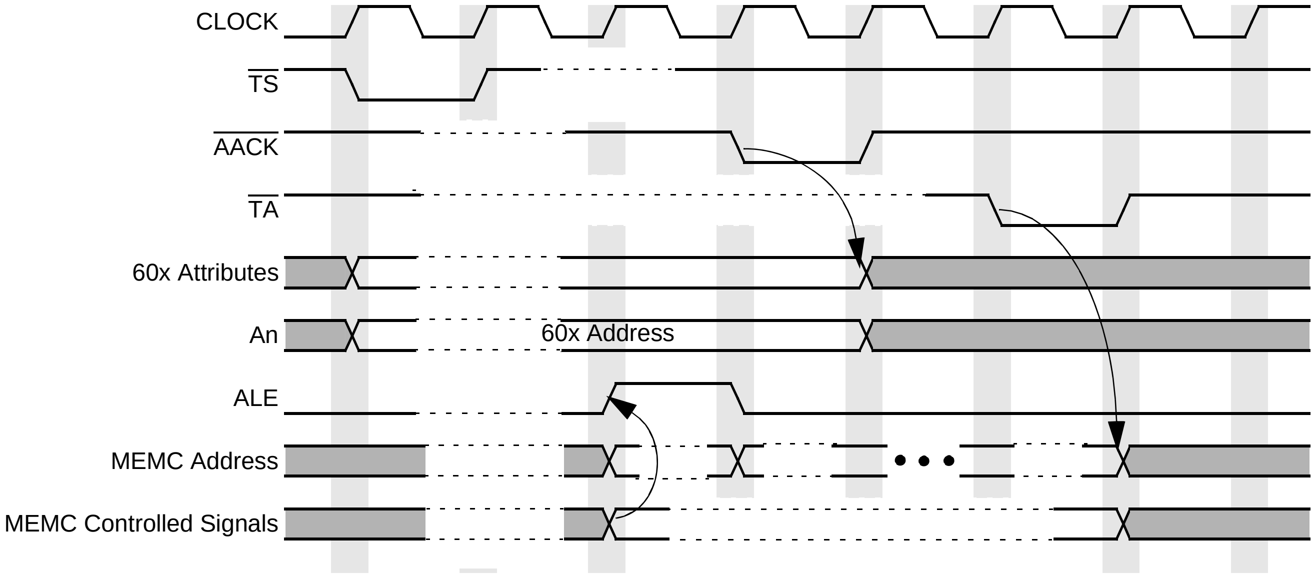


Figure 12-6. Schematic Timing Diagram for MEMC Access In Non 60x-Compatible Mode

In 60x-compatible mode there are several masters for the system bus. Therefore, the address bus is driven with the current access address only during the address phase (TS assertion to AACK assertion). The memory controller provides some additional control signals such as PSDMAMUX, ALE, and BADDR to support external address multiplexing and latching as well as burst incrementing. (60x protocol defines only the first address for the access without address increment). For details, see **Section 12.1.11** to **Section 12.1.14**, and **Section 12.6**. **Figure 12-7** shows a memory controller access in 60x-compatible mode and demonstrates the validity period of each group of signals involved in the access as well as the relationships between the various groups.



Note: The MEMC address is either BADDR and/or external latched address controlled by ALE and/or multiplexed address controlled by PSDMAMUX.

Figure 12-7. Timing Diagram for MEMC Access in 60x-Compatible Mode

The 60x attributes are the Address Transfer Attribute as described in **Section 13.1.4**, *Address Transfer Attribute*, on page 13-6. The *MEMC controlled signals* are the various signals controlled by the SDRAM or GPCM or UPM machines, as described throughout this chapter.

12.1.1 Address and Address Space Checking

The defined base address is written to the BR*x*. The bank size is written to the OR*x*. Each time a bus cycle access is requested on the system bus or local bus, addresses are compared with each bank. If a match is found on a memory controller bank, the attributes defined in the BR*x* and OR*x* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

Although system bus accesses to a bank allocated to the local bus are transferred to the local bus, local bus access hits to banks allocated to the system bus are ignored. The system-to-local 60x-compatible bus transactions have priority over regular memory bank hits.

12.1.2 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which you write to the OR*x* register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found together with bank match, the bus cycle is defined as a page hit. The SDRAM machine automatically closes an open page if the bus becomes idle, unless OR*x*[PMSEL] is set.

12.1.3 Parity Generation and Checking

Parity can be configured for any external bank. Parity is generated and checked on a per-byte basis using $\overline{\mathrm{DP}[0\text{–}7]}$ for the bank if BR*x*[DECC] = 01 for normal parity and 10 for RMW parity. BCR[EPAR] determines the global type of parity (odd or even).

Note: RMW parity can be used only for banks with a 32-bit or 64-bit port size. Using RMW parity on an SDRAM bank requires that either the system bus be placed in non-pipelined mode by writing a 1 to BCR[PLDP] or PSDMR[CL] = 10 for a CAS latency of 2. Also, using RMW parity on a bank with a 32-bit port size requires that the system bus be placed in strict 60x mode by setting BCR[ETM] to 0. See **Section 4.2**, *SIU Programming Model*.

12.1.4 Transfer Error Acknowledge ($\overline{\text{TEA}}$) Generation

The memory controller asserts the transfer error acknowledge signal ($\overline{\text{TEA}}$) (if enabled) in the following cases:

- An unaligned or burst access is attempted to internal MSC8113 space (registers).
- Any SC140 core or an external master attempts a burst access to the local bus address space.
- A bus monitor time-out.

12.1.5 Machine Check Interrupt ($\overline{\text{MCP}}$) Generation

The memory controller asserts machine check interrupt ($\overline{\text{MCP}}$) in the following cases:

- A parity error
- An ECC double-bit error
- An ECC single-bit error when the maximum number of ECC errors is reached

12.1.6 Data Buffer Controls ($\overline{\text{BCTL[0–1]}}$)

The memory controller provides two data buffer controls for the system bus ($\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$). These controls are activated when a GPCM- or UPM-controlled bank is accessed and are disabled by setting OR*x*[BCTLD]. Access to SDRAM-machine controlled bank does not activate the $\overline{\text{BCTL}x}$ controls.

The $\overline{\text{BCTL}x}$ signals have programmable polarity and functionality that are controlled by SIUMCR[BCTLC]. For details on possible polarity and functionality of $\overline{\text{BCTL}x\text{,}}$ see the discussion of the SIU Module Configuration Register (SIUMCR) in **Section 4.2**, *SIU Programming Model*.

The $\overline{\text{BCTL}x}$ signals are asserted on the rising edge of the external bus clock on the first cycle of the memory controller operation. They are deasserted on the rising edge of the external bus clock after the last assertion of $\overline{\text{PSDVAL}}$ if the access is asserted. See **Section 12.1.8**, *Partial Data Valid Indication (PSDVAL)*, on page 12-10. If back-to-back memory controller operations are pending, $\overline{\text{BCTL}x}$ signals are not deasserted.

12.1.7 Atomic Bus Operation

The MSC8113 device supports the following kinds of atomic bus operations BR*x*[ATOM]:

- *Read-after-write (RAWA)*. When a write access hits a memory bank in which BR*x*[ATOM] = 01, the MSC8113 locks the bus for the exclusive use of the accessing master (internal or external). During the lock period, no other device is granted the bus mastership. The lock is released when the master that created the lock accesses the same bank with a read transaction. If the master fails to release the lock within 256 bus clock cycles, the lock is released, and a special interrupt is generated. This feature is for CAM operations.
- *Write-after-read (WARA)*. When a read access hits a memory bank in which BR*x*[ATOM] = 10, the MSC8113 locks the bus for the exclusive use of the accessing master (internal or external). During the lock period, no other device is granted the bus mastership. The lock is released when the master that created the lock accesses the same bank with a write transaction. If the master fails to release the lock within 256 bus clock cycles, the lock is released, and a special interrupt is generated.

Note: This mechanism does not replace the 60x-compatible reservation mechanism.

12.1.8 Partial Data Valid Indication ($\overline{\text{PSDVAL}}$)

The system bus and the local bus have an internal 64-bit data bus. According to the 60x bus specification, $\overline{\text{TA}}$ is asserted when up to 64 bits (8 bytes) of data is transferred. Because the MSC8113 device memories can have port sizes smaller than 64 bits, there is a need for a partial data valid indication. The memory controller uses $\overline{\text{PSDVAL}}$ to indicate that data is latched by the memory on write accesses or that valid data is present on read accesses. The quantity of the data depends on the memory port size and the transfer size. The memory controller accumulates $\overline{\text{PSDVAL}}$ assertions, and when 64 bits (or the transfer size) are transferred, the memory controller asserts $\overline{\text{TA}}$ to indicate that a 60x data beat was transferred. **Table 12-1** shows the number of $\overline{\text{PSDVAL}}$ assertions needed for one $\overline{\text{TA}}$ assertion under various circumstances. **Figure 12-8** shows a 64-bit transfer on 32-bit port size memory.

Table 12-1. Number of $\overline{\text{PSDVAL}}$ Assertions Needed for $\overline{\text{TA}}$ Assertion

Port Size	Transfer Size	$\overline{\text{PSDVAL}}$ Assertions	$\overline{\text{TA}}$ Assertions
64	Any	1	1
32	64 bits (8 bytes)	2	1
32	32 bits (4 bytes) 32-bit aligned	1	1
16	64 bits (8 bytes)	4	1
16	32 bits (4 bytes)	2	1
16	16/8 bits (2 bytes or 1 byte)	1	1
8	64 bits (8 bytes)	8	1
8	32 bits (4 bytes)	4	1
8	16 bits (2 bytes)	2	1
8	8 bits (1 bytes)	1	1

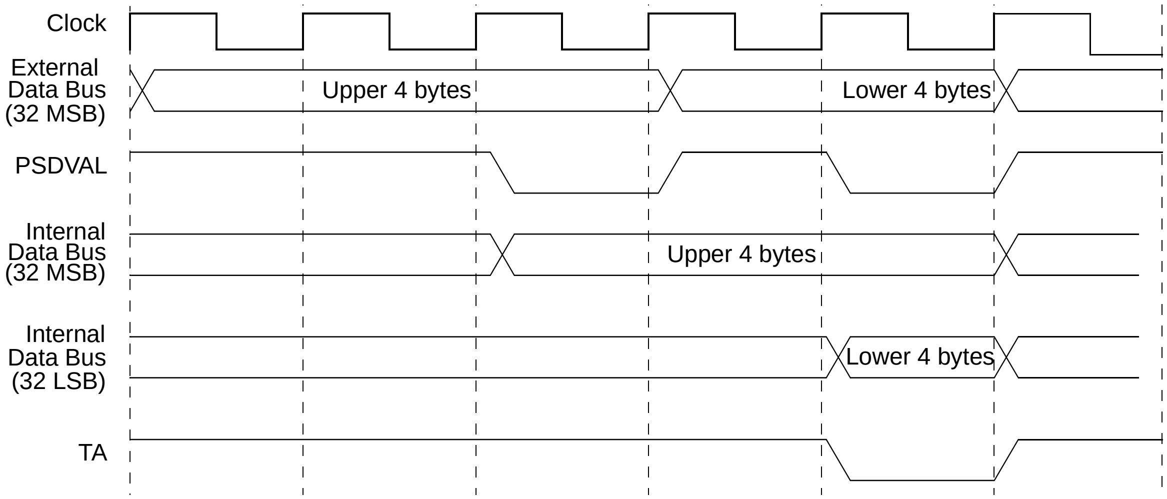

Clock
External Data Bus (32 MSB)
Upper 4 bytes
Lower 4 bytes
PSDVAL
Internal Data Bus (32 MSB)
Upper 4 bytes
Internal Data Bus (32 LSB)
Lower 4 bytes
TA

Figure 12-8. Partial Data Valid for 32-Bit Port Size Memory, 64-bit Transfer

12.1.9 ECC/Parity Byte-Select ($\overline{\text{PPBS}}$)

Systems that use ECC or RMW parity require an additional memory device that requires byte-select like a normal data device. ANDing $\overline{\text{PBS[0–7]}}$ through external logic to achieve the logical function of this byte-select can affect the memory access timing because it adds a delay to the byte-select path. The optional memory controller parity-byte-select signal is an internal AND of the eight byte-selects, allowing glueless and faster connection to ECC/RMW-parity devices. This option is enabled by setting SIUMCR[PBSE], as described in **Section 4.2**, *SIU Programming Model*.

12.1.10 Data Pipelining

Multiple-MSC8113 systems that use data checking, such as parity, face a timing problem when synchronous memories, such as SDRAM, are used. Because these devices can output data every cycle and because the data checking requires additional data set-up time, the timing constraints are extremely hard to meet. In such systems, you can eliminate the additional data set-up time requirement by setting the data pipelining bit, BR*x*[DR]. This creates data pipelining of one stage within the memory controller in which the data check calculations are done.

In systems that involve both PowerQUICC II-type masters and a 60x-compatible master, this feature can still be used on the system bus with the following restrictions:

- The arbiter and the memory controller are in the same MSC8113.
- The register field BCR[NPQM] is set correctly.

See **Section 12.6**, *External Master Support (60x-Compatible Mode)*, on page 12-83, and the discussion of the Bus Configuration Register (BCR) in **Section 4.2**, *SIU Programming Model*.

12.1.11 60x-Compatible Mode

The MSC8113 memory controller supports External master mode (60x-compatible mode), in which several 60x masters share the external system bus. The features described in **12.1.12**, **12.1.13**, and **12.1.14**, support this mode. For details see **Section 12.6**, *External Master Support (60x-Compatible Mode),* on page 12-83.

12.1.12 External Memory Controller Support

The MSC8113 device has an option to allocate specific banks (address spaces) to be controlled by an external memory controller or bus slave while retaining all the bank properties: port size, data check, atomic operation, and data pipelining. Programming BR*x* and OR*x*[AM] and setting the external memory controller bit, BR*x*[EMEMC] automatically assigns the bank to the system bus. For an access that hits the bank, all bus acknowledgment signals (such as $\overline{\text{AACK}}$, $\overline{\text{PSDVAL}}$, and $\overline{\text{TA}}$) and the memory-device control strobes are driven by an external memory controller or slave. If the device that initiates the transaction is internal to the MSC8113 device, the memory controller handles the port size, data checking, atomic locking, and data pipelining as if the access were governed by it.

This feature allows multiple MSC8113 systems to be connected in 60x-compatible mode without losing functionality and performance. It also makes it easy to connect other 60x-compatible slaves on the 60x bus.

12.1.13 External Address Latch Enable Signal (ALE)

The memory controller provides control for an external address latch needed on the system bus in 60x-compatible mode. ALE is asserted for one clock cycle on the first cycle of each memory controller transaction. In this section, whenever ALE is not on a timing diagram, assume that it is asserted on the first cycle in which $\overline{\text{CS}}$ can be asserted.

Note: ALE is relevant only in 60x-compatible mode.

12.1.14 BADDR[27–31] Signal Connections

Use BADDR[27–31] to generate addresses to memory or peripheral devices for burst accesses on system bus in 60x-compatible mode. In this mode, when a master initiates an external 60x-compatible bus transaction, it reflects the value of A[27–31] on the first clock cycle of the memory access. The memory controller latches these signals, and on subsequent clock cycles, BADDR[27–31] increments as programmed in the UPM or after each data beat is sampled in the GPCM or after each READ/WRITE command executes in the SDRAM machine. Not all the BADDR lines are necessarily used. **Table 12-2** shows which BADDR lines are needed for the device connection.

Table 12-2. BADDR Connections

BADDR*x*	64/72-Bit Port Size		32-Bit Port Size		Any 16-Bit Port Size Device	Any 8-Bit Port Size Device
	SDRAM	Non-SDRAM	SDRAM	Non-SDRAM		
BADDR27	Not connected	Connected	Not connected	Connected	Connected	Connected
BADDR28	Not connected	Connected	Not connected	Connected	Connected	Connected
BADDR29	Not connected	Not connected	Not connected	Connected	Connected	Connected
BADDR30	Not connected	Not connected	Not connected	Not connected	Connected	Connected
BADDR31	Not connected	Not connected	Not connected	Not connected	Not connected	Connected

12.2 SDRAM Machine

Note: To understand the operation of the Memory Controller SDRAM machine, you must be familiar with SDRAM devices protocol and behavior.

The MSC8113 device provides an SDRAM interface (machine) only for the system bus. The machine provides the necessary control functions and signals for JEDEC-compliant SDRAM devices. Each external memory bank can control an SDRAM device on the system bus. **Table 12-3** lists the SDRAM interface signals controlled by the memory controller.

Table 12-3. SDRAM Interface Signals

60x-compatible System Bus	Comments
$\overline{\text{CS}[0–7]}$	Device select
$\overline{\text{PSDRAS}}$	$\overline{\text{RAS}}$
$\overline{\text{PSDCAS}}$	$\overline{\text{CAS}}$
$\overline{\text{PSDWE}}$	$\overline{\text{WEN}}$
PSDA10	“A10” control
$\overline{\text{PSDDQM}[0–7]}$	Byte-select
PSDAMUX	External address multiplexing control (asserted = row, deasserted = column)

Additional controls are available in 60x-compatible mode:

- *ALE*. External address latch enable

Figure 12-9 shows an eight-bank, 128 MB system. Each bank consists of eight 2 × 1 Mb × 8 SDRAMs. The banks connect to the system bus. The SDRAM memory clock must operate at the same frequency as the system clock and be phase-aligned with it.

12.2.1 Supported SDRAM Configurations

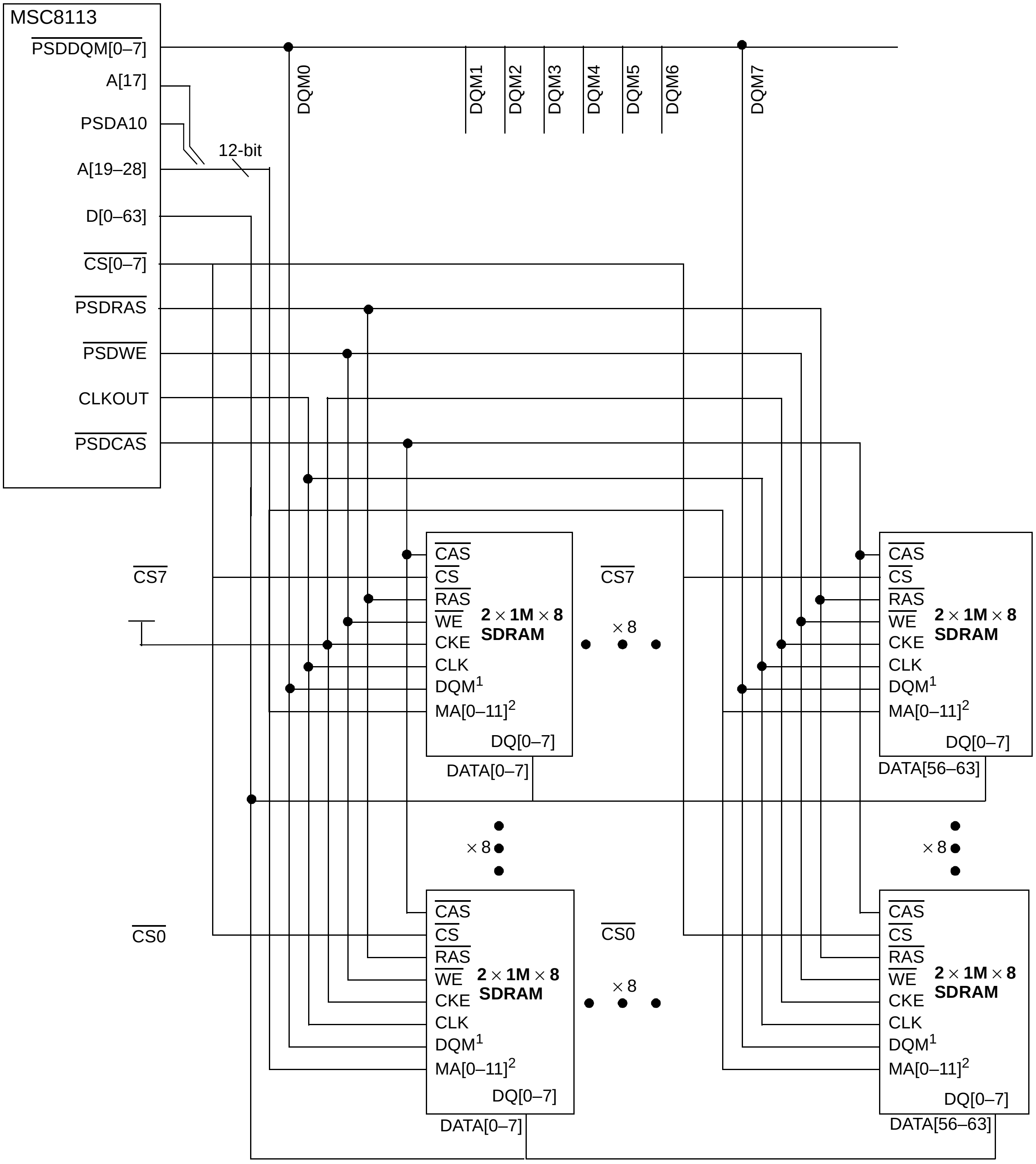
The MSC8113 memory controller supports any SDRAM configuration, but all SDRAM devices on the same bus must have the same port size and timing parameters.

12.2.2 SDRAM Power-On Initialization

At system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR*x*, BR*x*, PSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

1. Issue a PRECHARGE-ALL-BANKS command.
2. Issue eight CBR-REFRESH commands.
3. Issue a MODE-SET command to initialize the mode register.

The initial commands are executed by setting PSDMR[OP] and accessing the SDRAM with a single-byte transaction. See the discussion of the system bus SDRAM Mode Register (PSDMR) in **Section 12.8**, *Memory Controller Programming Model*, on page 12-95. Software should ensure that no memory operations begin until this process completes.



Notes: **1.** The DQM signal is defined as the byte-lane mask signal in some memory devices or the byte-lane enable signa other devices. As a mask signal, it is asserted high and a value of 1 masks the byte lane. As an enable signal, it is asserted low and a 0 enables the signal. For either definition, a 1 disables the byte lane and a 0 enables the byte lar

2. The Memory Address (MA[0–11]) signals in this figure are driven by the MSC8113 A[19–28], PSDA10, and A17 signals, respectively.

Figure 12-9. 128-Mb SDRAM (Eight-Bank Configuration, Banks 1 and 8 Shown)

12.2.3 JEDEC-Standard SDRAM Interface Commands

The MSC8113 device performs all accesses to SDRAM using JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device must be sampled on the rising edge of the bus clock. The MSC8113 provides the SDRAM interface commands listed in **Table 12-4**.

Table 12-4. SDRAM Interface Commands

Command	Description
BANK-ACTIVATE	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another BANK-ACTIVATE is issued.
MODE-SET	Sets SDRAM options—$\overline{\text{CAS}}$ latency, burst type, and burst length. $\overline{\text{CAS}}$ latency depends on the SDRAM device used (some SDRAMs provide CAS latency of 1, 2, or 3; some provide a latency of 1, 2, 3, or 4, and so on). Burst type must be chosen according to the 60x cache wrap (sequential). Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the MSC8113 supports only a 4-beat burst for a 64-bit port and an 8-beat burst for a 32-bit port. The MSC8113 does not support burst lengths of 1, 2, and a page for SDRAMs. The mode register data ($\overline{\text{CAS}}$ latency, burst length, and burst type) is programmed into the PSDMR register by initialization software at reset. After the PSDMR is set, the MSC8113 transfers the information to the SDRAM array by issuing a MODE-SET command. **Section 12.2.12**, *SDRAM Signals: mode-set Command Timing,* on page 12-28, gives timing information.
precharge (single bank/all banks)	Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers to prepare for reading another row in the SDRAM array. A PRECHARGE command must be issued after a read or write if the row address changes on the next access. The MSC8113 uses PSDA10 to distinguish the PRECHARGE-ALL-BANKS command. The SDRAMs must be compatible with this format.
read	Latches the column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each successive clock, additional data is output without additional READ commands. Burst size determines the amount of data transferred. At the end of the burst, the page remains open.
refresh	Causes a row to be read in both memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After it is read, a row is automatically rewritten into the memory array. Both banks must be in a precharged state before REFRESH executes.
write	Latches the column address and transfers data from the data signals to the selected sense amplifier as determined by the column address. During each successive clock, additional data is transferred to the sense amplifiers from the data signals without additional WRITE commands. Burst size determines the amount of data transferred. At the end of the burst, the page remains open.

12.2.4 Page-Mode Support and Pipeline Accesses

The SDRAM interface supports back-to-back page mode. A page (SDRAM row) remains open as long as back-to-back accesses that hit the page are generated on the bus. The page is closed once the bus becomes idle unless OR*x*[PMSEL] is set. If BCR[ETM] = 1, the use of SDRAM pipelining also allows for back-to-back data phases to occur as required by the 60x bus specification for data stream mode.

12.2.5 Bank Interleaving

Bank interleaving is used to achieve a high data rate during switches from one SDRAM row to another. The SDRAM machine has two modes of interleaving: bank based interleaving and page based interleaving. Interleaving mode defines which address bits are used as SDRAM bank selects. According to these address bits, the SDRAM machine performs the interleaving.

The SDRAM interface supports bank interleaving. If a missed page is in a different SDRAM bank than the currently open page, the SDRAM machine first issues an ACTIVATE command to the new page and later issues a DEACTIVATE command to the old page, thus eliminating the DEACTIVATE time overhead. Both pages must reside on different SDRAM devices or on different internal SDRAM banks. The second option can be disabled by setting OR*x*[IBID]. Set this bit if the BNKSEL signals are not used in 60x-compatible mode.

The bank interleaving feature allows for acceleration of the switching from one SDRAM bank to another SDRAM bank. If the SDRAM device supports activating new bank before deactivating the previous bank, the memory controller will use this feature to active the new bank and deactivate the old bank at optimal timing for maximal data rate. The address bits that select the bank are controlled by PSDMR[PBI].

The following two methods are used for internal bank interleaving:

- *Page-Based Interleaving*. For use when a long consecutive access to SDRAM is expected. The access spans more than one SDRAM row. This type of bank interleaving yields the best performance and is the preferred interleaving method. This method uses low address bits as the bank select for the SDRAM, thus allowing interleaving on every page boundary. During the access, the following row to one that is currently accessed is in different SDRAM bank. This enables page interleaving and maximal data rate. *Page-Based Interleaving* is activated by setting PSDMR[PBI] = 1. See **Section 12.2.14.1**, *SDRAM Configuration Example (Page-Based Interleaving)*, on page 12-29.
- *Bank-Based Interleaving*. For use when SDRAM accesses are shorter than one SDRAM row and not consecutive. The most-significant address bits are the bank select for the SDRAM, thus allowing interleaving only on bank boundaries. Bank-based interleaving is activated by clearing PSDMR[PBI]. See **Section 12.2.14.2**, *SDRAM Configuration Example (Bank-Based Interleaving)*, on page 12-31.

12.2.6 BNKSEL Signals in Single-MSC8113 Bus Mode

The BNKSEL signals provide the following functionality in single-MSC8113 bus mode:

- If bank-based interleaving is used, BNKSEL signals facilitate compatibility with SDRAMs that have different numbers of row or column address lines. The address lines of the MSC8113 bus and the BNKSEL lines can be routed independently to the address lines and BA lines of the DIMM. All SDRAMs populated on an MSC8113 bus must still have the same organization. This flexibility merely allows the SDRAMs to be populated as a group with larger or smaller devices as appropriate.

  If BNKSEL lines are not used, the number of row and column address lines of the SDRAMs affect which MSC8113 address bus lines on which the bank select signals are driven, and thus require that the BA signals of the SDRAMs be routed to those address lines, thus limiting flexibility.
- If BCR[EAV] is programmed, BNKSEL signals facilitate logic analysis of the system. Otherwise, the logic analyzer equipment must understand the address multiplexing scheme of the board and intelligently reconstruct the address of bus transactions.
- Configure SIUMCR[TCPC] to “10” to enable BNKSEL signal functionality.

12.2.7 SDRAM Address Multiplexing (SDAM and BSMA)

In single MSC8113 mode, the lower bits of the address bus connect to the device address port, and the memory controller multiplexes the row/column and the internal banks select lines, according to PSDMR[SDAM] and PSDMR[BSMA]. **Table 12-5** shows how PSDMR[SDAM] settings affect address multiplexing. PSDMR[BSMA] selects which address lines from the multiplexed address serve as the device bank selects. These bits are driven on to the BNKSEL lines if SIUMCR[TCPC] enables this (See **Section 4.1.6**, *SIU Multiplexing*, on page 4-8). If the BNKSEL is not enabled on to the external pins, the address bits selected by PSDMR[BSMA] should connect directly to the devices BA inputs. For an example of PSDMR[BSMA] functionality, see **Section 12.2.14**, *SDRAM Configuration Examples*, on page 12-29.

Note: In the 60x-compatible mode, the system bus address must be latched and multiplexed by glue logic that is controlled by ALE and PSDAMUX. ALE is asserted on the second to last cycle of the address phase, thus guarantying both set-up and hold times for the external latch. The external latch should sample all address bits needed for the generation of both row and column addresses. PSDAMUX controls the external multiplexor where PSDAMUX=1 means the row address should be multiplexed as output and PSDAMUX=0 means the column address should be the multiplexed as output. PSDAMUX is also available externally in single MSC8113 mode. However, you must still configure PSDMR[SDAM]. **Table 12-5** shows SDRAM address multiplexing for A[0–15], for both 60x compatible and non 60x compatible modes.

Table 12-5. SDRAM Address Multiplexing (A[0–15])

SDAM	External Bus Address Lines	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
000	**Signal driven on external lines when address multiplexing is enabled**	—	—	—	—	—	—	—	—	—	—	—	—	—	A5	A6	A7
001		—	—	—	—	—	—	—	—	—	—	—	—	—	—	A5	A6
010		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	A5
011		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
100		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
101		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 12-6 shows SDRAM address multiplexing for A[16–31].

Table 12-6. SDRAM Address Multiplexing (A[16–31])

SDAM	External Bus Address Lines	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	**Signal driven on external lines when address multiplexing is enabled**	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		—	—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Example 12-1. Address Multiplexing

This example demonstrates the impact of PSDMR[BSMA] and PSDMR[SDMA] on address multiplexing. Pay attention to the impact of PSDMR[SDMA] on PSDMR[BSMA]. The mutual impact is relevant both for single master and multi master. Bank interleaving is based on the address bits that select the SDRAM bank and impact SDRAM control signals timing in both modes: multi master and single master.

- *A[0–31] are the logic address.* A[0–31] are the address lines.
- *PSDMR[SDMA] configured to 011*. Multiplexes A[5–20] to A[16–31].
- *PSDMR[BSMA] configured to 100*. Selects A[16–18] as the bank select address. In this case A[5–7] are driven both on BNKSEL[0–2] and on A[16–18].
- *The SDRAM device inputs BA*. In this case, the lines can be connected either to A[16–18] or to BNKSEL[2–0].

12.2.8 SDRAM Read/Write Transactions

The SDRAM interface handles the following read/write transactions:

- Single-beat reads/writes up to 64 bits (8 bytes)
- Bursts of 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes)

SDRAM devices perform bursts for each transaction. The burst length depends on the port size. For 64-bit ports, it is a burst of 4. For 32-bit ports, it is a burst of 8. For reads that require less than the full burst length, extraneous data in the burst is ignored. For writes that require less than the full burst length, the MSC8113 protects non-targeted addresses by driving $\overline{\text{PSDDQM}x}$ high on the irrelevant cycles of the burst. However, system performance is not compromised since the MSC8113 immediately begins executing any pending transaction, effectively terminating the burst early.

12.2.9 SDRAM Refresh

The memory controller supplies auto (CBR) refreshes to SDRAM according to the interval specified in PSRT. This represents the time period required between refreshes. The value of PSRT depends on the specific SDRAM devices and the operating frequency of the MSC8113 bus. This value should allow for a potential collision between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

There are two levels of refresh request priority, low and high. The low-priority request is generated as soon as the refresh timer expires and is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high-priority and is served when the current memory controller operation finishes.

12.2.10 SDRAM Signals: Device-Specific Parameters

Software must assign correct values to some device-specific parameters that can be extracted from the data sheet. The values are stored in the OR*x* and PSDMR registers. These parameters include the following:

- *Precharge-to-Activate Interval*. Controlled by PSDMR[PRETOACT], defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command.

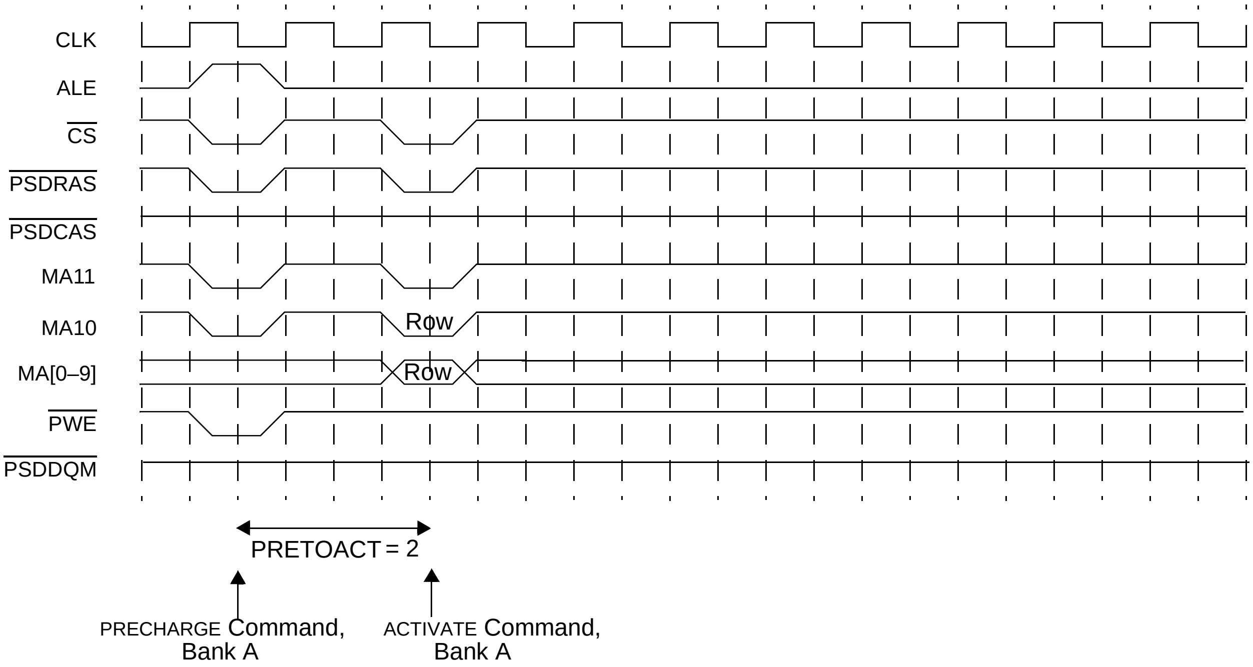


Figure 12-10. PRETOACT = 2 (2 Clock Cycles)

- *Activate to Read/Write Interval*. Controlled by PSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command.

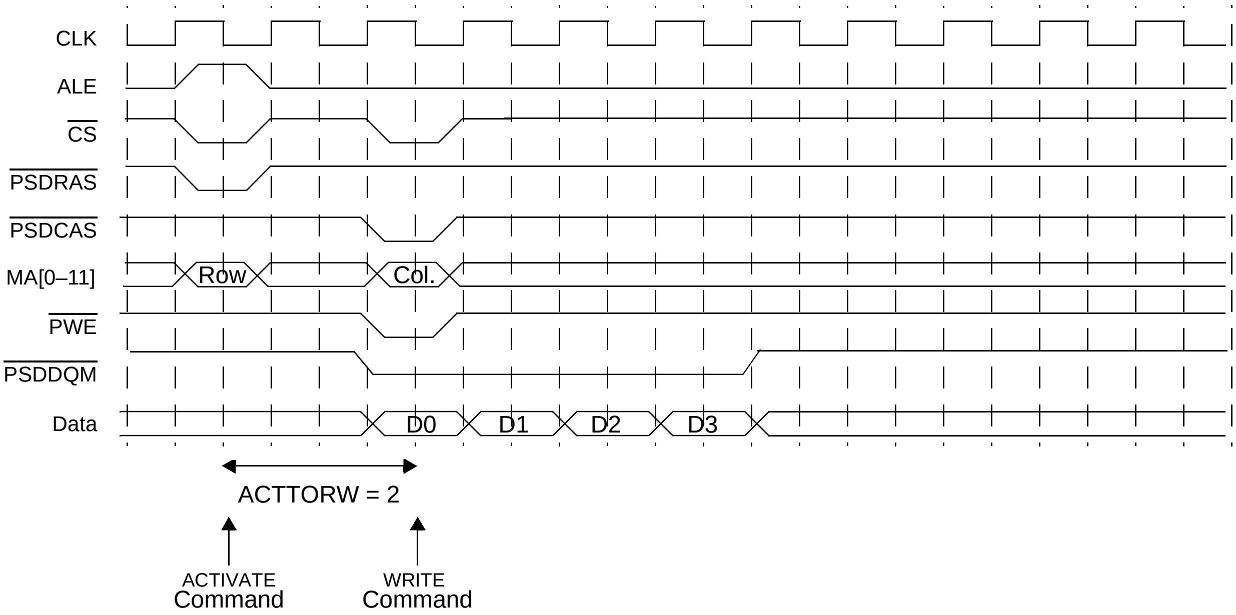


Figure 12-11. ACTTORW = 2 (2 Clock Cycles)

- *Column Address to First Data Out—$\overline{\text{CAS}}$ Latency*. Controlled by PSDMR[CL], defines the timing for first read data after SDRAM samples a column address.

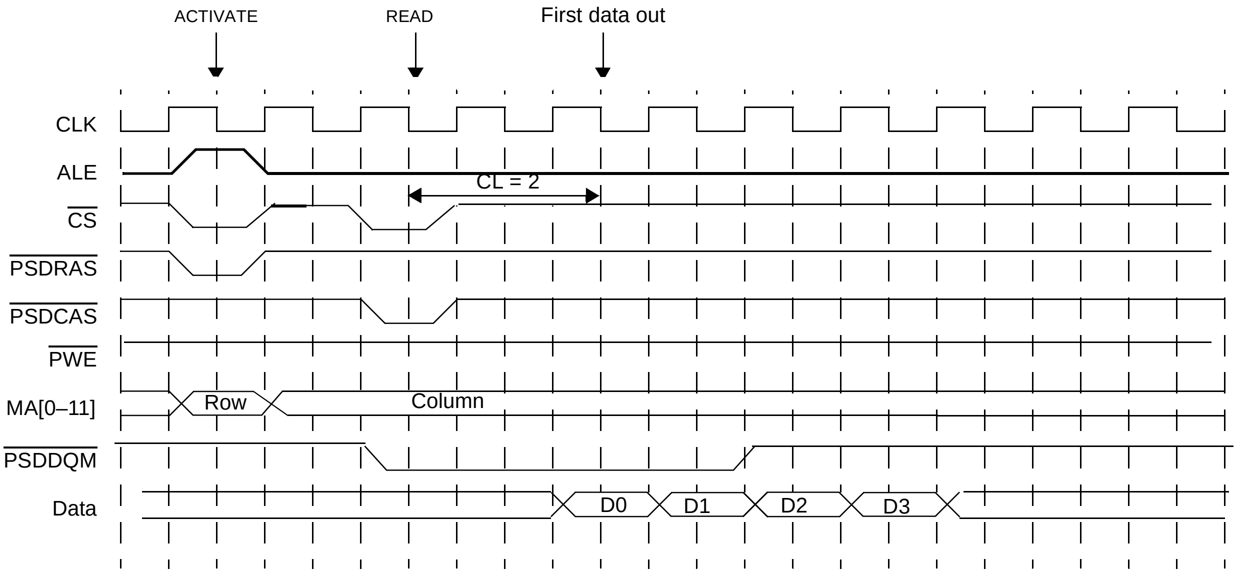


Figure 12-12. CL = 2 (2 Clock Cycles)

- *Last Data Out to Precharge*. Controlled by PSDMR[LFDOTOPRE], defines the earliest timing for PRECHARGE command after the last data is read from the SDRAM. It is always related to the CL parameter.

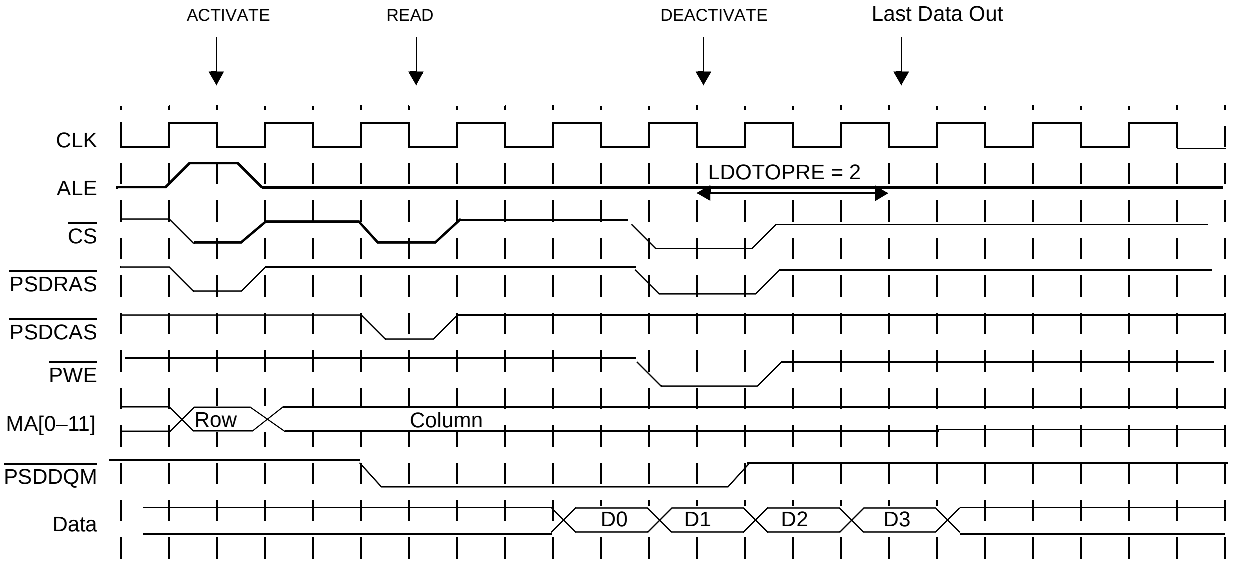


Figure 12-13. LDOTOPRE = 2 (–2 Clock Cycles)

- *Last Data In to Precharge—Write Recovery*. Controlled by PSDMR[WRC], defines the earliest timing for the PRECHARGE command after the last data is written to the SDRAM.

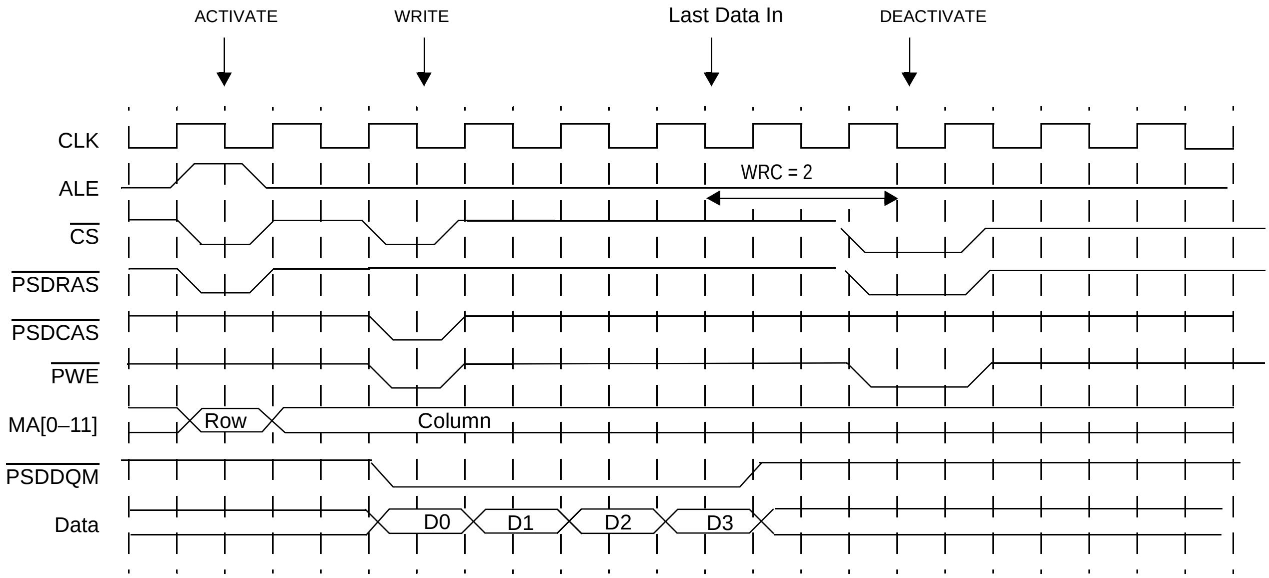


Figure 12-14. WRC = 2 (2 Clock Cycles)

- *Refresh Recovery Interval (RFRC)*. Controlled by PSDMR[RFRC], defines the earliest timing for an ACTIVATE command after a REFRESH command.

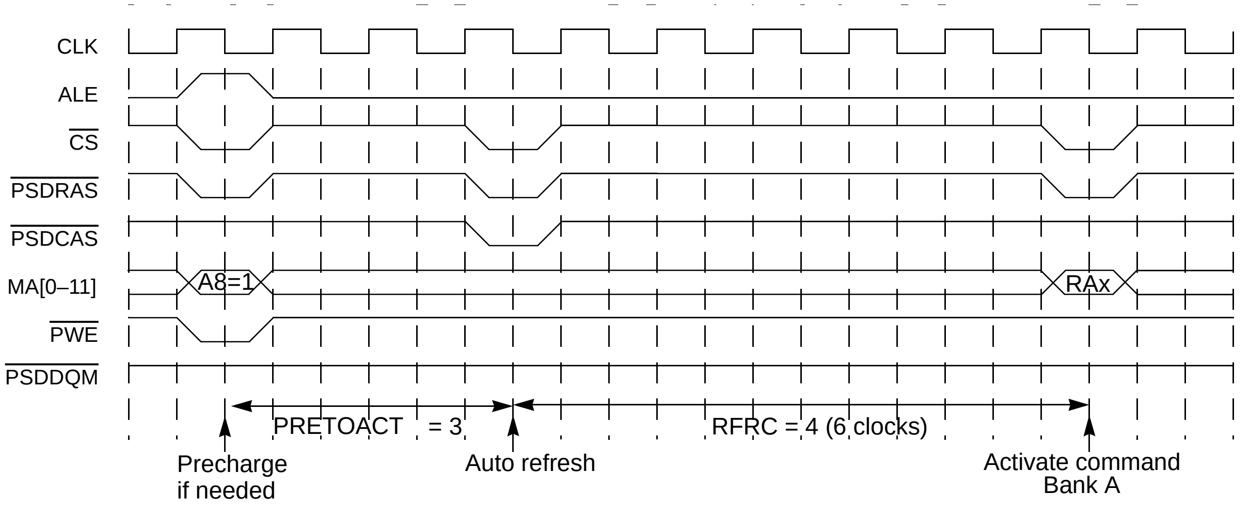


Figure 12-15. RFRC = 4 (6 Clock Cycles)

- *External Address Multiplexing Signal*. In 60x-compatible mode, external address multiplexing is placed on the address lines. If the additional delay of multiplexing is endangering the device set-up time, PSDMR[EAMUX] should be set. Setting this bit causes the memory controller to add another cycle for each address phase. Note that PSDMR[EAMUX] can also be set in any case of delays on the address lines, such as address buffers.

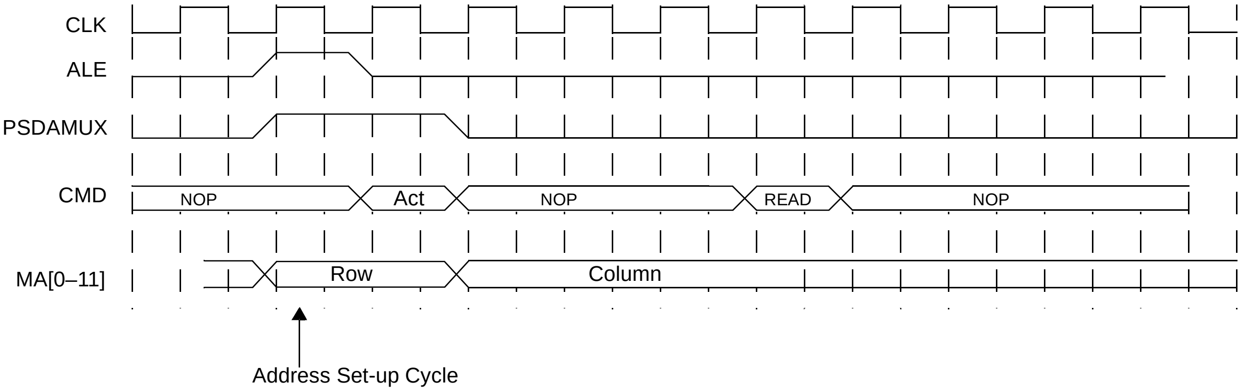


Figure 12-16. EAMUX = 1

- *External Address and Command Buffers (BUFCMD)*. In 60x-compatible mode, external buffers can be placed on the command strobes, except $\overline{\text{CS}}$, as well as on the address lines. If the additional delay of the buffers endangers the device set-up time, PSDMR[BUFCMD] should be set so the memory controller adds one cycle for each SDRAM command.

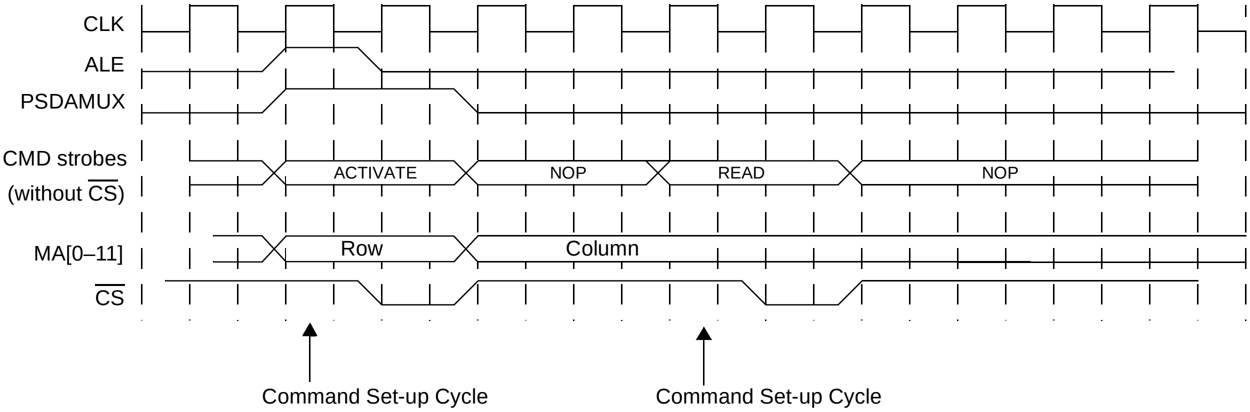


Figure 12-17. BUFCMD = 1

12.2.11 SDRAM Signals: General Interface Timing

The following figures show SDRAM timing for various types of accesses.

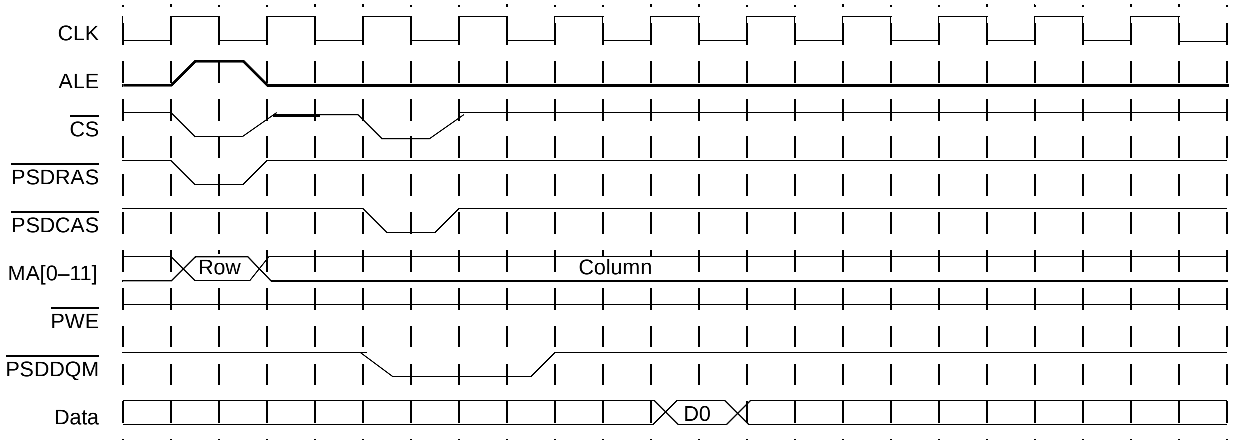


Figure 12-18. SDRAM Single-Beat Read, Page Closed, CL = 3

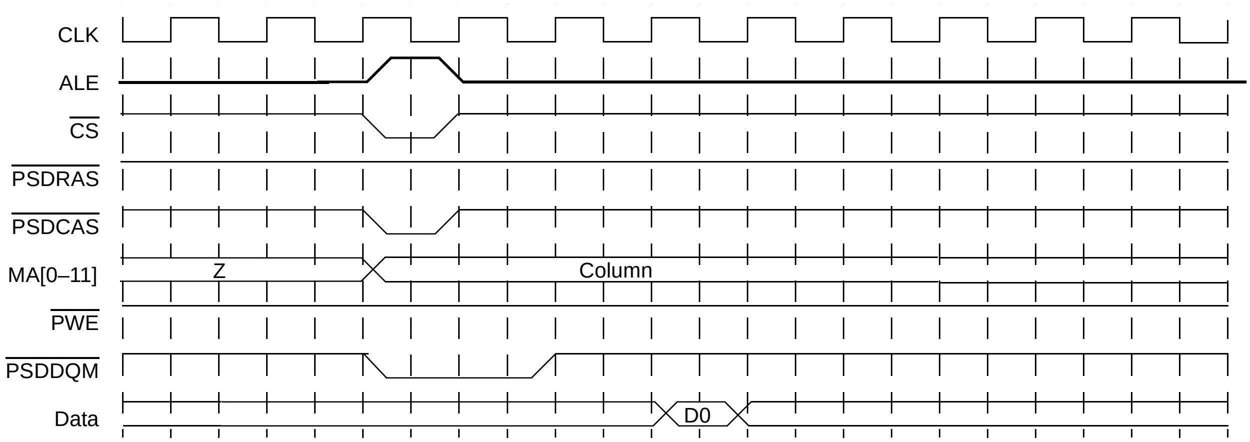


Figure 12-19. SDRAM Single-Beat Read, Page Hit, CL = 3

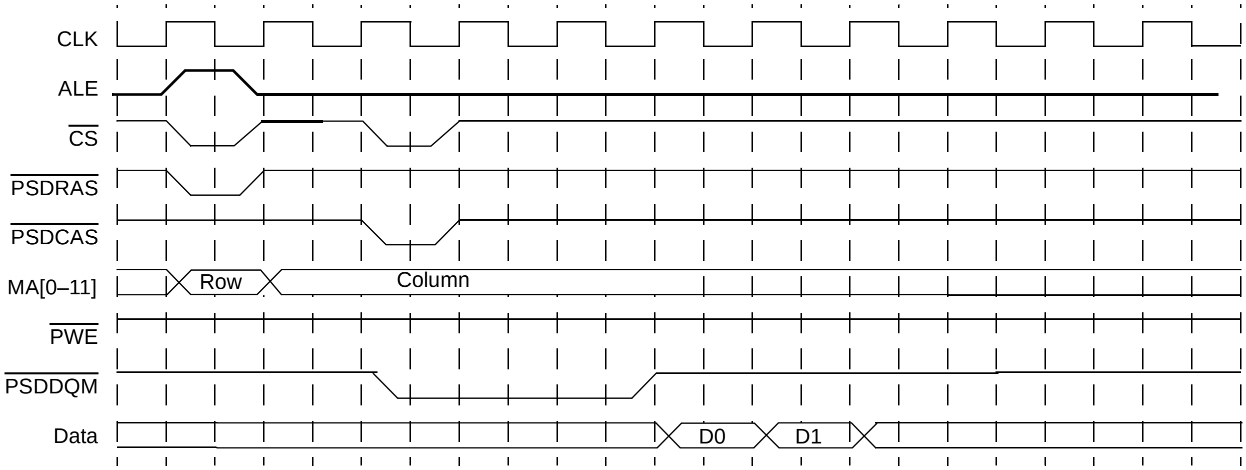
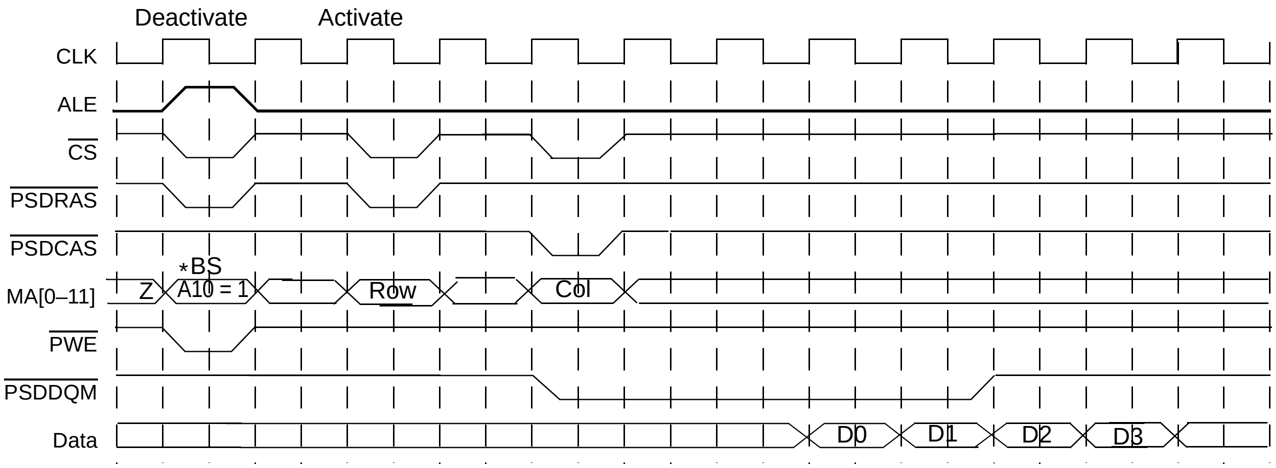


Figure 12-20. SDRAM Two-Beat Burst Read, Page Closed, CL = 3



* BS—Bank select according to SDRAM organization. A10 = 1 means all banks are precharged.
CAS Latency = 3

Figure 12-21. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

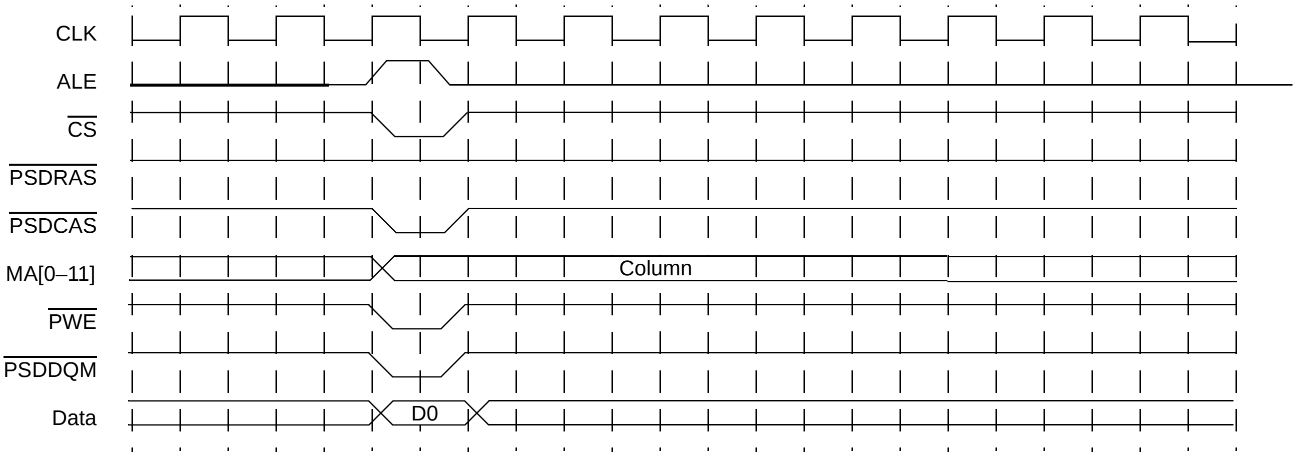


Figure 12-22. SDRAM Single-Beat Write, Page Hit

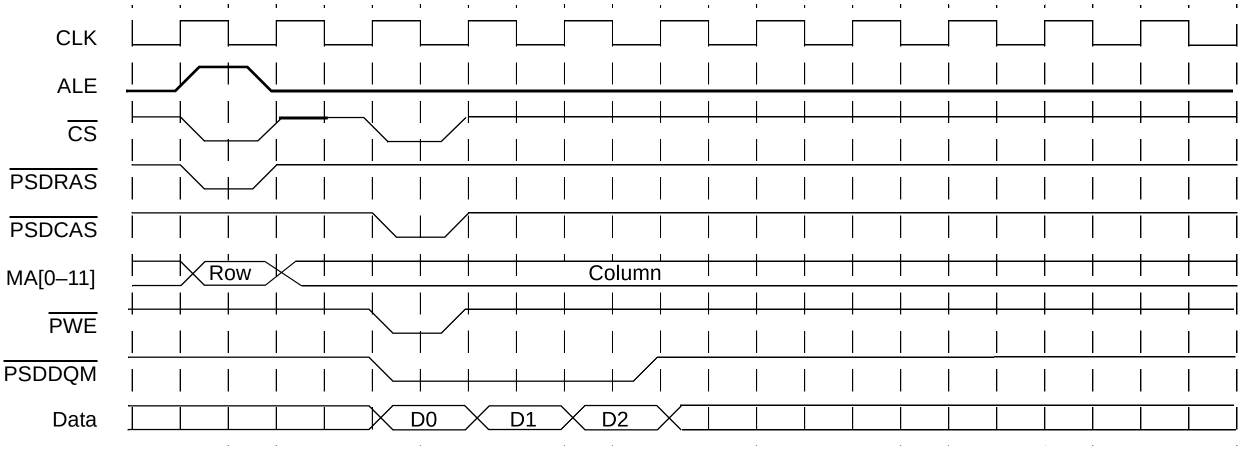


Figure 12-23. SDRAM Three-Beat Burst Write, Page Closed

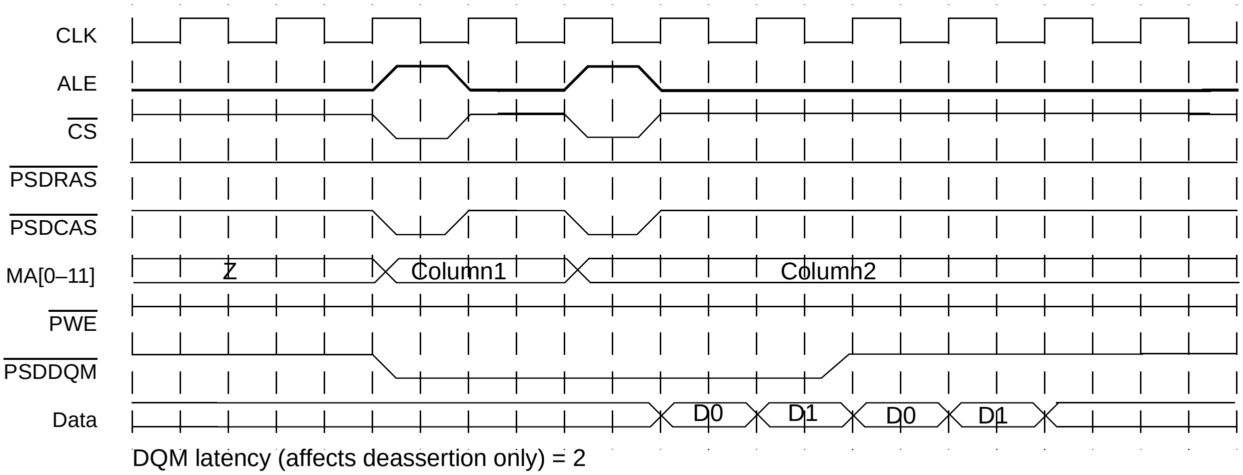


Figure 12-24. SDRAM Read-after-Read Pipeline, Page Hit, CL = 3

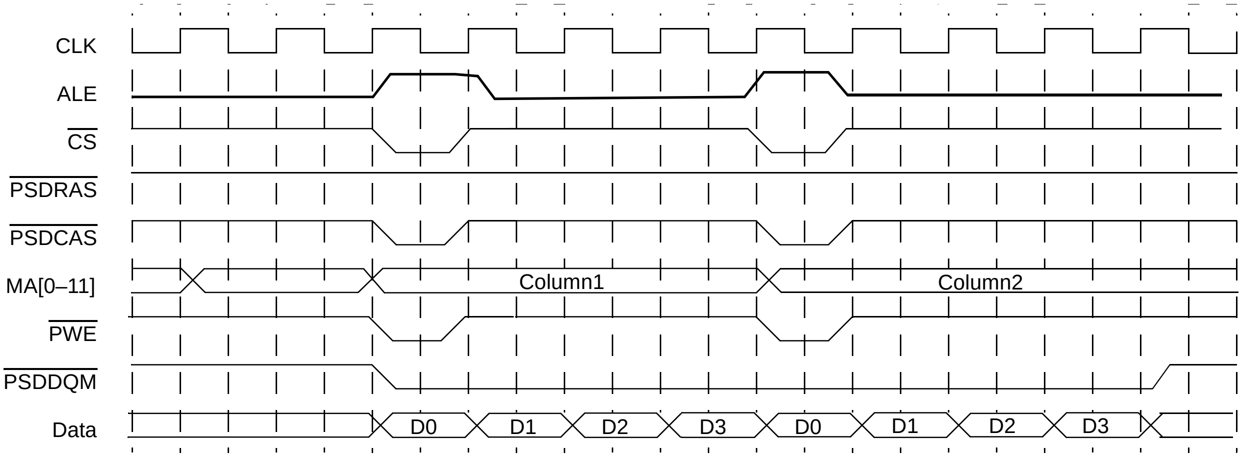


Figure 12-25. SDRAM Write-after-Write Pipelined, Page Hit

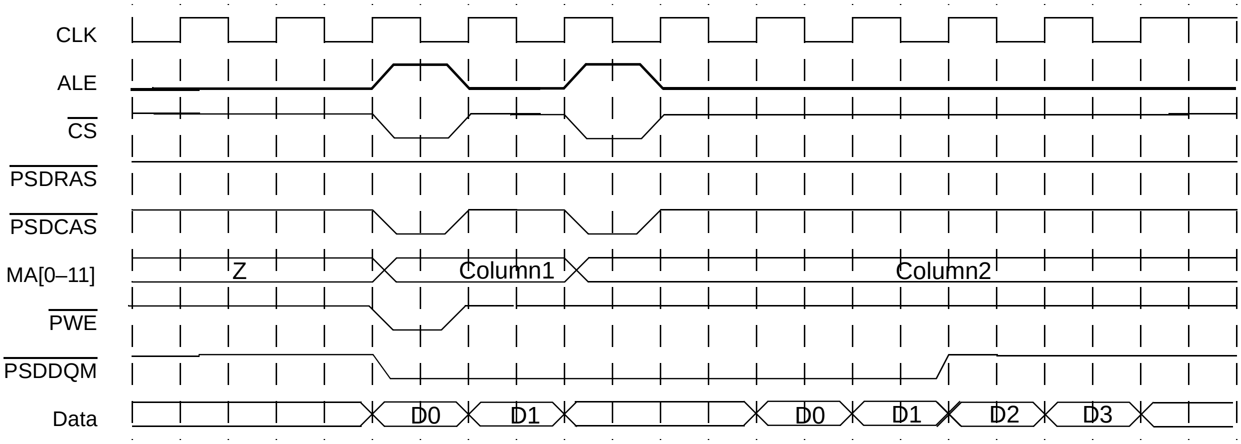
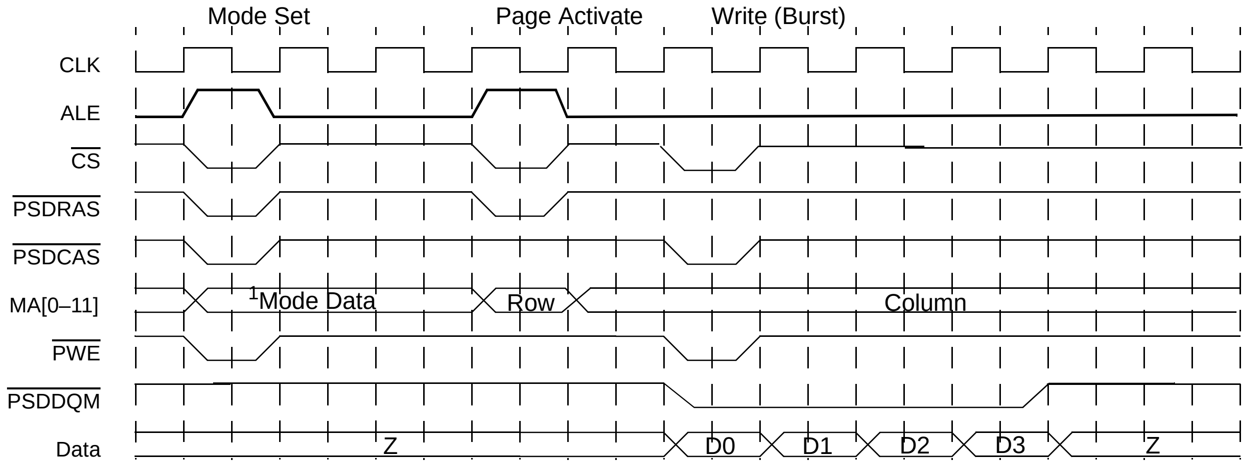


Figure 12-26. SDRAM Read-after-Write Pipelined, Page Hit

12.2.12 SDRAM Signals: MODE-SET Command Timing

The MSC8113 device transfers mode register data ($\overline{\text{CAS}}$ latency, burst length, burst type) stored in PSDMR to the SDRAM array using a MODE-SET command. **Figure 12-27** shows timings for the command.



Note: The mode data is the address value during a mode-set cycle. It is driven by the memory controller, in single MSC8113 mode, according to PSDMR[CL]. In 60x-compatible mode, software must drive the correct value on the address lines. **Figure 12-28** shows the actual value.

Figure 12-27. SDRAM MODE-SET Command Timing

Figure 12-28 shows mode data bit settings.

Bit number	0	1	2	3	4	5	6	7	8	9	10	11	lsb
	—					CL			0	BL			

Latency Mode—can be 1(001), 2(010), or 3(011).

Burst Length:
4(010) for 16- and 64-bit port sizes
8(011) for 8- and 32-bit port sizes

Figure 12-28. Mode Data Bit Settings

12.2.13 SDRAM Signals: Refresh Timing

The memory controller implements bank staggering for the auto refresh function, which reduces instantaneous current consumption for memory refresh operations. Once a refresh request is granted, the memory controller issues auto-refresh commands to each device associated with the refresh timer, in one clock intervals. After the last REFRESH command is issued, the memory controller waits for the number of clocks written in the SDRAM machine mode register (PSDMR[RFRC]). The timing is shown in **Figure 12-29**.

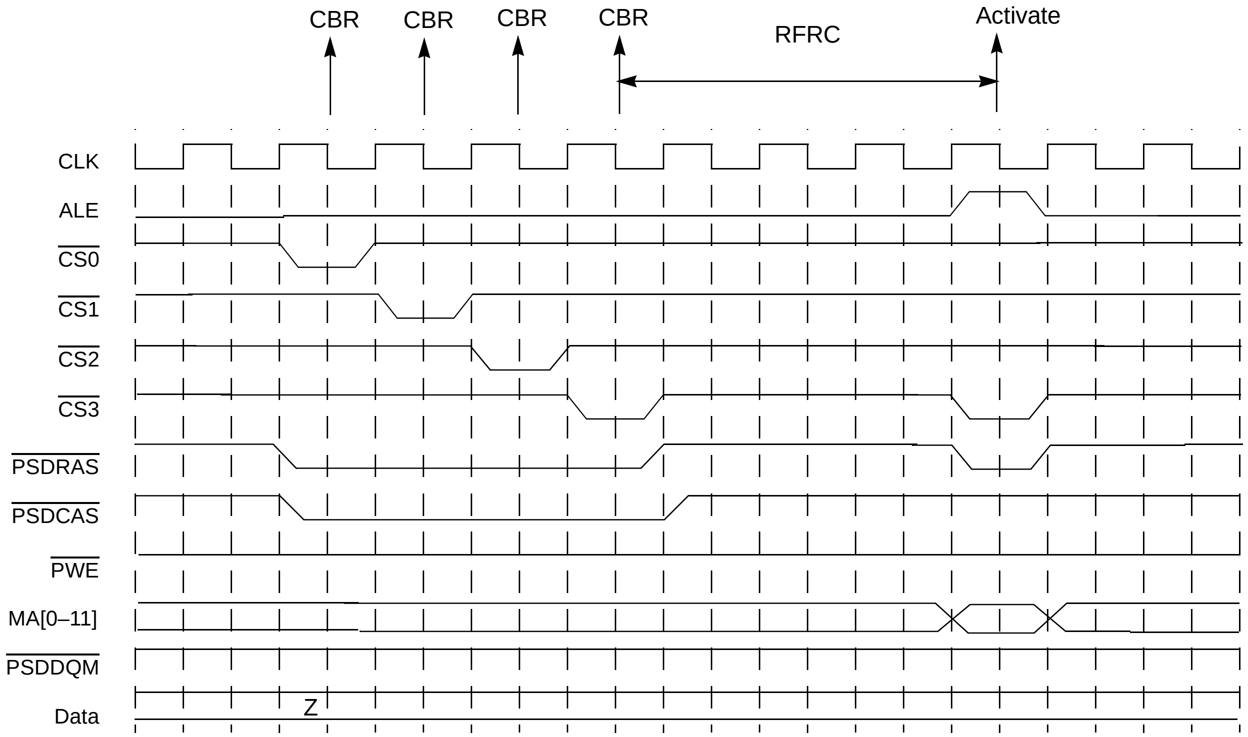


Figure 12-29. SDRAM Bank-Staggered CBR Refresh Timing

12.2.14 SDRAM Configuration Examples

The following sections provide SDRAM configuration examples for page- and bank-based interleaving.

12.2.14.1 SDRAM Configuration Example (Page-Based Interleaving)

Consider the following SDRAM organization:

- 32-bit port size organized as four 128-Mb devices, each organized as 16 M × 8 bits.
- Each device has four internal banks, 12 row address lines, and 10 column address lines

For page-based interleaving, the address bus is partitioned as shown in **Table 12-7**.

Table 12-7. 60x Address Bus Partition

A[0–5]	A[6–17]	A[18–19]	A[20–29]	A[30–31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- PSDMR[PBI] = 1, page-based interleaving
- OR*x*[BPD] = 01, four internal banks
- OR*x*[ROWST] = 0110, row address starts at A[6]
- OR*x*[NUMR] = 011, twelve row address lines

For the SDRAM device, during an ACTIVATE command, its address is as shown in **Table 12-8**.

Table 12-8. SDRAM Device Address Port During ACTIVATE Command

A[0–15]	A[16–17]	A[18–29]	A[30–31]
—	Internal bank select (A[18–19])	Row (A[6–17])	n.c.

Table 12-5 indicates that to multiplex A[6–17] over A[18–29], PSDMR[SDAM] must be 100 and, because the internal bank selects are multiplexed over A[16–17], PSDMR[BSMA] must be 011 (only the lower two bank select lines are used).

When using page-based interleaving, the internal bank-select signals that are multiplexed over the address lines are determined only by PSDMR[BSMA] during the ACTIVATE command. The output of the BNKSEL pins are not affected by the PSDMR[BSMA] value.

Note: In the preceding example, address lines A[18–19] are output on BNKSEL[1] and BNKSEL[2], accordingly.

During a READ/WRITE command, the address port is as shown in **Table 12-9**.

Table 12-9. SDRAM Device Address Port During READ/WRITE Command

A[0–15]	A[16–17]	A[18]	A[19]	A[20–29]	A[30–31]
—	Internal bank select	Don’t care	AP	Column	n.c.

Because AP alternates with A[7] of the row lines, set PSDMR[SDA10] = 011. This value drives A7 on the PSDA10 line during execution of the ACTIVATE command and AP during execution of the READ/WRITE and CBR commands. **Table 12-10** shows the register configuration. Not shown are PSRT and MPTPR, which should be programmed according to the device refresh requirements.

Table 12-10. Register Settings (Page-Based Interleaving)

Register	Settings			
BR*x*	BA	Base address	EMEMC	0
	PS	11 = 32-bit port size	ATOM	00
	DECC	00	DR	0
	WP	0	V	1
	MS	010 = SDRAM system bus		

Table 12-10. Register Settings (Page-Based Interleaving) (Continued)

Register	Settings			
ORx	SDAM LSDAM BPD ROWST	111111000000 00000 01 0110	NUMR PMSEL IBID	011 0 0
PSDMR	PBI RFEN OP SDAM BSMA SDA10 RFRC PRETOACT	1 1 000 100 011 011 from device data sheet from device data sheet	ACTTOROW BL LDOTOPRE WRC EAMUX BUFCMD CL	from device data sheet 1 from device data sheet from device data sheet 0 0 from device data sheet

12.2.14.2 SDRAM Configuration Example (Bank-Based Interleaving)

Consider the following SDRAM organization:

- 64-bit port size organized as eight 64 Mb devices, each organized as 8 M × 8 bits.
- Each device has four internal banks, 12 row address lines, and 9 column address lines

For bank-based interleaving, the address bus is partitioned as shown in **Table 12-11**.

Table 12-11. 60x Address Bus Partition

A[0–5]	A[6–7]	A[8–19]	A[20–28]	A[29–31]
msb of start address	Internal bank select	Row	Column	lsb

The following parameters can be extracted:

- PSDMR[PBI] = 0, bank-based interleaving
- OR*x*[BPD] = 01, four internal banks
- OR*x*[ROWST] = 0100, row address starts at A[8]
- OR*x*[NUMR] = 011, twelve row address lines

Table 12-12 shows the SDRAM address port during execution of an ACTIVATE command.

Table 12-12. SDRAM Device Address Port During ACTIVATE Command

A[0–14]	A[15–16]	A[17–28]	A[29–31]
—	Internal bank select (A[6–7])	Row (A[8–19])	n.c.

Table 12-5 indicates that to multiplex A[8–19] over A[17–28], the PSDMR[SDAM] field must contain a value of 001 and, because the internal bank selects are multiplexed over A[15–16], PSDMR[BSMA] must contain a value of 010 (only the lower two bank select lines are used).

When using bank-based interleaving, the internal bank-select signals that are multiplexed over the address lines should be adjacent to the row address during the ACTIVATE command (refer to **Table 12-12**). So, the value of PSDMR[BSMA] is selected according to the combination of PSDMR[SDAM], OR*x*[ROWST], and OR*x*[NUMR]. Otherwise, the output of the BNKSEL pins could be incorrect even if the device is connected to the BNKSEL pins. To ensure proper connection, note that BNKSEL[0] is msb and BNKSEL[2] is lsb.

Note: In the preceding example, address lines A[6–7] are driven on BNKSEL[1] and BNKSEL[2], accordingly.

When a READ/WRITE command executes, the address port appears as shown in **Table 12-13**.

Table 12-13. SDRAM Device Address Port During READ/WRITE Command

A[0–14]	A[15–16]	A[17]	A[18]	A[19]	A[20–28]	A[29–31]
—	Internal bank select	Don’t care	AP	Don’t care	Column	NC

Because AP alternates with A9 of the row lines, set PSDMR[SDA10] = 011. This setting drives A9 on the PSDA10 line when the ACTIVATE command executes and AP when the READ/WRITE and CBR commands execute. **Table 12-14** shows the register configuration. Not shown are PSRT and MPTPR, which should be programmed according to the device refresh requirements.

Table 12-14. Register Settings (Bank-Based Interleaving)

Register	Settings			
BR*x*	BA	Base address	EMEMC	0
	PS	00 = 64-bit port size	ATOM	00
	DECC	00	DR	0
	WP	0	V	1
	MS	010 = SDRAM system bus		
OR*x*	SDAM	111111000000	NUMR	011
	LSDAM	00000	PMSEL	0
	BPD	01	IBID	0
	ROWST	0100		
PSDMR	PBI	0	ACTTOROW	from device data sheet
	RFEN	1	BL	0
	OP	000	LDOTOPRE	from device data sheet
	SDAM	001	WRC	from device data sheet
	BSMA	010	EAMUX	0
	SDA10	011	BUFCMD	0
	RFRC	from device data sheet	CL	from device data sheet
	PRETOACT	from device data sheet		

12.3 General-Purpose Chip-Select Machine (GPCM)

The MSC8113 GPCM allows a glueless and flexible interface between the MSC8113, SRAM, EPROM, FEPROM, and ROM devices, and external peripherals. The GPCM contains two basic configuration register groups: BR*x* and OR*x*. **Table 12-15** lists the GPCM interface signals on the system bus.

Table 12-15. GPCM Interfaces Signals

60x-compatible System Bus	Comments
$\overline{\mathrm{CS}[0–7]}$	Device select[1]
$\overline{\mathrm{PWE}[0–7]}$	Write enables for write cycles
$\overline{\mathrm{POE}}$	Output enable for read cycles
Note:	There is an additional device select ($\overline{\mathrm{CS9}}$) on the local bus. It is internal only, and the local bus GPCM uses it to access IPBus peripherals.

Note: If you are familiar with the MPC8xx GPCM, you should first read **Section 12.3.4**, *Differences Between MPC8xx GPCM and MSC8113 GPCM*, on page 12-44.

GPCM-controlled devices can use $\overline{\mathrm{BCTL}}$*x* as read/write indicators. The $\overline{\mathrm{BCTL}}$*x* signals appear as R/$\overline{\mathrm{W}}$ in the timing diagrams. See **Section 12.1.6**, *Data Buffer Controls (BCTL[0–1])*, on page 12-9. Additional control is available in 60x-compatible mode (system bus only) via the external address latch enable (ALE) signal. **Figure 12-30** shows a simple connection between an SRAM device with a 32-bit port and the MSC8113. In the example, the SRAM connects to the system bus.

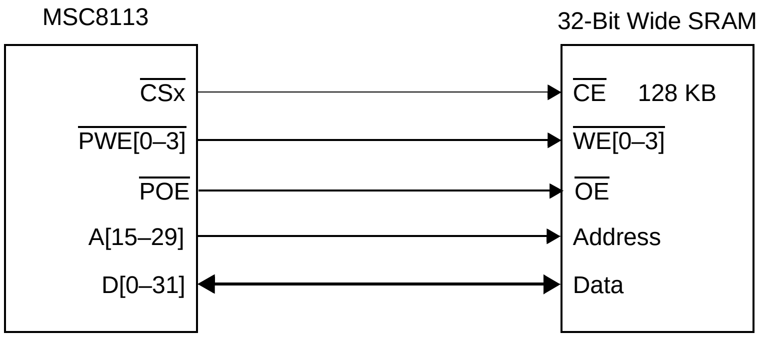


Figure 12-30. GPCM-to-SRAM Configuration

12.3.1 GPCM Signals: Timing Configuration

If BR*x*[MS] selects the GPCM, the attributes for the memory cycle are taken from OR*x*. These attributes include the OR*x*[CSNT], OR*x*[ACS], OR*x*[SCY], OR*x*[SETA], OR*x*[TRLX], and OR*x*[EHTR] fields. **Table 12-16** shows signal behavior and system response.

Table 12-16. GPCM Strobe Signal Behavior

Option Register Attributes				Signal Behavior			
TRLX	**Access**	**ACS**	**CSNT**	**Address to $\overline{\text{CS}}$ Asserted**	**$\overline{\text{CS}}$ Deasserted to Address Change**	**$\overline{\text{PWE}}$ Deasserted to Address/Data Invalid**	**Total Cycles**
0	Read	00	x	0	0	x	2 + SCY
0	Read	10	x	1/4 × Clock	0	x	2 + SCY
0	Read	11	x	1/2 × Clock	0	x	2 + SCY
0	Write	00	0	0	0	0	2 + SCY
0	Write	10	0	1/4 × Clock	0	0	2 + SCY
0	Write	11	0	1/2 × Clock	0	0	2 + SCY
0	Write	00	1	0	0	–1/4 × Clock	2 + SCY
0	Write	10	1	1/4 × Clock	–1/4 × Clock	–1/4 × Clock	2 + SCY
0	Write	11	1	1/2 × Clock	–1/4 × Clock	–1/4 × Clock	2 + SCY
1	Read	00	x	0	0	x	2 + (2 × SCY)
1	Read	10	x	(1 + 1/4) × Clock	0	x	3 + (2 × SCY)
1	Read	11	x	(1 + 1/2) × Clock	0	x	3 + (2 × SCY)
1	Write	00	0	0	0	0	2 + (2 × SCY)
1	Write	10	0	(1 + 1/4) × Clock	0	0	3 + (2 × SCY)
1	Write	11	0	(1 + 1/2) × Clock	0	0	3 + (2 × SCY)
1	Write	00	1	0	0	–(1 + 1/4) × Clock	3 + (2 × SCY)
1	Write	10	1	(1 + 1/4) × Clock	–(1 + 1/4) × Clock	–(1 + 1/4) × Clock	4 + (2 × SCY)
1	Write	11	1	(1 + 1/2) × Clock	–(1 + 1/4) × Clock	–(1 + 1/4) × Clock	4 + (2 × SCY)

SCY is the number of wait cycles from the option register.

12.3.1.1 Chip-Select Assertion Timing

From 0 to 30 wait states can be programmed for $\overline{\text{PSDVAL}}$ generation. Byte-write enable signals ($\overline{\text{PWE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{POE}}$) eliminates external glue logic. The memory banks selected to work with the GPCM have unique features. On system reset, a global (boot) chip-select provides a boot ROM chip-select before the system is fully configured. The banks selected to work with the GPCM support an option to output the $\overline{\text{CS}}$ line at different timings with respect to the external address bus. $\overline{\text{CS}}$ can be output in any of three configurations:

- Simultaneous with the external address
- One quarter of a clock cycle later
- One half of a clock cycle later

The GPCM does not deassert $\overline{\text{CS}}$ in back-to-back reads to the same device in single MSC8113 bus mode or in 60x-compatible bus mode with extended transfers enabled. In strict 60x bus mode, however, the GPCM does deassert $\overline{\text{CS}}$ in back-to-back reads. See the discussion of the Bus Configuration Register (BCR) in **Section 4.2**, *SIU Programming Model*.

Figure 12-31 shows a basic connection between the MSC8113 and an external peripheral device. Here, $\overline{\text{CS}}$ (the strobe output for the memory access) directly connects to the $\overline{\text{CE}}$ of the memory device, and $\overline{\text{BCTL0}}$ connects to the respective R/$\overline{\text{W}}$ in the peripheral device.

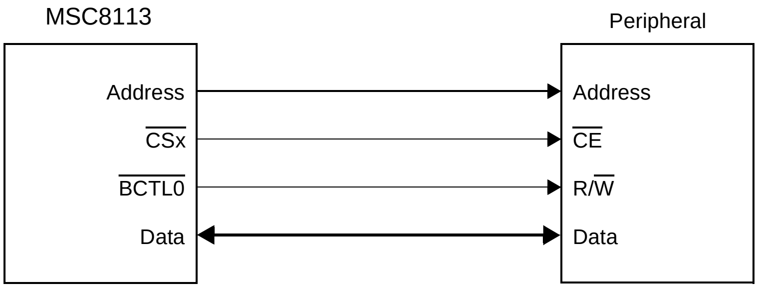


Figure 12-31. GPCM Peripheral Device Interface

Figure 12-32 shows $\overline{\text{CS}}$ as defined by the set-up time required between the address lines and $\overline{\text{CE}}$. You can configure OR*x*[ACS] to specify $\overline{\text{CS}}$ to meet this requirement.

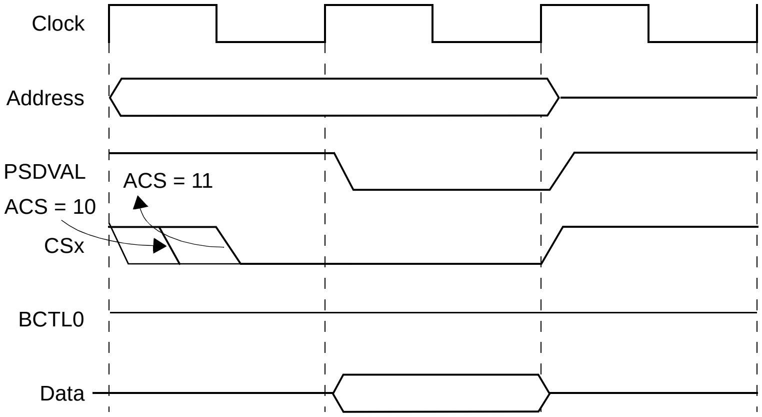


Figure 12-32. GPCM Peripheral Device Basic Timing (ACS = 1x and TRLX = 0)

12.3.1.2 Chip-Select and Write Enable Deassertion Timing

Figure 12-33 shows a basic connection between the MSC8113 device and a static memory device. Here, $\overline{\text{CS}}$ directly connects to the $\overline{\text{CE}}$ of the memory device. The $\overline{\text{PWE}}$ signals connect to the respective $\overline{\text{W}}$ signal in the memory device, where each $\overline{\text{PWE}}$ corresponds to a different data byte.

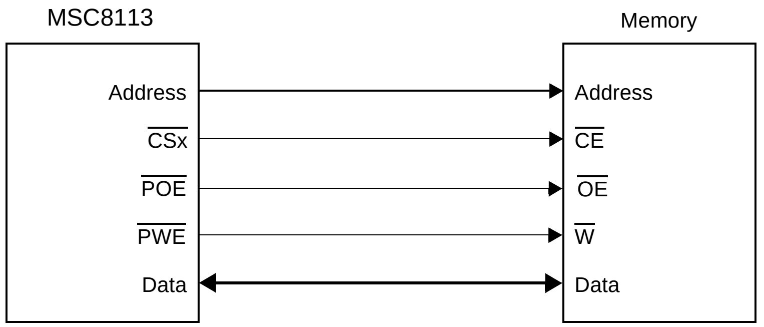


Figure 12-33. GPCM Memory Device Interface

As **Figure 12-35** shows, the timing for $\overline{\text{CS}}$ is the same as for the address lines. The strobes for the transaction are supplied by $\overline{\text{POE}}$ or $\overline{\text{PWE}}$, depending on the transaction direction (read or write). OR*x*[CSNT] controls the timing for the appropriate strobe deassertion in write cycles. When this attribute is asserted, the strobe is deasserted one quarter of a clock before the normal case.

For example, when OR*x*[ACS] = 00 and OR*x*[CSNT] = 1, $\overline{\text{PWE}}$ is deasserted one quarter of a clock earlier, as shown in **Figure 12-34**.

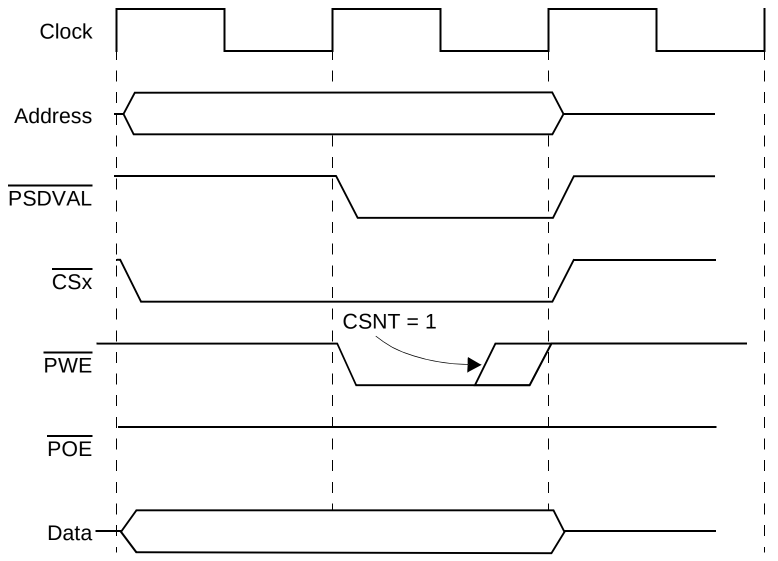


Figure 12-34. GPCM Memory Device Timing (ACS = 00, CSNT = 1, TRLX = 0)

When ORx[ACS] ≠ 00 and ORx[CSNT] = 1, $\overline{\text{PWE}}$ and $\overline{\text{CS}}$ are deasserted one quarter of a clock earlier, as shown in **Figure 12-35**.

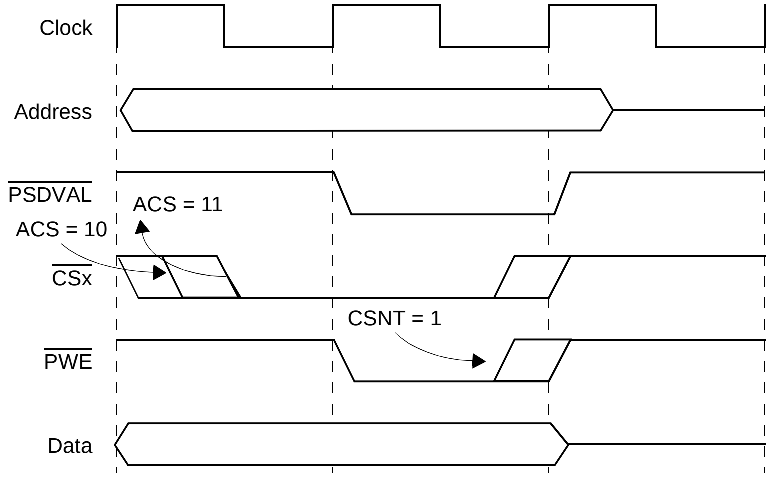


Figure 12-35. GPCM Memory Device Timing (ACS ≠ 00, CSNT = 1, TRLX = 0)

12.3.1.3 Relaxed Timing

ORx[TRLX] is provided for memory systems that require more relaxed timing between signals. When ORx[TRLX] = 1 and ORx[ACS] ≠ 00, an additional cycle between the address and strobes is inserted by the MSC8113 memory controller. See **Figure 12-36** and **Figure 12-37**.

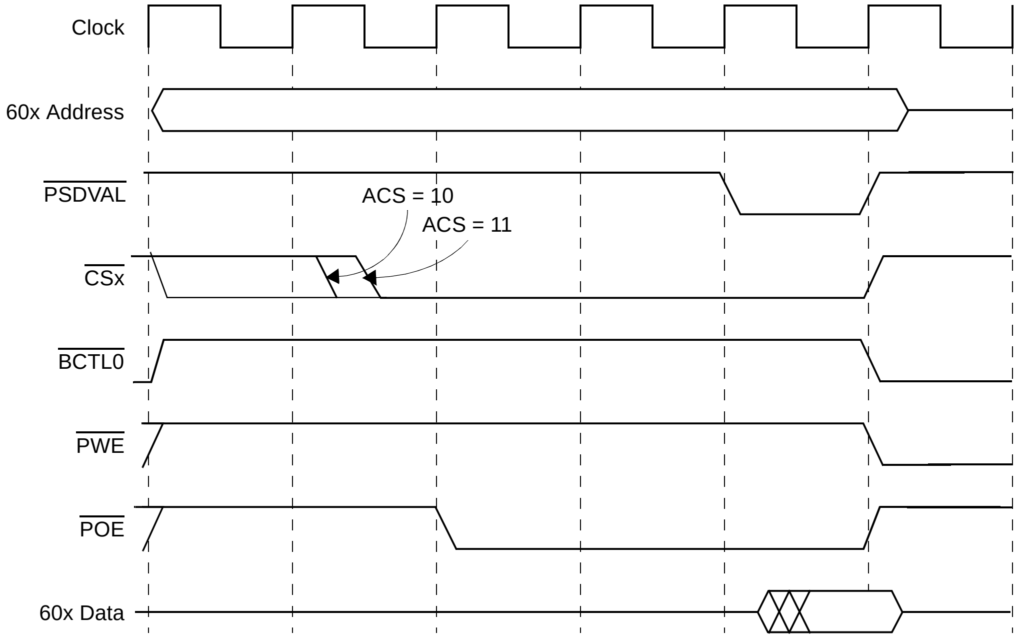


Figure 12-36. GPCM Relaxed Timing Read (ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1)

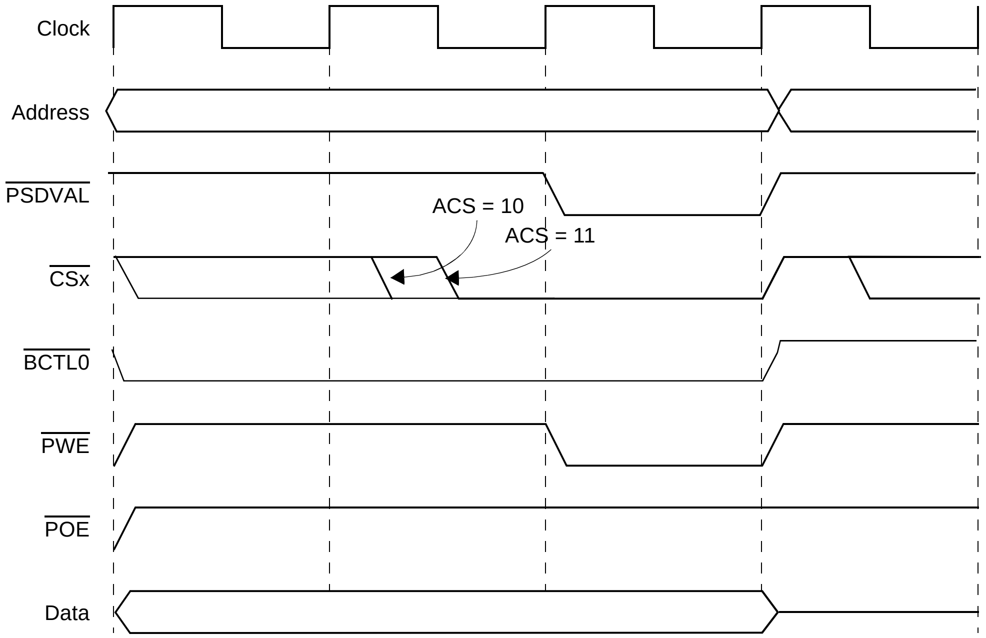


Figure 12-37. GPCM Relaxed-Timing Write (ACS = 1x, SCY = 0, CSNT = 0,TRLX = 1)

When OR*x*[TRLX] and OR*x*[CSNT] are set in a write-memory access, the strobe lines, $\overline{\text{PWE}}$[0–7] are deasserted one clock earlier than in the normal case. If OR*x*[ACS] ≠ 00, $\overline{\text{CS}}$ is also deasserted one clock earlier, as shown in **Figure 12-38** and **Figure 12-39**. When a bank is selected to operate with external transfer acknowledge (OR*x*[SETA] = 1 and OR*x*[TRLX] = 1), the memory controller does not support external devices that provide $\overline{\text{PSDVAL}}$ to complete the transfer with zero wait states. The minimum access duration in this case is three clock cycles.

12.3.1.4 Output Enable ($\overline{\text{POE}}$) Timing

The timing of the $\overline{\text{POE}}$ is affected only by TRLX. It always asserts and deasserts on the rising edge of the external bus clock. $\overline{\text{POE}}$ always asserts on the rising clock edge after $\overline{\text{CS}}$ is asserted, and therefore its assertion can be delayed (along with the assertion of $\overline{\text{CS}}$) by programming OR*x*[TRLX] = 1. $\overline{\text{POE}}$ deasserts on the rising clock edge coinciding with or immediately after $\overline{\text{CS}}$ deassertion.

12.3.1.5 Programmable Wait State Configuration

The GPCM supports internal $\overline{\text{PSDVAL}}$ generation. It allows fast accesses to external memory through an internal 60x-compatible bus master or a maximum 17-clock access by programming OR*x*[SCY]. The internal $\overline{\text{PSDVAL}}$ generation mode is enabled if OR*x*[SETA] = 0. If $\overline{\text{PGTA}}$ is asserted externally at least two clock cycles before the wait state counter expires, the current memory cycle is terminated. When OR*x*[TRLX] = 1, the number of wait states inserted by the memory controller is defined by 2 × SCY or a maximum of 30 wait states.

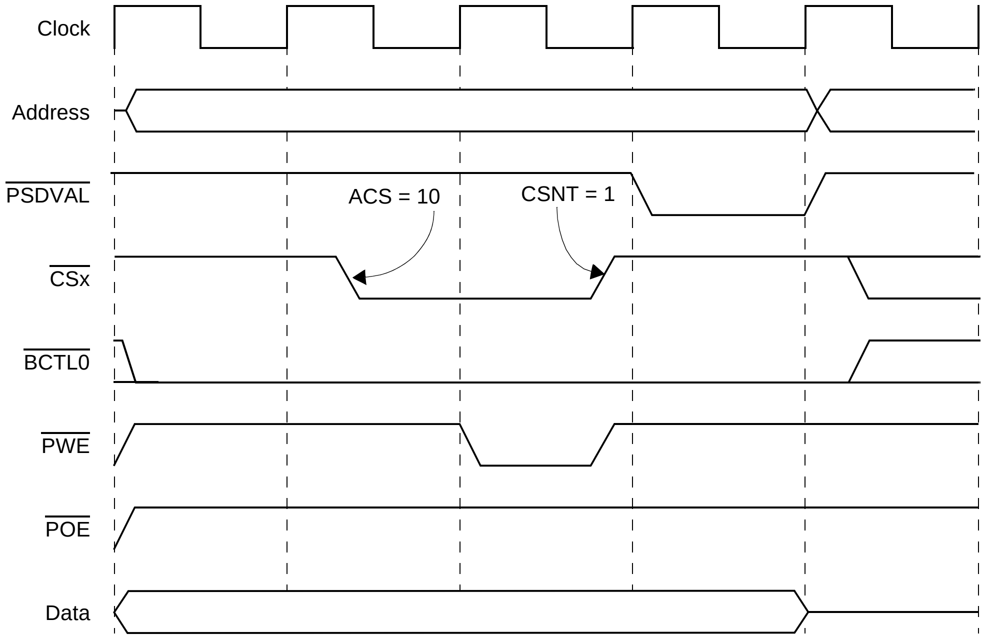


Figure 12-38. GPCM Relaxed-Timing Write (ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)

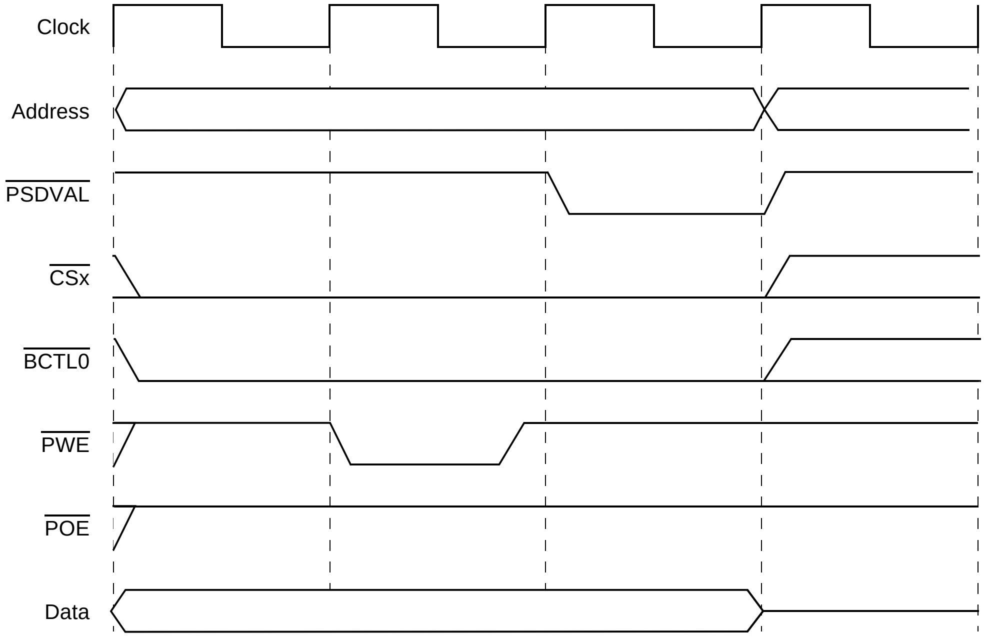


Figure 12-39. GPCM Relaxed-Timing Write (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1)

12.3.1.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should chose some combination of OR*x*[TRLX] and OR*x*[EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in **Table 12-17**. See **Figure 12-40** through **Figure 12-43** for timing examples.

Table 12-17. TRLX and EHTR Combinations

ORx[TRLX]	ORx[EHTR]	Number of Hold Time Clock Cycles
0	0	0
0	1	1
1	0	4
1	1	8

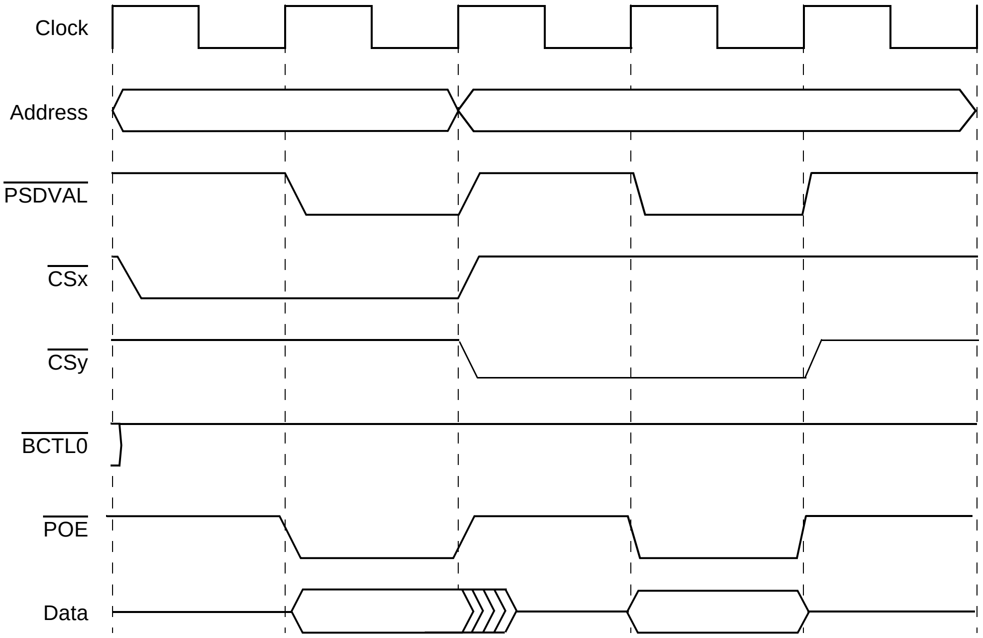


Figure 12-40. GPCM Read Followed by Read (OR*x*[29–30] = 00, Fastest Timing)

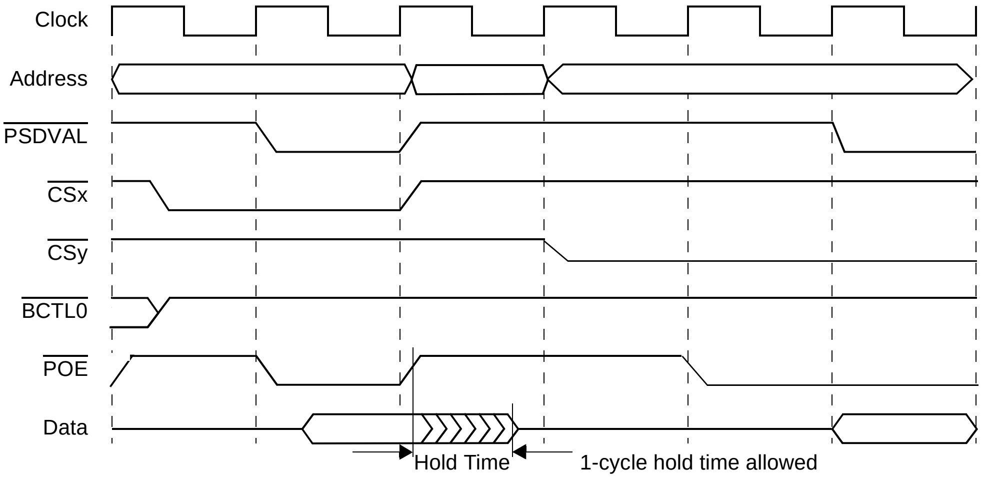


Figure 12-41. GPCM Read Followed by Read (OR*x*[29–30] = 01)

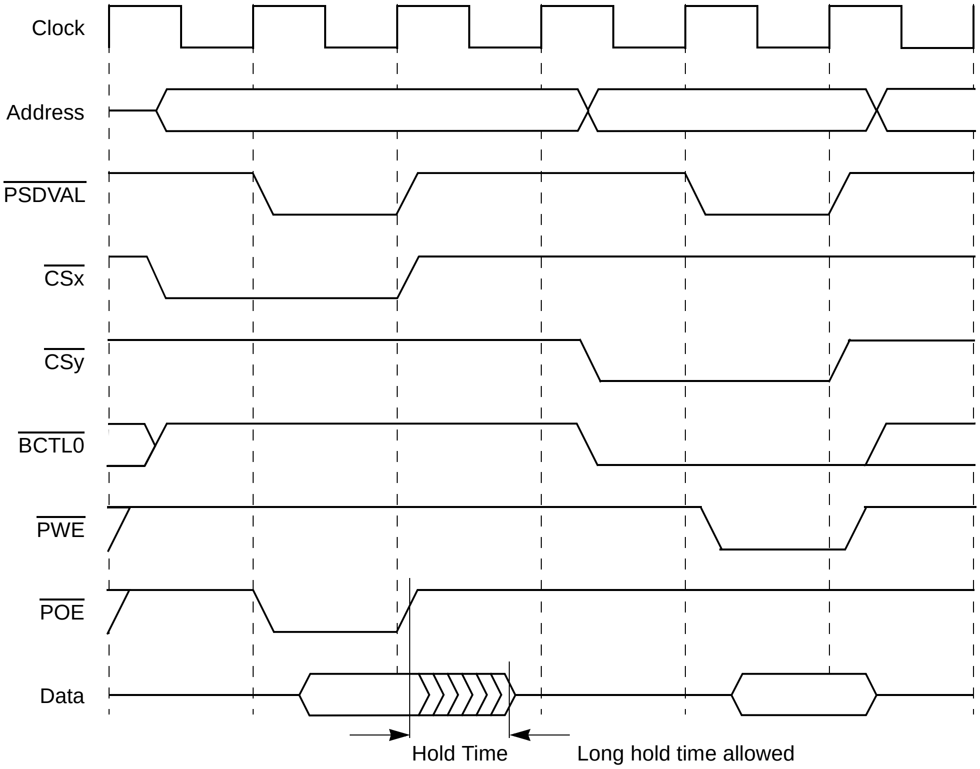


Figure 12-42. GPCM Read Followed by Write (OR*x*[29–30] = 01)

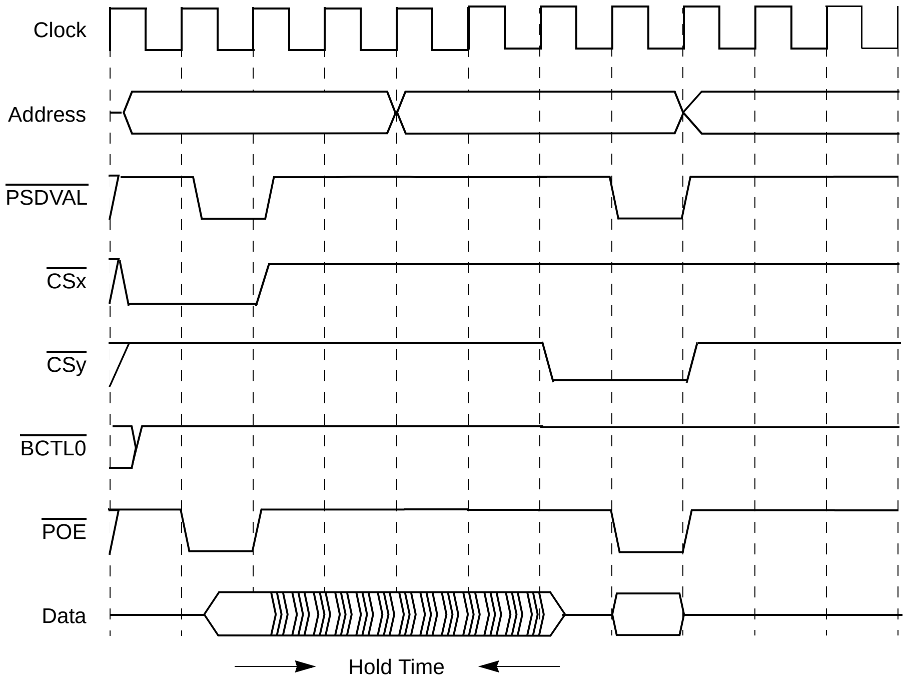


Figure 12-43. GPCM Read Followed by Read (OR*x*[29–30] = 10)

12.3.2 GPCM Signals: External Access Termination

The GPCM supports external access termination using PGTA, which the MSC8113 synchronizes and samples internally. If the sampled signal is asserted during a GPCM data phase (second cycle or later), it is converted to PSDVAL, which terminates the current GPCM access. PGTA should be asserted for one cycle. Because PGTA is internally synchronized, bus termination may occur up to three cycles after PGTA assertion, so for a read cycle, the device must still output data as long as POE is asserted. You select whether PSDVAL is generated internally or externally (by means of PGTA assertion) by resetting/setting ORx[SETA]. **Figure 12-44** shows how a GPCM access is terminated by PGTA assertion. Asserting PGTA terminates an access even if ORx[SETA] = 0 (internal PSDVAL generation).

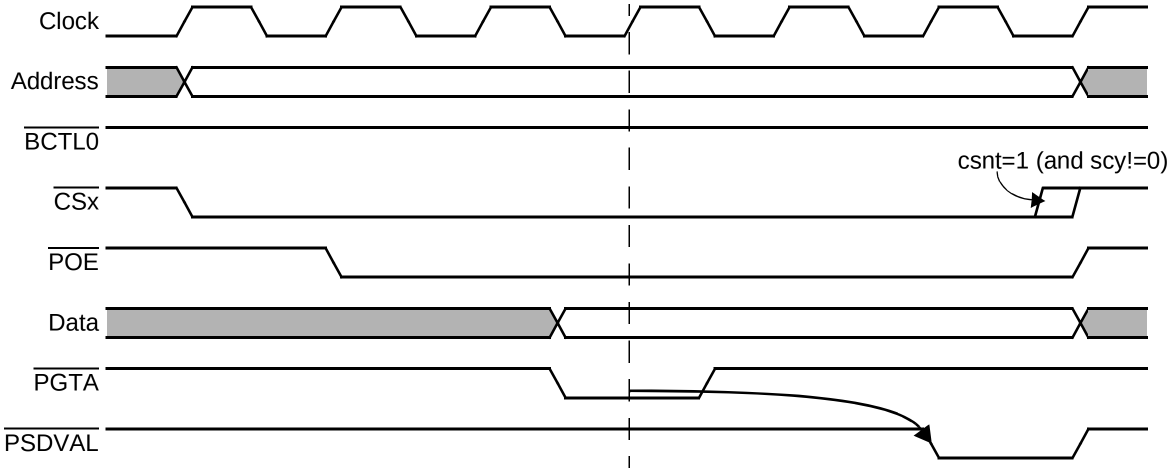


Figure 12-44. External Termination of GPCM Access

Figure 12-45 shows how a GPCM access is terminated internally after the cycle length defined in the ORx[SCY]. Assertion of PGTA before the defined cycle length will also terminate the access.

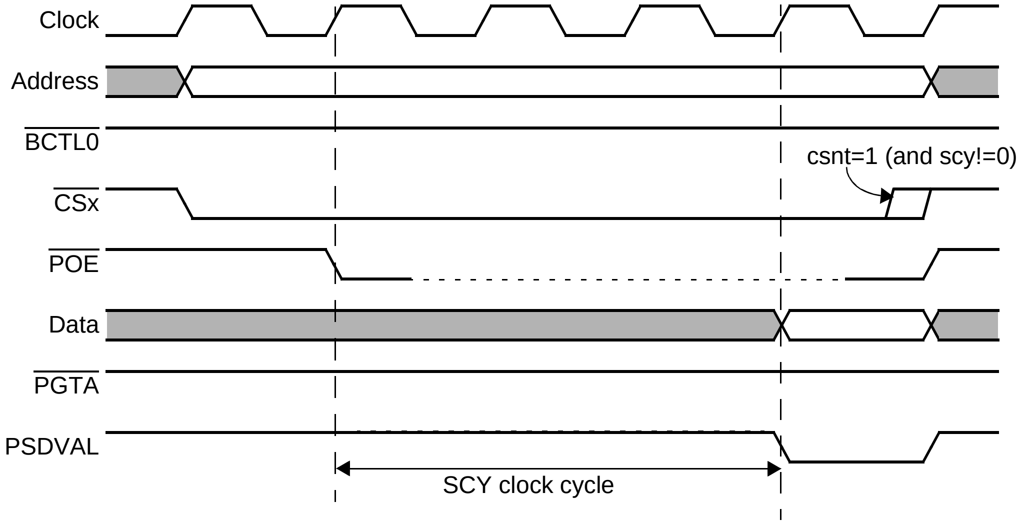


Figure 12-45. Internal Termination of GPCM Access

12.3.3 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. The $\overline{\mathrm{CS0}}$ signal is the boot chip-select output; its operation differs from the other external chip-select outputs on system reset. When the MSC8113 internal core begins accessing memory at system reset, $\overline{\mathrm{CS0}}$ is asserted for every address in the boot address range, unless an internal register is accessed. The address range is configured during reset. The boot chip-select also provides a programmable port size during system reset by using the configuration mechanism described in **Section 5.2**, *Reset Configuration*. The boot chip-select does not provide write protection. $\overline{\mathrm{CS0}}$ operates this way until the first write to OR0, and it can be used as any other chip-select register once the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only on hardware reset. **Table 12-18** describes the initial values of the boot bank in the memory controller.

Table 12-18. Boot Bank Field Values After Reset

Register	Settings	
BR0	BA[0–16]	11111110000000000
	PS [0–1]	From HRCW. See **Section 5.6.1**, *Hard Reset Configuration Word*.
	DECC[0–1]	00
	WP	0
	MS[0–2]	000
	EMEMC	From HRCW.
	ATOM[0–1]	00
	DR	0
	V	1
OR0	AM[0–16]	11111110000000000 (32 MB)
	BCTLD	0
	CSNT	1
	ACS[0–1]	11
	SCY[0–3]	1111
	SETA	0
	TRLX	1
	EHTR	0

12.3.4 Differences Between MPC8xx GPCM and MSC8113 GPCM

If you are familiar with the MPC8xx GPCM, you should know about the following differences between the MPC8xx GPCM and the MSC8113 GPCM:

- *External termination*. In the MPC8xx the external termination connects to the external system bus $\overline{\mathrm{TA}}$ signal and so must be asserted in sync with the system clock. In the MSC8113, this signal is separated from the bus and named $\overline{\mathrm{PGTA}}$. The signal is synchronized internally and sampled. The sampled signal is used to generate $\overline{\mathrm{TA}}$, which terminates the bus transaction.
- *Extended hold time*. Extended hold time for reads can be up to eight clock cycles (instead of one in the MPC8xx).

12.4 User-Programmable Machines (UPMs)

If you are familiar with the MPC8xx UPM, you should first read **Section 12.4.7**, *Differences Between MPC8xx UPM and MSC8113 UPM*, on page 12-82. **Table 12-19** lists the UPM interface signals on the system bus.

Table 12-19. UPM Interface Signals

60x-compatible System Bus	Comments
$\overline{\text{CS[0–7]}}$	Device select[1]
$\overline{\text{PBS[0–7]}}$	Byte-select
PGPL0	General-purpose line 0
PGPL1	General-purpose line 1
PGPL2	General-purpose line 2
PGPL3	General-purpose line 3
PGPL4/PUPMWAIT	General-purpose line 4/UPM WAIT
PGPL5	General-purpose line 5

Note: There is an additional device select ($\overline{\text{CS11}}$) on the local bus. It is internal only, and the UPMC uses it to access internal memories.

Additional control is available in 60x-compatible mode (system bus only) via the external address latch enable (ALE) signal. However, ALE is not a UPM-controlled signal; it toggles with chip-select signals.

Note: In this section, when a signal is named, the reference is to the system bus (prefix P is added, for example, $\overline{\text{BS}}$ -> $\overline{\text{PBS}}$) or local bus (prefix L is added, for example, $\overline{\text{BS}}$ -> $\overline{\text{LBS}}$) signal, according to the bank being accessed.

The three UPMs are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal-memory RAM array that specifies the logical value driven on the external memory controller signals for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock period on the byte-select and chip-select lines. **Figure 12-46** shows the basic operation of each UPM. The following events initiate a UPM cycle:

- Any internal or external device requests an external memory access, to an address space mapped to a chip-select serviced by the UPM.
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh.
- A transfer error or reset generates an exception request.

The RAM array contains sixty-four 32-bit RAM words. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN bit set, the memory controller synchronizes and samples the external PUPMWAIT signal, and the current request is frozen.

When any device on the system bus or local bus requests a new access to external memory, the addresses of the transfer are compared to each of the valid banks defined in the memory controller. When an address match occurs in one of the memory banks, BR*x*[MS] selects the UPM to handle this memory access. M*x*MR[BSEL] assigns the UPM to the system bus or to the local bus. System bus accesses to a bank allocated to the local bus are transferred to the internal local bus. However, local bus accesses to a bank allocated to the system bus are ignored. The last scenario occurs if one of the DMA controllers accesses the internal local bus, at an address mapped to the system bus in the bank register. Such accesses should be avoided.

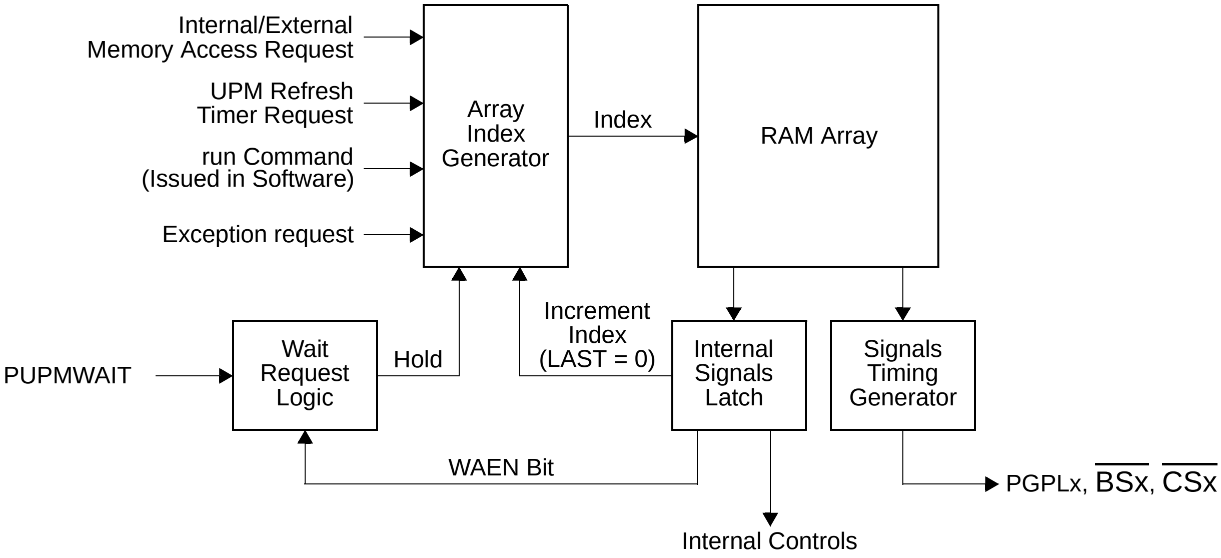


Figure 12-46. User-Programmable Machine Block Diagram

12.4.1 Requests

An internal or external device request for a memory access initiates one of the following patterns (M*x*MR[OP] = 00):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

These patterns are described in **Section 12.4.1.1**, *Memory Access Requests*. A UPM refresh timer request pattern initiates a refresh timer pattern (PTS). An exception (caused by a soft reset or the assertion of $\overline{\text{TEA}}$) while another UPM pattern is running initiates an exception condition pattern (EXS). A special pattern in the RAM array is associated with each of these cycle types. **Figure 12-47** shows the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*x*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

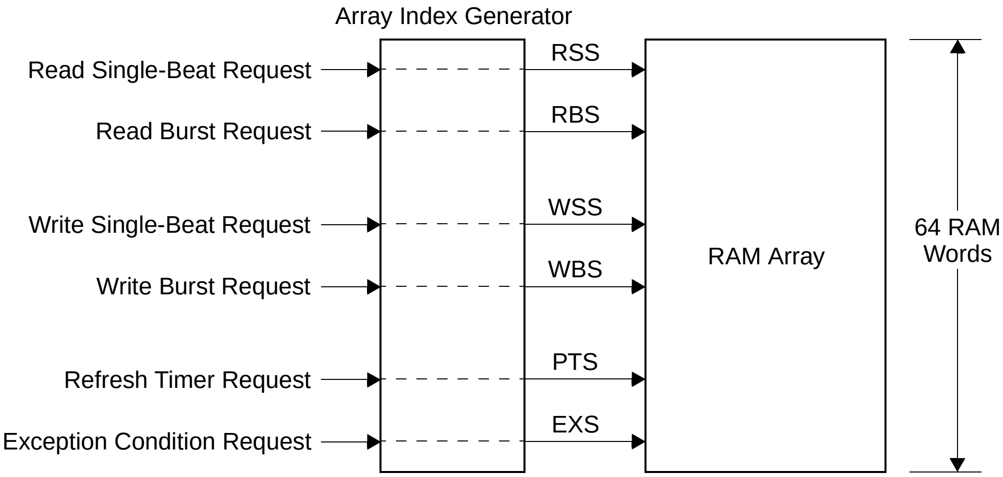


Figure 12-47. RAM Array Indexing

Table 12-20 shows the start address of each pattern.

Table 12-20. UPM Routine Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (PTS)	0x30
Exception condition (EXS)	0x3C

12.4.1.1 Memory Access Requests

When an internal device requests a new access to external memory, the address of the transfer is compared to each valid bank defined in BR*x*. The value in BR*x*[MS] selects the UPM to handle the memory access. You must ensure that the UPM is appropriately initialized before a request. The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand of up to 64 bits (8 bytes). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers 256 bits (32 bytes). For 64-bit accesses, the burst cycle starts with one transfer start and ends after four transfer acknowledges. A 32-bit device requires 8 data acknowledges; an 8-bit device requires 32. See **Section 12.1.8**, *Partial Data Valid Indication (PSDVAL)*, on page 12-10.

The MSC8113 device defines two additional transfer sizes: bursts of 128 bits (16 bytes) and 192 bits (24 bytes). The UPM treats these accesses as back-to-back, single-beat transfers.

12.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. **Figure 12-48** shows the hardware associated with memory refresh timer request generation. The system bus Assigned UPM Refresh Timer register (PURT) defines the period for the timers associated with UPM*x* on the system bus. See **Section 12.8**, *Memory Controller Programming Model*, on page 12-95.

All system bus refreshes use the refresh pattern of UPMA. If a refresh is required on the system bus, UPMA must be assigned to the system bus and MAMR[RFEN] must be set. Only one refresh routine should be programmed for the system bus and placed in the UPMA, which serves as the system bus refresh executor. If MAMR[RFEN] is set, the refresh timer of bus UPMA is assigned to request a transaction when the timer expires. There is no need to program refresh routines of UPMB and UPMC.

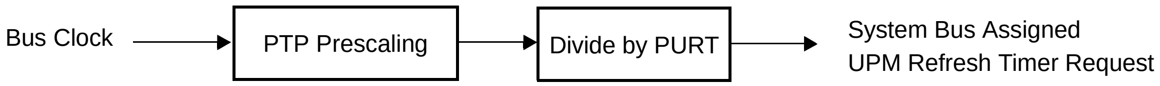


Figure 12-48. Memory Refresh Timer Request Block Diagram

12.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization. For these special cycles, you must create a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then the RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The RUN command is issued by setting M*x*MR[OP] = 11 and accessing the UPM*x* memory region with a single-byte transaction. The pattern must contain exactly one assertion of $\overline{\text{PSDVAL}}$ (UTA bit in the RAM word, described in **Table 12-21**). Otherwise, a bus time-out may occur.

12.4.1.4 Exception Requests

When the MSC8113 under UPM control initiates an access to a memory device, the external device may assert $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$. The UPM provides a mechanism by which memory control signals can meet the timing requirements of the device without losing data. The mechanism is the exception pattern that defines how the UPM deasserts its signals in a controlled manner.

12.4.2 Programming the UPMs

The UPM is a microsequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program the UPMs:

1. Set up BR*x* and OR*x*.
2. Write patterns into the RAM array.
3. Program MPTPR and PURT if refresh is required.
4. Program the Machine Mode Register (M*x*MR).

To write patterns to the RAM array, set M*x*MR[OP] = 01 and access the UPM with a single byte transaction.

12.4.3 Clock Timing

Fields in the RAM word specify the value of various external signals at each clock edge. The signal timing generator causes external signals to behave according to the timing specified in the current RAM word. **Figure 12-49** shows the clock schemes of the UPMs in the memory controller. Note that the width of T1/2/3/4 are equal. The clock phases shown reflect timing windows during which generated signals can change state.

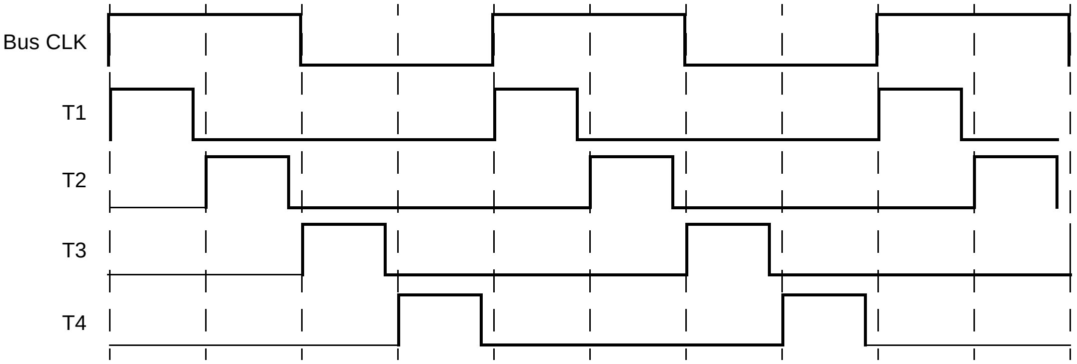


Figure 12-49. UPM Clock Scheme

The state of the external signals may change (if specified in the RAM array) at any positive edge of T1, T2, T3, or T4 (there is a propagation delay specified in **Section 2**, *Hardware Specifications*, of the *MSC8113 Data* sheet). However, only the $\overline{\text{CS}}$ signal corresponding to the currently accessed bank is manipulated by the UPM pattern when it runs. The $\overline{\text{BS}}$ signal assertion and deassertion timing is also specified for each cycle in the RAM word. The port size of the specified bank, the external address accessed, and the value of TSZ determine which of the eight $\overline{\text{BS}}$ signals are manipulated. The PGPL lines toggle as programmed for any access that initiates a particular pattern, but control resolution is limited to T1 and T3.

Figure 12-50 shows how $\overline{\text{CSx}}$, PGPL1, and PGPL2 are controlled. A word is read from the RAM which specifies the logical bits CST1, CST2, CST3, CST4, G1T1, G1T3, G2T1, and G2T3. These bits determine the corresponding output signal level for each clock.

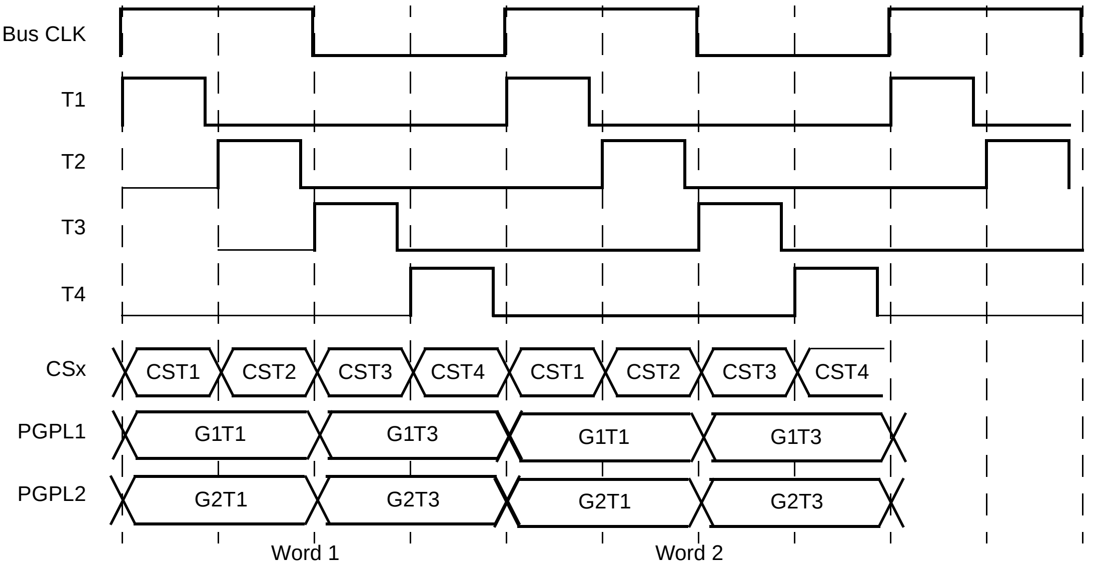
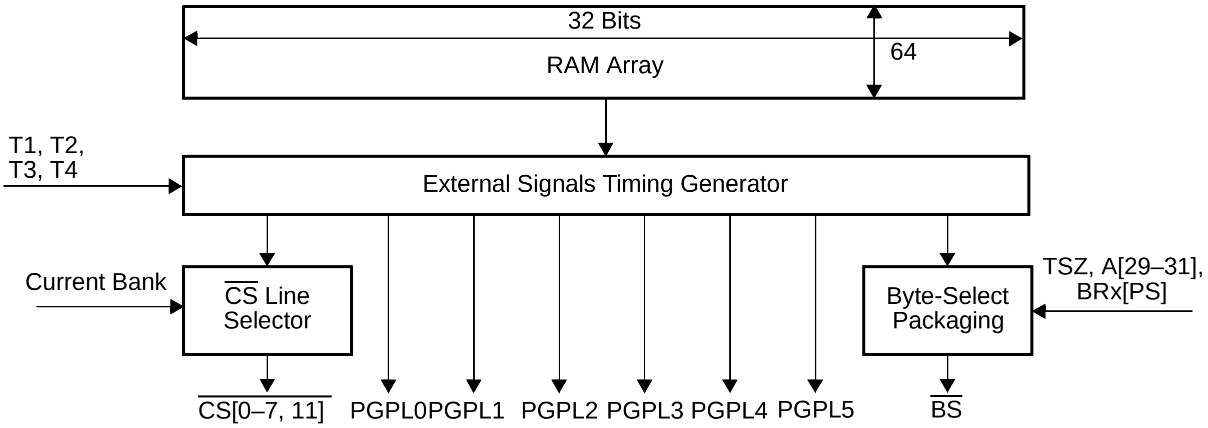


Figure 12-50. UPM Signals Timing Example

12.4.4 RAM Array

The RAM array for each UPM is 64 × 32 bits, as shown in **Figure 12-51**. The signals shown at the bottom are UPM outputs. The selected $\overline{\mathrm{CS}}$ is for the bank that matches the current address. The selected $\overline{\mathrm{BS}}$ is for the byte lanes read or written by the access.



Note: For details, see the discussion of the byte-select ($\overline{\mathrm{BS}}$) signals and the general-purpose (PGPL) signals on **page 1**

Figure 12-51. RAM Array and Signal Generation

12.4.4.1 RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM.

RAM Word RAM Word **M*x*MR[MAD] indirect addressing of 1 of 64 entries**

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H		G1T1	G1T3	G2T1	G2T3
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	G3T1	G3T3	G4T1/ DLT3	G4T3/ WAEN	G5T1	G5T3	REDO		LOOP	EXEN	AMX		NA	UTA	TODT	LAST
Type	R/W															
Reset	—	—	—	—	—	—	—		—	—	—		—	—	—	—

Table 12-21. RAM Word Bit Settings

Name	Reset	Description	Settings
CST1 0	—	**Chip-Select Timing 1** Defines the state of $\overline{\text{CS}}$ during clock phase 1.	0 The value of the $\overline{\text{CS}}$ line at the rising edge of T1 is zero. 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T1 is one.
CST2 1	—	**Chip-Select Timing 2** Defines the state of $\overline{\text{CS}}$ during clock phase 2.	0 The value of the $\overline{\text{CS}}$ line at the rising edge of T2 is zero. 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T2 is one.
CST3 2	—	**Chip-Select Timing 3** Defines the state of $\overline{\text{CS}}$ during clock phase 3.	0 The value of the $\overline{\text{CS}}$ line at the rising edge of T3 is zero. 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T3 is one.
CST4 3	—	**Chip-Select Timing 4** Defines the state of $\overline{\text{CS}}$ during clock phase 4.	0 The value of the $\overline{\text{CS}}$ line at the rising edge of T4 is zero. 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T4 is one.
BST1 4	—	**Byte-Select Timing 1** Defines the state of $\overline{\text{BS}}$ during clock phase 1. The final value of the $\overline{\text{BS}}$ lines depends on the values of BR*x*[PS], the TSZ lines, and A[29–31] for the access.	0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T1 is zero. 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T1 is one.
BST2 5	—	**Byte-Select Timing 2** Defines the state of $\overline{\text{BS}}$ during clock phase 2. The final value of the $\overline{\text{BS}}$ lines depends on the values of BR*x*[PS], TSZ, and A[29–31] for the access.	0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 is zero. 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 is one.

Table 12-21. RAM Word Bit Settings (Continued)

Name	Reset	Description	Settings
BST3 6	—	**Byte-Select Timing 3** Defines the state of $\overline{\text{BS}}$ during clock phase 3. The final value of the $\overline{\text{BS}}$ lines depends on the values of BR*x*[PS], TSZ, and A[29–31] for the access.	0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T3 is zero. 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T3 is one.
BST4 7	—	**Byte-Select Timing 4** Defines the state of $\overline{\text{BS}}$ during clock phase 4. The final value of the $\overline{\text{BS}}$ lines depends on the values of BR*x*[PS], TSZ, and A[29–31] for the access.	0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T4 is zero. 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T4 is one.
G0L 8–9	—	**General-Purpose Line 0 Lower** Defines the state of PGPL0 during phases 1–2.	00 The value of PGPL0 at the rising edge of T1 is as defined in M*x*MR[G0CL*x*]. 10 The value of the PGPL0 line at the rising edge of T1 is zero. 11 The value of the PGPL0 line at the rising edge of T1 is one.
G0H 10–11	—	**General-Purpose Line 0 Higher** Defines the state of PGPL0 during phase 3–4.	00 The value of PGPL0 at the rising edge of T3 is as defined in M*x*MR[G0CL*x*]. 10 The value of the PGPL0 line at the rising edge of T3 is zero. 11 The value of the PGPL0 line at the rising edge of T3 is one.
G1T1 12	—	**General-Purpose Line 1 Timing 1** Defines the state of PGPL1 during phase 1–2.	0 The value of the PGPL1 line at the rising edge of T1 is zero. 1 The value of the PGPL1 line at the rising edge of T1 is one.
G1T3 13	—	**General-Purpose Line 1 Timing 3** Defines the state of PGPL1 during phase 3–4.	0 The value of the PGPL1 line at the rising edge of T3 is zero. 1 The value of the PGPL1 line at the rising edge of T3 is one.
G2T1 14	—	**General-Purpose Line 2 Timing 1** Defines the state of PGPL2 during phase 1–2.	0 The value of the PGPL2 line at the rising edge of T1 is zero. 1 The value of the PGPL2 line at the rising edge of T1 is one.
G2T3 15	—	**General-Purpose Line 2 Timing 3** Defines the state of PGPL2 during phase 3–4.	0 The value of the PGPL2 line at the rising edge of T3 is zero. 1 The value of the PGPL2 line at the rising edge of T3 is one.
G3T1 16	—	**General-Purpose Line 3 Timing 1** Defines the state of PGPL3 during phase 1–2.	0 The value of the PGPL3 line at the rising edge of T1 is zero. 1 The value of the PGPL3 line at the rising edge of T1 is one.
G3T3 17	—	**General-Purpose Line 3 Timing 3** Defines the state of PGPL3 during phase 3–4.	0 The value of the PGPL3 line at the rising edge of T3 is zero. 1 The value of the PGPL3 line at the rising edge of T3 is one.

Table 12-21. RAM Word Bit Settings (Continued)

Name	Reset	Description	Settings
G4T1/DLT3 18	—	**General-Purpose Line 4 Timing 1/Delay Time 2** Function is determined by M*x*MR[GPL_x4DIS]. For an example, see **Section 12.4.4.4**, *Data Valid and Data Sample Control,* on page 12-59.	If M*x*MR defines PUPMWAIT/PGPL4 as an output (PGPL4), this bit functions as **G4T1**: 0 The value of the PGPL4 line at the rising edge of T1 is zero. 1 The value of the PGPL4 line at the rising edge of T1 is one. If M*x*MR[GPL_x4DIS] = 1, PUPMWAIT is chosen and this bit functions as **DLT3**. 0 In the current word, indicates that the data bus should be sampled at the rising edge of T1 (if a read burst or a single read service is executed). 1 In the current word, indicates that the data bus should be sampled at the rising edge of T3 (if a read burst or a single read service is executed).
G4T3/WAEN 19	—	**General-Purpose Line 4 Timing 3/Wait Enable** Function depends on the value of M*x*MR[GPL_x4DIS]. See **Section 12.4.4.7**, *Wait Mechanism,* on page 12-60.	If M*x*MR[GPL_x4DIS] = 0, **G4T3** is selected. 0 The value of the PGPL4 line at the rising edge of T3 is zero. 1 The value of the PGPL4 line at the rising edge of T3 is one. If M*x*MR[GPL_x4DIS] = 1, **WAEN** is selected. 0 The PUPMWAIT function is disabled. 1 A freeze in the external signals logical value occurs if the external UPMWAIT signal is detected asserted. This condition lasts until UPMWAIT is deasserted.
G5T1 20	—	**General-Purpose Line 5 Timing 1** Defines the state of PGPL5 during phase 1–2.	0 The value of the PGPL5 line at the rising edge of T1 is zero. 1 The value of the PGPL5 line at the rising edge of T1 is one.
G5T3 21	—	**General-Purpose Line 5 Timing 3** Defines the state of PGPL5 during phase 3–4.	0 The value of the PGPL5 line at the rising edge of T3 is zero. 1 The value of the PGPL5 line at the rising edge of T3 is one.
REDO 22–23	—	**Redo Current RAM Word** See the discussion of the repeat execution of the current RAM word on **page 12-58**.	00 Normal operation. 01 The current RAM word is executed twice. 10 The current RAM word is executed three times. 11 The current RAM word is executed four times.

Table 12-21. RAM Word Bit Settings (Continued)

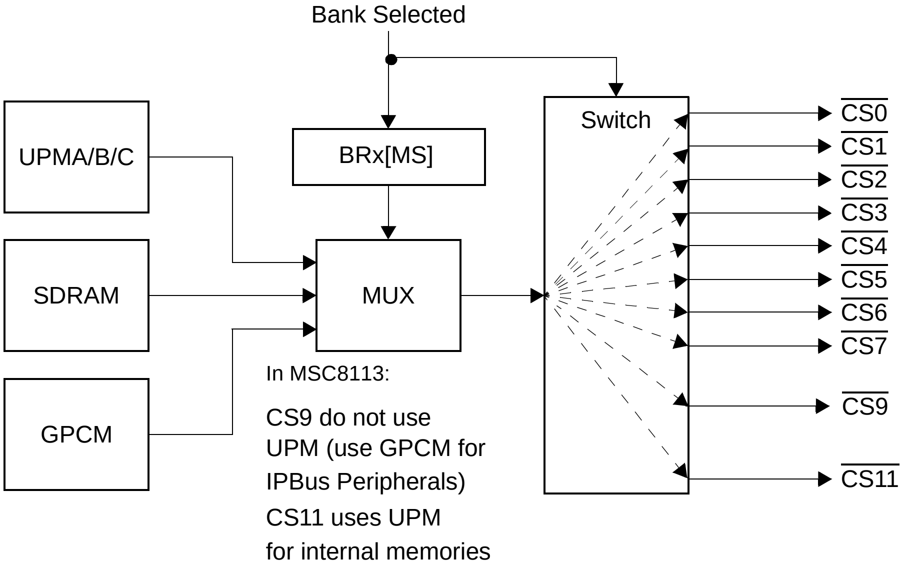
Name	Reset	Description	Settings
LOOP 24	—	**Loop Start or End** The first RAM word in the RAM array where LOOP is one is recognized as the loop start word. The next RAM word where LOOP is one is the loop end word. RAM words between the start and end are defined as the loop. The number of times the UPM executes this loop is defined in the corresponding loop field of the M*x*MR. See the discussion of loop control on **page 12-58**.	0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop.
EXEN 25	—	**Exception Enable** If an external device asserts $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$, EXEN allows branching to an exception pattern at the exception start address (EXS) at a fixed address in the RAM array. When the MSC8113 device under UPM control begins accessing a memory device, the external device may assert $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$. An exception occurs when one of these signals is asserted by an external device and the MSC8113 begins closing the memory cycle transfer. When one of these exceptions is recognized and EXEN in the RAM word is set, the UPM branches to the EXS and begins operating as the pattern defined there specifies. See **Table 12-20**. You should provide an exception pattern to deassert signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should deassert $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to prevent data corruption. If EXEN = 0, exceptions are deferred, and execution continues. After the UPM branches to the exception start address, it continues reading until the LAST bit is set in the RAM word.	0 The UPM continues executing the remaining RAM words. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected. The exception condition can be an external device asserting $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$.
AMX 26–27	—	**Address Multiplexing** Determines the source of A[0–31] at the rising edge of T1 (single-MSC8113 mode only). See **Section 12.4.4.3**, *Address Multiplexing,* on page 12-59.	00 A[0–31] is the non-multiplexed address. For example, column address. 01 Reserved. 10 A[0–31] is the address requested by the internal master multiplexed according to M*x*MR[AM*x*]. For example, row address. 11 A[0–31] is the contents of MAR. Used, for example, during SDRAM mode initialization.

Table 12-21. RAM Word Bit Settings (Continued)

Name	Reset	Description	Settings
NA 28	—	**Next Address** Determines when the address is incremented during a burst access. The value of NA is relevant only when the UPM serves a burst-read or burst-write request. NA is reserved under other patterns.	0 The address increment function is disabled. 1 The address is incremented in the next cycle. In conjunction with the BR*x*[PS], the increment value of A[27–31] and/or BADDR[27–31] at the rising edge of T1 is as follows: • If the accessed bank has a 64-bit port size, the value is incremented by 8. • If the accessed bank has a 32-bit port size, the value is incremented by 4. • If the accessed bank has a 16-bit port size, the value is incremented by 2. • If the accessed bank has an 8-bit port size, the value is incremented by 1.
UTA 29	—	**UPM Transfer Acknowledge** Indicates assertion of $\overline{\text{PSDVAL}}$, sampled by the bus interface in the current cycle.	0 $\overline{\text{PSDVAL}}$ is not asserted in the current cycle. 1 $\overline{\text{PSDVAL}}$ is asserted in the current cycle.
TODT 30	—	**Turn-On Disable Timer** The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a $\overline{\text{RAS}}$ precharge time. TODT turns the timer on to prevent another UPM access to the same bank until the timer expires. The disable timer period is determined in M*x*MR[DSx]. The disable timer does not affect memory accesses to different banks. **Note:** TODT must be set together with LAST. Otherwise, it is ignored.	0 The disable timer is turned off. 1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.
LAST 31	—	**Last** If this bit is set, it is the last RAM word in the program. When the LAST bit is read in a RAM word, the current UPM pattern terminates and the highest priority pending UPM request (if any) is serviced immediately in the external memory transactions. If the disable timer is activated (see TODT bit description) and the next access is to the same bank, the execution of the next UPM pattern is held off for the number of clock cycles specified in M*x*MR[DSx].	0 The UPM continues executing RAM words. 1 Service to the UPM request is complete.

Additional information on some of the RAM word fields is as follows:

- *Chip-Select Signals (CxTx)*. If BR*x*[MS] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{CS}}$ signal for that bank with timing as specified in the UPM RAM word. The selected UPM affects only assertion and deassertion of the appropriate $\overline{\text{CS}}$ signal. The state of the selected $\overline{\text{CS}x}$ signal of the corresponding bank depends on the value of each CST*x* bit. **Figure 12-52** and the timing diagrams in **Figure 12-50** show how UPMs control $\overline{\text{CS}}$ signals.



In MSC8113:

CS9 do not use UPM (use GPCM for IPBus Peripherals)

CS11 uses UPM for internal memories

Figure 12-52. $\overline{\text{CS}}$ Signal Selection

- *Byte-Select Signals (BxTx)*. If BR*x*[MS] of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects only the assertion and deassertion of the appropriate $\overline{\text{BS}}$ signals; their timing is specified in the RAM word. The $\overline{\text{BS}}$ signals are controlled by the port size of the accessed bank, the transfer size of the transaction, and the address accessed. **Figure 12-53** shows how UPMs control $\overline{\text{BS}}$ signals. **Table 12-22** shows how $\overline{\text{BS}}$ signals affect 64-, 32-, 16-, and 8-bit accesses. Note that for a refresh timer request, the UPM asserts/deasserts all the $\overline{\text{BS}}$ signals. The uppermost byte-select ($\overline{\text{BS[0]}}$) indicates that D[0–7] contains valid data during a cycle. Similarly, $\overline{\text{BS[[1]}}$ indicates that D[8–15] contains valid data, $\overline{\text{BS[2]}}$ indicates that D[16–23] contains valid data, and $\overline{\text{BS[3]}}$ indicates that D[24–31] contains valid data during a cycle, and so forth.

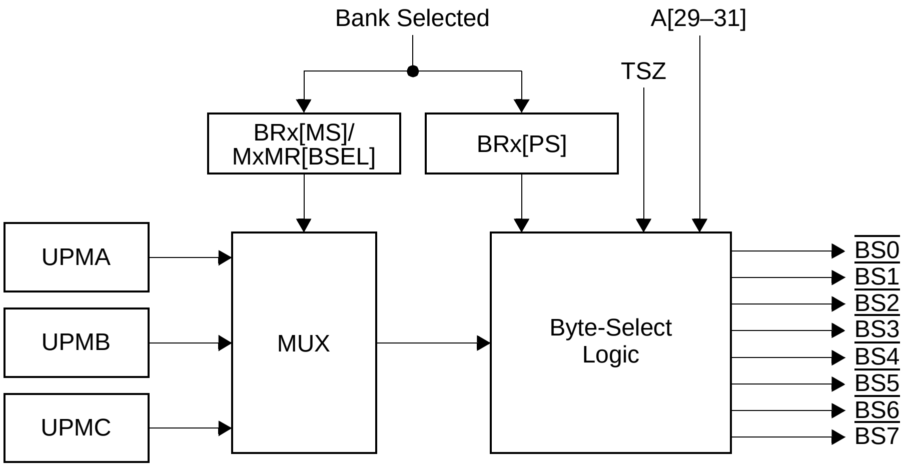


Figure 12-53. $\overline{\text{BS}}$ Signal Selection

Table 12-22. Byte-Select Enable Function

Transfer Size	Address State			Port Size														
				64-Bit								32-Bit				16-Bit		8-Bit
	A29	A30	A31	BS0	BS1	BS2	BS3	BS4	BS5	BS6	BS7	BS0	BS1	BS2	BS3	BS0	BS1	BS0
8 bits (1 bytes) TS=0001	0	0	0	A	—	—	—	—	—	—	—	A	—	—	—	A	—	A
	0	0	1	—	A	—	—	—	—	—	—	—	A	—	—	—	A	A
	0	1	0	—	—	A	—	—	—	—	—	—	—	A	—	A	—	A
	0	1	1	—	—	—	A	—	—	—	—	—	—	—	A	—	A	A
	1	0	0	—	—	—	—	A	—	—	—	A	—	—	—	A	—	A
	1	0	1	—	—	—	—	—	A	—	—	—	A	—	—	—	A	A
	1	1	0	—	—	—	—	—	—	A	—	—	—	A	—	A	—	A
	1	1	1	—	—	—	—	—	—	—	A	—	—	—	A	—	A	A
16 bits (2 bytes) TS=0010	0	0	0	A	A	—	—	—	—	—	—	A	A	—	—	A	A	A
	0	0	1	—	A	A	—	—	—	—	—	—	A	A	—	—	A	A
	0	1	0	—	—	A	A	—	—	—	—	—	—	A	A	A	A	A
	1	0	0	—	—	—	—	A	A	—	—	A	A	—	—	A	A	A
	1	0	1	—	—	—	—	—	A	A	—	—	A	A	—	—	A	A
	1	1	0	—	—	—	—	—	—	A	A	—	—	A	A	A	A	A
24 bits (3 bytes) TS=0011	0	0	0	A	A	A	—	—	—	—	—	A	A	A	—	A	A	A
	0	0	1	—	A	A	A	—	—	—	—	—	A	A	A	—	A	A
	1	0	0	—	—	—	—	A	A	A	—	A	A	A	—	A	A	A
	1	0	1	—	—	—	—	—	A	A	A	—	A	A	A	—	A	A
32 bits (4 bytes) TS=0100	0	0	0	A	A	A	A	—	—	—	—	A	A	A	A	A	A	A
	1	0	0	—	—	—	—	A	A	A	A	A	A	A	A	A	A	A
64 bits (16 bytes) TS=0000	0	0	0	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

Notes: **1.** A dash (—) denotes a byte-select (BS) is not used.

2. An “A” denotes a byte-select is used.

3. Address state is the calculated address for port size data tenure. The initial value is the address tenure A[29-31].

- *General-Purpose Signals (GxTx, GOx)*. PGPL[1–5] each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of T1 and/or at the rising edge of T3. PGPL0 offers enhancements beyond the other PGPL*x* lines. PGPL0 can be controlled by an address line specified in M*x*MR[G0CL*x*]. To use this feature, set G0H and G0L in the RAM word. For example, for a SIMM with multiple banks, this address line can switch between banks.
- *Loop Control*. The LOOP bit in the RAM word (bit 24) specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in **Table 12-23**. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter decrements by one. Continued loop execution depends on the loop counter. If the counter value is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can execute sequentially but cannot nest.

Table 12-23. M*x*MR Loop Field Usage

Request Serviced	Loop Field
Read single-beat cycle	M*x*MR[RLF*x*]
Read burst cycle	M*x*MR[RLF*x*]
Write single-beat cycle	M*x*MR[WLF*x*]
Write burst cycle	M*x*MR[WLF*x*]
Refresh timer expired	M*x*MR[TLF*x*]
RUN command	M*x*MR[RLF*x*]

- *Repeat Execution of Current RAM Word (REDO)*. The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words. Set the REDO bits of the RAM word to a nonzero value to cause the UPM to re-execute the current RAM word up to three times, according to **Table 12-21**. **Figure 12-68** shows an example of REDO use. Special care must be taken in the following cases:
  - — When UTA and REDO are both set, $\overline{\text{PSDVAL}}$ is asserted the number of times specified by the REDO function.
  - — When LOOP and REDO are both set, the loop mechanism works as usual, and the line is repeated according to the REDO function.
  - — LAST and REDO should not both be set.
  - — REDO should not be used within an exception routine.

12.4.4.2 Last Word (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern terminates, and the highest-priority pending UPM request (if any) is serviced immediately in the external memory transactions. If the disable timer is activated and the next access is to the same bank, the execution of the next UPM pattern is held off for the number of clock cycles specified in M*x*MR[DSx].

12.4.4.3 Address Multiplexing

The address lines are controlled by the pattern you provide in the UPM. The address multiplex bits can choose between outputting an address requested by the internal master as is or outputting it according to the multiplexing specified by the M*x*MR[AMx]. The last option is to output the contents of the Memory Address Register (MAR) on the external address bus. In 60x-compatible mode, MAR cannot be output on the 60x bus external address line. **Table 12-24** shows how M*x*MR[AMx] settings affect address multiplexing. Fro details, see **Section 12.2.14**, *SDRAM Configuration Examples*.

Table 12-24. UPM Address Multiplexing

AM*x*	External 60x-compatible Bus Address Line	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external line when address multiplexing is enabled	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		—	—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

12.4.4.4 Data Valid and Data Sample Control

When the UPM handles a read access and the UTA bit is 1, the value of the DLT3 bit in the same RAM word indicates when the data input is sampled by the internal 60x-compatible bus master, assuming that M*x*MR[GPL_x4DIS] = 1.

- If G4T4/DLT3 functions as DLT3 and DLT3 = 1 in the RAM word, data is latched on the falling edge of the external bus clock instead of the rising edge. The data is sampled by the internal master on the next rising edge as required by the MSC8113 bus. This feature lets you speed up the memory interface by latching data one-half clock early, which can be useful during burst reads. This feature should be used only in systems without external synchronous bus devices.
- If G4T4/DLT3 functions as G4T4, data is latched on the rising edge of the external bus clock, as is normal in MSC8113 bus operation.

Figure 12-54 shows UPM-controlled data sampling.

12.4.4.5 Disable Timer Mechanism (TODT)

The TODT bit in the RAM word turns the timer on to prevent another UPM access to the same bank until the timer expires. For details on this bit, see the discussion of Bit 30 in **Table 12-21**.

12.4.4.6 Signal Deassertion

When the LAST bit is read in a RAM word, the current UPM pattern terminates. On the next cycle all the UPM signals are deasserted unconditionally (driven to logic ‘1’). This deassertion does not occur if there is a back-to-back UPM request pending. In this case, the signals value on the cycle following the LAST bit is taken from the first line of the pending UPM routine.

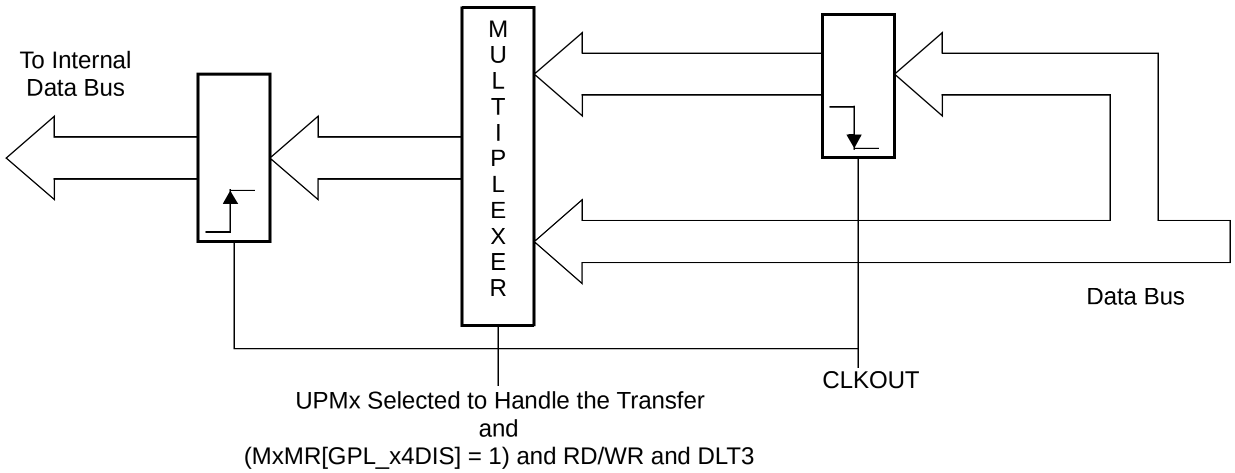

To Internal
Data Bus
MULTIPLEXER
Data Bus
CLKOUT
UPMx Selected to Handle the Transfer
and
(MxMR[GPL_x4DIS] = 1) and RD/WR and DLT3

Figure 12-54. UPM Read Access Data Sampling

12.4.4.7 Wait Mechanism

The WAEN bit in the RAM array word, shown in **Table 12-21**, enables the UPM wait mechanism in selected UPM RAM words. If the UPM reads two consecutive RAM words with the WAEN bit set, the external PUPMWAIT signal is sampled by the memory controller in the following cycle and the request is frozen. The PUPMWAIT signal is sampled at the rising edge of CLKOUT. If PUPMWAIT is asserted and WAEN = 1 in the previous UPM word, the UPM is frozen until PUPMWAIT is deasserted. The value of the external signal lines driven by the UPM remains as indicated in the previous word read by the UPM. When PUPMWAIT is deasserted, the UPM continues its normal functions. Note that during the WAIT cycles, the UPM deasserts $\overline{\text{PSDVAL}}$.

Figure 12-55 shows how the WAEN bit in the word read by the UPM and the PUPMWAIT signal hold the UPM in a particular state until PUPMWAIT is deasserted. As the example in **Figure 12-55** shows, the $\overline{\text{CS}x}$ and PGPL1 states (c12 and C) and the WAEN value (‘1’) are frozen until PUPMWAIT is recognized as deasserted. WAEN is typically set before the line that contains UTA = 1.

12.4.4.8 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some combination of OR*x*[EHTR]. Accesses after a read access to the slower memory bank is delayed by the number of clock cycles specified by **Table 12-17**, *TRLX and EHTR Combinations*, on page 12-40. For details, see **Section 12.3.1.6**, *Extended Hold Time on Read Accesses*, on page 12-40.

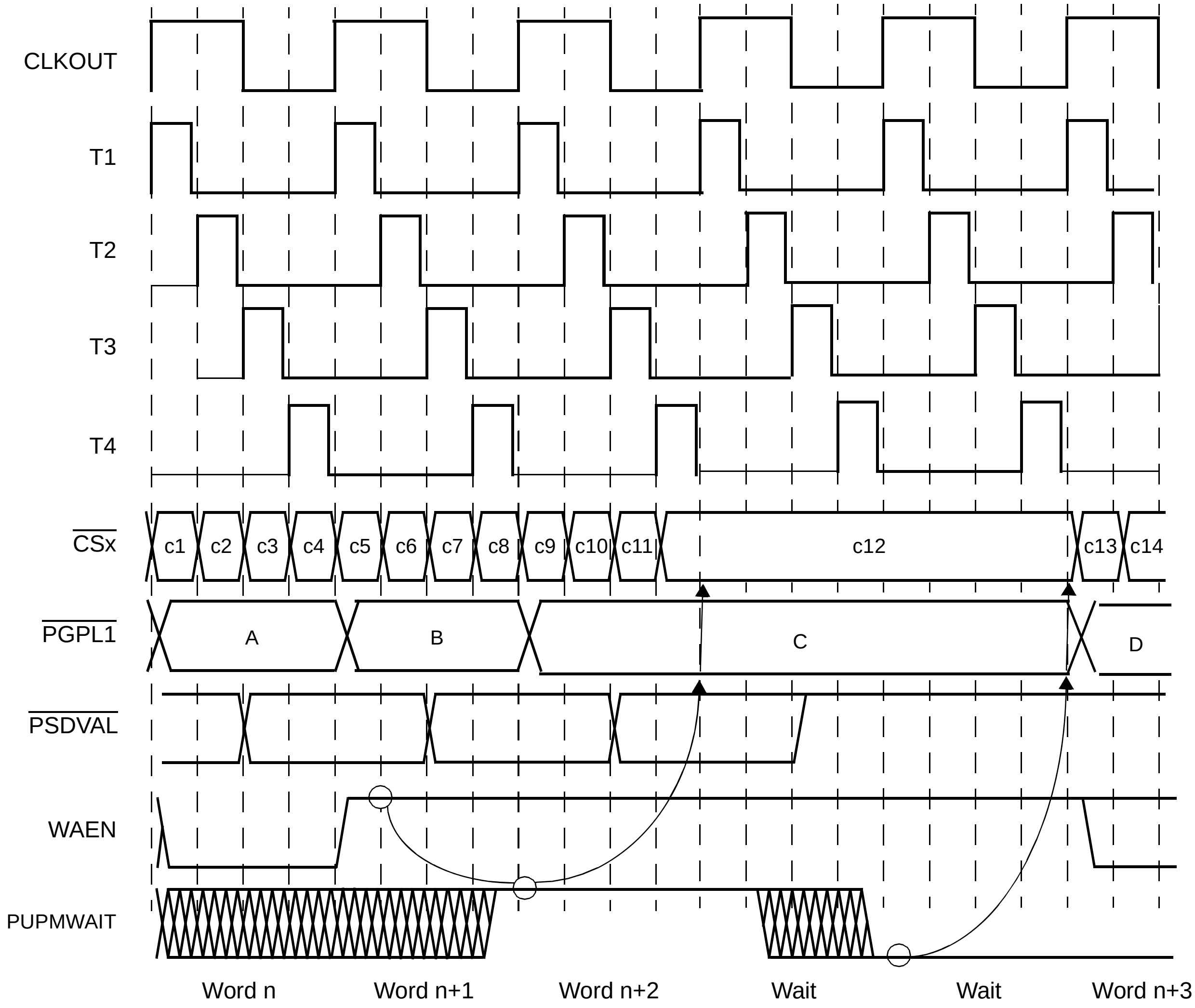


Figure 12-55. Wait Mechanism Timing for Internal and External Synchronous Masters

12.4.5 DRAM Configuration Example

In the following example, the DRAM is located on the external system bus. Consider the following DRAM organization:

- 64-bit port size organized as 8 × 8 × 2 Mb
- Each device has 12 row address lines and 9 column address lines.

The address bus is partitioned as shown in **Table 12-25**.

Table 12-25. System Address Bus Partition

A[0–7]	A[8–19]	A[20–28]	A[29–31]
msb of start address	Row	Column	lsb

During $\overline{\text{RAS}}$ assertion, the device address port should look contain the values shown in **Table 12-26**.

Table 12-26. DRAM Device Address Port During an ACTIVATE command

A[8–16]	A[17–28]	A[29–31]
—	Row (A[8–19])	n.c.

Table 12-24 indicates that to multiplex A[8–19] over A[17–28], choose M*x*MR[AM*x*] = 001. **Table 12-27** shows the register configuration. Not shown are PURT and MPTPR, which should be programmed according to the device refresh requirements.

Table 12-27. Register Settings

Register	Settings			
BR*x*	BA	msb of base address	EMEMC	0
	PS	00 = 64-bit port size	ATOM	00
	DECC	00	DR	0
	WP	0	V	1
	MS	100 = UPMA		
OR*x*	AM	11111111000000000 = 16 MB	BI	0
	BCTLD	0	EHTR	00
MAMR	BSEL	0 = 60x bus	GPL_*x*4DIS	0
	RFEN	1	RL F*x*	As needed
	OP	00	WLF*x*	As needed
	AM*x*	001	TLF*x*	As needed
	DS*x*	As needed	MAD	N/A
	G0CL*x*	N/A		

12.4.6 Interface Examples

Connecting the MSC8113 to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section presents timing diagrams for various UPM configurations.

12.4.6.1 Memory System Interface Example Using UPM

After timings are created, programming the UPM continues with translating these timings into tables representing the RAM array contents for each possible cycle. When a table is completed, the global parameters of the UPM must be defined for handling the disable timer (precharge) and the refresh timer relative to **Figure 12-56**.

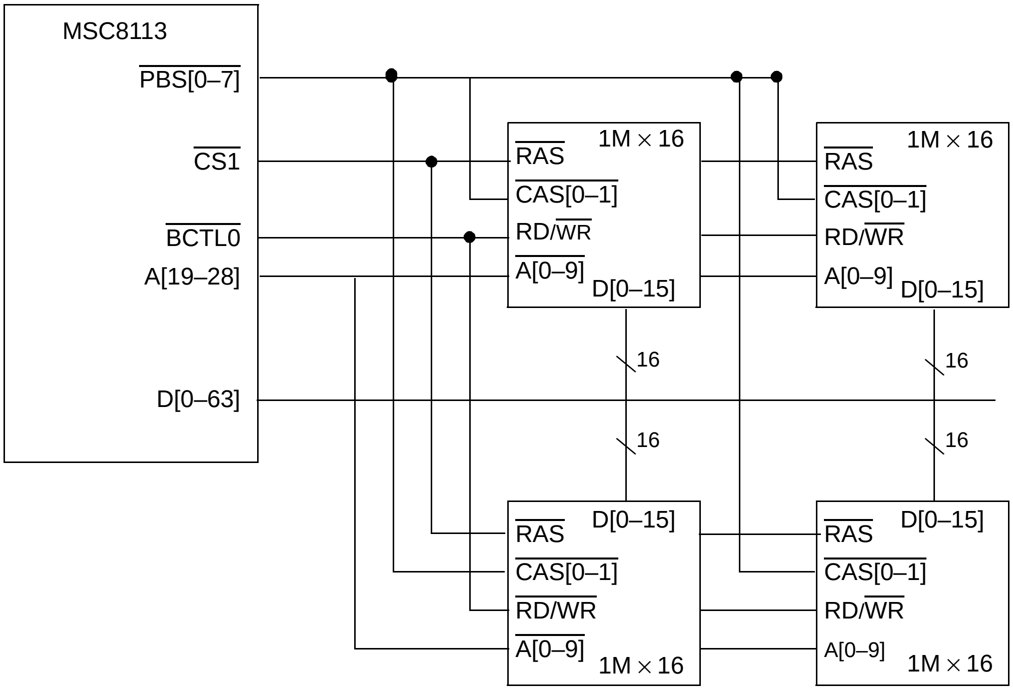


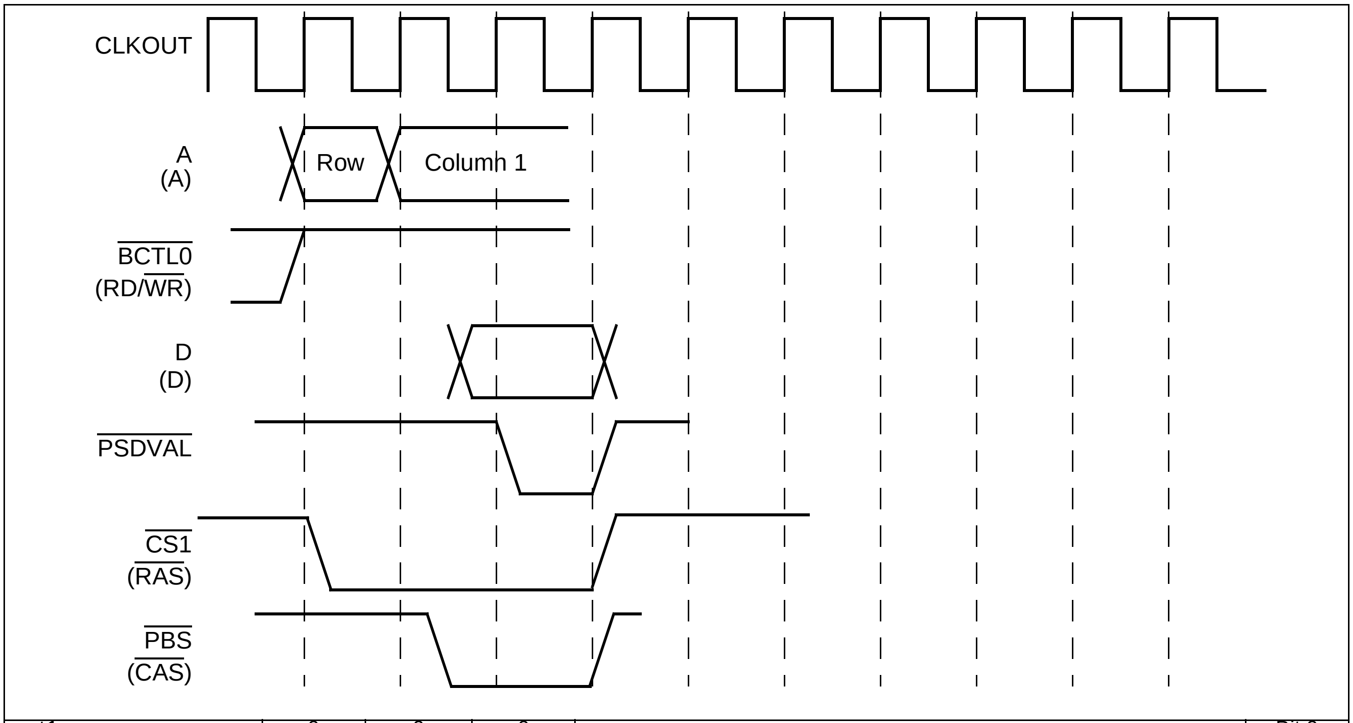
Figure 12-56. DRAM Interface Connection to the 60x bus (64-Bit Port Size)

Table 12-28 shows settings of different fields.

Table 12-28. UPMs Attributes Example

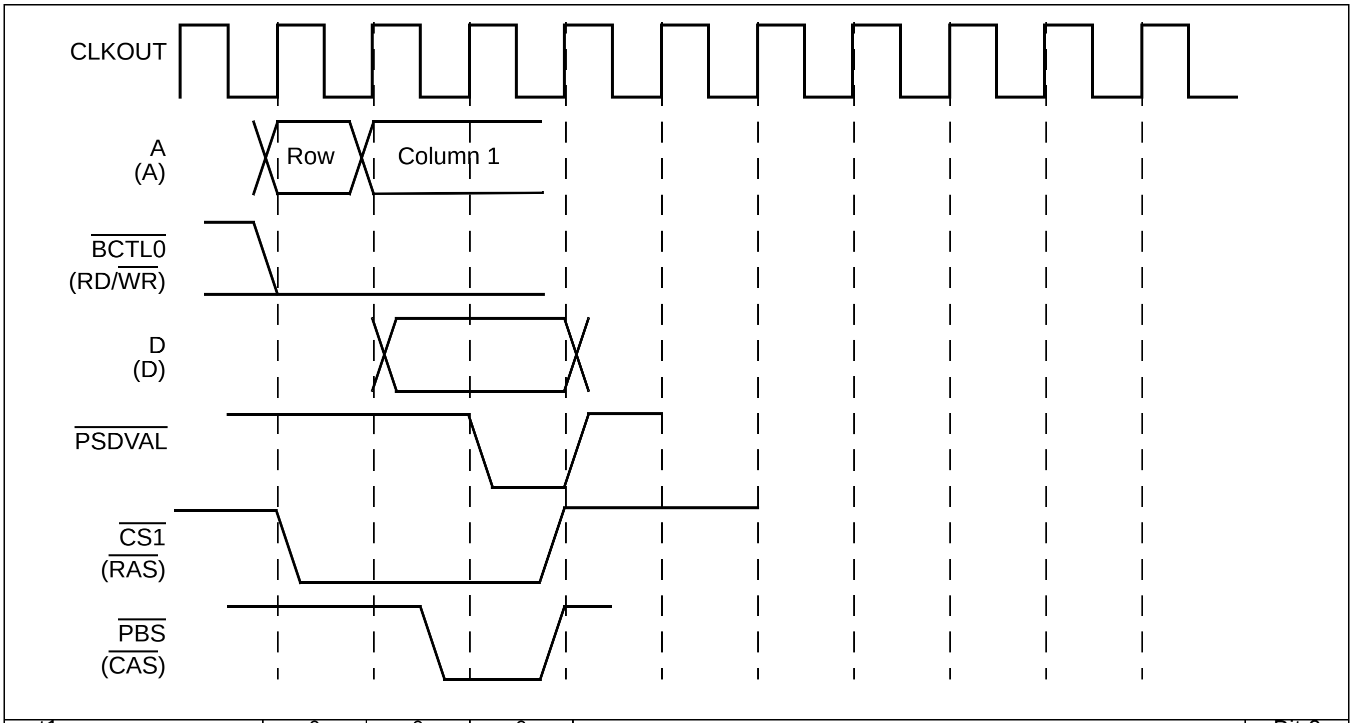
Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect (R/W)	BR*x*[WP]	0b0
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x0C
Refresh timer enable	MAMR[RFEN]	0b1
Address multiplex size	MAMR[AMx]	0b010
Disable timer period	MAMR[DSx]	0b01
Select between PGPL4 and Wait = PGPL4 data sample at clock rising edge	MAMR[GPL_x4DIS]	0b0
Burst inhibit device	OR*x*[BI]	0b0

The OR and BR of the specific bank must be initialized according to the address mapping of the DRAM device used. The BR*x*[MS] field should indicate the specific UPM selected to handle the cycle. The RAM array of the UPM can then be written through use of the M*x*MR[OP] = 01. **Figure 12-47** shows the first locations addressed by the UPM, according to the different services required by the DRAM.



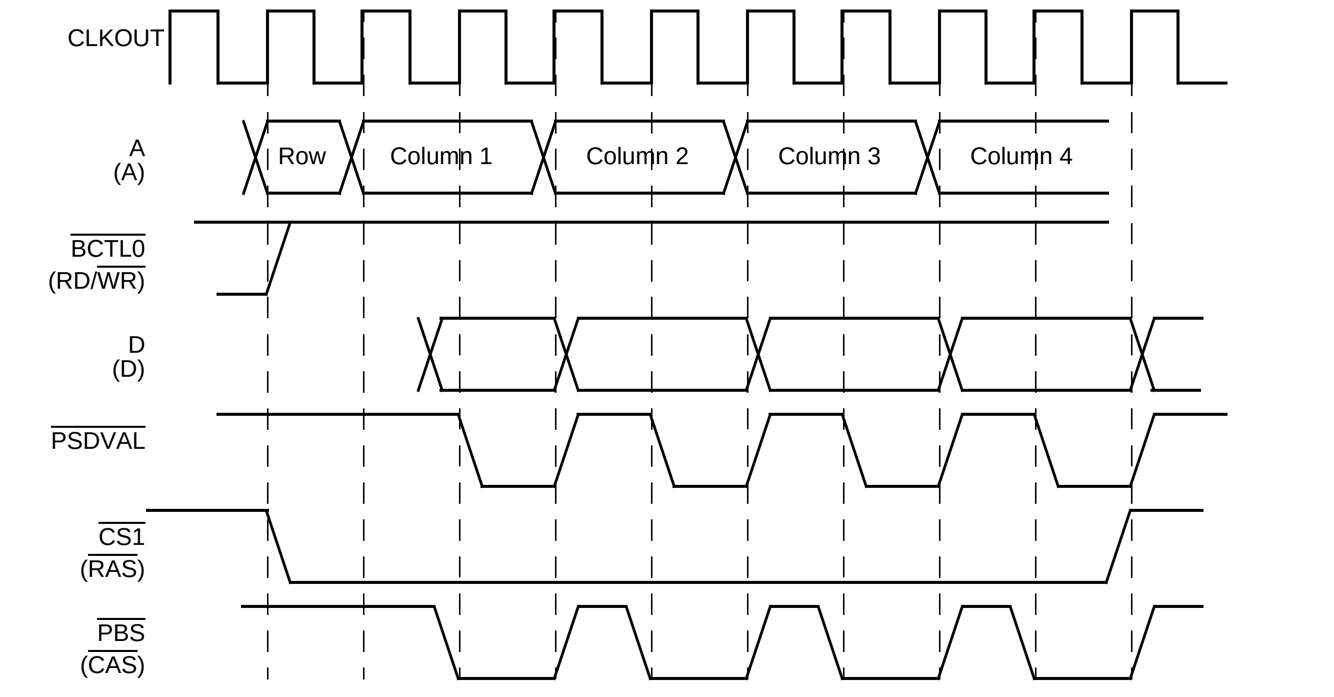
cst1	0	0	0		Bit 0
cst2	0	0	0		Bit 1
cst3	0	0	0		Bit 2
cst4	0	0	0		Bit 3
bst1	1	1	0		Bit 4
bst2	1	0	0		Bit 5
bst3	1	0	0		Bit 6
bst4	1	0	0		Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1					Bit 12
g1t3					Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo0					Bit 22
redo1					Bit 23
loop	0	0	0		Bit 24
exen	0	0	0		Bit 25
amx0	1	0	0		Bit 26
amx1	0	0	0		Bit 27
na	0	0	0		Bit 28
uta	0	0	1		Bit 29
todt	0	0	1		Bit 30
last	0	0	1		Bit 31
	RSS	**RSS+1**	**RSS+2**		

Figure 12-57. Single-Beat Read Access to FPM DRAM



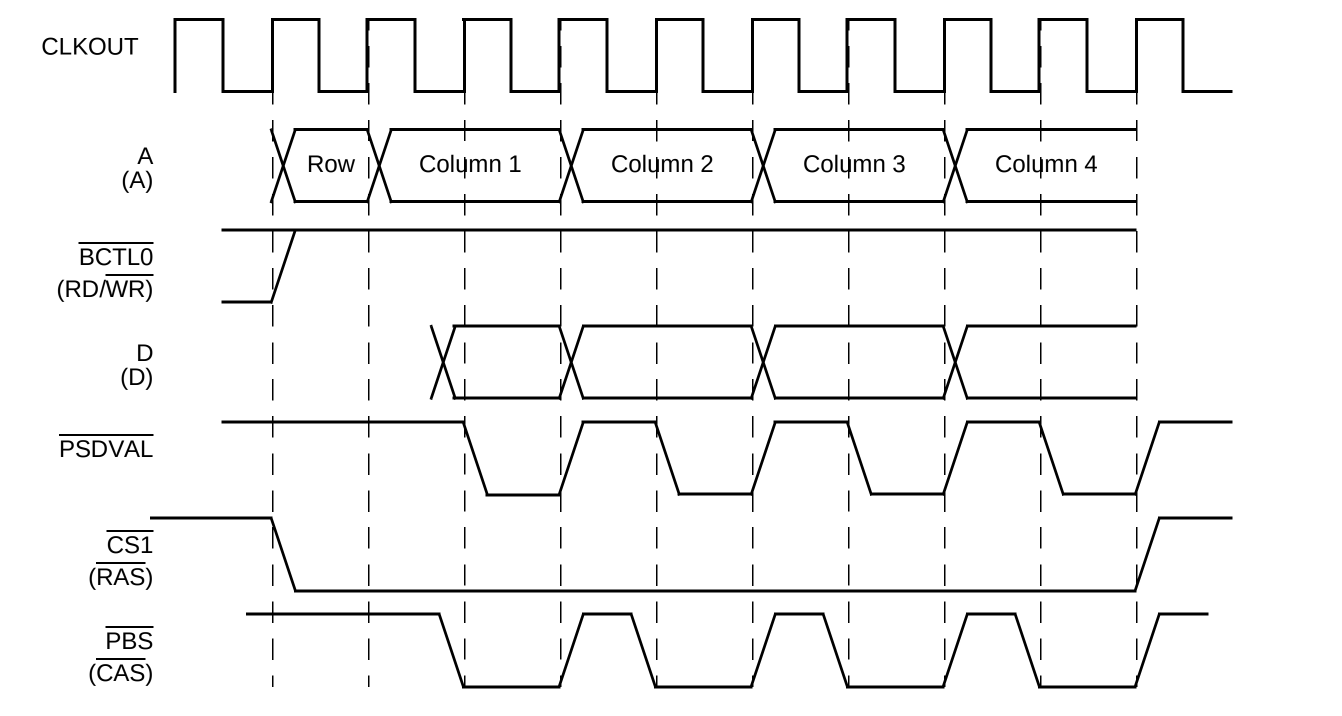
cst1	0	0	0		Bit 0
cst2	0	0	0		Bit 1
cst3	0	0	0		Bit 2
cst4	0	0	1		Bit 3
bst1	1	1	0		Bit 4
bst2	1	1	0		Bit 5
bst3	1	0	0		Bit 6
bst4	1	0	1		Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1					Bit 12
g1t3					Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo0					Bit 22
redo1					Bit 23
loop	0	0	0		Bit 24
exen	0	0	0		Bit 25
amx0	1	0	0		Bit 26
amx1	0	0	0		Bit 27
na	0	0	0		Bit 28
uta	0	0	1		Bit 29
todt	0	0	1		Bit 30
last	0	0	1		Bit 31
	WSS	**WSS+1**	**WSS+2**		

Figure 12-58. Single-Beat Write Access to FPM DRAM



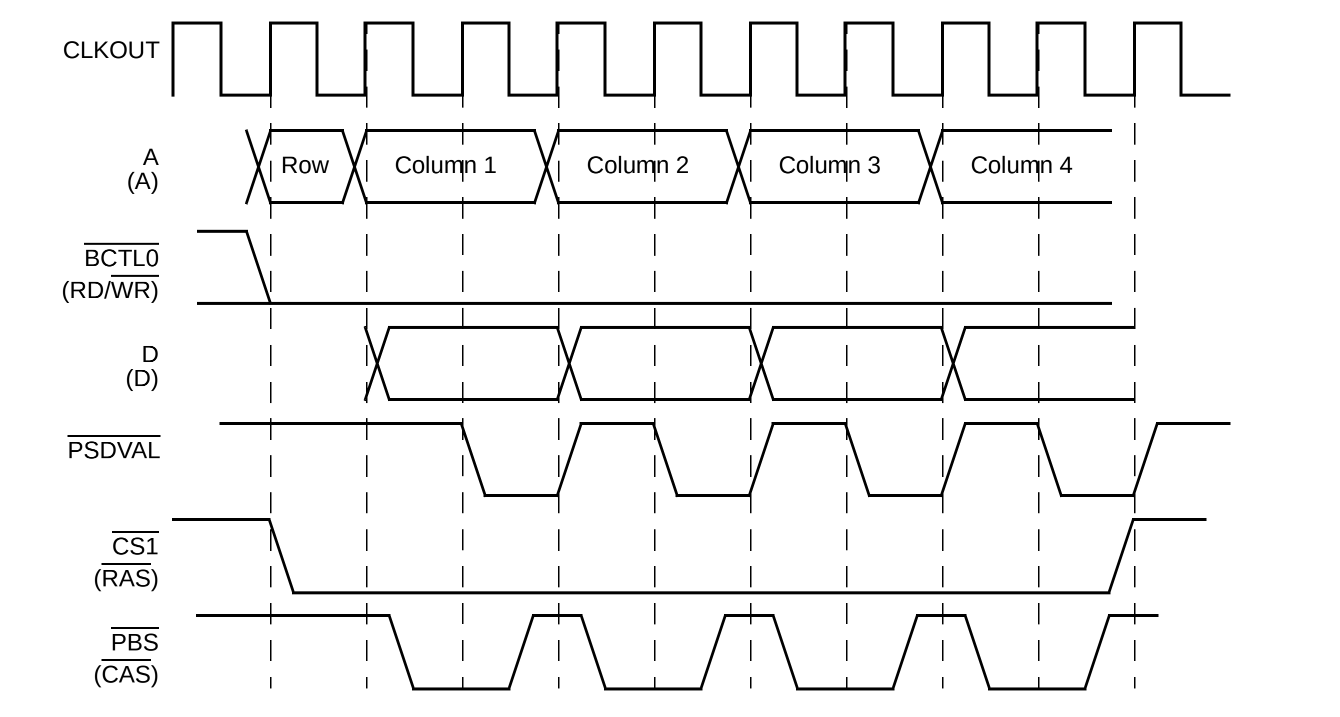
cst1	0	0	0	0	0	0	0	0	0		Bit 0
cst2	0	0	0	0	0	0	0	0	0		Bit 1
cst3	0	0	0	0	0	0	0	0	0		Bit 2
cst4	0	0	0	0	0	0	0	0	1		Bit 3
bst1	1	1	0	1	0	1	0	1	0		Bit 4
bst2	1	1	0	1	0	1	0	1	0		Bit 5
bst3	1	1	0	1	0	1	0	1	0		Bit 6
bst4	1	0	0	0	0	0	0	0	0		Bit 7
g0l0											Bit 8
g0l1											Bit 9
g0h0											Bit 10
g0h1											Bit 11
g1t1											Bit 12
g1t3											Bit 13
g2t1											Bit 14
g2t3											Bit 15
g3t1											Bit 16
g3t3											Bit 17
g4t1											Bit 18
g4t3											Bit 19
g5t1											Bit 20
g5t3											Bit 21
redo0											Bit 22
redo1											Bit 23
loop	0	0	0	0	0	0	0	0	0		Bit 24
exen	0	0	1	0	1	0	1	0	0		Bit 25
amx0	1	0	0	0	0	0	0	0	0		Bit 26
amx1	0	0	0	0	0	0	0	0	0		Bit 27
na	0	0	1	0	1	0	1	0	0		Bit 28
uta	0	0	1	0	1	0	1	0	1		Bit 29
todt	0	0	0	0	0	0	0	0	1		Bit 30
last	0	0	0	0	0	0	0	0	1		Bit 31
	RBS	**RBS+1**	**RBS+2**	**RBS+3**	**RBS+4**	**RBS+5**	**RBS+6**	**RBS+7**	**RBS+8**		

Figure 12-59. Burst Read Access to FPM DRAM (No LOOP)



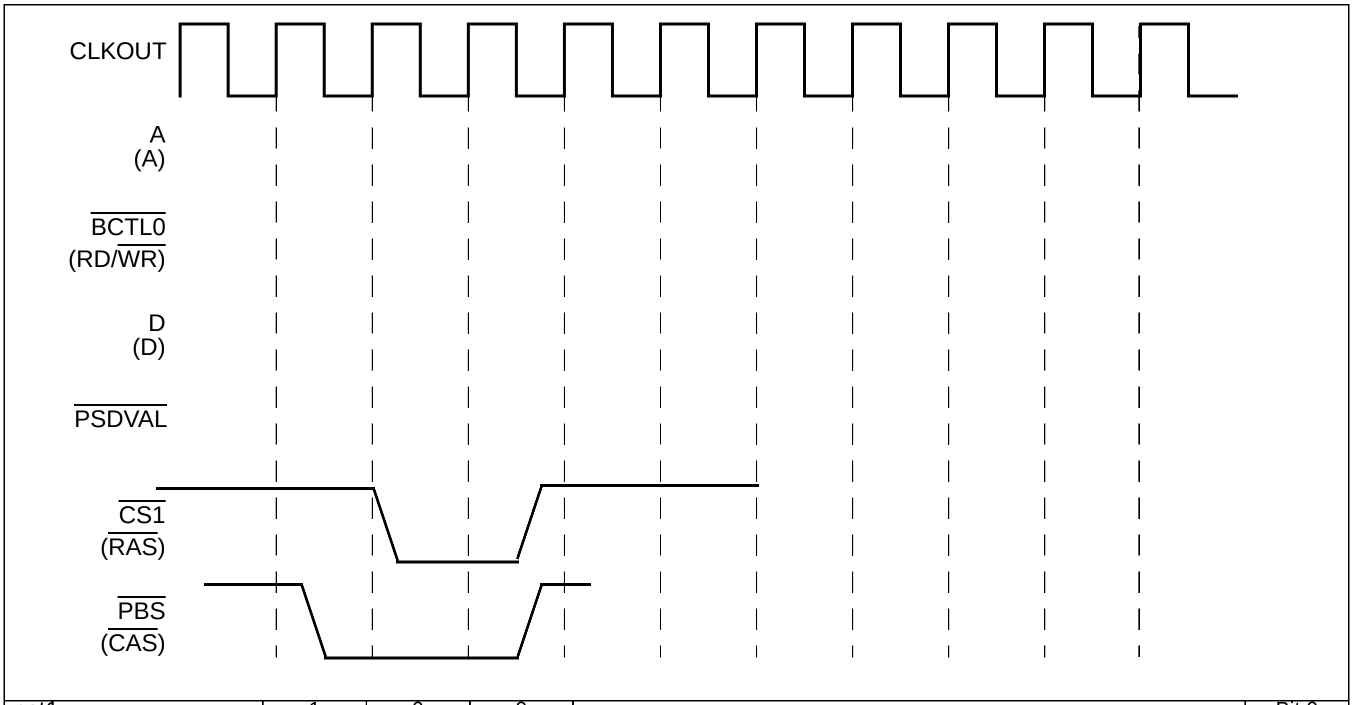
cst1	0	0	0				Bit 0
cst2	0	0	0				Bit 1
cst3	0	0	0				Bit 2
cst4	0	0	0				Bit 3
bst1	1	1	0				Bit 4
bst2	1	1	0				Bit 5
bst3	1	1	0				Bit 6
bst4	1	0	0				Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1							Bit 12
g1t3							Bit 13
g2t1							Bit 14
g2t3							Bit 15
g3t1							Bit 16
g3t3							Bit 17
g4t1							Bit 18
g4t3							Bit 19
g5t1							Bit 20
g5t3							Bit 21
redo0							Bit 22
redo1							Bit 23
loop	0	1	1				Bit 24
exen	0	0	1				Bit 25
amx0	1	0	0				Bit 26
amx1	0	0	0				Bit 27
na	0	0	1				Bit 28
uta	0	0	1				Bit 29
todt	0	0	1				Bit 30
last	0	0	1				Bit 31
	RBS	RBS+1	RBS+2				

Figure 12-60. Burst Read Access to FPM DRAM (LOOP)



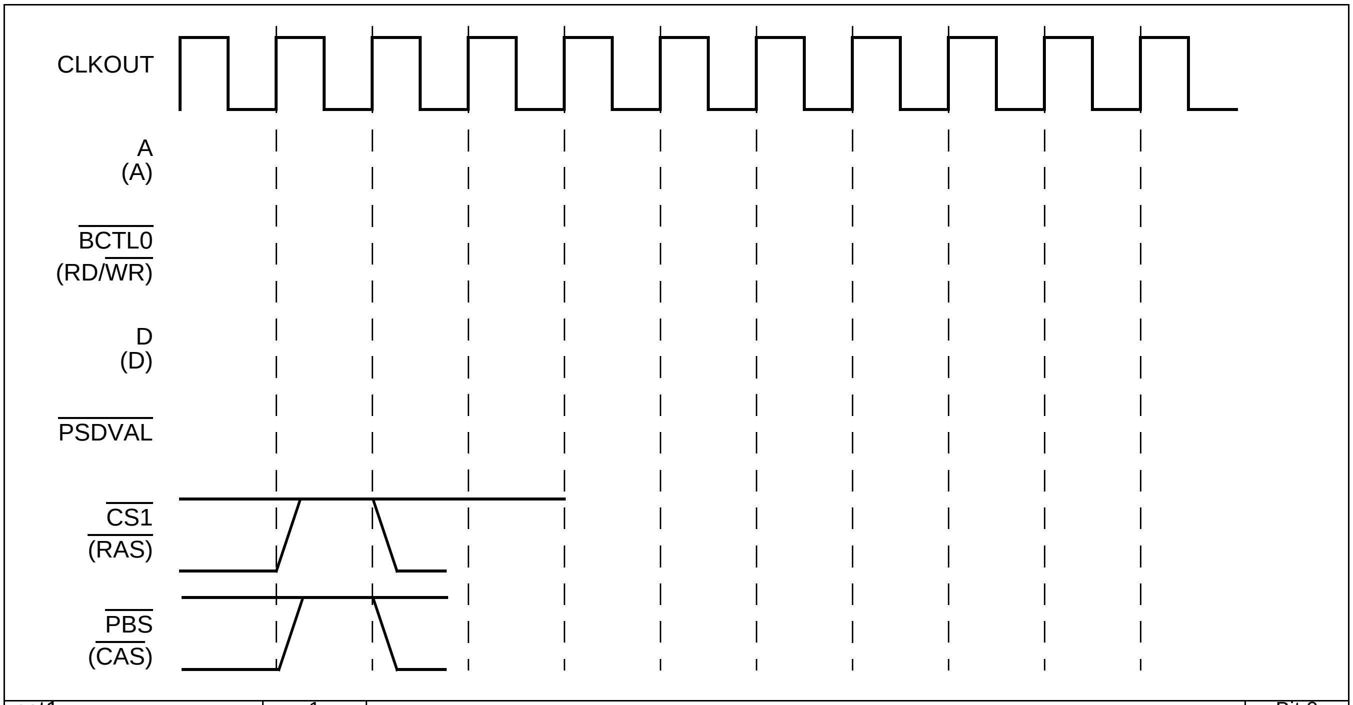
cst1	0	0	0	0	0	0	0	0	0		Bit 0
cst2	0	0	0	0	0	0	0	0	0		Bit 1
cst3	0	0	0	0	0	0	0	0	0		Bit 2
cst4	0	0	0	0	0	0	0	0	1		Bit 3
bst1	1	1	0	1	0	1	0	1	0		Bit 4
bst2	1	0	0	0	0	0	0	0	0		Bit 5
bst3	1	0	1	0	1	0	1	0	1		Bit 6
bst4	1	0	1	0	1	0	1	0	1		Bit 7
g0l0											Bit 8
g0l1											Bit 9
g0h0											Bit 10
g0h1											Bit 11
g1t1											Bit 12
g1t3											Bit 13
g2t1											Bit 14
g2t3											Bit 15
g3t1											Bit 16
g3t3											Bit 17
g4t1											Bit 18
g4t3											Bit 19
g5t1											Bit 20
g5t3											Bit 21
redo0											Bit 22
redo1											Bit 23
loop	0	0	0	0	0	0	0	0	0		Bit 24
exen	0	0	1	0	1	0	1	0	0		Bit 25
amx0	1	0	0	0	0	0	0	0	0		Bit 26
amx1	0	0	0	0	0	0	0	0	0		Bit 27
na	0	0	1	0	1	0	1	0	0		Bit 28
uta	0	0	1	0	1	0	1	0	1		Bit 29
todt	0	0	0	0	0	0	0	0	1		Bit 30
last	0	0	0	0	0	0	0	0	1		Bit 31
	WBS	WBS+1	wBS+2	WBS+3	WBS+4	WBS+5	WBS+6	WBS+7	WBS+8		

Figure 12-61. Burst Write Access to FPM DRAM (No LOOP)



cst1	1	0	0		Bit 0
cst2	1	0	0		Bit 1
cst3	1	0	1		Bit 2
cst4	1	0	1		Bit 3
bst1	1	0	0		Bit 4
bst2	0	0	0		Bit 5
bst3	0	0	1		Bit 6
bst4	0	0	1		Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1					Bit 12
g1t3					Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo0					Bit 22
redo1					Bit 23
loop	0	0	0		Bit 24
exen	0	0	0		Bit 25
amx0	0	0	0		Bit 26
amx1	0	0	0		Bit 27
na	0	0	0		Bit 28
uta	0	0	0		Bit 29
todt	0	0	1		Bit 30
last	0	0	1		Bit 31
	PTS	PTS+1	PTS+2		

Figure 12-62. Refresh Cycle (CBR) to FPM DRAM



cst1	1		Bit 0
cst2	1		Bit 1
cst3	1		Bit 2
cst4	1		Bit 3
bst1	1		Bit 4
bst2	1		Bit 5
bst3	1		Bit 6
bst4	1		Bit 7
g0l0			Bit 8
g0l1			Bit 9
g0h0			Bit 10
g0h1			Bit 11
g1t1			Bit 12
g1t3			Bit 13
g2t1			Bit 14
g2t3			Bit 15
g3t1			Bit 16
g3t3			Bit 17
g4t1			Bit 18
g4t3			Bit 19
g5t1			Bit 20
g5t3			Bit 21
redo0			Bit 22
redo1			Bit 23
loop	0		Bit 24
exen	0		Bit 25
amx0	0		Bit 26
amx1	0		Bit 27
na	0		Bit 28
uta	0		Bit 29
todt	1		Bit 30
last	1		Bit 31
	EXS		

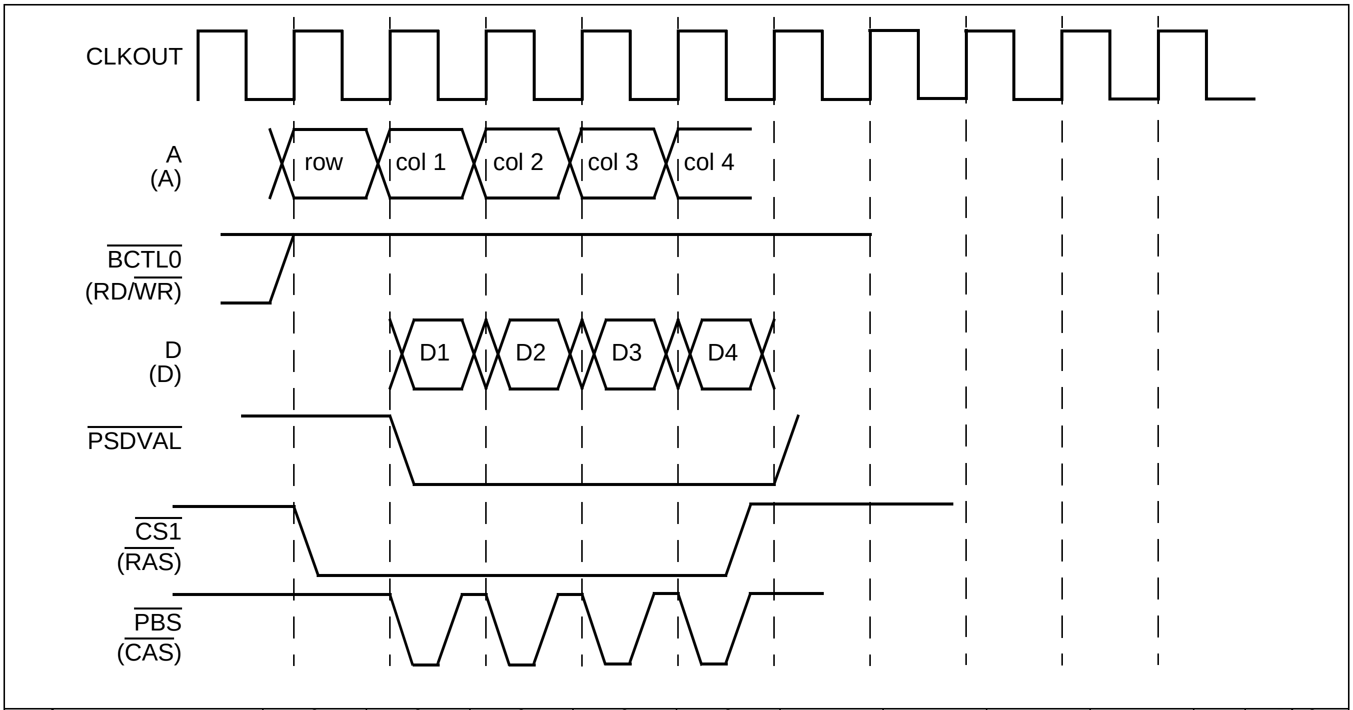
Figure 12-63. Exception Cycle

If PGPL4 is not used as an output, the performance for a page read access can be improved by setting M*x*MR[GPL_x4DIS]. The following example shows how the burst read access to FPM DRAM (no LOOP) is modified using this feature. The configuration registers are defined as shown in **Table 12-29**.

Table 12-29. UPMs Attributes Example

Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect	BR*x*[WP]	0b0
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x0C
Refresh timer enable	MAMR[RFEN]	0b1
Address multiplex size	MAMR[AMx]	0b010
Disable timer period	MAMR[DSx]	0b01
Select between PGPL4 and Wait = Wait, data sampled at clock negative edge	MAMR[GPL_x4DIS]	0b1
Burst inhibit device	OR*x*[BI]	0b0

The timing diagram in **Figure 12-64** shows how the burst-read access in **Figure 12-59** can be reduced.



cst1 cst2 cst3 cst4	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1						Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1						Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1											Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3											Bit 12 Bit 13
g2t1 g2t3											Bit 14 Bit 15
g3t1 g3t3											Bit 16 Bit 17
g4t1 -> DLT3 g4t3	1 0	1 0	1 0	1 0	1 0						Bit 18 Bit 19
g5t1 g5t3											Bit 20 Bit 21
redo0 redo1											Bit 22 Bit 23
loop exen	0 0	0 0	0 0	0 0	0 0						Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0	0 0 1 1 0 0	0 0 1 1 0 0	0 0 1 1 0 0	0 0 0 1 1 1						Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	RBS	RBS+1	RBS+2	RBS+3	RBS+4						

Figure 12-64. FPM DRAM Burst Read Access (Data Sampling on Falling Edge of CLKIN)

12.4.6.2 EDO Interface Example

Figure 12-65 shows a memory connection to extended data-out type devices. For this connection, PGPL1 connects to the memory device OE signals.

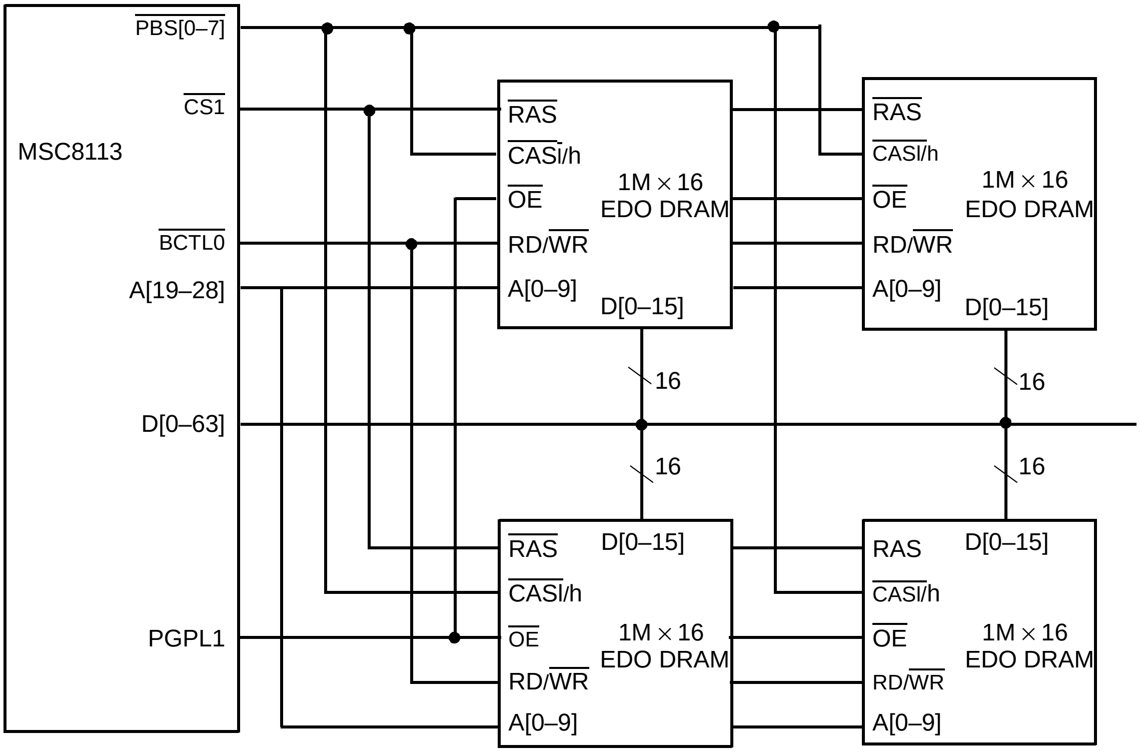
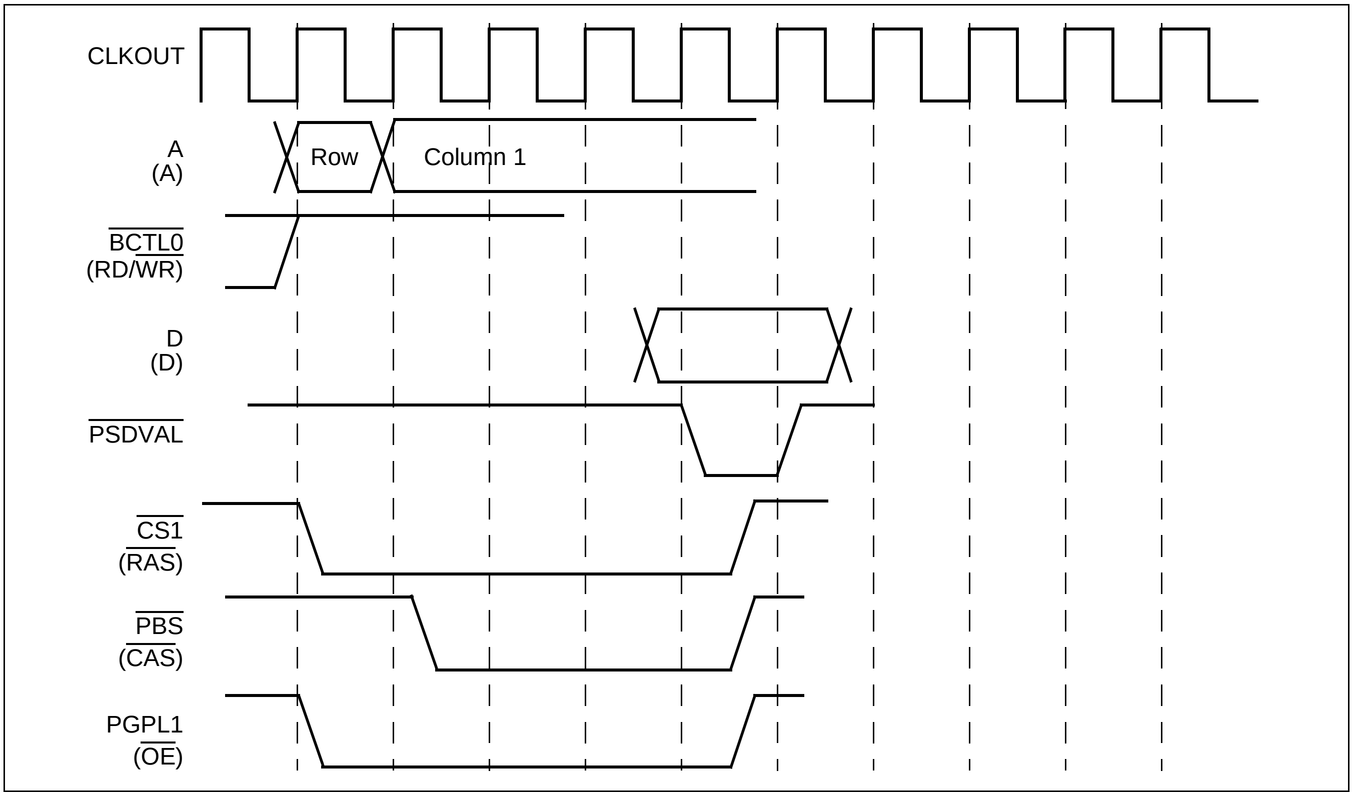


Figure 12-65. MSC8113/EDO Interface Connection to the System Bus (64-Bit Port Size)

Table 12-30 shows the programming of the register field to support the configuration shown in **Figure 12-65**. The example assumes a CLKOUT frequency of 66 MHz and that the device needs a 1,024-cycle refresh every 10 µs.

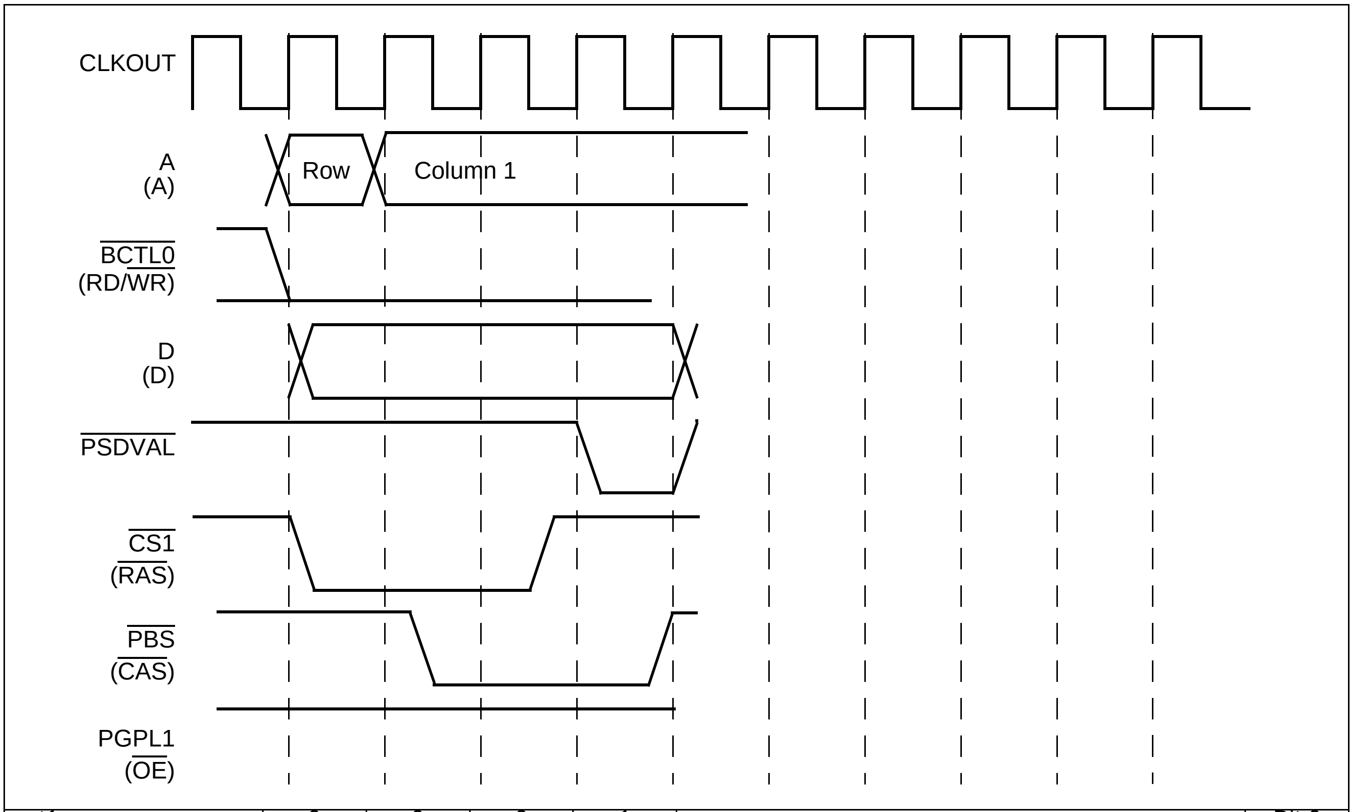
Table 12-30. EDO Connection Field Value Example

Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect	BR*x*[WP]	0b0
Refresh timer prescaler	MPTPR	0x04
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x07
Refresh timer enable	MAMR[RFEN]	0b1
Address multiplex size	MAMR[AMx]	0b001
Disable timer period	MAMR[DSx]	0b10
Burst inhibit device	OR*x*[BI]	0b0


CLKOUT
A
(A)
Row
Column 1
$\overline{\text{BCTL0}}$
(RD/$\overline{\text{WR}}$)
D
(D)
$\overline{\text{PSDVAL}}$
$\overline{\text{CS1}}$
($\overline{\text{RAS}}$)
$\overline{\text{PBS}}$
($\overline{\text{CAS}}$)
PGPL1
($\overline{\text{OE}}$)

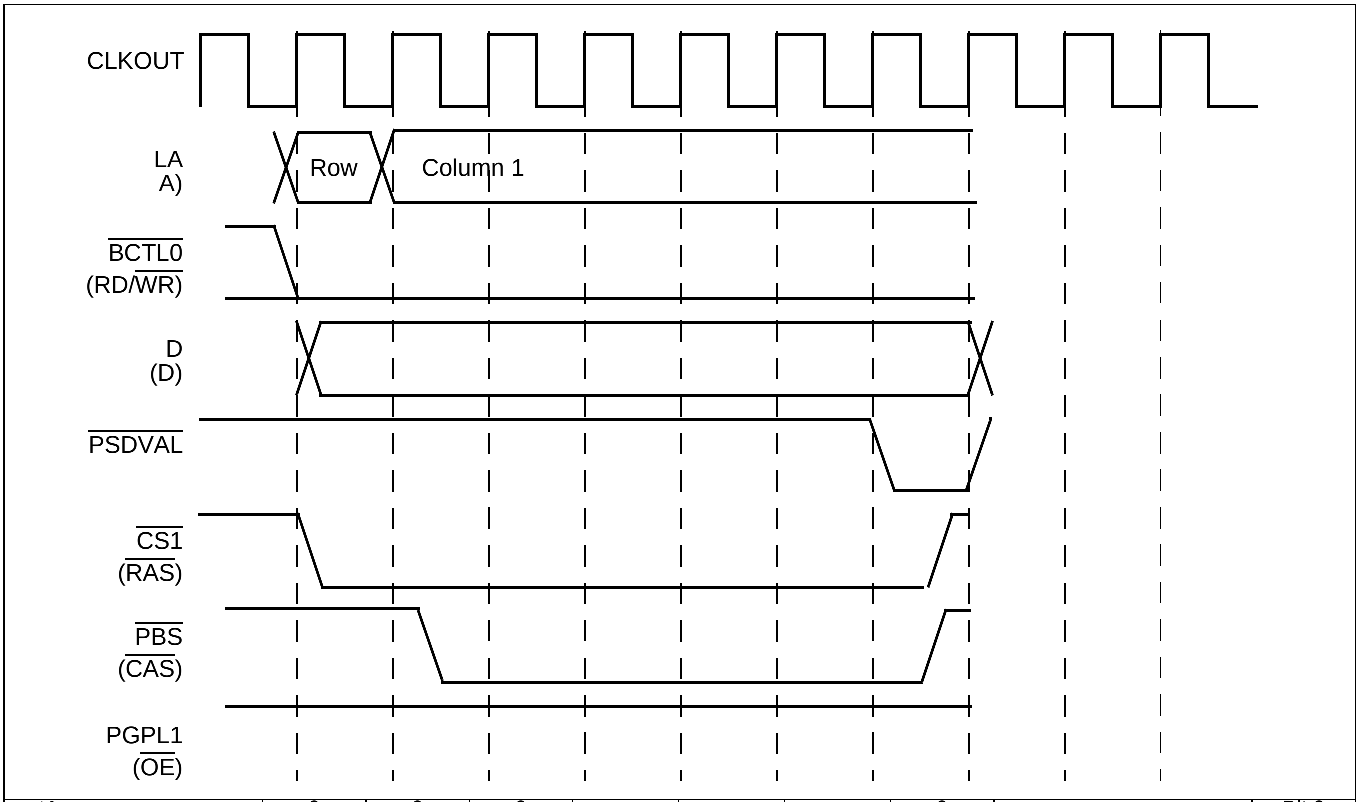
cst1	0	0	0	0	0		Bit 0
cst2	0	0	0	0	0		Bit 1
cst3	0	0	0	0	1		Bit 2
cst4	0	0	0	0	1		Bit 3
bst1	1	1	0	0	0		Bit 4
bst2	1	0	0	0	0		Bit 5
bst3	1	0	0	0	1		Bit 6
bst4	1	0	0	0	1		Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1	0	0	0	0	0		Bit 12
g1t3	0	0	0	0	1		Bit 13
g2t1							Bit 14
g2t3							Bit 15
g3t1							Bit 16
g3t3							Bit 17
g4t1							Bit 18
g4t3							Bit 19
g5t1							Bit 20
g5t3							Bit 21
redo0							Bit 22
redo1							Bit 23
loop	0	0	0	0	0		Bit 24
exen	0	0	0	0	0		Bit 25
amx0	1	0	0	0	0		Bit 26
amx1	0	0	0	0	0		Bit 27
na	0	0	0	0	0		Bit 28
uta	0	0	0	0	1		Bit 29
todt	0	0	0	0	1		Bit 30
last	0	0	0	0	1		Bit 31
	RSS	RSS+1	RSS+2	RSS+3	RSS+4		

Figure 12-66. Single-Beat Read Access to EDO DRAM



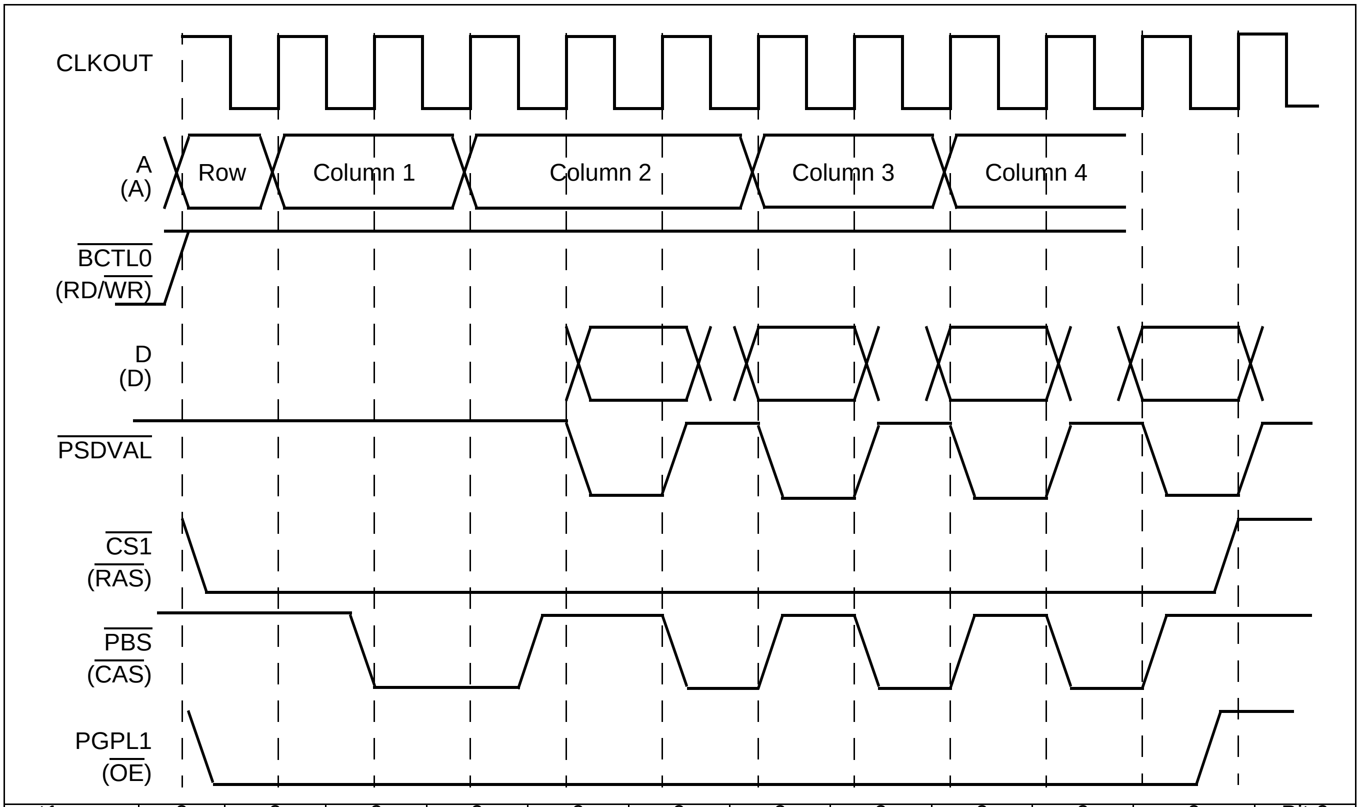
cst1	0	0	0	1		Bit 0
cst2	0	0	0	1		Bit 1
cst3	0	0	1	1		Bit 2
cst4	0	0	1	1		Bit 3
bst1	1	1	0	0		Bit 4
bst2	1	0	0	0		Bit 5
bst3	1	0	0	0		Bit 6
bst4	1	0	0	1		Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1	1	1	1		Bit 12
g1t3	1	1	1	1		Bit 13
g2t1						Bit 14
g2t3						Bit 15
g3t1						Bit 16
g3t3						Bit 17
g4t1						Bit 18
g4t3						Bit 19
g5t1						Bit 20
g5t3						Bit 21
redo0						Bit 22
redo1						Bit 23
loop	0	0	0	0		Bit 24
exen	0	0	0	0		Bit 25
amx0	1	0	0	0		Bit 26
amx1	0	0	0	0		Bit 27
na	0	0	0	0		Bit 28
uta	0	0	0	1		Bit 29
todt	0	0	0	1		Bit 30
last	0	0	0	1		Bit 31
	WSS	WSS+1	WSS+2	WSS+3		

Figure 12-67. Single-Beat Write Access to EDO DRAM



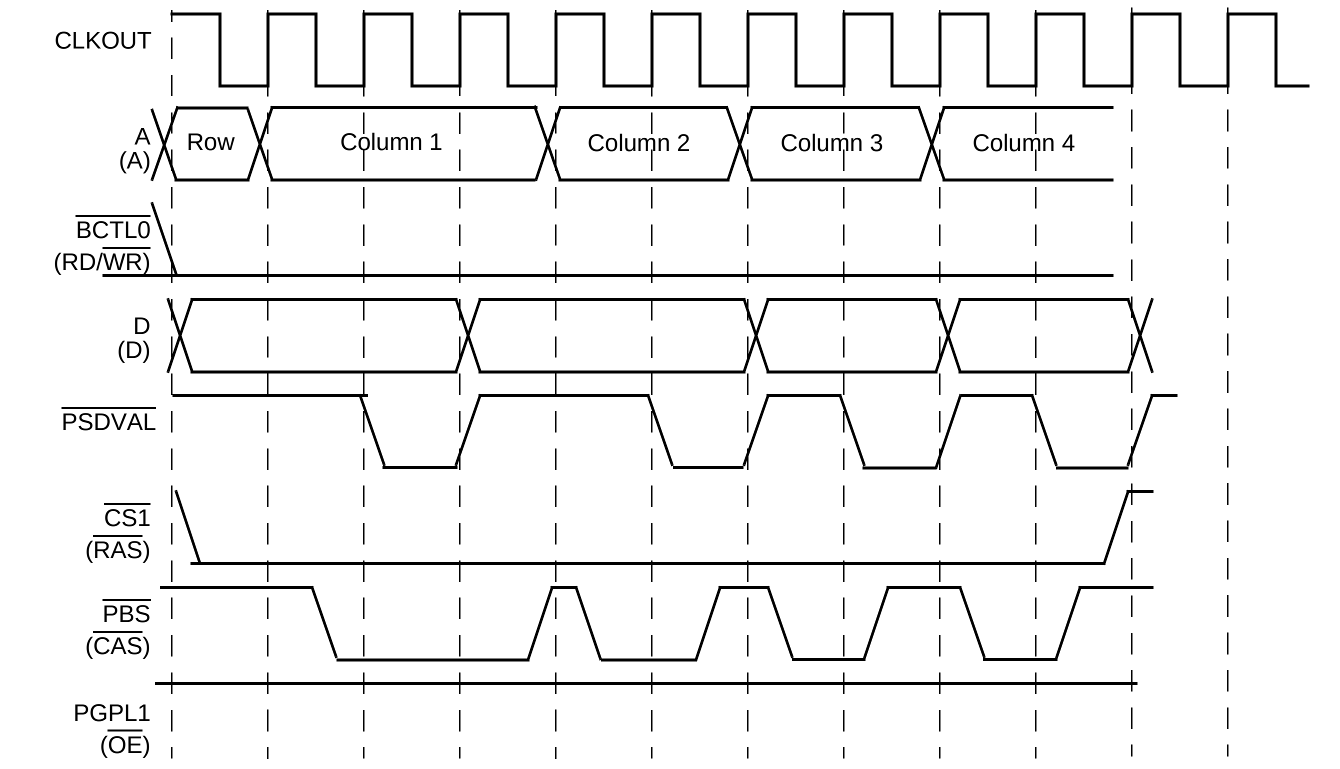
cst1	0	0	0				0		Bit 0
cst2	0	0	0				0		Bit 1
cst3	0	0	0				1		Bit 2
cst4	0	0	0				1		Bit 3
bst1	1	1	0				0		Bit 4
bst2	1	0	0				0		Bit 5
bst3	1	0	0				1		Bit 6
bst4	1	0	0				1		Bit 7
g0l0									Bit 8
g0l1									Bit 9
g0h0									Bit 10
g0h1									Bit 11
g1t1	1	1	1				1		Bit 12
g1t3	1	1	1				1		Bit 13
g2t1									Bit 14
g2t3									Bit 15
g3t1									Bit 16
g3t3									Bit 17
g4t1									Bit 18
g4t3									Bit 19
g5t1									Bit 20
g5t3									Bit 21
redo0	0	0	1				0		Bit 22
redo1	0	0	1				0		Bit 23
loop	0	0	0				0		Bit 24
exen	0	0	0				0		Bit 25
amx0	1	0	0				0		Bit 26
amx1	0	0	0				0		Bit 27
na	0	0	0				0		Bit 28
uta	0	0	0				1		Bit 29
todt	0	0	0				1		Bit 30
last	0	0	0				1		Bit 31
	WSS	WSS+1	WSS+2	REDO1	REDO2	REDO3	WSS+3		

Figure 12-68. Single-Beat Write to EDO DRAM, REDO Inserts Three Wait States



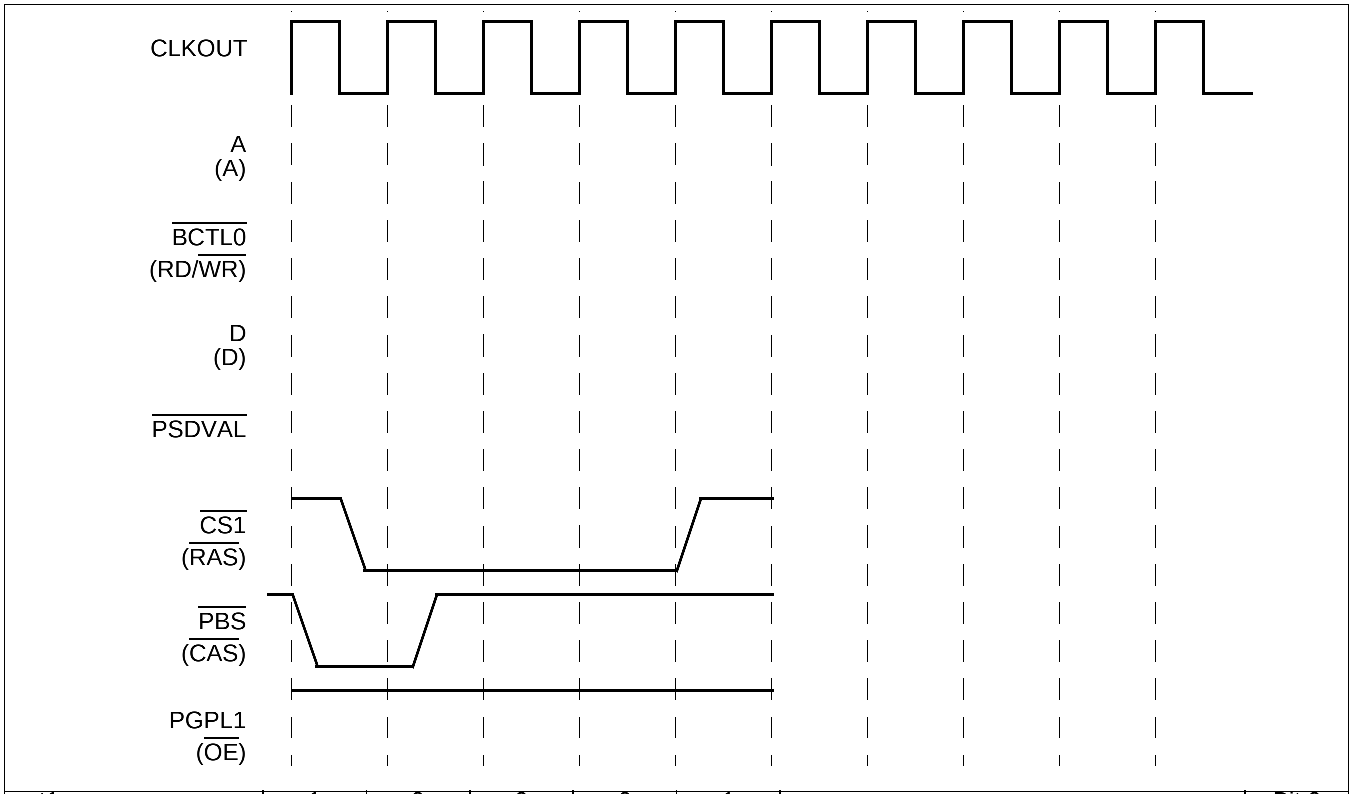
cst1	0	0	0	0	0	0	0	0	0	0	0	Bit 0
cst2	0	0	0	0	0	0	0	0	0	0	0	Bit 1
cst3	0	0	0	0	0	0	0	0	0	0	0	Bit 2
cst4	0	0	0	0	0	0	0	0	0	0	1	Bit 3
bst1	1	1	0	0	1	0	1	0	1	0	1	Bit 4
bst2	1	1	0	0	1	0	1	0	1	0	1	Bit 5
bst3	1	1	0	1	1	0	1	0	1	0	1	Bit 6
bst4	1	0	0	1	1	0	1	0	1	0	1	Bit 7
g0l0												Bit 8
g0l1												Bit 9
g0h0												Bit 10
g0h1												Bit 11
g1t1	0	0	0	0	0	0	0	0	0	0	0	Bit 12
g1t3	0	0	0	0	0	0	0	0	0	0	1	Bit 13
g2t1												Bit 14
g2t3												Bit 15
g3t1												Bit 16
g3t3												Bit 17
g4t1												Bit 18
g4t3												Bit 19
g5t1												Bit 20
g5t3												Bit 21
redo0												Bit 22
redo1												Bit 23
loop	0	0	0	0	0	0	0	0	0	0	0	Bit 24
exen	0	0	0	1	0	1	0	1	0	0	0	Bit 25
amx0	1	0	0	0	0	0	0	0	0	0	0	Bit 26
amx1	0	0	0	0	0	0	0	0	0	0	0	Bit 27
na	0	0	1	0	0	1	0	1	0	0	0	Bit 28
uta	0	0	0	0	1	0	1	0	1	0	1	Bit 29
todt	0	0	0	0	0	0	0	0	0	0	1	Bit 30
last	0	0	0	0	0	0	0	0	0	0	1	Bit 31
	RBS	RBS+1	RBS+2	RBS+3	RBS+4	RBS+5	RBS+6	RBS+7	RBS+8	RBS+9	RBS+ 10	

Figure 12-69. Burst Read Access to EDO DRAM



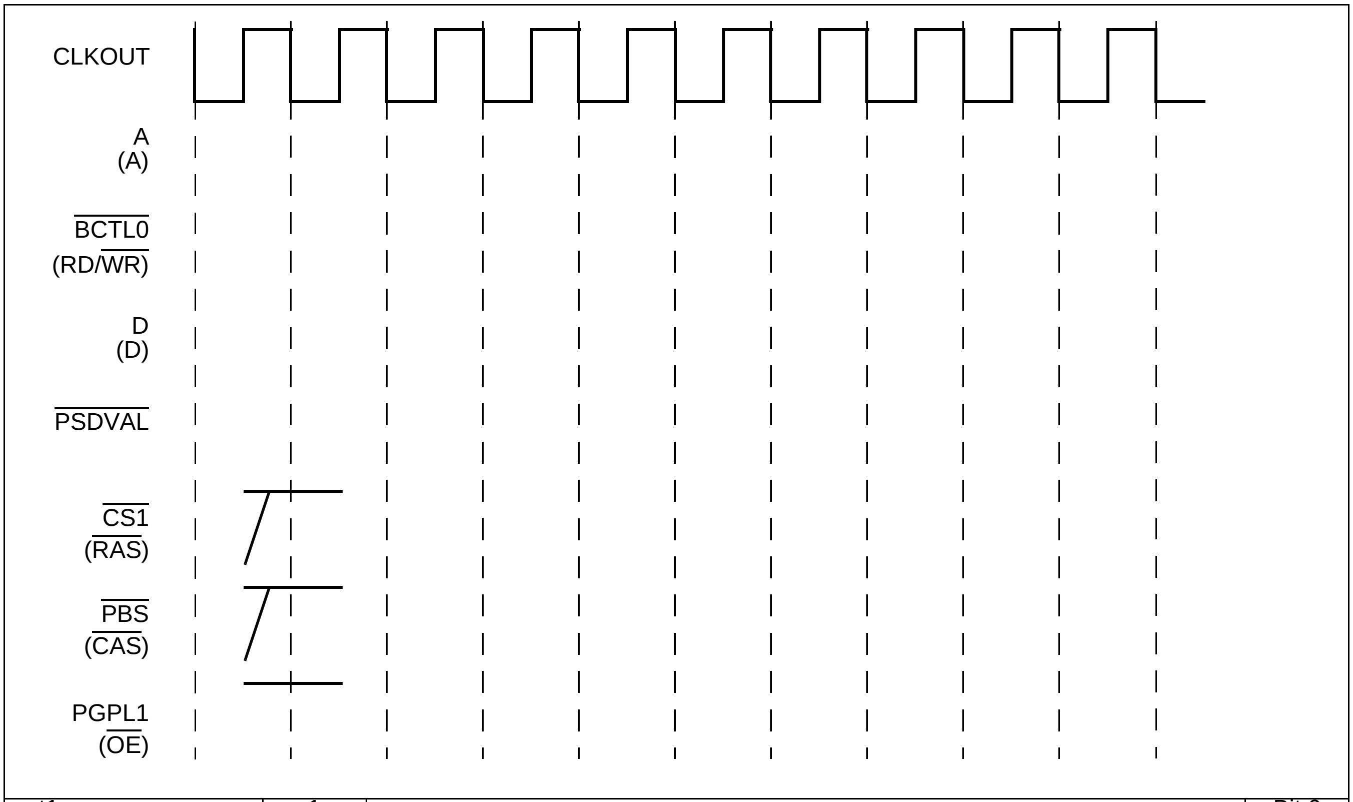
cst1 cst2 cst3 cst4	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1		Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	1 1 0 0	0 0 0 0	0 0 0 1	1 0 0 0	0 0 1 1	1 0 0 0	0 1 1 1	1 0 0 0	0 1 1 1		Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1												Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1		Bit 12 Bit 13
g2t1 g2t3												Bit 14 Bit 15
g3t1 g3t3												Bit 16 Bit 17
g4t1 g4t3												Bit 18 Bit 19
g5t1 g5t3												Bit 20 Bit 21
redo0 redo1												Bit 22 Bit 23
loop exen	0 0	0 0	0 0	0 1	0 0	0 1	0 0	0 1	0 0	0 0		Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0	0 0 0 0 0 0	0 0 0 1 0 0	0 0 1 0 0 0	0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 0 0	0 0 0 1 1 1		Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	WBS	WBS+ 1	WBS+ 2	WBS+ 3	WBS+ 4	WBS+ 5	WBS+ 6	WBS+ 7	WBS+ 8	WBS+ 9		

Figure 12-70. Burst Write Access to EDO DRAM



cst1	1	0	0	0	1		Bit 0
cst2	1	0	0	0	1		Bit 1
cst3	0	0	0	0	1		Bit 2
cst4	0	0	0	0	1		Bit 3
bst1	0	0	1	1	1		Bit 4
bst2	0	1	1	1	1		Bit 5
bst3	0	1	1	1	1		Bit 6
bst4	0	1	1	1	1		Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1	1	1	1	1	1		Bit 12
g1t3	1	1	1	1	1		Bit 13
g2t1							Bit 14
g2t3							Bit 15
g3t1							Bit 16
g3t3							Bit 17
g4t1							Bit 18
g4t3							Bit 19
g5t1							Bit 20
g5t3							Bit 21
redo0							Bit 22
redo1							Bit 23
loop	0	0	0	0	0		Bit 24
exen	0	0	0	0	0		Bit 25
amx0	0	0	0	0	0		Bit 26
amx1	0	0	0	0	0		Bit 27
na	0	0	0	0	0		Bit 28
uta	0	0	0	0	0		Bit 29
todt	0	0	0	0	1		Bit 30
last	0	0	0	0	1		Bit 31
	PTS	PTS+1	PTS+2	PTS+3	PTS+4		

Figure 12-71. Refresh Cycle (CBR) to EDO DRAM



cst1	1		Bit 0
cst2	1		Bit 1
cst3	1		Bit 2
cst4	1		Bit 3
bst1	1		Bit 4
bst2	1		Bit 5
bst3	1		Bit 6
bst4	1		Bit 7
g0l0			Bit 8
g0l1			Bit 9
g0h0			Bit 10
g0h1			Bit 11
g1t1	1		Bit 12
g1t3	1		Bit 13
g2t1			Bit 14
g2t3			Bit 15
g3t1			Bit 16
g3t3			Bit 17
g4t1			Bit 18
g4t3			Bit 19
g5t1			Bit 20
g5t3			Bit 21
redo0			Bit 22
redo1			Bit 23
loop	0		Bit 24
exen	0		Bit 25
amx0	0		Bit 26
amx1	0		Bit 27
na	0		Bit 28
uta	0		Bit 29
todt	1		Bit 30
last	1		Bit 31
	EXS		

Figure 12-72. Exception Cycle for EDO DRAM

12.4.7 Differences Between MPC8xx UPM and MSC8113 UPM

Users familiar with the MPC8xx UPM should know about the major differences between the MPC8xx devices and the MSC8113 device:

- *First cycle timing transferred to the UPM array*. In the MPC8xx UPM, the first cycle value of some signals is determined from OR*x*[SAM,G5LA,G5LS]. In the MSC8113, all signals are controlled only by the pattern written to the array.
- *Timing of PGPL[0–5]*. In the MPC8xx, the PGPL lines could change on the positive edge of T2 or T3. In the MSC8113, these signals can change on the positive edge of T1 or T3 to allow connection to high-speed synchronous devices such as burst SRAM.
- *UPM controlled signals deasserted at end of an access*. In the MPC8xx UPM, if you do not deassert the UPM signals at the end of an access, those signals keep their previous value. In the MSC8113, all UPM signals are deasserted ($\overline{\text{CS}}$, $\overline{\text{BS}}$, PGPL[0–4] driven to logic 1 and PGPL5 driven to logic 0) at the end of that cycle, unless there is a back-to-back UPM cycle pending. In many cases this allows the UPM routine to finish one cycle earlier because it is now possible and preferable to assert both UTA and LAST.
- *MCR is eliminated*. In the MSC8113, MCR is eliminated. The function of RAM read/write and RUN occurs via the M*x*MR.
- *UTA polarity is reversed*. In the MSC8113, UTA is active high.
- *TODT signal*. The disable timer control (TODT) and LAST bit in the RAM array word must be set together. Otherwise, TODT is ignored.
- *Refresh timer value is in a separate register*. In the MSC8113, the refresh timer value has moved to the system bus Assigned UPM Refresh Timer Register (PURT), which can serve multiple UPMs.
- *System bus refresh*. Refresh on the system bus must occur in UPMA.
- *New feature*. Repeated execution of the current RAM word (REDO).
- *Extended hold time*. Extended hold time on reads can be up to eight clock cycles instead of one as in the MPC8xx.

12.5 Handling Devices With Slow or Variable Access Times

The memory controller provides two ways to interface with slave devices that are very slow (access time is greater than the maximum allowed by the user-programming model) or cannot guarantee a predefined access time (for example some FIFO, hierarchical bus interface, or dual-port memory devices):

- *Wait mechanism*. Used only in accesses controlled by the UPM. Setting M*x*MR[GPL_*x*4DIS] enables this mechanism.
- *External termination ($\overline{\text{PGTA}}$) mechanism*. Used only in accesses controlled by the GPCM. OR*x*[SETA] specifies whether the access is terminated internally or externally.

The following examples show how the two mechanisms work.

12.5.1 Hierarchical Bus Interface Example

Assume that one of the SC140 cores initiates a system bus read cycle that addresses the DSI of another MSC8113. The programmer cannot predict when the SC140 core can latch valid data because the internal local bus of another MSC8113 may be occupied (by the DMA controller, for example).

- *The wait solution (UPM)*. The external device (DSI of another MSC8113) asserts PUPMWAIT to the memory controller to indicate that data is not ready. The memory controller synchronizes this signal because the wait signal is asynchronous. As a result of the wait signal assertion, the UPM enters a freeze mode at the rising edge of CLKOUT upon encountering the WAEN bit being set in the UPM word. The UPM stays in freeze mode until PUPMWAIT is deasserted, and then continues executing from the next entry to the end of the pattern (LAST bit is set).
- *The external termination solution (GPCM)*. The external device (DSI of another MSC8113) asserts $\overline{\text{PGTA}}$ to the memory controller when it can sample data. Note that $\overline{\text{PGTA}}$ is also synchronized.

12.5.2 Slow Devices Example

When an SC140 core initiates a read cycle from a device with an access time that exceeds the maximum allowed by the user programming model, there are two solutions:

- *The wait solution (UPM)*. The SC140 core generates a read access from the slow device. The device in turn asserts the wait signal until the data is ready. The SC140 core samples data only after the wait signal is deasserted.
- *The external termination solution (GPCM)*. The SC140 core generates a read access from the slow device, which must generate the asynchronous $\overline{\text{PGTA}}$ when it is ready.

12.6 External Master Support (60x-Compatible Mode)

The memory controller supports internal and external 60x-compatible bus masters. Accesses from the MSC8113 internal system bus master are internal while accesses from an external bus master are considered external. External bus master support is available only if the BCR[EBM] bit is set. For strict 60x-compatible mode the BCR[ETM] bit must be clear; For further details see **Section 4.2**, *SIU Programming Model*. There are two types of external bus masters.

- Strict 60x-compatible device using a 64-bit data bus, such as MPC603e, MPC604e, MPC750.
- MSC8113-type devices, such as an MPC8260, MSC8101, MSC8102 or another MSC8113.

12.6.1 Strict 60x-Compatible External Masters

Any 60x-compatible devices that use a 64-bit data bus can access the MSC8113 internal registers and local bus. These devices can also use memory controller services under the following restrictions, which apply only to system bus-assigned memory banks accessed by the external device:

- 64-bit port size only
- No ECC or RMW-parity
- No data pipelining

For 60x bus compatibility, the following connections should be observed:

- Connect MSC8113 TSZ[1–3] to the external master TSZ[0–2]
- Pull down MSC8113 TSZ0
- Pull up MSC8113 $\overline{\text{PSDVAL}}$

12.6.2 MSC8113-Type External Masters

An MSC8113-type external master is a 60x-compatible master with additional functionality. It has fewer restrictions than other 60x-compatible masters:

- Any port size (64, 32, 16, 8)
- ECC and RMW-parity
- Data pipelining

12.6.3 Extended Controls in 60x-Compatible Mode

In 60x-compatible mode, the memory controller provides extended controls for the glue logic. The extended controls consist of the following:

- Memory address latch (ALE) to latch the system bus address for memory use
- The address multiplex signal (PGPL5/PSDAMUX), which controls external multiplexing for DRAM and SDRAM devices. PSDAMUX is also available externally in single MSC8113 mode.
- LSB address lines (BADDR[27–31]) for incrementing memory addresses
- $\overline{\text{PSDVAL}}$ as a termination to a partial transaction (such as port-size beat access).
- Internal SDRAM bank selects (BNKSEL[0–2]) to allow SDRAM bank interleaving, as described in **Section 12.2.6**, *BNKSEL Signals in Single-MSC8113 Bus Mode*.

12.6.4 Address Incrementing for External Bursting Masters

In external master mode the address bus has several masters which causes the current transaction address to be valid only during the 60x address phase. Therefore, for generating burst accesses in external master mode (BCR[EBM] bit is set), two extended signals are used. The ALE signal is used to control an external latch which samples all address bits from A[0–26] needed for current transaction generation, and BADDR[27-31] is used to generate the lsb address bits to the memory devices in burst accesses. In external master mode, when a master initiates an external 60x-compatible bus transaction, BADDR[27–31] reflects the value of A[27–31] on the first clock cycle of the memory access (when ALE was asserted). The memory controller latches these signals, and on subsequent clock cycles, BADDR[27–31] increments as programmed in the UPM or after each data beat is sampled in the GPCM or after each READ/WRITE command in the SDRAM machine (the SDRAM machine uses BADDR only for port sizes of 16 or 8 bits).

Note: BADDR[27–31] signals are multiplexed with other signals (see **Section 3.4**, *Direct Slave Interface, System Bus, Ethernet, and Interrupt Signals,* on page 3-4). In cases where BADDR[27–31] are not valid, external latching and incrementing logic must be use for burst address generation.

12.6.5 External Masters Timing

External and internal masters have similar memory access timings. However, because more time is required to decode the addresses of external masters, memory accesses by external masters start one cycle later than those of internal masters. As soon as the external master asserts $\overline{\text{TS}}$, the memory controller compares the address with each of its defined valid banks. If a match is found, the memory controller asserts the address latch enable (ALE) and control signals to the memory devices. In low bus frequencies, the additional cycle which is used for external master access address decoding can be eliminated by setting the BCR[EXDD] bit (see **Table 4-3**).

Figure 12-73 shows the one-cycle delay for external master access. For systems that use the system bus with low frequency, the one-cycle delay for external masters can be eliminated by setting BCR[EXDD].

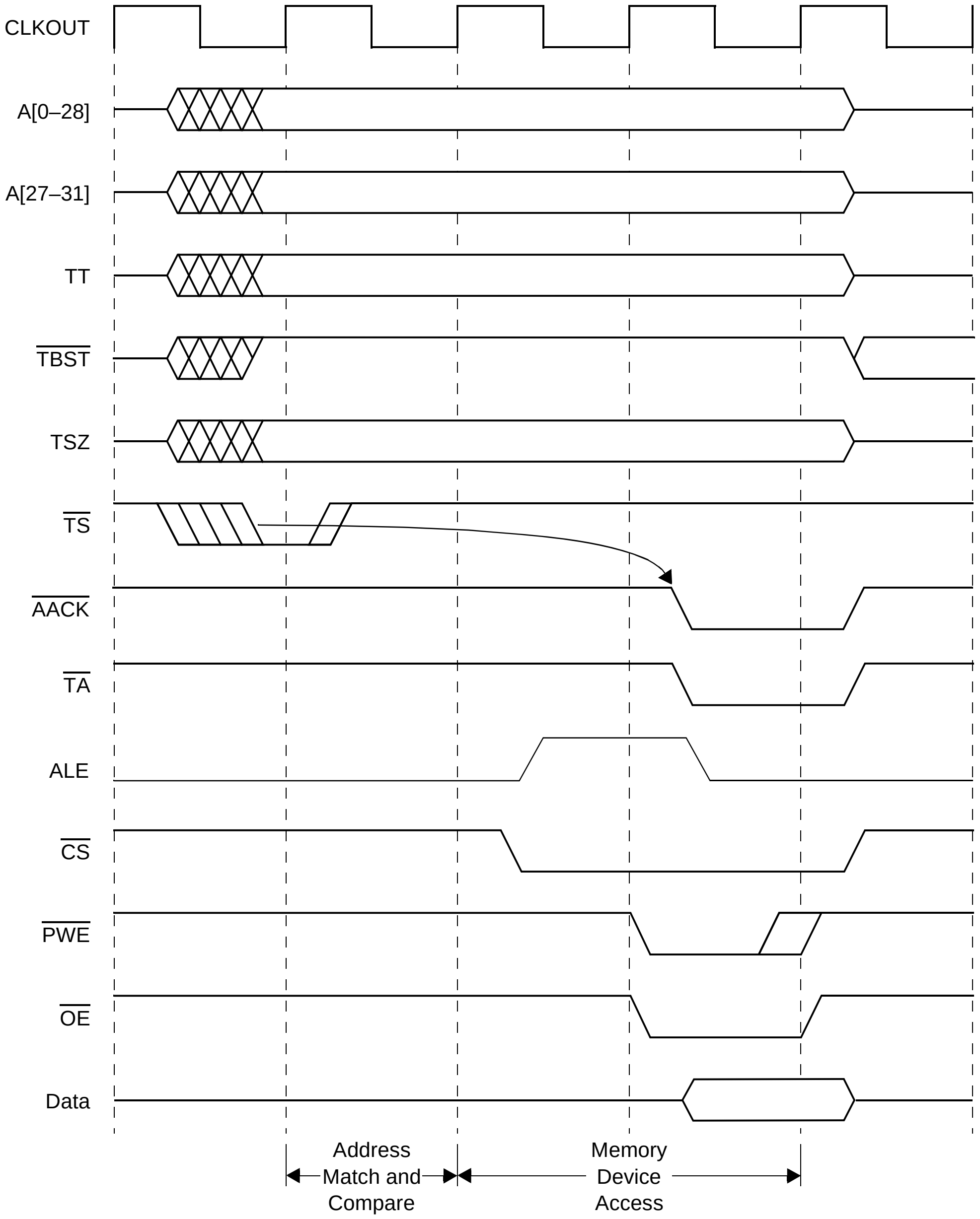


Figure 12-73. External Master Access (GPCM)

The memory controller asserts $\overline{\text{PSDVAL}}$ for each data beat as to indicate data beat termination on write transactions and data valid on read transactions.

Figure 12-74 shows an interconnection in which a 60x-compatible external master and the MSC8113 can share access to a 1MB 64-bit port SDRAM device. Note that the address multiplexer is controlled by PSDAMUX, while the address latch is controlled by ALE. Also, because the 64-bit port SDRAM has burst address increment logic, BADDR is not needed.

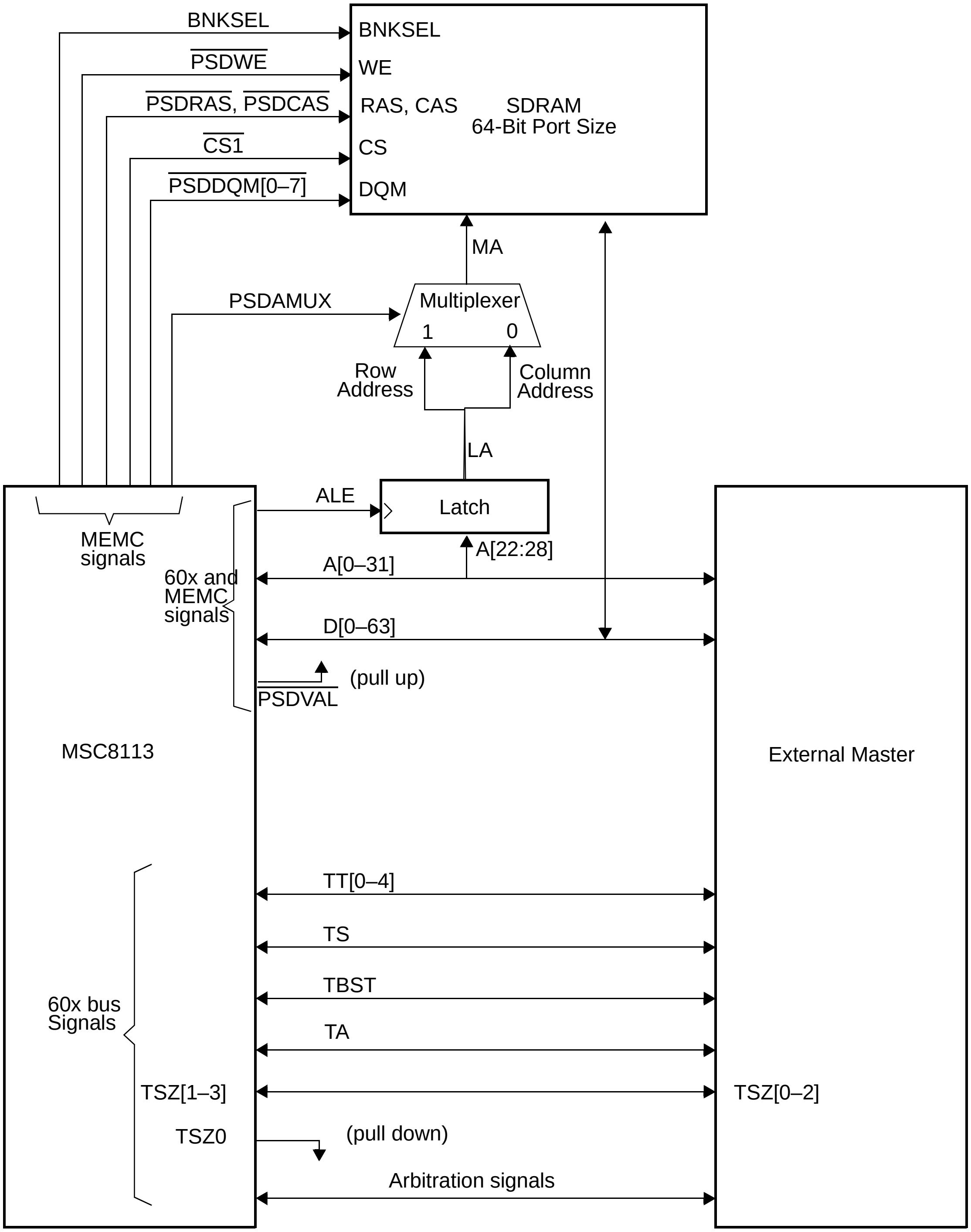


Figure 12-74. External Master Configuration With SDRAM Device (Example)

Figure 12-75 shows an example of a 60x-compatible mode write access to the shared SDRAM using the memory controllers SDRAM machine.

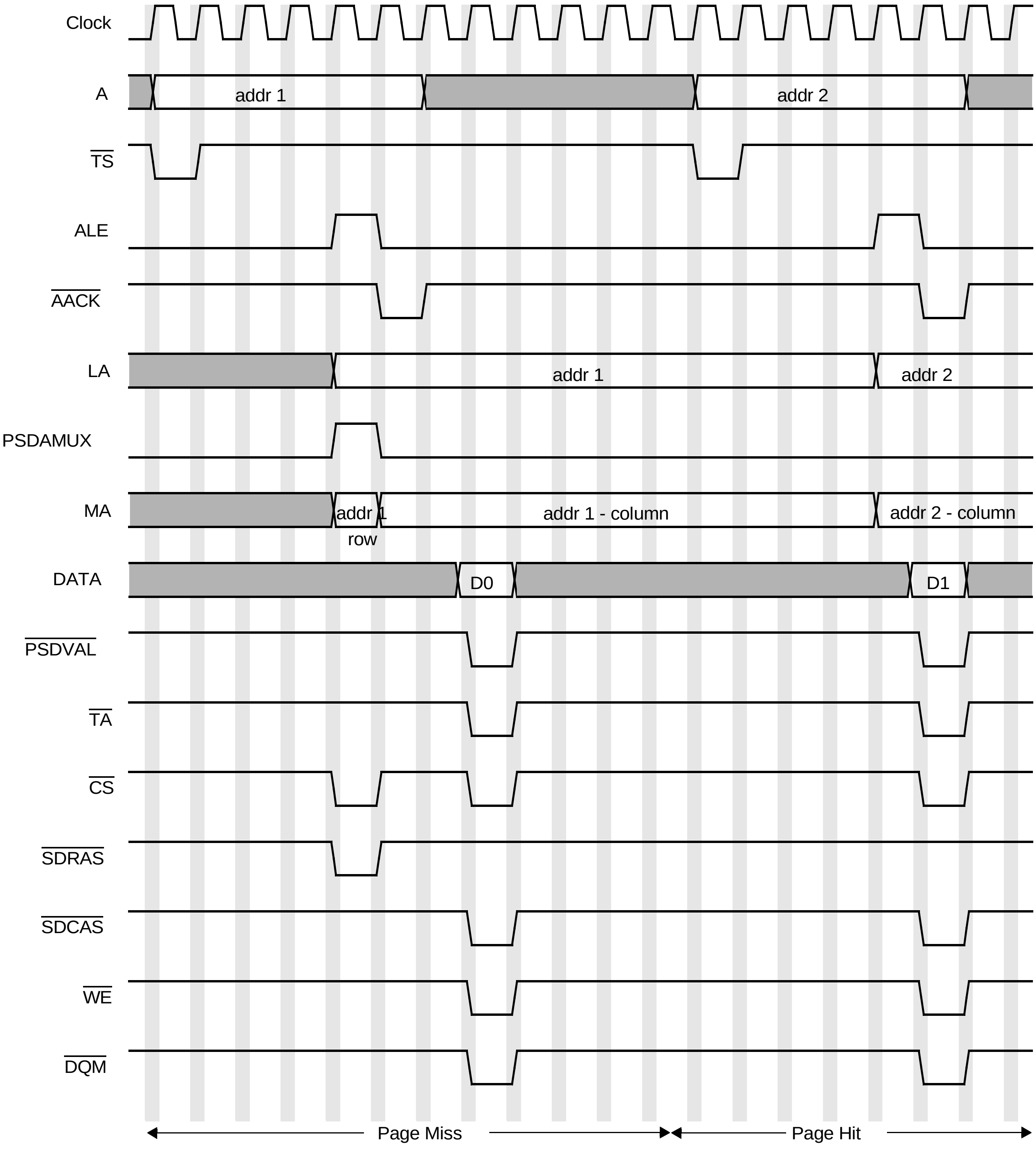


Figure 12-75. 60x-Compatible Mode SDRAM Access

Figure 12-76 shows an interconnection in which a 60x-compatible external master and the MSC8113 can share access to an external 1 MB 64-bit port device using the UPM memory controller machines. Note that while the address latch controlled by ALE latches A[0–26], the lsb of the address are driven by the BADDR[27–31].

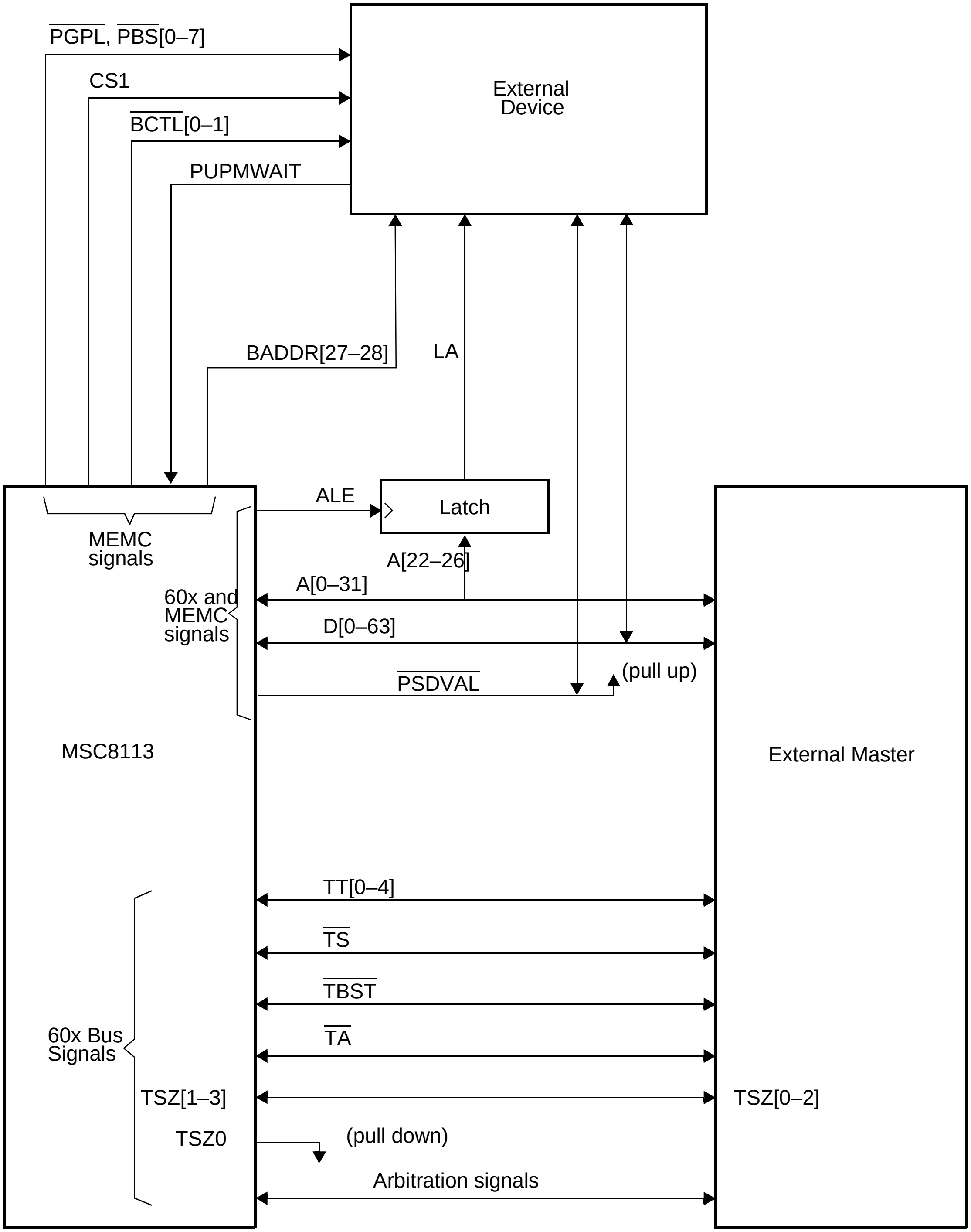
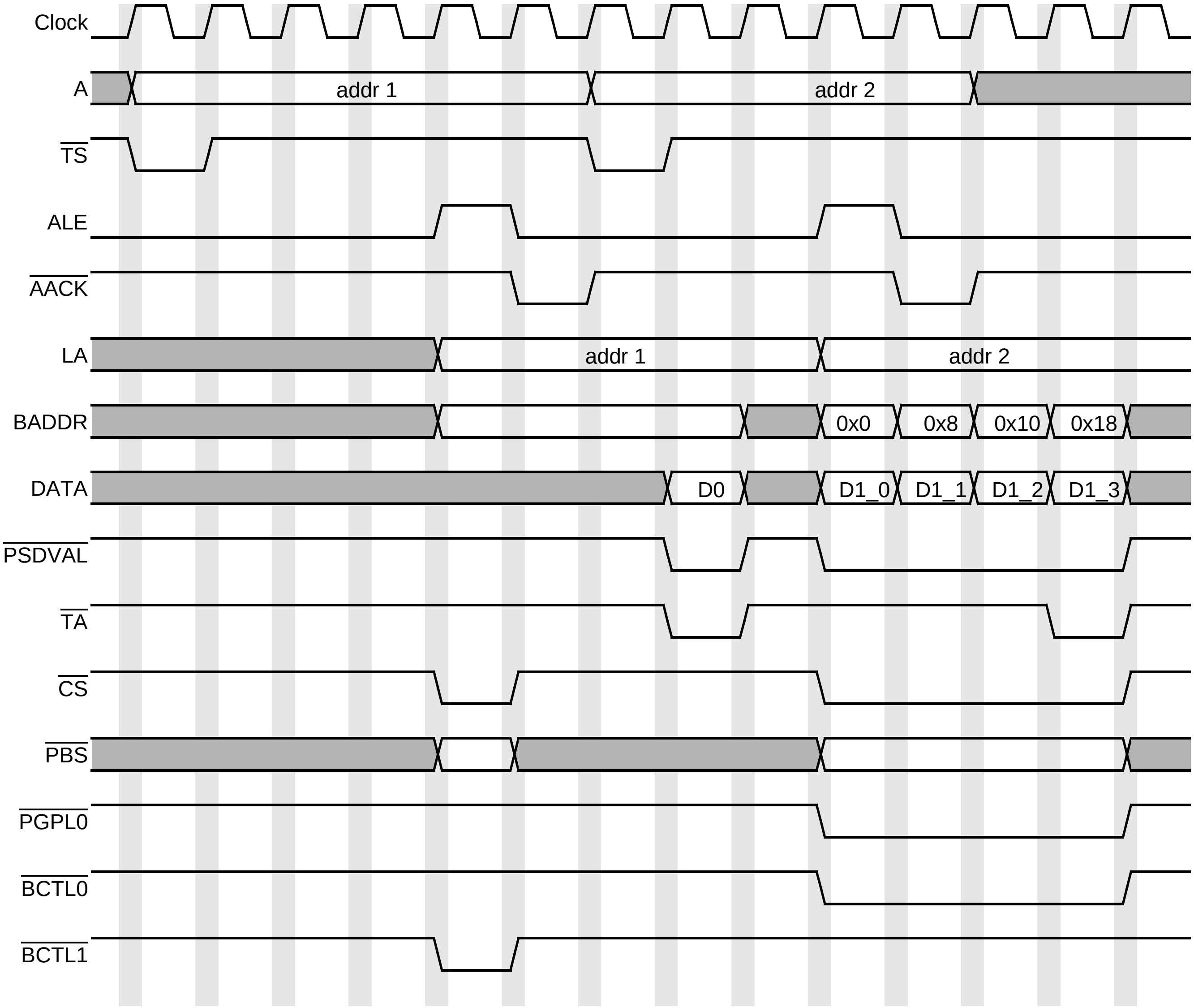


Figure 12-76. External Master Configuration With UPM (Example)

Figure 12-77 shows an example of a 60x-compatible mode read access followed by burst write access to a SRAM like device with write latency=0 cycles and read latency=3 cycles, using the memory controller UPM machine.



Note: PGPL0 is a general-purpose line using as an indicator for burst accesses.

Figure 12-77. 60x-Compatible Mode UPM Access

Figure 12-78 shows an interconnection in which a 60x-compatible external master and the MSC8113 can share access to an external 1 MB 64-bit port device using the GPCM memory controller machines. Note that the address latch controlled by ALE latches A[0–31].

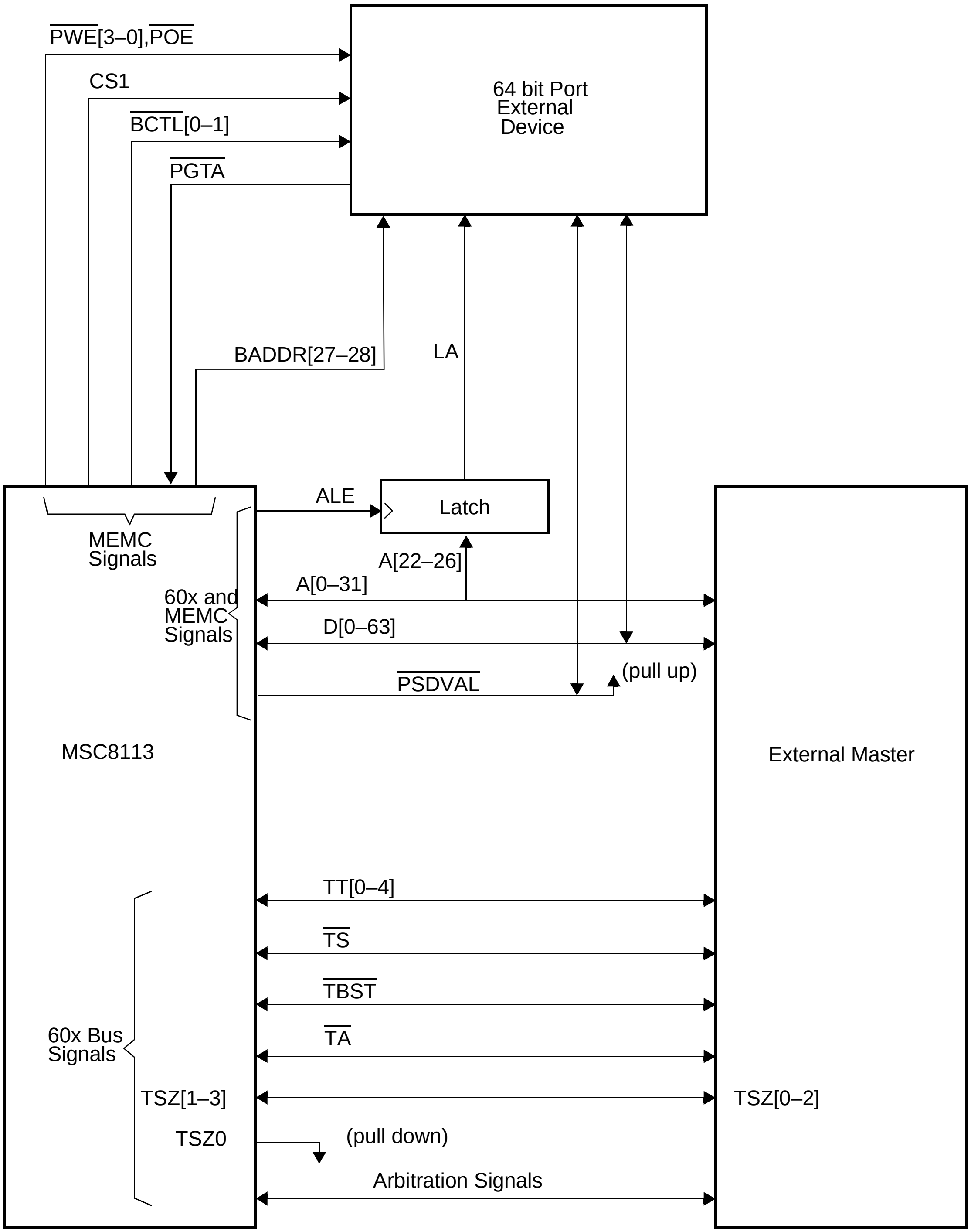


Figure 12-78. External Master Configuration With GPCM (Example)

Figure 12-79 shows an example of a 60x-compatible mode write follow by read, using the memory controllers GPCM machine.

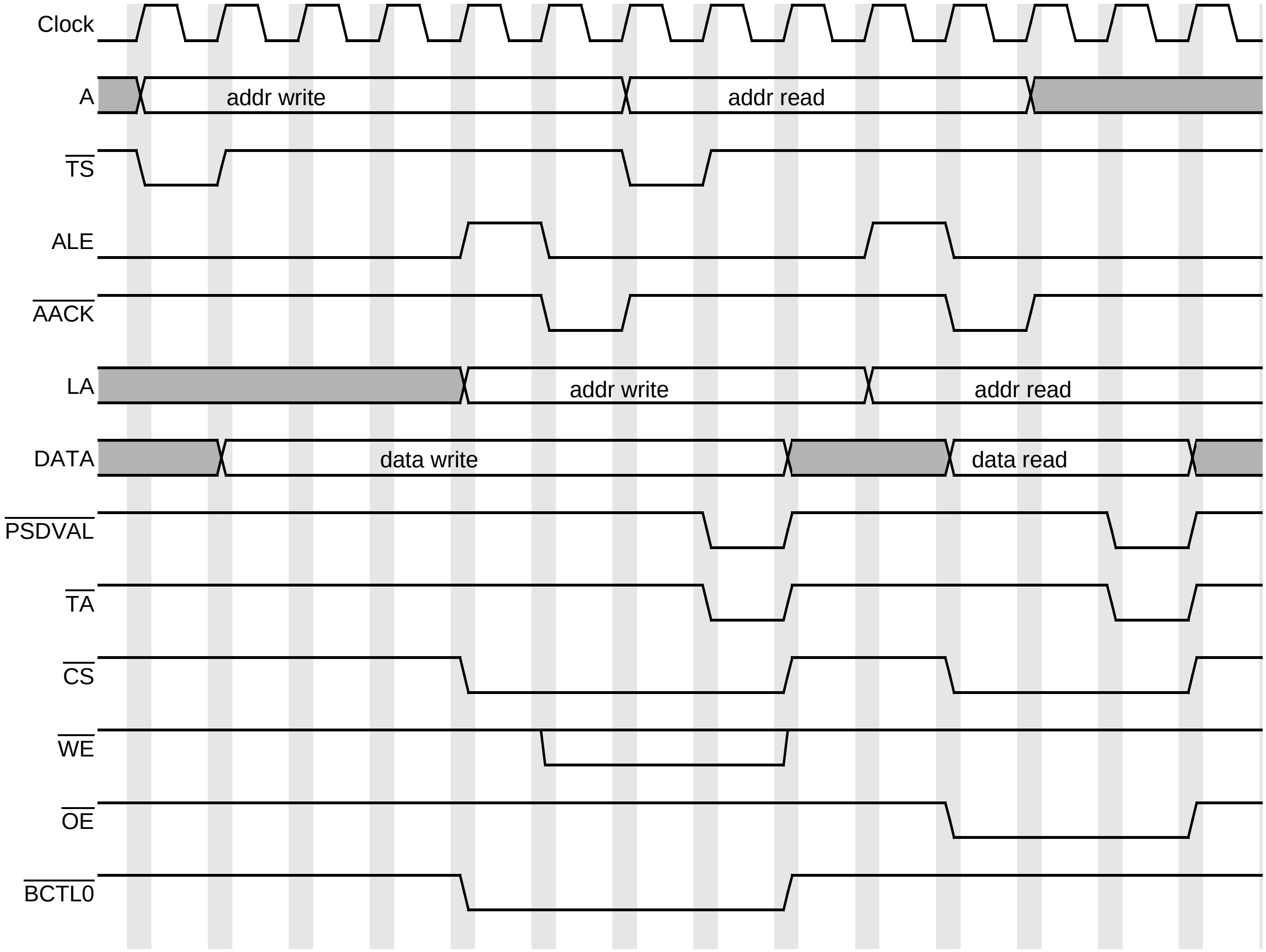


Figure 12-79. 60x-Compatible Mode GPCM Access

Note: A burst access is split in to several single GPCM accesses, with BADDR incrementing between the accesses.

12.7 Internal SRAM and IPBus Peripherals Support

The memory controller handles the IPBus peripherals and internal memories exactly like any other devices or memories. The internal SRAM is accessed by the UPMC and mapped to $\overline{\text{CS11}}$. The IPBus peripherals are accessed by the GPCM and mapped to $\overline{\text{CS9}}$.

12.7.1 UPM Programming Example — Internal SRAM

The SRAM is accessed via the UPMC on the local bus. The code below is an example of UPM programming. In this example, the notation is based on the following:

- You can change the base address.
- The r5 register holds the base address of the SRAM.
- BR11, OR11, MDR, MCMR are the addresses of the UPM registers.

```
   move.l #$020000c1,d1 ; base address for SRAM is 0x02000000
   move.l #$ffe00000,d7 ;
   move.l d7,OR11 ; SET bank size to 2MB
   move.l d1,BR11 ; SET bank to UPMC
;; -------------- READ SINGLE ------------------------------
   move.l #$90051240,d7 ;
   move.l d7,MCMR ;

   move.l #$00030040,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00030045,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;
;; -------------- READ BURST ------------------------------------
   move.l #$90051248,d7 ;
   move.l d7,MCMR ;

   move.l #$00030c48,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00030c4c,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00030c4c,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00030044,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00030045,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;
;; -------------- WRITE SINGLE -----------------------------------
   move.l #$90051258,d7 ;
   move.l d7,MCMR ;

   move.l #$00000040,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;

   move.l #$00000045,d7 ;
   move.l d7,MDR ;
   move.w #$0,(r5) ;
;; -------------- WRITE BURST ------------------------------------
   move.l #$90051260,d7 ;
```

```
    move.l d7,MCMR ;

    move.l #$00000c48,d7 ;
    move.l d7,MDR ;
    move.w #$0,(r5) ;

    move.l #$00000c4c,d7 ;
    move.l d7,MDR ;
    move.w #$0,(r5) ;

    move.l #$00000c4c,d7 ;
    move.l d7,MDR ;
    move.w #$0,(r5) ;

    move.l #$00000044,d7 ;
    move.l d7,MDR ;
    move.w #$0,(r5) ;

    move.l #$00000045,d7 ;
    move.l d7,MDR ;
     move.w #$0,(r5) ;
;; -------------- EXCEPTION ------------------------------------
    move.l #$9005127c,d7 ;
    move.l d7,MCMR ;

    move.l #$ff000001,d7 ;
    move.l d7,MDR ;
    move.w #$0,(r5) ;
;; -------------- RESUME NORMAL OPERATION -------------------------
    move.l #$80011240,d7 ;
    move.l d7,MCMR ;
```

12.7.2 GPCM Programming Example, IPBus Peripherals

A data transfer through the internal interface to the IPBus peripherals can occur in only a single access. The following example shows how to program the GPCM for data transfer to the IPBus peripherals. In this example, the notation is based on the following:

- You can change the base address.
- BR9 and OR9 are the addresses of the GPCM registers.

```
    move.l #$02181821,d0 ; base address for IPBus peripherals is 0x02180000
    move.l #$fffc0008,d1 ;
    move.l d1,OR9 ; SET bank size to 256 KB
    move.l d0,BR9 ; SET bank to GPCM-local bus
```

12.7.3 Flyby Mode

Data can be transferred between internal memories using DMA Flyby mode. In this mode, the DMA controller, and not the memory controller, effects the transfer. For details on DMA flyby mode, refer to **Chapter 16**, *Direct Memory Access (DMA) Controller*.

12.8 Memory Controller Programming Model

This section describes the memory controller registers in detail. The registers discussed are listed as follows:

- Base Registers (BR[0–7, 9, 11]), **page 12-95**
- Option Registers —SDRAM Mode (OR[0–7, 9, 11]), **page 12-98**
- Option Registers —GPCM Mode (OR[0–7, 9, 11]), **page 12-101**
- Option Registers —UPM Mode (OR[0–7, 9, 11]), **page 12-103**
- System Bus SDRAM Mode Register (PSDMR), **page 12-104**
- UPM Machine Mode Registers (MAMR, MBMR, MCMR), **page 12-107**
- Memory Data Register (MDR), **page 12-110**
- Memory Address Register (MAR), **page 12-110**
- System Bus Assigned UPM Refresh Timer (PURT), **page 12-111**
- System Bus Assigned SDRAM Refresh Timer (PSRT), **page 12-111**
- Memory Refresh Timer Prescaler Register (MPTPR), **page 12-112**
- System Bus Error Status and Control Registers (TESCRx), **page 12-112**
- Local bus Error Status and Control Register (L_TESCR1), **page 12-112**

BR[0–7] Base Registers 0–7

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BA															
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
	BA	—		PS		DECC		WP	MS			EMEMC	ATOM		DR	V
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: After a system reset:
- the BA field has value 0b11111110000000000 in BR0 and 0b00000000000000000 in BR[1–7];
- the PS field has value of BPS (reset) in BR0 and 00 in BR[1–7];
- the EMEMC field has value of EXMC (reset) in BR0 and 0 in BR[1–7];
- the V field has value 1 in BR0 and 0 in BR[1–7].

BR9 Base Register 9

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	See **Table 8-7**, *Banks 9 and 11 Address Space,* on page 8-28															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	BA	—		PS		DECC		WP	MS			EMEMC	ATOM		DR	V
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1

BR11 Base Register 11

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	See **Table 8-7**, *Banks 9 and 11 Address Space,* on page 8-28															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	BA	—		PS		DECC		WP	MS			EMEMC	ATOM		DR	V
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

BR[0–7, 9, 11] contain the base address and address types that the memory controller uses to compare the address bus value with the current address accessed. Each register also includes a memory attribute and selects the machine for memory operation handling.

Note: When you write BR[9, 11] and OR[9, 11], you must also update the DSI Internal Base Address Registers (DIBAR*x*) and DSI Internal Address Mask Registers (DIAMR*x*) in the correct sequence. BR*x*[V] should be set only after OR*x* is programmed.

Table 12-31 describes the BR*x* fields.

Table 12-31. BR*x* Bit Descriptions

Name	Reset	Description	Settings
BA 0–16	0x0	**Base Address** The upper 17 bits of each base address register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. BR*x*[BA] is used with OR*x*[AM]. **Note:** After system reset, BR0[BA] is 0b11111110000000000.	
— 17–18	00	Reserved. Write to zero for future compatibility.	

Table 12-31. BR*x* Bit Descriptions (Continued)

Name	Reset	Description	Settings
PS 19–20	00	**Port Size** Specifies the port size of this memory region. **Note:** After system reset, BR0[PS] is set to the value of the BPS field from the HRCW.	01 8-bit 10 16-bit 11 32-bit 00 64-bit
DECC 21–22	00	**Data Error Correction and Checking** Specifies the method for data error checking and correction. See **Section 12.1**, *Basic Architecture*.	00 Data errors checking disabled 01 Normal parity checking 10 Read-modify-write parity checking 11 ECC correction and checking
WP 23	0	**Write Protect** Restricts write accesses within the address range of a BR. An attempt to write to this address range while WP = 1 can cause the bus monitor logic (if enabled) to assert $\overline{\text{TEA}}$, which terminates the cycle. When WP is set, the memory controller does not assert $\overline{\text{CS}x}$ and $\overline{\text{PSDVAL}}$ on write cycles to this memory bank. TESCR1[WP] or L_TESCR1[WP] is set if you attempt a write to this memory bank. **Notes:** **1.** Two banks should not overlap if one needs write protection and the other does not. **2.** Due to the existing overlap between Banks 9 and 11, the value of BR9[WP] should equal the value of BR11[WP] when Bank 11 is valid (that is, when BR11[V] = 1).	0 Read and write accesses are allowed. 1 Only read access is allowed.
MS 24–26	000	**Machine Select** Specifies machine select for the memory operations handling and assigns the bank to the system bus or local bus, if GPCM or SDRAM is selected. If UPM*x* is selected, M*x*MR[BSEL] determines the bus assignment.	000 GPCM—system bus (reset value) 001 GPCM—local bus 010 SDRAM—system bus 011 Reserved 100 UPMA 101 UPMB 110 UPMC 111 Reserved
EMEMC 27	0	**External MEMC Enable** Overrides BR*x*[MS] and assigns the bank to the system bus. However, other BR fields remain in effect. See **Section 12.1**, *Basic Architecture*, **page 12-12**. When this bit is set, the external memory controller is expected to assert $\overline{\text{AACK}}$, $\overline{\text{TA}}$, and $\overline{\text{PSDVAL}}$. **Note:** After a system reset, the BR0[EMEMC] is set to the value of the EXMC field from the HRCW.	0 Accesses are handled by the memory controller according to BR*x*[MS]. 1 Accesses are handled by an external memory controller (or other slave) on the system bus.

Table 12-31. BR*x* Bit Descriptions (Continued)

Name	Reset	Description	Settings
ATOM 28–29	00	**Atomic Operation** See **Section 12.1, *Basic Architecture***, **page 12-10**. **RAWA.** Writes to the address space handled by the memory controller bank cause the MSC8113 to lock the bus for the exclusive use of the master. The lock is released when the master performs a read operation from this address space. This feature is intended for CAM operations. **WARA.** Reads from the address space handled by the memory controller bank cause the MSC8113 to lock the bus for the exclusive use of the accessing device. The lock is released when the device performs a write operation to this address space. **Note:** If the device fails to release the bus, the lock is released after 256 clock cycles.	00 The address space controlled by the memory controller bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA). 10 Write-after-read-atomic (WARA). 11 Reserved
DR 30	0	**Data Pipelining** See **Section 12.1**, *Basic Architecture*, **page 12-11**. This feature is for memory regions that use parity checks and need to improve data set-up time.	0 No data pipelining is done. 1 Data beats of accesses to the address space controlled by the memory controller bank are delayed by one cycle.
V 31	0	**Valid Bit** Indicates that the contents of the BR*x* and OR*x* pair are valid. The $\overline{\text{CS}}x$ signal does not assert until V is set. An access to a region with no V bit set may cause a bus monitor time-out. **Note:** After a system reset, the BR0[V] is 1.	0 This bank is invalid. 1 This bank is valid.

ORx Option Registers—SDRAM Mode

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="12">SDAM</td><td colspan="4">LSDAM</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset[1]</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td>LSDAM</td><td colspan="2">BPD</td><td colspan="4">ROWST</td><td colspan="3">NUMR</td><td>PMSEL</td><td>IBID</td><td colspan="4">—</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset[1]</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
</table>

Notes:
1. After a system reset the OR0 has value 0xFE000EF4.
2. The boot sequence sets OR9 to 0xFFFC0008 (see **Section 12.7.2**) and OR11 to 0xFFE00000 (see **Section 12.7.1**).
3. The other OR registers are not initialized at reset.

OR*x* define the size of memory banks and access attributes. The OR*x* attribute bits support three modes of operation as defined by BR*x*[MS]: SDRAM mode, GPCM mode, and UPM mode.

Table 12-32 describes OR*x* fields in SDRAM mode. For details see **Section 12.2.14**, *SDRAM Configuration Examples*, on page 12-29.

Table 12-32. OR*x* Bit Descriptions (SDRAM Mode)

Name	Reset	Description	Settings
SDAM 0–11	—	**SDRAM Address Mask** Masks corresponding BR*x* bits. Masking address bits independently allows SDRAM devices of different size address ranges to be used. Clearing bits masks the corresponding address bit. Setting bits causes the corresponding address bits to be compared with the address lines. Address mask bits can be set or cleared in any order, allowing a resource to reside in more than one area of the address map. SDAM can be read or written at any time. **Note:** If PSDMR[PBI]=0, the maximum size of the memory bank should not exceed 128 MB.	000000000000 = 4 GB 100000000000 = 2 GB 110000000000 = 1 GB 111000000000 = 512 MB 111100000000 = 256 MB 111110000000 = 128 MB 111111000000 = 64 MB 111111100000 = 32 MB 111111110000 = 16 MB 111111111000 = 8 MB 111111111100 = 4 MB 111111111110 = 2 MB 111111111111 = 1 MB
LSDAM 12–16	—	**Lower SDRAM Address Mask** **Note:** Reset LSDAM to 0x0 to implement a minimum size of 1 MB when using SDRAM.	
		SDRAM Page Information	
BPD 17–18	—	**Banks Per Device** Sets the number of internal banks per SDRAM device. Note that for 128-Mb SDRAMs, BPD must be 00 or 01.	00 2 internal banks per device. 01 4 internal banks per device. 10 8 internal banks per device (not valid for 128-Mb SDRAMs). 11 Reserved.
ROWST 19–22	—	**Row Start Address Bit** Sets the demultiplexed row start address bit. The value of ROWST depends on PSDMR[PBI].	**For PSDMR[PBI] = 0**: 0010 A7 0100 A8 0110 A9 1000 A10 1010 A11 1100 A12 1110 A13 Other values are reserved. **For PSDMR[PBI] = 1**: 0000 A0 0001 A1 ... 1100 A12 1101–1111 Reserved

Table 12-32. ORx Bit Descriptions (SDRAM Mode) (Continued)

Name	Reset	Description	Settings
NUMR 23–25	—	**Number of Row Address Lines** Sets the number of row address lines in the SDRAM device.	000 9 row address lines 001 10 row address lines 010 11 row address lines 011 12 row address lines 100 13 row address lines 101 14 row address lines 110 15 row address lines 111 16 row address lines
PMSEL 26	—	**Page Mode Select** Selects page mode for the SDRAM connected to the memory controller bank.	0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.
IBID 27	—	**Internal Bank Interleaving within Same Device Disable** Disables bank interleaving between internal banks of a SDRAM device connected to the chip-select line. IBID should be set in 60x-compatible mode if the SDRAM device is not connected to the BNKSEL signals.	0 Enables bank interleaving. 1 Disables bank interleaving.
— 28–31	—	Reserved. Write to zero for future compatibility.	

ORx Option Registers—GPCM Mode

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	AM															
Type	R/W															
Reset[1]	See note.															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	AM	—		BCTLD	CSNT	ACS		—	SCY				SETA	TRLX	EHTR	—
Type	R/W															
Reset[1]	See note.															

Notes: **1.** After a system reset the OR0 has value 0xFE000EF4.

2. The boot sequence sets OR9 to 0xFFFC0008 (see **Section 12.7.2**) and OR11 to 0xFFE00000 (see **Section 12.7.1**).

3. The other OR registers are not initialized at reset.

Table 12-33. OR*x* Bit Descriptions (GPCM Mode)

Name	Reset	Description	Settings
AM 0–16	—	**Address Mask** Masks corresponding BR*x* bits. Masking address bits independently allows external devices of different size address ranges to be used. When AM is set, address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. **Note:** After system reset, OR0[AM] is 0b11111110000000000. The boot sequence sets OR9[AM] to 0b11111111111111000.	0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address lines.
— 17–18	—	Reserved. Write to zero for future compatibility.	
BCTLD 19	—	**Data Buffer Control Disable** Disables the assertion of $\overline{\text{BCTL}x}$ during access to the current memory bank. See **Section 12.1**, *Basic Architecture*, on **page 12-9**. **Note:** After system reset OR0[BCTLD] is cleared. The boot sequence clears OR9[BCTLD].	0 $\overline{\text{BCTL}x}$ is asserted upon access to the current memory bank. 1 $\overline{\text{BCTL}x}$ is not asserted upon access to the current memory bank.
CSNT 20	—	**Chip-Select Deassertion Time** Determines when $\overline{\text{CS}}$/$\overline{\text{PWE}}$ are deasserted during an external memory write access handled by the GPCM. This helps meet address/data hold times for slow memories and peripherals. **Note:** After system reset OR0[CSNT] is set. The boot sequence clears OR9[CSNT].	0 $\overline{\text{CS}}$/$\overline{\text{PWE}}$ are deasserted normally. 1 $\overline{\text{CS}}$/$\overline{\text{PWE}}$ are deasserted a quarter of a clock earlier.
ACS 21–22	—	**Address to Chip-Select Set-Up** Can be used when the external memory access is handled by the GPCM. It allows the delay of the $\overline{\text{CS}}$ assertion relative to the address change. **Note:** After a system reset, OR0[ACS] is 11. The boot sequence writes 00 to OR9[ACS].	00 $\overline{\text{CS}}$ is output at the same time as the address lines. 01 Reserved. 10 $\overline{\text{CS}}$ is output a quarter of a clock after the address lines. 11 $\overline{\text{CS}}$ is output half a clock after the address lines.

Table 12-33. ORx Bit Descriptions (GPCM Mode) (Continued)

Name	Reset	Description	Settings
— 23	—	Reserved. Write to zero for future compatibility.	
SCY 24–27	—	**Cycle Length in Clocks** Determines the number of wait states inserted in the cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. The total memory access length is (2 + SCY) x Clocks. **Notes:** **1.** If both SETA and CSNT fields are set, then SCY field should be written to a non zero value. **2.** After a system reset, OR0[SCY] is 1111. The boot sequence writes 0000 to OR9[SCY].	0000 0 clock cycle wait states 0001 1 clock cycle wait states 0010 2 clock cycle wait states 0011 3 clock cycle wait states 0100 4 clock cycle wait states 0101 5 clock cycle wait states 0110 6 clock cycle wait states 0111 7 clock cycle wait states 1000 8 clock cycle wait states 1001 9 clock cycle wait states 1010 10 clock cycle wait states 1011 11 clock cycle wait states 1100 12 clock cycle wait states 1101 13 clock cycle wait states 1110 14 clock cycle wait states 1111 15 clock cycles wait states
SETA 28	—	**External Access Termination** (PSDVAL generation) Specifies that when the GPCM is selected to handle the memory access initiated to this memory region, the access is terminated externally by asserting the external PGTA signal. In this case, PSDVAL is asserted one or two clocks later on the bus, depending on the synchronization of PGTA. See **Section 12.3.2**, **Note:** After a system reset, the OR0[SETA] is cleared. The boot sequence sets OR9[SETA].	0 PSDVAL is generated internally by the memory controller unless PGTA is asserted earlier externally. 1 PSDVAL is generated after external logic asserts PGTA.
TRLX 29	—	**Timing Relaxed** Works in conjunction with EHTR (bit 30). **Note:** After a system reset, the OR0[TRLX] is set. The boot sequence clears OR9[TRLX].	
EHTR 30	—	**Extended Hold Time on Read Accesses** Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next write access to the same bank, or any type of access to another bank. It does not affect subsequent read accesses to the same bank. **Note:** After a system reset, the OR0[EHTR] is cleared. The boot sequence clears OR9[EHTR].	TRLX and EHTR work together and are interpreted as follows: 00 Normal timing is generated by the memory controller. No additional cycles are inserted. 01 One idle clock cycle is inserted. 10 Four idle clock cycles are inserted. 11 Eight idle clock cycles are inserted.
— 31	—	Reserved. Write to zero for future compatibility.	

ORx Option Registers—UPM Mode

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	AM															
Type	R/W															
Reset	See notes.															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	AM	—		BCTLD	—			BI	—					EHTR		—
Type	R/W															
Reset	See notes.															

Notes:
1. After a system reset, OR0 has a value of 0xFE000EF4.
2. The boot sequence sets OR9 to 0xFFFC0008 (see **Section 12.7.2**) and OR11 to 0xFFE00000 (see **Section 12.7.1**).
3. The other OR registers are not initialized at reset.

Table 12-34. ORx—UPM Mode Bit Descriptions

Name	Reset	Description	Settings
AM 0–16	—	**Address Mask** Masks corresponding BR*x* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. **Note:** The boot sequence sets OR11[AM] to 0b11111111111000000.	0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address lines.
— 17–18	—	Reserved. Write to zero for future compatibility.	
BCTLD 19	—	**Data Buffer Control Disable** Disables the assertion of $\overline{\mathrm{BCTL}}x$ during access to the current memory bank. See **Section 12.1**, *Basic Architecture*, **page 12-9**. **Note:** The boot sequence clears OR11[BCTLD].	0 $\overline{\mathrm{BCTL}}x$ is asserted upon access to the current memory bank. 1 $\overline{\mathrm{BCTL}}x$ is not asserted upon access to the current memory bank.
— 20–22	—	Reserved. Write to zero for future compatibility.	
BI 23	—	**Burst Inhibit** Indicates if the memory bank supports burst accesses. **Note:** The boot sequence clears OR11[BI].	0 The bank supports burst accesses. 1 The bank does not support burst accesses. The UPM*x* executes burst accesses as series of single accesses.
— 24–28	—	Reserved. Write to zero for future compatibility.	

Table 12-34. ORx—UPM Mode Bit Descriptions (Continued)

Name	Reset	Description	Settings
EHTR 29–30	—	**Extended Hold Time on Read Accesses** Indicates the number of cycles inserted between the current bank read access and the next access. **Note:** The boot sequence writes 00 to OR11[EHTR].	00 Normal timing is generated by the memory controller. No additional cycles are inserted. 01 One idle clock cycle is inserted. 10 Four idle clock cycles are inserted. 11 Eight idle clock cycles are inserted.
— 31	—	Reserved. Write to zero for future compatibility.	

PSDMR System Bus SDRAM Mode Register

	Bit 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PBI	RFEN	OP			SDAM			BSMA			SDA10			RFRC	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	RFRC	PRETOACT			ACTTORW			BL	LDOTOPRE		WRC		EAMUX	BUFCMD	CL	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSDMR configures operations pertaining to SDRAM machine on the system bus.

Table 12-35. PSDMR Bit Descriptions

Name	Reset	Description	Settings
PBI 0	0	**Page-Based Interleaving** Selects the address multiplexing method. PSDMR[PBI] works in conjunction with PSDMR[SDA10]. **Note:** See **Section 12.2.5**, *Bank Interleaving,* on page 12-17.	0 Bank-based interleaving. 1 Page-based interleaving (normal operation).
RFEN 1	0	**Refresh Enable** Indicates that the SDRAM needs refresh services. **Note:** See the discussion of PSRT on **page 12-111**.	0 Refresh services are not required. 1 Refresh services are required.
OP 2–4	000	**SDRAM Operation** Determines which operation occurs when the SDRAM device is accessed. **Note:** If 60x-compatible mode is in effect on the system bus or the SDRAM port size is 8/16 or the SDRAM is connected to the BADDR lines (not needed for 64/32 port size), the bus master must supply the mode register data on the low bits of the address during the access.	000 Normal operation. 001 CBR refresh, used in SDRAM initialization. 010 Self refresh (for debug purpose). 011 Mode Register write, used in SDRAM initialization. 100 Precharge bank (for debug purpose). 101 Precharge all banks, used in SDRAM initialization. 110 Activate bank (for debug purpose). 111 Read/write (for debug purpose).

Table 12-35. PSDMR Bit Descriptions (Continued)

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th><th>Settings</th></tr>
<tr><td><b>SDAM</b>
5–7</td><td>000</td><td><b>Address Multiplex Size</b>
Determines how the address of the current memory cycle can be output on the address lines during RAS cycle.
<b>Note:</b> See <b>Section 12.2.7</b>, <i>SDRAM Address Multiplexing (SDAM and BSMA),</i> on page 12-18.</td><td><table>
<tr><th>SDAM</th><th>External System Bus Address Line</th><th>Signal Driven on External Line</th></tr>
<tr><td>000</td><td>A[13–31]</td><td>A[5–23]</td></tr>
<tr><td>001</td><td>A[14–31]</td><td>A[5–22]</td></tr>
<tr><td>010</td><td>A[15–31]</td><td>A[5–21]</td></tr>
<tr><td>011</td><td>A[16–31]</td><td>A[5–20]</td></tr>
<tr><td>100</td><td>A[17–31]</td><td>A[5–19]</td></tr>
<tr><td>101</td><td>A[18–31]</td><td>A[5–18]</td></tr>
</table></td></tr>
<tr><td><b>BSMA</b>
8–10</td><td>000</td><td><b>Bank Select Multiplexed Address Line</b>
Selects the address lines to serve as bank select address for the system bus SDRAM. The selected address can also be output on the BNKSEL signals (optional).
<b>Note:</b> See <b>Section 12.2.7</b>, <i>SDRAM Address Multiplexing (SDAM and BSMA),</i> on page 12-18 for details.</td><td>000 A[12–14]
001 A[13–15]
010 A[14–16]
011 A[15–17]
100 A[16–18]
101 A[17–19]
110 A[18–20]
111 A[19–21]</td></tr>
<tr><td><b>SDA10</b>
11–13</td><td>000</td><td><b>A10 Control</b>
With PSDMR[PBI], determines which address line can be output to PSDA10 during an ACTIVATE command, when SDRAM is selected, to control the memory access.
<b>Note:</b> See <b>Section 12.2.14.1</b>, <i>SDRAM Configuration Example (Page-Based Interleaving),</i> on page 12-29 for details.</td><td><b>For PSDMR[PBI] = 0</b>:
000 A12
001 A11
010 A10
011 A9
100 A8
101 A7
110 A6
111 A5

<b>For PSDMR[PBI] = 1</b>:
000 A10
001 A9
010 A8
011 A7
100 A6
101 A5
110 A4
111 A3</td></tr>
</table>

Table 12-35. PSDMR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SDRAM Device-Specific Parameters:			
RFRC 14–16	000	**Refresh Recovery** Defines the earliest timing for an ACTIVATE command after a REFRESH command. Sets the refresh recovery interval in clock cycles.	000 Reserved 001 3 clocks 010 4 clocks 011 5 clocks 100 6 clocks 101 7 clocks 110 8 clocks 111 16 clocks
PRETOACT 17–19	000	**Precharge to Activate Interval** Defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command.	001 1 clock-cycle wait states. 010 2 clock-cycle wait states. ... 111 7 clock-cycle wait states. 000 8 clock-cycle wait states.
ACTTORW 20–22	000	**Activate to Read/Write Interval** Defines the earliest timing for a READ/WRITE command after an ACTIVATE command.	001 1 clock cycle. 010 2 clock cycles. ... 111 7 clock cycles. 000 8 clock cycles.
BL 23	0	**Burst Length** Defines the SDRAM burst length.	0 SDRAM burst length is 4. Use this value if the device port size is 64 or 16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
LDOTOPRE 24–25	00	**Last Data Out to Precharge** Defines the earliest timing for PRECHARGE command after the last data was read from the SDRAM.	00 0 clock cycles. 01 –1 clock cycle. 10 –2 clock cycles. 11 Reserved.
WRC 26–27	00	**Write Recovery Time** Defines the earliest timing for PRECHARGE command after the last data was written to the SDRAM.	01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles. 00 4 clock cycles.
EAMUX 28	0	**External Address Multiplexing Enable/Disable** If this bit is set, PSDMR[ACTTORW] should be a minimum of two clock cycles. In 60x-compatible mode, external address multiplexing is placed on the address lines. If the additional delay of the multiplexing endangers the device set-up time, PSDMR[EAMUX] should be set. Setting this bit causes the memory controller to add another cycle for each address phase. Note that PSDMR[EAMUX] can also be set in any case of delays on the address lines, such as address buffers.	0 No external address multiplexing. Fastest timing. 1 The memory controller asserts PSDAMUX for an extra cycle before issuing an ACTIVATE command to the SDRAM. This is useful when external address multiplexing can cause a delay on the address lines.

Table 12-35. PSDMR Bit Descriptions (Continued)

Name	Reset	Description	Settings
BUFCMD 29	0	**Command Buffer** If external buffers are placed on the control lines going to both the SDRAM and address lines, setting the PSDMR[BUFCMD] bit causes all SDRAM control lines except $\overline{\text{CS}}$ to be asserted for two cycles, instead of one. In 60x-compatible mode, external buffers may be placed on the command strobes, except $\overline{\text{CS}}$, as well as on the address lines. If the additional delay of the buffers endangers the device set-up time, PSDMR[BUFCMD] should be set, which causes the memory controller to add a cycle for each SDRAM command.	0 Normal timing for the control lines. 1 All control lines except $\overline{\text{CS}}$ are asserted for two cycles.
CL 30–31	00	**CAS Latency** Defines the timing for first read data after SDRAM samples a column address.	00 Reserved 01 1 10 2 11 3
Note: See **Section 12.2.10**, *SDRAM Signals: Device-Specific Parameters* for additional recommendations for configuring this register.			

M*x*MR Machine A/B/C Mode Registers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BSEL	RFEN	OP		—	AMx			DSx		G0CLx			GPL_x4DIS	RLFx	
Type	R/W															
Reset[1]	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
	RLFx		WLFx				TLFx				MAD					
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: The boot sequence sets the MCMR to 0x80011240 (see **Section 12.7.1**).

M*x*MR configures the UPMs.

Table 12-36. M*x*MR Bit Descriptions

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th><th>Settings</th></tr>
<tr><td><b>BSEL</b>
0</td><td>0</td><td><b>Bus Select</b>
Assigns banks that select UPM<i>x</i> to the system bus or local bus. UPMC is assigned to the local bus and controls accesses to the internal memories. UPMA and UPMB control external devices residing on the system bus. If 60x bus refresh is required, UPMA should be assigned to system bus.
<b>Note:</b> The boot sequence sets MCMR[BSEL].</td><td>0 Banks that select UPM<i>x</i> are assigned to the system bus.
1 Banks that select UPM<i>x</i> are assigned to the local bus.</td></tr>
<tr><td><b>RFEN</b>
1</td><td>0</td><td><b>Refresh Enable</b>
Indicates that the UPM needs refresh services. See the discussion of the system bus Assigned UPM Refresh Timer (PURT) in <b>Section 12.8</b>.</td><td>0 Refresh services are not required.
1 Refresh services are required.</td></tr>
<tr><td><b>OP</b>
2–3</td><td>00</td><td><b>Command Opcode</b>
Determines the command executed by the UPM<i>x</i> when a memory access hits a UPM assigned bank.

For Write to array, on the next memory access to a UPM assigned bank, write the contents of the MDR into the RAM location pointed by MAD. After the access, the MAD field is automatically incremented.

For Read from array, on the next memory access to a UPM assigned bank, read the contents of the RAM location pointed by MAD into the MDR. After the access, the MAD field is automatically incremented.

For Run pattern, on the next memory access to a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM.
<b>Note:</b> RLF determines the number of times a loop executes during a pattern run.</td><td>00 Normal operation.
01 Write to array.
10 Read from array.
11 Run pattern.</td></tr>
<tr><td>—
4</td><td>0</td><td colspan="2">Reserved. Write to zero for future compatibility.</td></tr>
<tr><td><b>AMx</b>
5–7</td><td>000</td><td><b>Address Multiplex Size</b>
Determines how the address of the current memory cycle can be output on the address lines. The address output is controlled by the contents of the UPM<i>x</i> RAM array. This field is useful when the MSC8113 connects to DRAM devices requiring row and column addresses multiplexed on the same lines. See <b>Section 12.4.4.3</b>, <i>Address Multiplexing,</i> on page 12-59.</td><td>
<table>
<tr><th>AMx</th><th>External System Bus Address Line</th><th>Signal Driven on External Line</th></tr>
<tr><td>000</td><td>A[16–31]</td><td>A[8–23]</td></tr>
<tr><td>001</td><td>A[16–31]</td><td>A[7–22]</td></tr>
<tr><td>010</td><td>A[16–31]</td><td>A[6–21]</td></tr>
<tr><td>011</td><td>A[16–31]</td><td>A[5–20]</td></tr>
<tr><td>100</td><td>A[17–31]</td><td>A[5–19]</td></tr>
<tr><td>101</td><td>A[18–31]</td><td>A[5–18]</td></tr>
</table>
</td></tr>
</table>

Table 12-36. M*x*MR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DSx 8–9	00	**Disable Timer Period** Guarantees a minimum time between accesses to the same memory bank if it is controlled by the UPM*x*. The disable timer is turned on by the TODT bit in the RAM array, and when expired, the UPM*x* allows the machine access to handle a memory pattern to the same memory region. Accesses to a different memory region by the same UPM*x* is allowed. **Note:** To avoid conflicts between successive accesses to different memory regions, the minimum pattern in the RAM array for a request serviced should not be shorter than the period established by DSx.	00 1-cycle disable period. 01 2-cycle disable period. 10 3-cycle disable period. 11 4-cycle disable period.
G0CLx 10–12	000	**General Line 0 Control** Determines which address line can be output to PGPL0 when the UPM*x* is selected to control the memory access.	000 A12 001 A11 010 A10 011 A9 100 A8 101 A7 110 A6 111 A5
GPL_x4DIS 13	0	**GPL_A4 Output Line Disable** Determines if the PUPMWAIT/$\overline{\text{PGTA}}$/PGPL4 behaves as an output line controlled by the corresponding bits in the UPM*x* array (GPL4*x*).	0 PUPMWAIT/$\overline{\text{PGTA}}$/PGPL4 behaves as PGPL4. UPM*x*[G4T4/DLT3] is interpreted as G4T4. The UPM*x*[G4T3/WAEN] is interpreted as G4T3. 1 PUPMWAIT/$\overline{\text{PGTA}}$/PGPL4 behaves as PUPMWAIT. UPM*x*[G4T4/DLT3] is interpreted as DLT3. UPM*x*[G4T3/WAEN] is interpreted as WAEN.
RLFx 14–17	0000	**Read Loop Field** Determines the number of times a loop defined in the UPM*x* executes for a burst- or single-beat read pattern or when M*x*MR[OP] = 11 (RUN command). **Note:** The boot sequence sets MCMR[RLFx] to 0010.	0001 The loop executes 1 time. 0010 The loop executes 2 times. ... 1111 The loop executes 15 times. 0000 The loop executes 16 times.
WLFx 18–21	0000	**Write Loop Field** Determines the number of times a loop defined in the UPM*x* executes for a burst- or single-beat write pattern. **Note:** The boot sequence sets MCMR[WLFx] to 0010.	0001 The loop executes 1 time. 0010 The loop executes 2 times. ... 1111 The loop executes 15 times. 0000 The loop executes 16 times.

Table 12-36. M*x*MR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TLFx 22–25	0000	**Refresh Loop Field** Determines the number of times a loop defined in the UPM*x* executes for a refresh service pattern. **Note:** The boot sequence sets MCMR[TLFx] to 1001.	0001 The loop executes 1 time. 0010 The loop executes 2 times. ... 1111 The loop executes 15 times. 0000 The loop executes 16 times.
MAD 26–31	0x0	**Machine Address** RAM address pointer for the command executed. This field is incremented by one, each time the UPM is accessed and the OP field is set to WRITE or READ.	

MDR Memory Data Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	MD															
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
	MD															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MDR contains data written to or read from the RAM array for UPM READ or WRITE commands. MDR must be set up before a WRITE command is issued to the UPM.

Table 12-37. MDR Bit Descriptions

Name	Reset	Description
MD 0–31	0x0	**Memory Data** The data to be read from or written into the RAM array when a WRITE or READ command is supplied to the UPM.

MAR Memory Address Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	A															
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
	A															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-38. MAR Bit Descriptions

Name	Reset	Description
A 0–31	0x0	**Memory Address** The memory address register can be output to the address lines under control of the AMX bits in the UPM.

PURT System Bus Assigned UPM Refresh Timer

Bit	0	1	2	3	4	5	6	7
	PURT							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Table 12-39. PURT Bit Descriptions

Name	Reset	Description
PURT 0–7	0	**Refresh Timer Period** Determines the timer period. Compute the value of PURT according to the following equation: $\text{PURT} < \frac{F_{Bus} \times RefreshRate}{\text{MPTPR[PTP]} + 1} - 1$ This timer generates a refresh request for all valid banks that selected a UPM machine assigned to the system bus (M*x*MR[BSEL] = 0) and is refresh-enabled (M*x*MR[RFEN] = 1). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks are rotating their requests. *Example:* For a 25-MHz bus clock (F_{BUS}) and a required refresh rate (RefreshRate) of 15.6 µs, given MPTPR[PTP] = 32, the PURT value should be 10 decimal, which is the next lower integer value.

PSRT System Bus Assigned SDRAM Refresh Timer

Bit	0	1	2	3	4	5	6	7
	PSRT							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Table 12-40. PSRT Bit Descriptions

Name	Reset	Description
PSRT 0–7	0	**Refresh Timer Period** Determines the timer period. Compute the value of PSRT according to the following equation: $\text{PSRT} < \frac{F_{Bus} \times RefreshRate}{\text{MPTPR[PTP]} + 1} - 1$ This timer generates refresh requests for all valid banks that selected a SDRAM machine assigned to the system bus and is refresh-enabled (PSDMR[RFEN] = 1). Each time the timer expires, all banks that qualify generate a bank staggering auto refresh request using the SDRAM machine. See **Section 12.2.9**, *SDRAM Refresh,* on page 12-20. *Example:* For a 25-MHz bus clock (F_{BUS}) and a required refresh rate (RefreshRate) of 15.6 µs, given MPTPR[PTP] = 32, the PSRT value should be 10 decimal, which is the next lower integer value.

MPTPR Memory Refresh Timer Prescaler Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PTP								—							
Type	R/W															
Reset	0	0	0	0	0	0	1	*x*	0	0	0	0	0	0	0	0

Table 12-41. MPTPR Bit Descriptions

Name	Reset	Description
PTP 0–5	0	**Memory Refresh Timers Prescaler** Determines the period of the memory refresh timers input clock. It divides the bus clock.
PTP 6	1	
PTP 7	*x*	
— 8–15	0	Reserved. Write to zero for future compatibility.
Note: Where *x* can be a zero or one—that is, 00000010 or 00000011.		

TESCR*x* System Bus Error Status and Control Registers

Indicate the source of an error that causes assertion of $\overline{\text{TEA}}$ or $\overline{\text{MCP}}$ on the system bus. See the discussion of system bus Transfer Error Status and Control Registers (TESCRx) in **Section 4.2**, *SIU Programming Model*.

L_TESCR1 Local Bus Error Status and Control Register

Indicates the source of an error that causes assertion of $\overline{\text{TEA}}$ or $\overline{\text{MCP}}$ on the local bus. See the discussion of the Local Bus Transfer Error Status and Control Register 1 (L_TESCR1) in **Section 4.2**, *SIU Programming Model*.

System Bus 13

The 60x-compatible system bus provides flexible support for the internal SC140 DSP cores as well as other internal and external 60x-compatible bus masters. The system bus allows 32-bit addressing, a 64-bit data bus, and burst operations that transfer as many as 256 bits of data in a four-beat burst. The data bus is accessed in 8-bit, 16-bit, 32-bit, and 64-bit data ports. The system bus handles accesses of 1, 2, 3, and 4 bytes, aligned or unaligned, on 4-byte boundaries; it also handles 64-bit, 128-bit, 192-bit, and 256-bit accesses. The address and data buses handle synchronous, one-level pipeline transactions. The system bus interface can be configured to support both external and internal masters or internal masters only. This chapter describes the system bus signals and the operational protocols.

13.1 System Bus Signals

This section describes the external signals of the MSC8113 system bus. It describes the individual signals, showing behavior when a signal is asserted and deasserted, when the signal is an input and an output, and the differences in how signals work in external-master or internal-only configurations.

Note: A bar over a signal name indicates that the signal is active low–for example, $\overline{\text{ARTRY}}$ (address retry) and $\overline{\text{TS}}$ (transfer start). Active-low signals are referred to as asserted (active) when they are low and deasserted when they are high. Signals that are not active-low, such as TSIZ[0–3] (transfer size signals) and TT[0–4] (transfer type signals) are referred to as asserted when they are high and deasserted when they are low.

Table 13-1 shows the functional groupings of the MSC8113 system bus signals.

Table 13-1. MSC8113 System Bus Signal Groupings

Signal Group	Descriptions	Location
Address Arbitration	The MSC8113 device uses these signals in external arbiter mode to arbitrate for address bus mastership. The MSC8113 arbiter uses these signals to enable an external device to arbitrate for address bus mastership.	**Table 13-2** on page 13-6
Address Start	Indicates that a bus master has begun a transaction on the address bus.	**Table 13-3** on page 13-8
Address Bus	Transfer the address.	**Table 13-4** on page 13-8
Address Transfer Attribute	Provides information about the type of transfer, such as the transfer size and whether the transaction is single, single extended, or burst.	**Table 13-5** on page 13-9
Address Transfer Termination	Acknowledges the end of the address phase of the transaction; also indicates whether a condition exists that requires the address phase to be repeated.	**Table 13-6** on page 13-11
Data Arbitration	The MSC8113 device uses these signals in external arbiter mode to arbitrate for data bus mastership. The MSC8113 arbiter uses these signals to enable an external device to arbitrate for data bus mastership.	**Table 13-7** on page 13-12
Data Transfer	Transfer the data and ensure its integrity. This signal group consists of the data bus and data parity signals.	**Table 13-8** on page 13-16
Data Transfer Termination	Required after each data beat in a data transfer. In a single-beat transaction, the data transfer termination signals also indicate the end of the tenure. For burst accesses or extended port-size accesses, these signals apply to individual beats and indicate the end of the tenure only after the final data beat.	**Table 13-9** on page 13-20

Figure 13-1 shows the grouping of the MSC8113 system bus signal configuration, as well as pin numbers. **Chapter 3**, *External Signals* describes the MSC8113 external signals in greater detail.

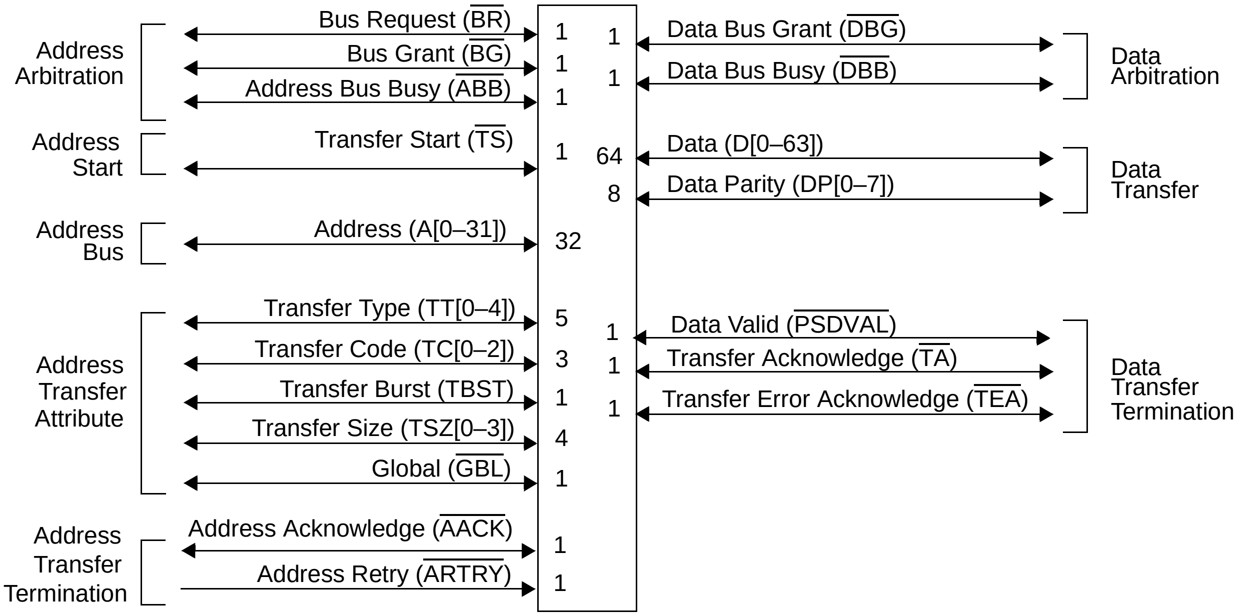


Figure 13-1. MSC8113 System Bus Groupings

13.1.1 Address Arbitration

Devices use these input and output signals to request address bus mastership, recognize when the request is granted, and indicate to other devices when mastership is granted. For details on how these signals interact, see **Section 13.2.3.1**, *Address Arbitration*.

Table 13-2. Address Arbitration Signals

Name	Type	Description	
BR	Input/Output	**Bus Request** Use in external master mode. BR has no meaning in internal-only mode.	
	Output	State Meaning	*Asserted*. Indicates that the MSC8113 device is requesting mastership of the address bus. BR can be asserted for one or more cycles and then deasserted due to an internal cancellation of the bus request. See **Section 13.2.3.1**, *Address Arbitration*. *Deasserted*. Indicates that the MSC8113 device is not requesting the address bus. The MSC8113 may have no bus operation pending. It may be parked, or the ARTRY input may have been asserted on the previous bus clock cycle.
		Timing Comments	*Assertion*. Occurs on any cycle; does not occur if the MSC8113 device is parked and the address bus is idle (BG asserted and ABB input deasserted). *Deassertion*. Occurs for at least one cycle following a qualified BG even if another transaction is pending. Deassertion also occurs for at least one cycle following any qualified ARTRY on the bus. It may also occur if the MSC8113 device cancels the bus request internally before receiving a qualified BG. *High Impedance*. Occurs during a hard reset or checkstop condition.
	Input	State Meaning	*Asserted*. Indicates that the external master has a bus transaction to perform and is waiting for a qualified BG to begin the address tenure. BR may be asserted even if the two possible pipelined address tenures have already been granted. *Deasserted*. Indicates that the external master has no bus transaction to perform, or if the device is parked, that it is potentially ready to start a bus transaction on the next clock cycle (with proper qualification, see BG).
		Timing Comments	*Assertion*. Occurs on any cycle; does not occur if the external master is parked and the address bus is idle (BG asserted and ABB input deasserted). *Deassertion*. Occurs for at least one cycle after a qualified BG even if another transaction is pending. Deassertion also occurs for at least one cycle following any qualified ARTRY on the bus. It may also occur if the external master cancels a bus request internally before receiving a qualified BG. *High Impedance*. Occurs during a hard reset or checkstop condition.

Table 13-2. Address Arbitration Signals (Continued)

Name	Type	Description
BG	Input/ Output	**Bus Grant** Use in external master mode. BG has no meaning in internal-only mode.
	Input	State Meaning: *Asserted*. Indicates that, with the proper qualification, the MSC8113 device can begin a bus transaction and assume ownership of the address bus. A bus grant is qualified if BG is asserted and ABB and ARTRY are deasserted (where ARTRY is asserted only during the cycle after AACK). The assertion of BR is not required for a qualified bus grant (to allow bus parking). *Deasserted*. Indicates that the MSC8113 is not granted next address ownership. Timing Comments: *Assertion*. Occurs on any cycle. Once the MSC8113 has assumed address bus ownership, it does not begin checking for BG again until the cycle after AACK. *Deassertion*. Occurs whenever the MSC8113 must be prevented from using the address bus. The MSC8113 can still assume address bus ownership on the cycle BG is deasserted if it was asserted the previous cycle with other bus grant qualifications.
	Output	State Meaning: *Asserted*. Indicates that, with the proper qualification, the MSC8113 device can begin a bus transaction and assume ownership of the address bus. A bus grant is qualified if BG is asserted and ABB and ARTRY are deasserted (where ARTRY is asserted only during the cycle after AACK). The assertion of BR is not required for a qualified bus grant (to allow bus parking). *Deasserted*. Indicates that the external device is not granted next address ownership. Timing Comments: *Assertion*. Can occur on any cycle. Once the external device assumes address bus ownership, it does not begin checking for BG again until the cycle after AACK. *Deassertion*. Can occur when an external device must be kept from using the address bus. The external device may still assume address bus ownership on the cycle that BG is deasserted if it was asserted the previous cycle with other bus grant qualifications.
ABB	Input/ Output	**Address Bus Busy** Use in external master mode. ABB has no meaning in internal-only mode.
	Output	State Meaning: *Asserted*. Indicates that the MSC8113 device is the current address bus master. The MSC8113 may not assume address bus ownership in case a bus request is internally cancelled by the cycle in which a qualified BG would have been recognized. *Deasserted*. The MSC8113 is not the current address bus master. Timing Comments: *Assertion*. Occurs the cycle after the MSC8113 device accepts a qualified BG and remains asserted for the duration of the address tenure. *Turn-Off Sequencing*. Deasserts for a fraction of a bus cycle (1/2 minimum, depends on clock mode) starting the cycle following the assertion of AACK. It then goes to the high impedance state.
	Input	State Meaning: *Asserted*. Indicates that external device is the address bus master. *Deasserted*. Indicates that the address bus may be available for use by the MSC8113 (see BG). The MSC8113 also tracks the state of ABB on the bus from the TS and AACK inputs. (See section on address arbitration phase.) Timing Comments: *Assertion*. Can occur when the MSC8113 device must be prevented from using the address bus. *Deassertion*. Can occur when the MSC8113 device can use the address bus.

13.1.2 Address Start

Table 13-3. Address Start Signal

Name	Type	Description	
TS	Input/ Output	**Transfer Start** Indicates that an address bus transfer has begun. Use in external master mode. $\overline{\text{TS}}$ has no meaning in internal-only mode.	
	Output	State Meaning	*Asserted*. Indicates that the MSC8113 device has started a bus transaction and that the address bus and transfer attribute signals are valid. It is also an implied data bus request if the transfer attributes TT[0–4] indicate that a data tenure is required for the transaction. *Deasserted*. Has no special meaning during a normal transaction.
		Timing Comments	*Assertion/Deassertion*. Driven and asserted on the cycle after the MSC8113 accepts a qualified $\overline{\text{BG}}$; remains asserted for one clock only. Deasserted for the remainder of the address tenure. Assertion is coincident with the first clock that $\overline{\text{ABB}}$ is asserted. *High Impedance*. Occurs the cycle following the assertion of $\overline{\text{AACK}}$ (same cycle as $\overline{\text{ABB}}$ deassertion).
	Input	State Meaning	*Asserted*. Indicates that another device has begun a bus transaction and that the address bus and transfer attribute signals are valid for snooping. *Deasserted*. Has no special meaning.
		Timing Comments	*Assertion/Deassertion*. Must be asserted for one cycle only and then immediately deasserted. Assertion may occur at any time during assertion of $\overline{\text{ABB}}$.

13.1.3 Address Bus

Table 13-4. Address Bus Signals

Name	Type	Description	
A[0–31]	Input/ Output	**Address Bus** The address transfer signals transmit the address. In internal-only mode the memory controller uses these signals for glueless address transfers to memory and I/O devices.	
	Output	State Meaning	*Content*. Specifies the physical address of the bus transaction. For burst or extended operations, the address is 32 bits.
		Timing Comments	*Assertion/Deassertion*. Driven valid (by the MSC8113) on the same cycle that $\overline{\text{TS}}$ is driven/asserted; remains driven/valid for the duration of the address tenure. *High Impedance*. Occurs the cycle following the assertion of $\overline{\text{AACK}}$; no precharge action performed on release.
	Input	State Meaning	*Content*. Specifies the physical address of the bus transaction. For burst or extended operations, the address is 32 bits.
		Timing Comments	*Assertion/Deassertion*. Must be valid (driven by other device) on the same cycle that $\overline{\text{TS}}$ is asserted; sampled by the MSC8113 only on this cycle.

13.1.4 Address Transfer Attribute

The address transfer attribute signals further characterize the transfer, such as the size of the transfer, whether it is a read or write operation, and whether it is a burst or single-beat transfer. For details on how these signals interact, see **Section 13.2.3.3**, *Address Transfer Attribute Signals*.

Table 13-5. Address Transfer Attribute Signals

Name	Type	Description
TT[0–4]	Input/Output Output: Input	**Transfer Type** Use in external master mode. The TT[0–4] signals have no meaning in internal-only mode. For a complete description of TT[0–4] signals and transfer type encoding. See **Section n**, *Transfer type signals (TT[0–4]). The transfer type signals define the nature of the transfer requested (Read or Write). Table 13-10 describes the MSC8113 action as master, slave, and snooper.*. State Meaning: *Asserted/Deasserted*. Specifies the type of transfer in progress. Timing Comments: *Assertion/Deassertion*. Same as A[0–31]. *High Impedance*. Same as A[0–31]. State Meaning: *Asserted/Deasserted*. Specifies the type of transfer in progress for snooping by the MSC8113 device. Timing Comments: *Assertion/Deassertion*. Same as A[0–31].
TC[0–2]	Input/Output	**Transfer Code** Use in external master mode. The TC[0–2] signals have no meaning in internal-only mode. For a complete description of TC[0–2] signals and transfer code encoding. See **Section n**, *Transfer Code signals (TC[0–2]). The transfer code signals give supplemental information about the corresponding address, mainly the source of the transaction, as listed in Table 13-11.*. State Meaning: *Asserted/Deasserted*. Gives supplemental information about the corresponding address, mainly the source of the transaction. Timing Comments: *Assertion/Deassertion*. Same as A[0–31]. *High Impedance*. Same as A[0–31].
TBST	Input/Output	**Transfer Burst** Use in external master mode. $\overline{\text{TBST}}$ has no meaning in internal-only mode. State Meaning: *Asserted*. Indicates that a burst transfer is in progress (see **Section n**, *Transfer burst and size signals (TBST and TSIZ[0–3]). These signals together indicate the size of the requested data transfer. The signals can be used with address bits A[27–31] and the device port size to determine which portion of the data bus contains valid data for a write transaction or which portion of the bus should contain valid data for a read transaction. The MSC8113 uses four 32-bit burst transactions for transferring cache blocks. For these transactions, TSIZ[0–3] are encoded as 0b0010, TBST is asserted, and address bits A[27–28] determine which 32 bits are sent first. The MSC8113 supports critical-first burst transactions (32-bit-aligned) from the processor. The MSC8113 transfers the critical 32 bits of data first, followed by 32 bits from increasing addresses, wrapping back to the beginning of the 8-level block as required.*). *Deasserted*. Indicates that a burst transfer is not in progress. Timing Comments: *Assertion/Deassertion*. Same as A[0–31]. *High Impedance*. Same as A[0–31].

Table 13-5. Address Transfer Attribute Signals (Continued)

Name	Type	Description
TSIZ[0–3]	Input/Output	**Transfer Size** Use in external master mode. The $\overline{\text{TSIZ[0–3]}}$ signals have no meaning in internal-only mode. State Meaning *Asserted/Deasserted.* Specifies the data transfer size for the transaction (see Section **Section n**, *Transfer burst and size signals (TBST and TSIZ[0–3]). These signals together indicate the size of the requested data transfer. The signals can be used with address bits A[27–31] and the device port size to determine which portion of the data bus contains valid data for a write transaction or which portion of the bus should contain valid data for a read transaction. The MSC8113 uses four 32-bit burst transactions for transferring cache blocks. For these transactions, TSIZ[0–3] are encoded as 0b0010, TBST is asserted, and address bits A[27–28] determine which 32 bits are sent first. The MSC8113 supports critical-first burst transactions (32-bit-aligned) from the processor. The MSC8113 transfers the critical 32 bits of data first, followed by 32 bits from increasing addresses, wrapping back to the beginning of the 8-level block as required.*). Timing Comments *Assertion/Deassertion*. Same as A[0–31]. *High Impedance*. Same as A[0–31].
GBL	Input/Output	**Global** Use in external master mode. $\overline{\text{GBL}}$ has no meaning in internal-only mode. State Meaning *Asserted*. Indicates that the transaction is global and should be snooped by other devices. *Deasserted*. Indicates that the transaction is not global and should not be snooped by other devices. Timing Comments *Assertion/Deassertion*. Same as A[0–31]. *High Impedance*. Same as A[0–31].

13.1.5 Address Transfer Termination

The address transfer termination signals indicate either that the address phase of the transaction has completed successfully or must be repeated, and when it should be terminated. For details on how these signals interact, see **Section 13.2.3.9**, *Address Transfer Termination*.

Table 13-6. Address Transfer Termination Signals

Name	Type	Description
AACK	Input/Output	**Address Acknowledge** Use in external master mode. AACK has no meaning in internal-only mode.
	Output	State Meaning — *Asserted*. Indicates that the address tenure of a transaction is terminated. On the cycle following the assertion of AACK, the bus master releases the address-tenure-related signals to the high-impedance state and samples ARTRY. *Deasserted*. Indicates that the address bus and the transfer attributes must remain driven, if deasserted during ABB.
		Timing Comments — *Assertion*. Occurs a programmable number of clocks after TS or whenever ARTRY conditions are resolved. *Deassertion*. Occurs one clock after assertion.
	Input	State Meaning — *Asserted*. Indicates that a 60x-compatible system bus slave is terminating the address tenure. On the cycle following the assertion of AACK, the bus master releases the address tenure related signals to the high-impedance state and samples ARTRY. *Deasserted*. Indicates that the address tenure must remain active and the address tenure related signals driven.
		Timing Comments — *Assertion*. Occurs during the 60x-compatible system bus slave access, at least two clocks after TS. *Deassertion*. Occurs one clock after assertion.
ARTRY	Input	**Address Retry** Use in external master mode. ARTRY has no meaning in internal-only mode.
		State Meaning — *Asserted*. If the MSC8113 is the address bus master, ARTRY indicates that the MSC8113 must retry the preceding address tenure and immediately deassert BR (if asserted). If the associated data tenure has started, the MSC8113 also aborts the data tenure immediately, even if the burst data has been received. If the MSC8113 is not the address bus master, this input indicates that the MSC8113 should deassert BR for one bus clock cycle immediately after external device asserts ARTRY to permit a copy-back operation to main memory. Note that the subsequent address presented on the address bus may not be the one that generated the assertion of ARTRY. *Deasserted/High Impedance*. Indicates that the MSC8113 does not need to retry the last address tenure.
		Timing Comments — *Assertion*. Occurs as early as the second cycle after TS is asserted and must occur by the bus clock cycle immediately after AACK is asserted if an address retry is required. *Deassertion*. Must occur during the second cycle after AACK is asserted.

13.1.6 Data Arbitration

Like the address arbitration signals, data arbitration signals maintain an orderly process for determining data bus mastership. There is no data arbitration signal equivalent to the address arbitration signal BR, because, except for address-only transactions, TS implies data bus requests. For details on how these signals interact, see **Section 13.2.4.1**, *Data Bus Arbitration*.

Table 13-7. Data Arbitration Signals

Name	Type	Description
DBG	Input/ Output	**Data Bus Grant** Use in external master mode. DBG has no meaning in internal-only mode.
	Input	DBG is an input when MSC8113 is configured to external arbiter. State Meaning *Asserted*. Indicates that, with the proper qualification, the MSC8113 can assume mastership of the data bus. The MSC8113 device derives a qualified data bus grant when DBG is asserted and DBB and ARTRY are deasserted; that is, the data bus is not busy (DBB is deasserted), and there is no outstanding attempt to perform an ARTRY of the associated address tenure. *Deasserted*. Indicates that the MSC8113 must hold off its data tenures.
		Timing Comments *Assertion*. Occurs any time to indicate that the MSC8113 is free to take data bus mastership. It is not sampled until TS is asserted. *Deassertion*. Occurs at any time to indicate the MSC8113 cannot assume data bus mastership.
	Output	DBG is an output when the MSC8113 is configured to internal arbiter. State Meaning *Asserted*. Indicates that the external device can, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of DBG, deassertion of DBB, and deassertion of ARTRY. The requirement for the ARTRY signal is only for the address bus tenure associated with the data bus tenure about to be granted (that is, not for another address tenure available because of address pipelining). *Deasserted*. Indicates that an external device is not granted mastership of the data bus.
		Timing Comments *Assertion*. Occurs on the first clock in which the data bus is not busy and the processor has the highest priority outstanding data transaction. *Deassertion*. Occurs one clock after assertion.
DBB	Input/ Output	**Data Bus Busy** Use in external master mode. DBB has no meaning in internal-only mode.
	Output	State Meaning *Asserted*. Indicates that the MSC8113 device is the data bus master. The MSC8113 always assumes data bus mastership if it needs the data bus and determines a qualified data bus grant (see DBG). *Deasserted*. Indicates that the MSC8113 device is not using the data bus.
		Timing Comments Assertion. Occurs during the bus clock cycle following a qualified DBG. *Deassertion*. Occurs for a minimum of one-half bus clock cycle following the assertion of the final TA following TEA or certain ARTRY cases. *High Impedance*. Occurs after DBB is deasserted.
	Input	State Meaning Asserted. Indicates that another device is bus master. *Deasserted*. Indicates that the data bus is free (with proper qualification, see DBG) for use by the MSC8113.
		Timing Comments *Assertion*. Must occur when the MSC8113 device must be prevented from using the data bus. *Deassertion*. May occur whenever the data bus is available.

13.1.7 Data Transfer

Data transfer signals are used in the same way in both internal-only and external master modes. Like the address transfer signals, the data transfer signals transmit data and generate and monitor parity for the data transfer. For details how data transfer signals interact, see **Section 13.2.4.3**, *Data Bus Transfers and Normal Termination*.

Table 13-8. Data Transfer Signals

<table>
<tr><th>Name</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td rowspan="7">D[0–63]</td><td>Input/
Output</td><td colspan="2"><b>Data Bus</b>
The D[0–63] signals have the same meanings in both internal-only mode and external master mode.

<b>Note:</b> The MSC8113 device has either a 32-bit or a 64-bit external port for the system bus.</td></tr>
<tr><td rowspan="4">Output</td><td>State Meaning</td><td>The data bus holds eight byte lanes assigned as shown below.
<table>
<tr><th>Data Bus Signals</th><th>Byte Lane</th></tr>
<tr><td>D[0–7]</td><td>0</td></tr>
<tr><td>D[8–15]</td><td>1</td></tr>
<tr><td>D[16–23]</td><td>2</td></tr>
<tr><td>D[24–31]</td><td>3</td></tr>
<tr><td>D[32–39]</td><td>4</td></tr>
<tr><td>D[40–47]</td><td>5</td></tr>
<tr><td>D[48–55]</td><td>6</td></tr>
<tr><td>D[56–63]</td><td>7</td></tr>
</table></td></tr>
<tr><td>Timing Comments</td><td>The number of times the data bus is driven depends on the transfer size, port size, and whether the transfer is a single-beat or burst operation.</td></tr>
<tr><td>State Meaning</td><td><i>Asserted/Deasserted</i>. Represents the state of data during a data write. Byte lanes not selected for data transfer do not supply valid data. MSC8113 duplicates data to enable valid data to be sent to different port sizes.</td></tr>
<tr><td>Timing Comments</td><td><i>Assertion/Deassertion</i>. Initial beat coincides with $\overline{\text{DBB}}$, for bursts, transitions on the bus clock cycle following each assertion of $\overline{\text{TA}}$ and, for port size, transitions on the bus clock cycle following each assertion of $\overline{\text{PSDVAL}}$.
<i>High Impedance</i>. Occurs on the bus clock cycle after the final assertion of $\overline{\text{TA}}$, $\overline{\text{TEA}}$, or certain $\overline{\text{ARTRY}}$ cases.</td></tr>
<tr><td rowspan="2">Input</td><td>State Meaning</td><td><i>Asserted/Deasserted</i>. Represents the state of data during a data read transaction.</td></tr>
<tr><td>Timing Comments</td><td><i>Assertion/Deassertion</i>. Data must be valid on the same bus clock cycle that $\overline{\text{TA}}$ and/or $\overline{\text{PSDVAL}}$ is asserted.</td></tr>
</table>

Table 13-8. Data Transfer Signals (Continued)

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td><b>DP[0–7]</b></td><td>Input/
Output</td><td>Data Bus Parity
The DP[0–7] signals have the same meanings in both internal-only mode and external master mode.
<b>Note:</b> The MSC8113 has either a 32-bit or a 64-bit external port for the system bus.</td></tr>
<tr><td></td><td>Output</td><td>State Meaning <i>Asserted/Deasserted</i>. Represents odd parity for each of eight bytes of data write transactions. Odd parity means that an odd number of bits, including the parity bit, are driven high. The signal assignments are listed in the following table.
<table>
<tr><th>Signal Name</th><th>Data Bus Signal Assignments</th></tr>
<tr><td>DP0</td><td>D[0–7]</td></tr>
<tr><td>DP1</td><td>D[8–15]</td></tr>
<tr><td>DP2</td><td>D[16–23]</td></tr>
<tr><td>DP3</td><td>D[24–31]</td></tr>
<tr><td>DP4</td><td>D[32–39]</td></tr>
<tr><td>DP5</td><td>D[40–47]</td></tr>
<tr><td>DP6</td><td>D[48–55]</td></tr>
<tr><td>DP7</td><td>D[56–63]</td></tr>
</table>
Timing Comments <i>Assertion/Deassertion</i>. The same as the data bus.
<i>High Impedance</i>. The same as the data bus.</td></tr>
<tr><td></td><td>Input</td><td>State Meaning <i>Asserted/Deasserted</i>. Represents odd parity for each byte of read data. Parity is checked on all data byte lanes, regardless of the size of the transfer.
Timing Comments <i>Assertion/Deassertion</i>. The same as D[0–63].</td></tr>
</table>

13.1.8 Data Transfer Termination

Data transfer termination signals are required after each data beat in a data transfer. In a single-beat transaction that is not a port-size transfer, the data transfer termination signals also indicate the end of the tenure. In burst or port size accesses, the data transfer termination signals apply to individual beats and indicate the end of the tenure only after the final data beat. For details on how these signals interact, see **Section 13.2.4**, *Data Tenure Operations*.

Table 13-9. Data Transfer Termination Signals

Name	Type	Description	
PSDVAL	Input/ Output	**Partial Data Valid Indication** TA asserts with PSDVAL to indicate the termination of the current transfer as well as each complete data beat in burst transactions.	
	Input	State Meaning	*Asserted*. Indicates that a beat data transfer completed successfully. Note that PSDVAL must be asserted for each data beat in a single beat, port size and burst transaction. See **Section 13.2.4.5**, *Port Size Data Bus Transfers and PSDVAL Termination*. *Deasserted*. (During DBB) indicates that, until PSDVAL is asserted, the MSC8113 must continue to drive the data for the current write or must wait to sample the data for reads.
		Timing Comments	*Assertion*. Must not occur before the cycle after the assertion of AACK for the current transaction (if the address retry mechanism is to be used to prevent invalid data from being used by the MSC8113); otherwise, assertion can occur at any time during the assertion of DBB. The system can withhold assertion of PSDVAL to indicate that the MSC8113 should insert wait states to extend the duration of the data beat. *Deassertion*. Must occur after the bus clock cycle of the final (or only) data beat of the transfer. For a burst and/or port size transfer, the system can assert PSDVAL for one bus clock cycle and then deassert it to insert wait states during the next beat.
	Output	State Meaning	*Asserted*. Indicates that the data has been latched for a write operation or that the data is valid for a read operation, thus terminating the current data beat. If it is the last or only data beat, this also terminates the data tenure. *Deasserted*. Indicates that the master must extend the current data beat (insert wait states) until data can be provided or accepted by the MSC8113 device.
		Timing Comments	*Assertion*. Occurs on the clock in which the current data transfer can be completed. *Deassertion*. Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, PSDVAL may be deasserted between beats to insert one or more wait states before the completion of the next beat.

Table 13-9. Data Transfer Termination Signals (Continued)

Name	Type	Description	
$\overline{\text{TA}}$	Input/ Output	**Transfer Acknowledge**	
	Input	State Meaning	*Asserted*. Indicates that a single-beat data transfer completed successfully or that a data beat in a burst transfer completed successfully. Note that $\overline{\text{TA}}$ must be asserted for each data beat in a burst transaction. For more information, see **Section 13.2.4.3**, *Data Bus Transfers and Normal Termination*. *Deasserted*. (During assertion of $\overline{\text{DBB}}$) indicates that, until $\overline{\text{TA}}$ is asserted, the MSC8113 must continue to drive the data for the current write or must wait to sample the data for reads.
		Timing Comments	*Assertion*. Must not occur before the cycle after the assertion of $\overline{\text{AACK}}$ for the current transaction if the address retry mechanism is to be used to prevent invalid data from being used by the MSC8113. Otherwise, assertion may occur at any time during the assertion of $\overline{\text{DBB}}$. The system can withhold assertion of $\overline{\text{TA}}$ to indicate that the MSC8113 should insert wait states to extend the duration of the data beat. *Deassertion*. Must occur after the bus clock cycle of the final (or only) data beat of the transfer. For a burst transfer, the system can assert $\overline{\text{TA}}$ for one bus clock cycle and then deassert it to advance the burst transfer to the next beat and insert wait states during the next beat.
	Output	State Meaning	*Asserted*. Indicates that the data has been latched for a write operation, or that the data is valid for a read operation, thus terminating the current data beat. If it is the last or only data beat, this also terminates the data tenure. *Deasserted*. Indicates that master must extend the current data beat (insert wait states) until data can be provided or accepted by the MSC8113 device.
		Timing Comments	*Assertion*. Occurs on the clock in which the current data transfer can be completed. *Deassertion*. Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, $\overline{\text{TA}}$ may be deasserted between beats to insert one or more wait states before the completion of the next beat.
$\overline{\text{TEA}}$	Input/Output	**Transfer Error Acknowledge**	
	Input	State Meaning	*Asserted*. Indicates that a bus error occurred. The assertion of $\overline{\text{TEA}}$ causes the deassertion/high impedance of $\overline{\text{DBB}}$ in the next clock cycle. However, data entering the MSC8113 internal memory resources such as GPRs or caches are not invalidated. *Deasserted*. Indicates that no bus error was detected.
		Timing Comments	*Assertion*. May be asserted while $\overline{\text{DBB}}$ is asserted and for the cycle after if $\overline{\text{TA}}$ is asserted during a read operation. $\overline{\text{TEA}}$ should be asserted for one cycle only. *Deassertion*. $\overline{\text{TEA}}$ must be deasserted no later than the deassertion of $\overline{\text{DBB}}$.
	Output	State Meaning	*Asserted*. Indicates that a bus error has occurred. Assertion of $\overline{\text{TEA}}$ terminates the transaction in progress; that is, asserting $\overline{\text{TA}}$ is unnecessary because it is ignored by the target device. An unsupported memory transaction, such as a direct-store access or a graphics read or write, causes the assertion of $\overline{\text{TEA}}$ (provided $\overline{\text{TEA}}$ is enabled and the address transfer matches the MSC8113 memory map). *Deasserted*. Indicates that no bus error was detected.
		Timing Comments	*Assertion*. Occurs on the first clock after the bus error is detected. *Deassertion*. Occurs one clock after assertion.

13.2 60x-Compatible Bus Protocols

This section describes the general 60x protocol for a 64-bit data bus. In the MSC8113 this protocol is true under the following conditions:

- The system bus has a 32-bit or 64-bit external data bus port. The internal part of the system bus is a 64-bit data bus.

Note: In 32-bit external data bus mode, an external master can perform only 32-bit single accesses to the internal address space.

- The system bus supports up to three external masters, as well as internal masters, using an internal arbiter, or it functions as a client of an external arbiter.
- The MSC8113 device has a local memory bus with 64-bit internal data bus. Internally, the SIU holds a 64-bit local bus.
- The MSC8113 local bus supports only internal masters with an internal arbiter and functions as a single master on the local memory bus.

13.2.1 System Bus Operating Modes

The system bus supports separate bus configurations for internal and external 60x-compatible bus masters.

- Single-MSC8113 bus mode connects external devices by using only the memory controller.
- The 60x-compatible bus mode enables connections to other masters and 60x-compatible slaves.

The figures in the following sections show how the MSC8113 is connected in these two configurations.

13.2.1.1 Single MSC8113 Bus Mode

In single-MSC8113 bus mode, the MSC8113 is the only bus device in the system. The internal memory controller controls all devices on the external bus. **Figure 13-2** shows the signal connections for single-MSC8113 bus mode. Notice that the MSC8113 uses the address bus as a memory address bus. Slaves cannot use the 60x-compatible bus signals because the addresses have memory timing, not address tenure timing.

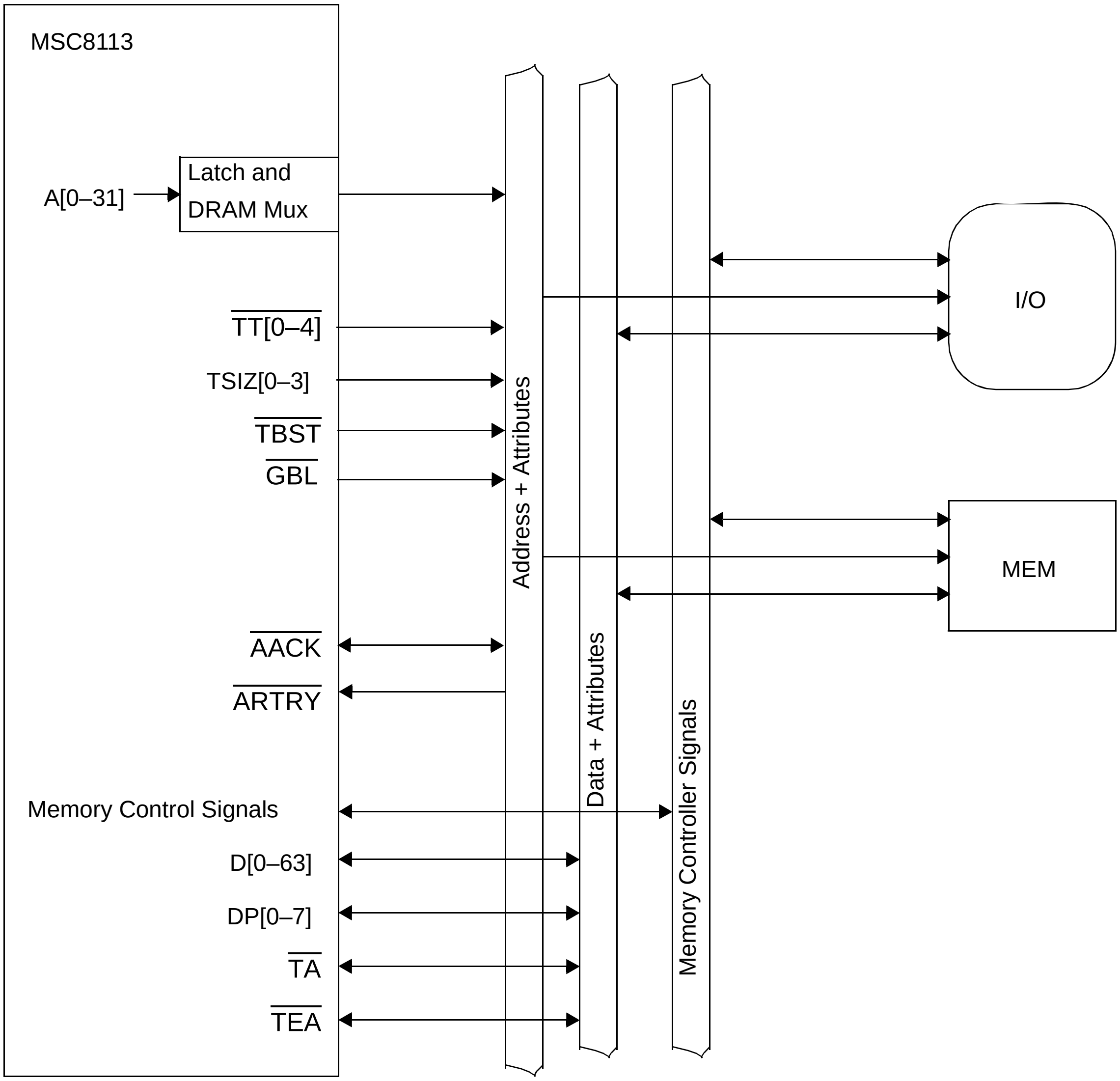


Figure 13-2. Single-MSC8113 Bus Mode

13.2.1.2 60x-Compatible Bus Mode

The 60x-compatible bus mode includes one or more potential external masters (for example, ASIC DMA controllers, high-end PowerQUICC II devices, and/or additional MSC81XXs). **Figure 13-3** shows how an external processor attaches to the MSC8113.

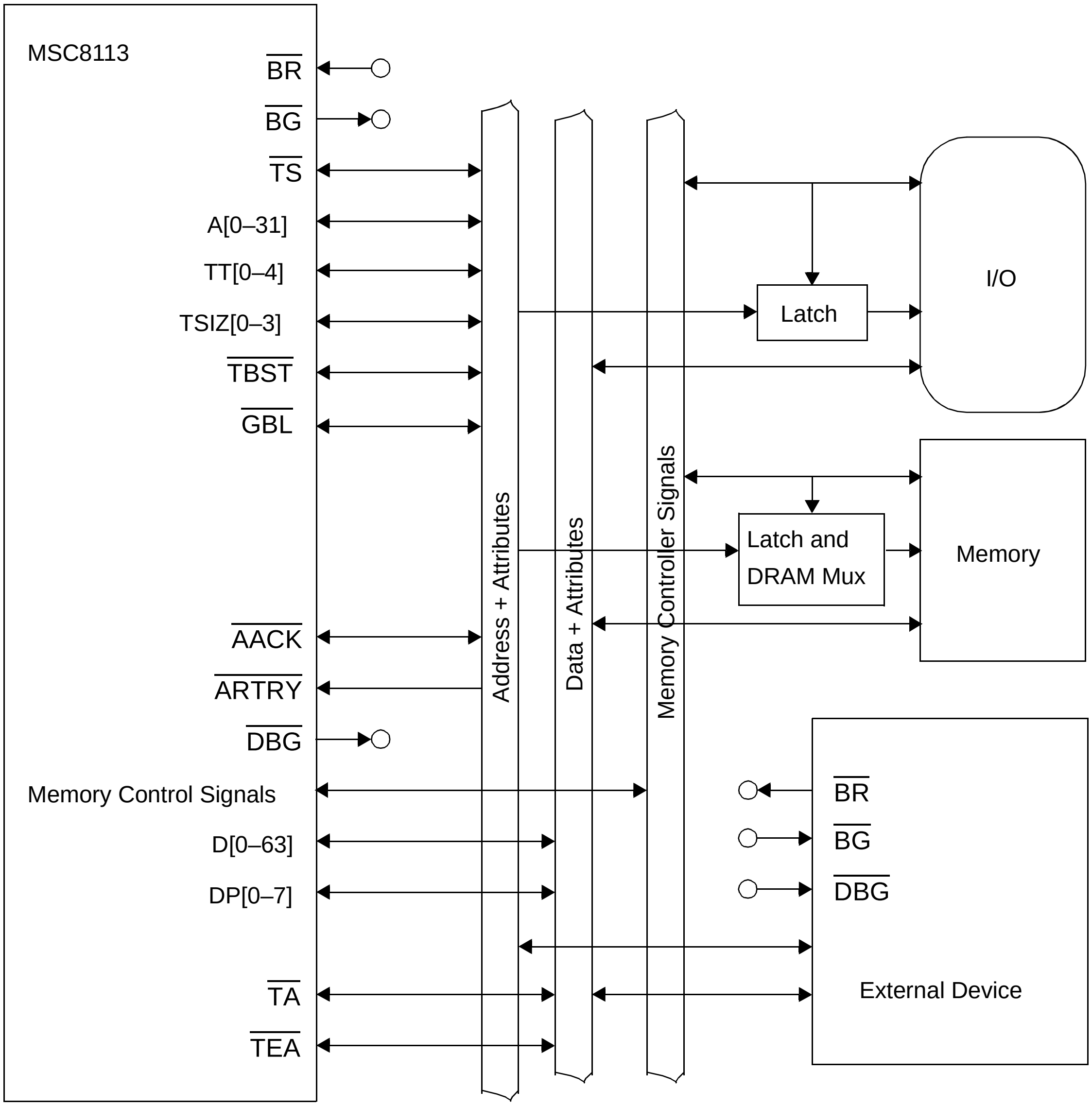


Figure 13-3. 60x-Compatible Bus Mode

13.2.2 System Bus Protocols

Typically, system bus accesses consist of address and data tenures, which in turn each consist of three phases—arbitration, transfer, and termination, as shown in **Figure 13-4**. The independence of the tenures is indicated by the data tenure overlap with the next address tenure, which allows split-bus transactions at the system level in multiprocessor systems. **Figure 13-4** shows a single-beat data transfer of up to 256 bits. Notice that the MSC8113 supports port sizes of 8, 16, 32 and 64 bits and requires the additional bus signal, $\overline{\text{PSDVAL}}$ (not defined by the 60x bus specification) to support different port sizes fully. For details, see **Section 13.2.4.5**.

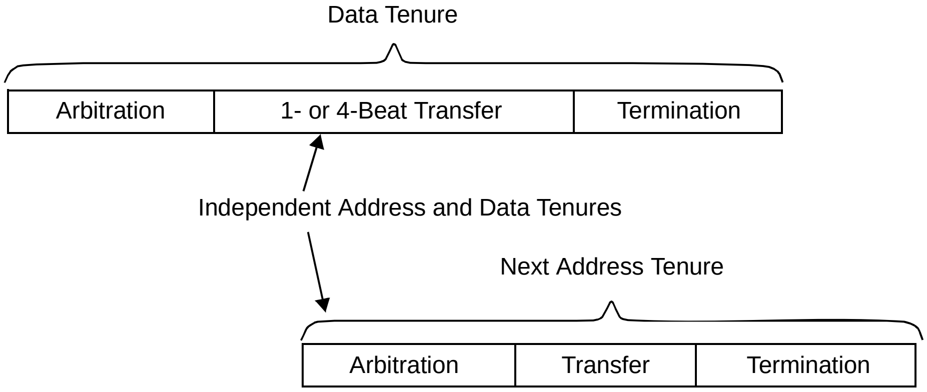


Figure 13-4. Basic Transfer Protocol

The basic functions of the address and data tenures are as follows:

- Address tenure:
  - — *Arbitration*. Address bus arbitration signals request and grant address bus mastership.
  - — *Transfer*. After a device is granted address bus mastership, it transfers the address. The address signals and the transfer attribute signals control the address transfer.
  - — *Termination*. After the address transfer, the system acknowledges that the address tenure is complete or that it must be repeated, signalled by the assertion of the address retry signal ($\overline{\text{ARTRY}}$).
- Data tenure:
  - — *Arbitration*. After address tenure begins, the bus device arbitrates for data bus mastership.
  - — *Transfer*. After the device is granted data bus mastership, it samples the data bus for read operations or drives the data bus for write operations.
  - — *Termination*. Acknowledgment of a successful data transfer is required after each beat in a data transfer. In single-beat transactions, the data termination signals also indicate the end of the tenure. In burst or port-size accesses, data termination signals indicate the completion of individual beats and, after the final data beat, the end of the tenure.

13.2.2.1 Arbitration Phase

The external system bus design permits one device, either the MSC8113 or a bus-attached external device, to be granted bus mastership at a time. Bus arbitration is handled either by an external central bus arbiter or by the internal arbiter. In the latter case, the system is optimized for three external 60x-compatible bus masters besides the MSC8113. The arbitration configuration (external or internal) is determined at system reset by sampling the Hard Reset Configuration Word (HRCW[EARB]). Alternatively, it is defined by programming PPC_ACR[EARB]. See **Section 4.2.1**.

The MSC8113 controls bus access through the bus request ($\overline{\text{BR}}$) and bus grant ($\overline{\text{BG}}$) signals. It determines the state of the address and data bus busy signals by monitoring data bus grant ($\overline{\text{DBG}}$), transfer start ($\overline{\text{TS}}$), address acknowledge ($\overline{\text{AACK}}$), and transfer acknowledge ($\overline{\text{TA}}$), and it qualifies them with address bus busy ($\overline{\text{ABB}}$), and data bus busy ($\overline{\text{DBB}}$).

The following signals are for address bus arbitration:

- $\overline{\text{BR}}$ (bus request). A device asserts $\overline{\text{BR}}$ to request address bus mastership.
- $\overline{\text{BG}}$ (bus grant). Assertion indicates that a bus device may, with proper qualification, assume mastership of the address bus. A qualified bus grant occurs when $\overline{\text{BG}}$ is asserted while $\overline{\text{ABB}}$ and address retry ($\overline{\text{ARTRY}}$) are deasserted.
- $\overline{\text{ABB}}$ (address bus busy). A device asserts $\overline{\text{ABB}}$ to indicate it is the current address bus master. Note that if all devices assert $\overline{\text{AACK}}$ with $\overline{\text{TS}}$ and would normally deassert $\overline{\text{ABB}}$ after $\overline{\text{AACK}}$ is asserted, the devices can ignore $\overline{\text{ABB}}$ because the MSC8113 can internally generate $\overline{\text{ABB}}$. The MSC8113 $\overline{\text{ABB}}$, if enabled, must be tied to a pull-up resistor.

The following signals are for data bus arbitration:

- $\overline{\text{DBG}}$ (data bus grant). Indicates that a bus device can, with the proper qualification, assume data bus mastership. A qualified data bus grant occurs when $\overline{\text{DBG}}$ is asserted while $\overline{\text{DBB}}$ and $\overline{\text{ARTRY}}$ are deasserted.
- $\overline{\text{DBB}}$ (data bus busy). Assertion by the device indicates that the device is the current data bus master. The device master always assumes data bus mastership if it needs the data bus and is given a qualified data bus grant (see $\overline{\text{DBG}}$). Note that if all devices assert $\overline{\text{DBB}}$ in conjunction with a qualified data bus grant and would normally deassert $\overline{\text{DBB}}$ after the last $\overline{\text{TA}}$ is asserted, the devices can ignore $\overline{\text{DBB}}$ because the MSC8113 generates $\overline{\text{DBB}}$ internally. The MSC8113 $\overline{\text{DBB}}$ signal, if enabled, must be tied to a pull-up resistor.

The following is a summary of rules for arbitration:

- Preference among devices is determined at the request level. The MSC8113 device supports 16 levels of bus requests (see **Section 4.2.1**).
- When no bus device requests the address bus, the MSC8113 device parks the device selected in the arbiter configuration register on the bus.

13.2.2.2 Address Pipelining and Split-Bus Transactions

The 60x bus protocol provides independent address and data bus capability to handle pipelined and split-bus transaction system organizations. Address pipelining allows the next address tenure to begin before the current data tenure finishes. Although this ability does not inherently reduce memory latency, support for address pipelining and split-bus transactions can greatly improve effective bus/memory throughput. These benefits are most fully realized in shared-memory, multiple-master implementations in which bus bandwidth is critical to system performance.

External arbitration, as provided by the MSC8113 device, is required in systems with multiple devices sharing the system bus. The MSC8113 uses the $\overline{\text{AACK}}$ signal to control pipelining. The MSC8113 supports both one- and zero-level bus pipelining. One-level pipelining is achieved by asserting $\overline{\text{AACK}}$ to the current address bus master and granting mastership of the address bus to the next requesting master before the current data bus tenure completes. Two address tenures can occur before the current data bus tenure completes. The MSC8113 device also supports non-pipelined accesses (see **Section 13.2.3.12**, *Pipeline Control*).

13.2.2.3 Memory Coherency

Asserting the global ($\overline{\text{GBL}}$) output signal indicates whether the current transaction must be snooped by other snooping devices on the bus. Address bus masters assert $\overline{\text{GBL}}$ to indicate that the current transaction is a global access (that is, an access to a memory shared by more than one device). If $\overline{\text{GBL}}$ is not asserted, that transaction is not snooped. When other devices detect the $\overline{\text{GBL}}$ input asserted, they respond by snooping any addresses broadcast. In MSC8113 a transaction with $\overline{\text{GBL}}$ indication is generated by one of the following sources:

- *SC140 cores*. The transaction is mapped to a cacheable area by EQBS programming (see **Section 9.3.9**, *EQBS Programming Model*).
- *DMA Controller*. There is an indication in each channel parameter.

Minimize the number of sources marked as global because the bus retry protocol for enforcing coherency may require significant bus bandwidth. See **Section 13.2.3.10**, *Address Retried With ARTRY Signal*, on page 13-33.

13.2.3 Address Tenure Operations

This section describes the three phases of the address tenure: address bus arbitration, address transfer, and address termination.

13.2.3.1 Address Arbitration

The arbitration configuration (external or internal) is chosen at system reset. For internal arbitration, the MSC8113 provides arbitration for the 60x-compatible address bus and the system is optimized for three external 60x-compatible bus masters besides the MSC8113. The bus request ($\overline{\text{BR}}$) for the external device is an external input to the arbiter. The bus grant signal for the external device ($\overline{\text{BG}}$) is an output to the external device. The $\overline{\text{BG}}$ signal is asserted by the MSC8113 internal arbiter one clock after the current master on the bus asserts $\overline{\text{AACK}}$. Therefore, it is a qualified $\overline{\text{BG}}$. Assuming that all potential masters deassert $\overline{\text{ABB}}$ one clock after receiving $\overline{\text{AACK}}$, the device receiving $\overline{\text{BG}}$ can start the address tenure by asserting $\overline{\text{TS}}$ one clock after receiving $\overline{\text{BG}}$. In addition to the external signals, there are internal request and grant signals for the MSC8113 internal devices.

Bus accesses are prioritized, with programmable priority. When an MSC8113 internal master needs the system bus, it asserts the internal bus request along with the request level. The arbiter asserts the internal 60x-compatible bus grant for the highest-priority request.

The MSC8113 supports address bus parking via the parked master bits in the arbiter configuration register. The MSC8113 parks the address bus (asserts the address bus grant signal in anticipation of an address bus request) to the external master or internal masters. When a device is parked, the arbiter can hold $\overline{\text{BG}}$ asserted for a device even if that device has not requested the bus. Therefore, when the parked device needs to perform a bus transaction, it skips the bus request delay and assumes address bus mastership on the next cycle. $\overline{\text{BR}}$ is not asserted, and the access latency is shortened by one cycle.

The MSC8113 and external device bus devices qualify $\overline{\text{BG}}$ by sampling $\overline{\text{ARTRY}}$ in the deasserted state prior to taking address bus mastership. The deassertion of $\overline{\text{ARTRY}}$ during the address retry window (one cycle after the assertion of $\overline{\text{AACK)}}$ indicates that no address retry is requested. If a device detects $\overline{\text{ARTRY}}$ asserted, it cannot accept an address bus grant during the $\overline{\text{ARTRY}}$ cycle or the cycle following. A device that asserts $\overline{\text{ARTRY}}$ asserts its bus request during the cycle after the assertion of $\overline{\text{ARTRY}}$ and assumes bus mastership when it is given a bus grant.

The series of address transfers in **Figure 13-5** shows the transfer protocol when the MSC8113 device is configured in 60x-compatible bus mode. In this example, the MSC8113 is initially parked on the bus with $\overline{\text{BG INT}}$ asserted, which lets it start an address bus tenure by asserting $\overline{\text{TS}}$. Note that $\overline{\text{BG INT}}$ is an internal signal that is not reflected externally. During the same clock cycle, the external master bus request is asserted to request access to the system bus, thereby causing the deassertion of $\overline{\text{BG INT}}$ internally and the assertion of $\overline{\text{BG}}$ externally. Following MSC8113 address tenure, the external master takes the bus and initiates its address transaction. The internal arbiter samples $\overline{\text{BR}}$ during the clock cycle in which $\overline{\text{AACK}}$ is asserted; if $\overline{\text{BR}}$ is not asserted (no pending request), it deasserts $\overline{\text{BG}}$ and asserts the parked bus grant ($\overline{\text{BG\_INT}}$ in this example). The master can assert $\overline{\text{BR}}$ and receive a qualified bus grant without subsequently using the bus. It can deassert (cancel) $\overline{\text{BR}}$ before accepting a qualified bus grant.

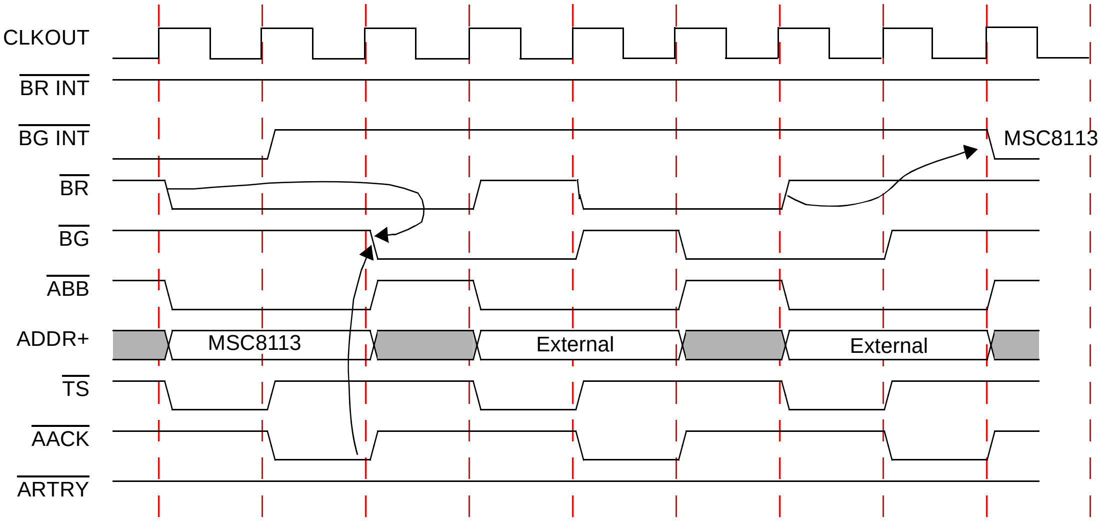


Figure 13-5. Address Bus Arbitration With External 60x-Compatible Bus Master

13.2.3.2 Address Pipelining

The MSC8113 device supports one-level address pipelining by asserting AACK to the current bus master when its data tenure starts and by granting the address bus to the next requesting device before the current data bus tenure completes. Address pipelining improves data throughput by allowing the memory-control hardware to decode a new set of address and control signals while the current data transaction finishes. The MSC8113 pipelines data bus operations in strict order with the associated address operations. **Figure 13-6** shows how address pipelining allows address tenures to overlap the associated data tenures.

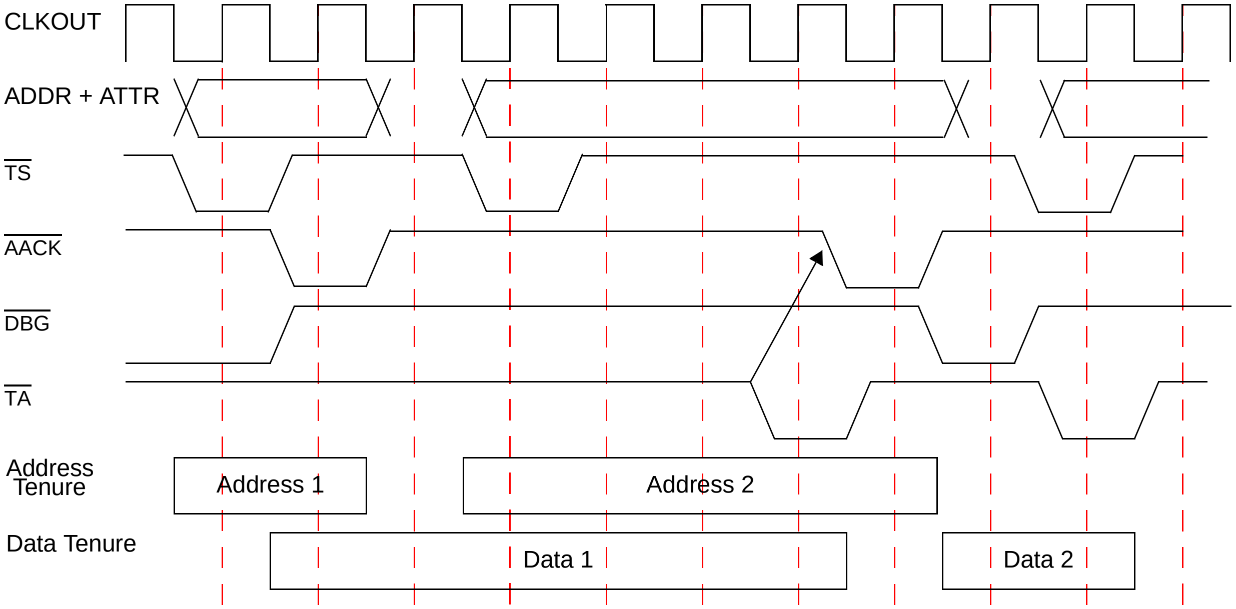


Figure 13-6. Address Pipelining

13.2.3.3 Address Transfer Attribute Signals

During the address transfer, the address is placed on the address signals, A[0–31]. The bus master provides the following signals that characterize the address transfer: transfer type (TT[0–4]), transfer code (TC[0–2]), transfer size (TSIZ[0–3]), and transfer burst ($\overline{\text{TBST}}$) signals. These signals are discussed in the following sections.

- *Transfer type signals (TT[0–4])*. The transfer type signals define the nature of the transfer requested (Read or Write). **Table 13-10** describes the MSC8113 action as master, slave, and snooper.

Table 13-10. Transfer Type Encoding

TT[0–4]^{1,2}	60x Bus Specification^3		MSC8113 as Bus Master		MSC8113 as Snooper	MSC8113 as Slave
	Command	**Transaction**	**Bus Transaction**	**Transaction Source**	**Action on Hit**	**Action on Slave Hit**
00010	Write	Single-beat or burst write	Single-beat or burst write	Master	Cancel reservation	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
01010	Read	Single-beat or burst read	Single-beat or burst read	Master	Not applicable to MSC8113	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
else	Reserved	—	Not applicable to MSC8113	Not applicable to MSC8113	Not applicable to MSC8113	Illegal

Notes:
1. TT1 can be interpreted as a read-versus-write indicator for the bus.
2. The MSC8113 can use a reduced mode in which only TT1 is an external signal. This mode is used in configurations with all external masters supporting TT1 only mode. The MSC8113 builds the complete transfer type internally.
3. This column specifies the TT encoding for the general 60x protocol. The processor generates or snoops only a subset of those encodings.

- *Transfer Code signals (TC[0–2])*. The transfer code signals give supplemental information about the corresponding address, mainly the source of the transaction, as listed in **Table 13-11**.

Note: The TCx signals can be used with the TT[0–4] and $\overline{\text{TBST}}$ signals to further define the current transaction.

Table 13-11. Transfer Code Encoding

TC[0–2]	System Bus	Local Bus
000	Reserved	System-Local bus bridge
001	DSI	DSI
010	Reserved	TDM
011	Ethernet Controller	Ethernet Controller
100	Reserved	Reserved
101	SC140 cores	Reserved

Table 13-11. Transfer Code Encoding (Continued)

TC[0–2]	System Bus	Local Bus
110	DMA	DMA
111	DMA	DMA

- *Transfer burst and size signals ($\overline{\text{TBST}}$ and TSIZ[0–3])*. These signals together indicate the size of the requested data transfer. The signals can be used with address bits A[27–31] and the device port size to determine which portion of the data bus contains valid data for a write transaction or which portion of the bus should contain valid data for a read transaction. The MSC8113 uses four 32-bit burst transactions for transferring cache blocks. For these transactions, TSIZ[0–3] are encoded as 0b0010, $\overline{\text{TBST}}$ is asserted, and address bits A[27–28] determine which 32 bits are sent first. The MSC8113 supports critical-first burst transactions (32-bit-aligned) from the processor. The MSC8113 transfers the critical 32 bits of data first, followed by 32 bits from increasing addresses, wrapping back to the beginning of the 8-level block as required.

Table 13-12. Transfer Size Encoding

$\overline{\text{TBST}}$	TSIZ[0–3]	Transfer Size	Comments
Deasserted	0001	1 Byte	—
Deasserted	0010	2 Bytes	—
Deasserted	0011	3 Bytes	—
Deasserted	0100	4 Bytes	—
Deasserted	0101	5 Bytes	Extended 5 bytes
Deasserted	0110	6 Bytes	Extended 6 bytes
Deasserted	0111	7 Bytes	Extended 7 bytes
Deasserted	0000	8 Bytes	Maximum data bus size
Deasserted	1001	16 Bytes	Extended 16 bytes
Deasserted	1010	24 Bytes	Extended 24 bytes
Asserted	0010	32 Bytes	Burst
Note: For details on extended mode, see **Section 13.2.3.8**, *Extended Transfer Mode,* on page 13-31			

The basic coherency size of the bus is 32 bytes for the processor. Data transfers that cross an aligned 32-byte boundary must present a new address to the bus at that boundary for proper snoop operation or must operate as non-coherent with respect to the MSC8113.

Note: In case of a 60x-compatible bus error or a local bus error, the TC and TT fields are captured in the SIU TESCR1 or L_TESRC1 registers, respectively. See the descriptions of TESCR1 and L_TESCR1 in **Section 4.2**, *SIU Programming Model*, on page 4-10.

13.2.3.4 Burst Ordering During Data Transfers

During burst data transfer operations, 32 bytes of data are transferred to or from the cache. Burst write transfers are performed zero 8-bytes-first. However, because burst reads are performed critical-8-bytes-first, a burst-read transfer may not start with the first 8 bytes. **Table 13-13** describes MSC8113 burst ordering.

Table 13-13. Burst Ordering

Data Transfer	8-Byte Starting Address			
	A[27–28] = 00[2]	A[27–28] = 01	A[27–28] = 10	A[27–28] = 11
First data beat[1]	8B0[3]	8B1	8B2	8B3
Second data beat	8B1	8B2	8B3	8B0
Third data beat	8B2	8B3	8B0	8B1
Fourth data beat	8B3	8B0	8B1	8B2

Notes:
1. Each data beat terminates with one valid assertion of $\overline{\text{TA}}$.
2. A[27–28] specifies the first 8 bytes of the 32-byte block being transferred; any subsequent 8 bytes must wrap around the block. A[29–31] are always 0b000 for burst transfers by the MSC8113.
3. DWx represents the 8 bytes that would be addressed by A[27–28] = *x* if a nonburst transfer was performed.

13.2.3.5 Effect of Alignment on Data Transfers

Table 13-14 lists the aligned transfers that can occur to and from the MSC8113. These are transfers in which the data is aligned to an address that is an integer multiple of the size of the data. For example, **Table 13-14** shows that 1-byte data is always aligned; however, a 4-byte data must reside at an address that is a multiple of four to be aligned.

Table 13-14. Aligned Data Transfers

Program Transfer Size	TSIZ[0–3]	A[29–31]	Data Bus Byte Lanes							
			D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
			B0	B1	B2	B3	B4	B5	B6	B7
Byte	0 0 0 1	0 0 0	OP0[1]	—[2]	—	—	—	—	—	—
	0 0 0 1	0 0 1	—	OP1	—	—	—	—	—	—
	0 0 0 1	0 1 0	—	—	OP2	—	—	—	—	—
	0 0 0 1	0 1 1	—	—	—	OP3	—	—	—	—
	0 0 0 1	1 0 0	—	—	—	—	OP4	—	—	—
	0 0 0 1	1 0 1	—	—	—	—	—	OP5	—	—
	0 0 0 1	1 1 0	—	—	—	—	—	—	OP6	—
	0 0 0 1	1 1 1	—	—	—	—	—	—	—	OP7

Table 13-14. Aligned Data Transfers (Continued)

Program Transfer Size	TSIZ[0–3]	A[29–31]	Data Bus Byte Lanes							
			D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
			B0	B1	B2	B3	B4	B5	B6	B7
2 Bytes	0 0 1 0	0 0 0	OP0	OP1	—	—	—	—	—	—
	0 0 1 0	0 1 0	—	—	OP2	OP3	—	—	—	—
	0 0 1 0	1 0 0	—	—	—	—	OP4	OP5	—	—
	0 0 1 0	1 1 0	—	—	—	—	—	—	OP6	OP7
4 Bytes	0 1 0 0	0 0 0	OP0	OP1	OP2	OP3	—	—	—	—
	0 1 0 0	1 0 0	—	—	—	—	OP4	OP5	OP6	OP7
8 Bytes	0 0 0 0	0 0 0	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7

Notes: **1.** OP*x*: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a 2-byte operand and OP7 is the least-significant byte.

2. —: These lanes are ignored during reads and driven with undefined data during writes.

The MSC8113 device supports misaligned memory operations, although they may degrade performance substantially. A misaligned memory address is not aligned to the size of the data being transferred. For example, it could be 4 bytes read from an odd byte address. The MSC8113 processor bus interface supports misaligned transfers within a 4-byte (32-bit aligned) boundary, as shown in **Table 13-15**. The four-byte transfer in **Table 13-15** is only one example of misalignment. As long as the attempted transfer does not cross a 4-byte boundary, the MSC8113 can transfer the data to the misaligned address within a single bus transfer—for example, 2 bytes read from an odd byte-aligned address. It takes two bus transfers to access data that crosses a 4-byte boundary. Because of the performance degradation, misaligned memory operations should be avoided. It is strongly recommended that you align code and data through software where possible.

Table 13-15. Unaligned Data Transfer Example (Four-Byte Example)

Program Size of 4 Bytes	TSIZ[1–3]	A[29–31]	Data Bus Byte Lanes							
			D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
			B0	B1	B2	B3	B4	B5	B6	B7
Aligned	1 0 0	0 0 0	A	A	A	A	—	—	—	—
Misaligned—First access	0 1 1	0 0 1	—	A	A	A	—	—	—	—
Second access	0 0 1	1 0 0	—	—	—	—	A	—	—	—
Misaligned—First access	0 1 0	0 1 0	—	—	A	A	—	—	—	—
Second access	0 1 0	1 0 0	—	—	—	—	A	A	—	—
Misaligned—First access	0 0 1	0 1 1	—	—	—	A	—	—	—	—
Second access	0 1 1	1 0 0	—	—	—	—	A	A	A	—

Table 13-15. Unaligned Data Transfer Example (Four-Byte Example) (Continued)

Program Size of 4 Bytes	TSIZ[1–3]	A[29–31]	Data Bus Byte Lanes							
			D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
			B0	B1	B2	B3	B4	B5	B6	B7
Aligned	1 0 0	1 0 0	—	—	—	—	A	A	A	A
Misaligned—First access	0 1 1	1 0 1	—	—	—	—	—	A	A	A
Second access	0 0 1	0 0 0	A	—	—	—	—	—	—	—
Misaligned—First access	0 1 0	1 1 0	—	—	—	—	—	—	A	A
Second access	0 1 0	0 0 0	A	A	—	—	—	—	—	—
Misaligned—First access	0 0 1	1 1 1	—	—	—	—	—	—	—	A
Second access	0 1 1	0 0 0	A	A	A	—	—	—	—	—
Note: A = Byte lane used; — = Byte lane not used										

13.2.3.6 Effect of Port Size on Data Transfers

The MSC8113 device transfers operands through its 32/64-bit data ports. If the internal memory controller performs the transfer, the MSC8113 supports 8-bit, 16-bit, 32-bit, and 64-bit data port sizes.

The MSC8113 system bus has either a 32-bit or a 64-bit external data bus port. The internal part of the system bus is a 64-bit data bus. The MSC8113 local bus has a 64-bit internal data bus. The bus requires that the portion of the data bus allocated for a transfer to or from a particular port size be fixed. A 64-bit port must reside on data bus bits D[0–63], a 32-bit port must reside on bits D[0–31], a 16-bit port must reside on bits D[0–15], and an 8-bit port must reside on bits D[0–7]. The MSC8113 always tries to transfer the maximum amount of data on all bus cycles: for a 2-byte operation, it always assumes that the port is 64 bits wide when beginning the bus cycle; for burst and extended byte cycles, a 64-bit bus is assumed. In **Figure 13-7**, **Table 13-16**, and **Table 13-17**, OP0 is the MSB of a 2-byte operand and OP7 is the LSB. **Figure 13-7** shows the device connections on the data bus.

Table 13-16 lists the bytes required on the data bus for read cycles.

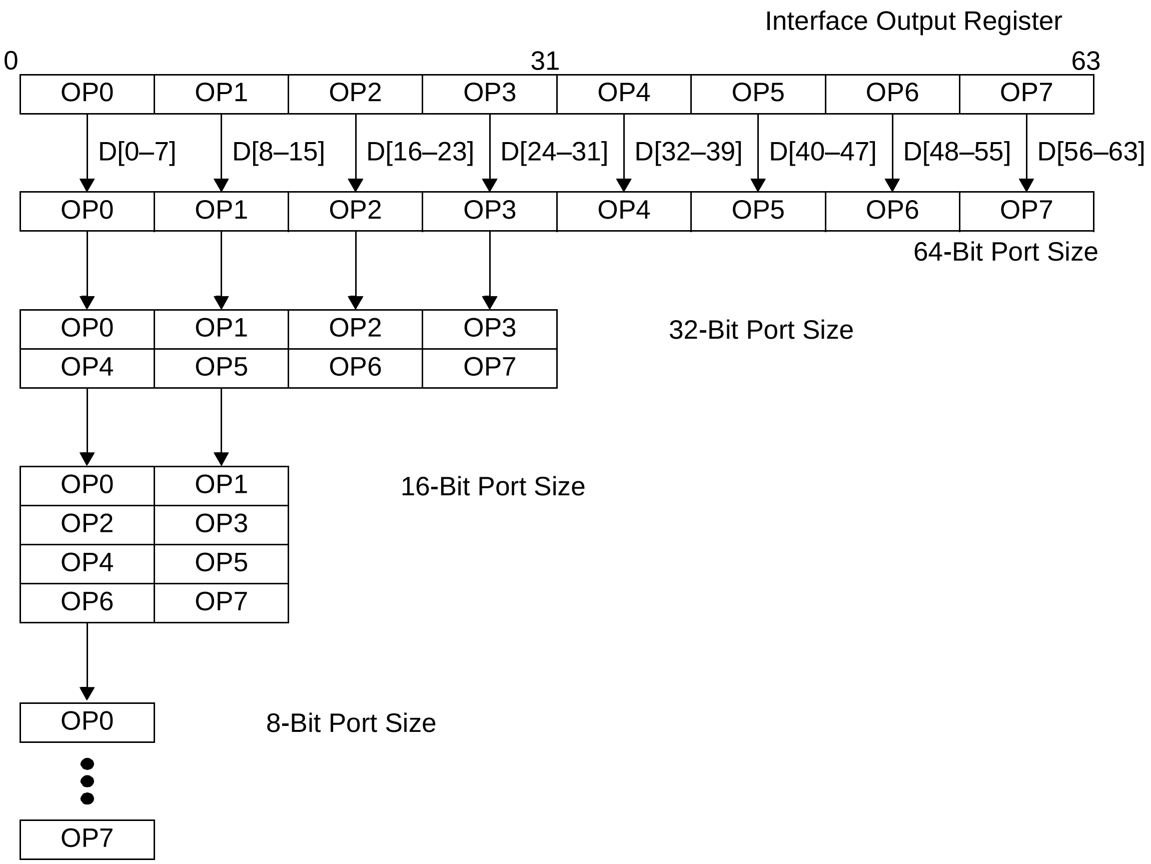


Figure 13-7. Interface to Different Port Size Devices

Table 13-16. Data Bus Requirements for Read Cycle

Transfer Size TSIZ[0–3]	Address State[1] A[29–31]	Port Size/Data Bus Assignments														
		64-Bit								32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	32–39	40–47	48–55	56–63	0–7	8–15	16–23	24–31	0–7	8–15	0–7
Byte (0001)	000	OP0[2]	—[3]	—	—	—	—	—	—	OP0	—	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	—	—	—	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	—	—	—	—	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	—	—	—	—	—	—	OP3	—	OP3	OP3
	100	—	—	—	—	OP4	—	—	—	OP4	—	—	—	OP4	—	OP4
	101	—	—	—	—	—	OP5	—	—	—	OP5	—	—	—	OP5	OP5
	110	—	—	—	—	—	—	OP6	—	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	—	—	—	—	OP7	—	—	—	OP7	—	OP7	OP7

Table 13-16. Data Bus Requirements for Read Cycle (Continued)

Transfer Size TSIZ[0–3]	Address State[1] A[29–31]	Port Size/Data Bus Assignments														
		64-Bit								32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	32–39	40–47	48–55	56–63	0–7	8–15	16–23	24–31	0–7	8–15	0–7
2 Bytes (0010)	000	OP0	OP1	—	—	—	—	—	—	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	—	—	—	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	—	—	—	—	—	—	OP2	OP3	OP2	OP3	OP2
	100	—	—	—	—	OP4	OP5	—	—	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	—	—	—	—	OP5	OP6	—	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	—	—	—	—	OP6	OP7	—	—	OP6	OP7	OP6	OP7	OP6
3 Bytes (0011)	000	OP0	OP1	OP2	—	—	—	—	—	OP0	OP1	OP2	—	OP0	OP1	OP0
	001	—	OP1	OP2	OP3	—	—	—	—	—	OP1	OP2	OP3	—	OP1	OP1
	100	—	—	—	—	OP4	OP5	OP6	—	OP4	OP5	OP6	—	OP4	OP5	OP4
	101	—	—	—	—	—	OP5	OP6	OP7	—	OP5	OP6	OP7	—	OP5	OP5
4 Bytes (0100)	000	OP0	OP1	OP2	OP3	—	—	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	—	—	—	—	OP4	OP5	OP6	OP7	OP4	OP5	OP6	OP7	OP4	OP5	OP4
8 Bytes (0000)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP0	OP1	OP2	OP3	OP0	OP1	OP0

Notes:
1. Address state is the calculated address for port size.
2. OPx: These lanes are read or written during that bus transaction. OP0 is the MSB of a 2-byte operand and OP7 is the LSB.
3. — Denotes a byte not required during that read cycle.

Table 13-17 lists data transfer patterns for write cycles for accesses initiated by the MSC8113 device.

Table 13-17. Data Bus Contents for Write Cycles

Transfer Size TSIZ[0–3]	Address State[1] A[29–31]	Data Bus Pattern[4]							
		0–7	**8–15**	**16–23**	**24–31**	**32–39**	**40–47**	**48–55**	**56–63**
Byte (0001)	000	OP0[2]	—[3]	—	—	—	—	—	—
	001	OP1	OP1	—	—	—	—	—	—
	010	OP2	—	OP2	—	—	—	—	—
	011	OP3	OP3	—	OP3	—	—	—	—
	100	OP4	—	—	—	OP4	—	—	—
	101	OP5	OP5	—	—	—	OP5	—	—
	110	OP6	—	OP6	—	—	—	OP6	—
	111	OP7	OP7	—	OP7	—	—	—	OP7
2 Bytes (0010)	000	OP0	OP1	—	—	—	—	—	—
	001	OP1	OP1	OP2	—	—	—	—	—
	010	OP2	OP3	OP2	OP3	—	—	—	—
	100	OP4	OP5	—	—	OP4	OP5	—	—
	101	OP5	OP5	OP6	—	—	OP5	OP6	—
	110	OP6	OP7	OP6	OP7	—	—	OP6	OP7
3 Bytes (0011)	000	OP0	OP1	OP2	—	—	—	—	—
	001	OP1	OP1	OP2	OP3	—	—	—	—
	100	OP4	OP5	OP6	—	OP4	OP5	OP6	—
	101	OP5	OP5	OP6	OP7	—	OP5	OP6	OP7
4 Bytes (0100)	000	OP0	OP1	OP2	OP3	—	—	—	—
	100	OP4	OP5	OP6	OP7	OP4	OP5	OP6	OP7
8 Bytes (0000)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7

Notes:
1. Address state is the calculated address for port size.
2. OPx: These lanes are read or written during that bus transaction. OP0 is the MSB of a 2-byte operand and OP7 is the LSB.
3. — Denotes a byte not driven during that write cycle.
4. Including the required duplications for 8/16/32/64-bit port size, according to the definition in **Table 13-16**.

13.2.3.7 60x-Compatible System Bus Mode—Size Calculation

To comply with the requirements listed in **Table 13-16** and **Table 13-17**, the transfer size and a new address must be calculated at the termination of each beat of a port-size transaction. In single-MSC8113 bus mode, these address and size calculations are internal and do not constrain the system. In 60x-compatible bus mode, the external slave or master must determine the new address and size. **Table 13-18** describes the address and size calculation state machine. Note that the address and size states are for internal use and are not transferred on the address or TSIZ values. Extended transactions (16-byte and 24-byte) are not described here but can be determined by extending this table for 9-byte, 10-byte, 16-byte, 23-byte, and 24-byte transactions.

Table 13-18. Address and Size State Calculations

Size State	Address State[0–4]					Port Size	Next Size State	Next Address State[0–4]				
Byte	x	x	x	x	x	x	Stop					
2 Bytes	x	x	x	x	0	Byte	Byte	x	x	x	x	1
	x	x	0	0	1		Byte	x	x	0	1	0
	x	x	1	0	1		Byte	x	x	1	1	0
	x	x	x	0	1	2 Bytes	Byte	x	x	x	1	0
	x	x	x	x	0		Stop					
3 Bytes	x	x	0	0	0	Byte	2 Bytes	x	x	0	0	1
	x	x	0	0	1		2 Bytes	x	x	0	1	0
	x	x	1	0	0		2 Bytes	x	x	1	0	1
	x	x	1	0	1		2 Bytes	x	x	1	1	0
	x	x	0	0	0	2 Bytes	Byte	x	x	0	1	0
	x	x	0	0	1		2 Bytes	x	x	0	1	0
	x	x	1	0	0		Byte	x	x	1	1	0
	x	x	1	0	1		2 Bytes	x	x	1	1	0
	x	x	x	x	x	4 Bytes	Stop					
4 Bytes	x	x	x	0	0	Byte	3 Bytes	x	x	x	0	1
	x	x	x	0	0	2 Bytes	2 Bytes	x	x	x	1	0
	x	x	x	x	x	4 Bytes	Stop					
5 Bytes	x	x	0	1	1	Byte	4 Bytes	x	x	1	0	0
6 Bytes	x	x	0	1	0	Byte	5 Bytes	x	x	0	1	1
	x	x	0	1	0	2 Bytes	4 Bytes	x	x	1	0	0
7 Bytes	x	x	0	0	1	Byte	6 Bytes	x	x	0	1	0
8 Bytes	x	x	0	0	0	Byte	7 Bytes	x	x	0	0	1
	x	x	0	0	0	2 Bytes	6 Bytes	x	x	0	1	0
	x	x	0	0	0	4 Bytes	4 Bytes	x	x	1	0	0
	x	x	0	0	0	8 Bytes	Stop					

13.2.3.8 Extended Transfer Mode

The MSC8113 extended transfer mode improves bus performance. This mode should not be confused with the extended bus protocol for direct-store operations in some earlier processors. The MSC8113 device generates 5-byte, 6-byte, 7-byte, 16-byte, or 24-byte extended transfers. These transactions are compatible with the 60x-compatible system bus, but some slaves or masters do not support these features. To disable this type of transaction, clear BCR[ETM]. This places the MSC8113 in strict 60x-Compatible bus mode. The following tables are extensions to **Table 13-16**, **Table 13-17**, and **Table 13-18**.

Table 13-19 lists the bytes required on the data bus for extended read cycles. The 16-byte and 24-byte transfers are always 8-byte aligned and use a maximum 64-bit port size.

Table 13-19. Data Bus Requirements for Extended Read Cycles

Transfer Size TSIZ[0–3]	Address State A[29–31]	Port Size/Data Bus Assignments														
		64-Bit								32-Bit				16-Bit		8- Bit
		0–7	8–15	16–23	24–31	32–39	40–47	48–55	56–63	0–7	8–15	16–23	24–31	0–7	8–15	0–7
5 Bytes (0101)	000	OP0	OP1	OP2	OP3	OP4	—	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	011	—	—	—	OP3	OP4	OP5	OP6	OP7	—	—	—	OP3	—	OP3	OP3
6 Bytes (0110)	000	OP0	OP1	OP2	OP3	OP4	OP5	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	010	—	—	OP2	OP3	OP4	OP5	OP6	OP7	—	—	OP2	OP3	OP2	OP3	OP2
7 Bytes (0111)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	001	—	OP1	OP2	OP3	OP4	OP5	OP6	OP7	—	OP1	OP2	OP3	—	OP1	OP1

Table 13-20 lists the patterns of the extended data transfer for write cycles when the MSC8113 initiates an access. The 16-byte and 24-byte transfers are always 8-byte aligned and use a maximum 64-bit port size.

Table 13-20. Data Bus Contents for Extended Write Cycles

Transfer Size TSIZ[0–3]	Address State A[29–31]	Data Bus Pattern							
		D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
5 Bytes (0101)	000	OP0	OP1	OP2	OP3	OP4	—	—	—
	011	OP3	OP3	—	OP3	OP4	OP5	OP6	OP7
6 Bytes (0110)	000	OP0	OP1	OP2	OP3	OP4	OP5	—	—
	010	OP2	OP3	OP2	OP3	OP4	OP5	OP6	OP7
7 Bytes (0111)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	—
	001	OP1	OP1	OP2	OP3	OP4	OP5	OP6	OP7

Table 13-21 includes added states to the transfer size calculation state machine. Only extended transfers use these states. Extended transfer mode is enabled by setting BCR[ETM].

Table 13-21. Address and Size State for Extended Transfers

Size State [0–3]	Address State[0–4]					Port Size	Next Size State [0–3]	Next Address State[0–4]				
1 Byte	x	x	x	1	1	Byte	Byte	x	x	1	0	0
	x	x	1	0	1			x	x	1	1	0
	x	x	x	x	x	Byte	Stop					
3 Bytes	x	x	0	1	0	Byte	Byte	x	x	0	1	1
	x	x	1	0	0			x	x	1	0	1
	x	x	0	1	0	Byte	Byte	x	x	1	0	0
	x	x	1	0	0			x	x	1	1	0
2 bytes	x	x	0	0	1	Byte	3 Bytes	x	x	0	1	0
	x	x	0	1	1			x	x	1	0	0
5 Bytes	x	x	0	0	0	Byte	2 Bytes	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	1	0			x	x	0	1	1
	x	x	0	1	1			x	x	1	0	0
	x	x	0	0	0	Byte	3 Bytes	x	x	0	1	0
	x	x	0	1	0			x	x	1	0	0
	x	x	0	1	1		2 Bytes	x	x	1	0	0
	x	x	0	0	0	2 Bytes	Byte	x	x	1	0	0
	x	x	0	1	1		2 Bytes	x	x	1	0	0
	x	x	x	x	x	4 Bytes	Stop					
6 Bytes	x	x	0	0	0	Byte	5 Bytes	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	1	0			x	x	0	1	1
	x	x	0	0	0	Byte	2 Bytes	x	x	0	1	0
	x	x	0	1	0			x	x	1	0	0
	x	x	0	0	0	2 Bytes	Byte	x	x	1	0	0
	x	x	0	1	0		2 Bytes	x	x	1	0	0
	x	x	x	x	x	4 Bytes	Stop					

Table 13-21. Address and Size State for Extended Transfers (Continued)

Size State [0–3]	Address State[0–4]					Port Size	Next Size State [0–3]	Next Address State[0–4]				
7 Bytes	x	x	0	0	0	Byte	6 Bytes	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	0	0	Byte	5 Bytes	x	x	0	1	0
	x	x	0	0	1		6 Bytes	x	x	0	1	0
	x	x	0	0	0	2 Bytes	3 Bytes	x	x	1	0	0
	x	x	0	0	1		4 Bytes	x	x	1	0	0
	x	x	x	x	x	4 Bytes	Stop					

13.2.3.9 Address Transfer Termination

Address transfer termination occurs with the assertion of $\overline{\text{AACK}}$ or $\overline{\text{ARTRY}}$. $\overline{\text{ARTRY}}$ remains asserted until one clock after $\overline{\text{AACK}}$; the bus clock cycle after $\overline{\text{AACK}}$ is called the $\overline{\text{ARTRY}}$ window. The MSC8113 uses $\overline{\text{AACK}}$ to enforce a pipeline depth of one to its internal slaves. The MSC8113 controls assertion of $\overline{\text{AACK}}$ unless an external slave claims the cycle. The external slave asserts $\overline{\text{AACK}}$ for one clock cycle and then deasserts it for one clock cycle before entering a high-impedance state. The MSC8113 holds $\overline{\text{AACK}}$ in a high-impedance state until it is required to assert $\overline{\text{AACK}}$ to terminate the address cycle.

13.2.3.10 Address Retried With $\overline{\text{ARTRY}}$ Signal

The address transfer can be terminated with the requirement to retry if $\overline{\text{ARTRY}}$ is asserted during the address tenure and through the cycle following $\overline{\text{AACK}}$. The assertion causes the entire transaction (address and data tenure) to repeat. As a bus master, the MSC8113 responds to an assertion of $\overline{\text{ARTRY}}$ by aborting the bus transaction and requesting the bus again, as shown in **Figure 13-8**. Note that after recognizing an assertion of $\overline{\text{ARTRY}}$ and aborting the current transaction, the MSC8113 cannot run the same transaction until the next time the bus is granted.

As a bus master, the MSC8113 recognizes either an early or qualified $\overline{\text{ARTRY}}$ and prevents the data tenure associated with the retried address tenure. If the data tenure has begun, the MSC8113 terminates the data tenure immediately even if the burst data has been received. If the assertion of $\overline{\text{ARTRY}}$ is received up to or on the same bus cycle as the first (or only) assertion of $\overline{\text{TA}}$ for the data tenure, the MSC8113 ignores the first data beat. If it is a read operation, the MSC8113 does not forward data internally to the MSC8113 internal storage. This retry mechanism allows the memory system to begin operating in parallel with the bus snoopers, provided external devices do not present data sooner than the bus cycle before all snoop responses can be determined and asserted on the bus.

The system must ensure that the first (or only) assertion of TA does not occur sooner than the cycle of the first assertion of ARTRY on the bus, or conversely, that ARTRY is never asserted later than the cycle of the first or only assertion of TA. This guarantees the relationship between TA and ARTRY such that, in case of an address retry, the data can be cancelled in the device before it can be forwarded to the internal storage locations. Generally, the memory system must also detect this event and abort any transfer in progress. If this TA/ARTRY relationship is not met, the device master may enter an undefined state. You can use PPC_ACR[DBGD] to ensure correct operation of the system.

During the clock of a qualified ARTRY, each device master determines whether it should deassert BR and ignore BG on the following cycle. The following cycle is the window-of-opportunity for the snooping master. During this window, only the snooping master that asserted ARTRY and requires a snoop copyback operation is allowed to assert BR. This guarantees the snooping master a window of opportunity to request and be granted the bus before the just-retried master can restart its transaction. BG is also blocked in the window-of-opportunity, so the arbiter has a chance to deassert BG to an already granted potential bus master to perform a new arbitration.

13.2.3.11 Address Tenure Timing Configuration

During address tenures initiated by 60x-compatible bus devices, the timing of the MSC8113 assertion of AACK is determined by the BCR[APD] bit and the pipeline status of the system bus. Because the MSC8113 device can support one level of pipelining, it uses AACK to control the system bus pipeline condition. To maintain the one-level pipeline, AACK is not asserted for a pipelined address tenure until the current data tenure ends. The MSC8113 device also delays asserting AACK until no more address retry conditions can occur. The earliest the MSC8113 can assert AACK is the clock cycle when the wait-state values set by BCR[APD] have expired.

BCR[APD] specifies the minimum number of address tenure wait states for address operations initiated by 60x-compatible bus devices. BCR[APD] indicates how many cycles the MSC8113 should wait for ARTRY, but because it is assumed that ARTRY can be asserted (by other masters) only on cacheable address spaces, BCR[APD] is considered only on transactions that hit a 60*x*-assigned memory controller bank and have GBL asserted during the address phase.

Extra wait states may occur because of other MSC8113 configuration parameters. In systems with multiple potential masters, the number of wait states configured by BCR[APD] should be at least as large as the value the slowest master would need to assert a snoop response. For example, additional wait states are required when the internal processor is running in 1:1 clock mode; this case requires at least one wait state to generate the ARTRY response.

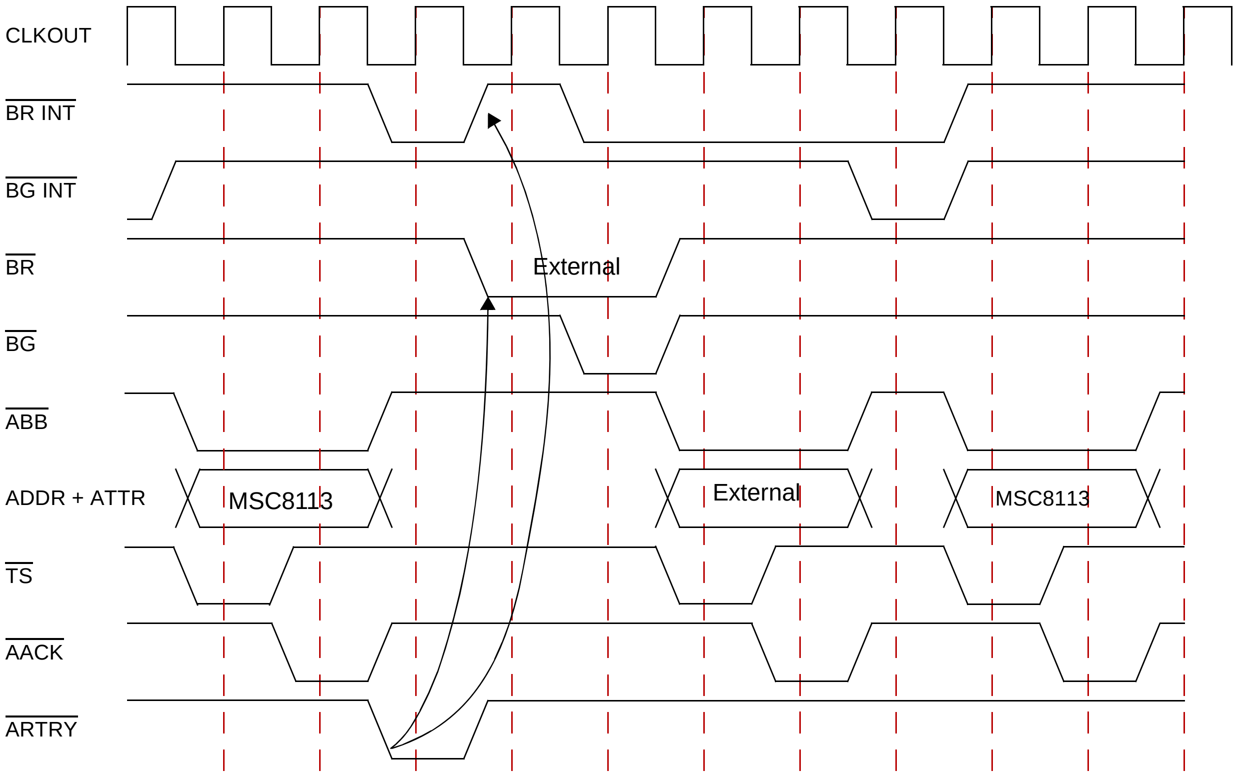


Figure 13-8. Retry Cycle

13.2.3.12 Pipeline Control

The MSC8113 device supports the following two modes:

- *One-level pipeline mode*. To maintain the one-level pipeline, $\overline{\text{AACK}}$ is not asserted for a pipelined address tenure until the current data tenure ends. In 60x-compatible bus mode, a two-level pipeline depth can occur (for example, when an external 60x-bus slave does not support one-level pipelining). When the internal arbiter counts a pipeline depth of two (two assertions of $\overline{\text{AACK}}$ before the assertion of the current data tenure), it deasserts all address ($\overline{\text{BG}}$) signals.
- *No-pipeline mode*. The MSC8113 does not assert $\overline{\text{AACK}}$ until the corresponding data tenure ends.

The pipeline mode of operation is determined by the BCR[PLDP] bit. See **Section 4.2**, *SIU Programming Model*, on page 4-10.

13.2.4 Data Tenure Operations

This section describes the operation of the MSC8113 device during the data bus arbitration, transfer, and termination phases of the data tenure.

13.2.4.1 Data Bus Arbitration

The beginning of an address transfer, marked by the assertion of transfer start ($\overline{\text{TS}}$), is also an implicit data bus request if the transfer type signals (TT[0–4]) indicate that the transaction is not address-only. (The address-only transactions are not applicable for MSC8113). The MSC8113 arbiter supports three external masters and uses $\overline{\text{DBG}}$ signals to grant the external master data bus. The $\overline{\text{DBG}}$ signals are not asserted if the data bus, which is shared with memory, is busy with a transaction. A qualified data bus grant occurs if $\overline{\text{DBG}}$ is asserted while the data bus operation signals $\overline{\text{DBB}}$ and $\overline{\text{ARTRY}}$ are deasserted. The MSC8113 arbiter should assert $\overline{\text{DBG}}$ only when the first $\overline{\text{TA}}$ is asserted with or after the associated $\overline{\text{ARTRY}}$. The MSC8113 $\overline{\text{DBG}}$ is asserted with $\overline{\text{TS}}$ if the data bus is free and if PPC_ACR[DBGD] = 0. If PPC_ACR[DBGD] = 1 and the data bus is not busy, $\overline{\text{DBG}}$ is asserted one cycle after $\overline{\text{TS}}$. The $\overline{\text{DBG}}$ delay should ensure that $\overline{\text{ARTRY}}$ is not asserted after the first or only $\overline{\text{TA}}$ assertion. For the programming model, see the discussion of the 60x Bus Arbiter Configuration Register (PPC_ACR) in **Section 4.2.1**, *System Configuration and Protection Registers*.

Note: $\overline{\text{DBB}}$ should not be asserted after the data tenure is finished. Assertion of $\overline{\text{DBB}}$ after the last $\overline{\text{TA}}$ causes improper operation of the bus. MSC8113 internal masters do not assert $\overline{\text{DBB}}$ after the last $\overline{\text{TA}}$.

If the data bus is not busy with the data of a previous transaction on the bus, the external arbiter must assert $\overline{\text{DBG}}$ in the same cycle in which $\overline{\text{TS}}$ is asserted (by a master that was granted the bus) or in the following cycle. If the external arbiter asserts $\overline{\text{DBG}}$ on the cycle in which $\overline{\text{TS}}$ is asserted, PPC_ACR[DBGD] should be set to zero. Otherwise, PPC_ACR[DBGD] should be set to one. External masters connected to the system bus must assert $\overline{\text{DBB}}$ only for the duration of their data tenure. External masters should not use $\overline{\text{DBB}}$ to prevent other masters from using the data bus after their data tenure has ended.

13.2.4.2 Data Streaming Mode

A special MSC8113 data streaming mode improves bus performance in some conditions. Generally, the bus protocol requires one idle cycle between any two data tenures to prevent contention on the data bus when the driver of the data is changing. However, when the driver on the data bus is the same for both data tenures, this idle cycle can be omitted. In data streaming mode, the MSC8113 omits the idle cycle where possible. MSC8113 applications often require data stream transfers of more than 4 × 64 bits. For example, the ATM cell payload is 6 × 64 (12 × 32) bits. All this data is driven from a single device on the bus, so data-streaming saves a cycle for such a transfer. When data-streaming mode is enabled, transactions initiated by bus masters within the device omit the idle cycle if the data driver is the same. Note that data streaming mode cannot be enabled when the MSC8113 is in 60x-compatible bus mode and a device that uses $\overline{\text{DBB}}$ is connected to the bus. This restriction is necessary because MSC8113 in data streaming mode may leave $\overline{\text{DBB}}$ asserted after the last $\overline{\text{TA}}$ of a transaction, thus violating the strict bus protocol. Data streaming mode is enabled by setting BCR[ETM].

13.2.4.3 Data Bus Transfers and Normal Termination

The data transfer signals include D[0–63] and DP[0–7]. For memory accesses, the data signals form a 64-bit data path, D[0–63], for read and write operations. The MSC8113 handles data transfers in either single-beat or burst operations. Single-beat operations transfer from 1 to 24 bytes of data at a time. Burst operations always transfer 256 bits in four 64-bit beats. A burst transaction is indicated when the bus master asserts $\overline{\text{TBST}}$. A transaction terminates normally when $\overline{\text{TA}}$ is asserted.

The $\overline{\text{TA}}$, $\overline{\text{TEA}}$, and $\overline{\text{ARTRY}}$ signals terminate the individual data beats of the data tenure and the data tenure itself:

- $\overline{\text{TA}}$ indicates normal termination of data transactions. It must always be asserted on the bus cycle coincident with the data that it is qualifying. The slave can withhold it for any number of clocks until valid data is ready to be supplied or accepted.
- Asserting $\overline{\text{TEA}}$ indicates a nonrecoverable bus error event. Upon receiving a final (or only) termination condition, the MSC8113 always deasserts $\overline{\text{DBB}}$ for one cycle, except when fast data bus grant is performed.
- Asserting $\overline{\text{ARTRY}}$ causes the data tenure to terminate immediately if the $\overline{\text{ARTRY}}$ is for the address tenure associated with the data tenure in operation (the data tenure may not be terminated due to address pipelining). The earliest allowable assertion of $\overline{\text{TA}}$ depends directly on the latest possible assertion of $\overline{\text{ARTRY}}$.

Figure 13-9 shows both a single-beat and burst data transfer. The MSC8113 asserts $\overline{\text{TA}}$ to mark the cycle in which data is accepted. In a normal burst transfer, the fourth assertion of $\overline{\text{TA}}$ signals the end of a transfer.

13.2.4.4 Effect of ARTRY Assertion on Data Transfer and Arbitration

The MSC8113 device allows an address tenure to overlap with its associated data tenure. The MSC8113 internally guarantees that the first TA of the data tenure is delayed to be at the same time or after the ARTRY window (the clock after the assertion of AACK).

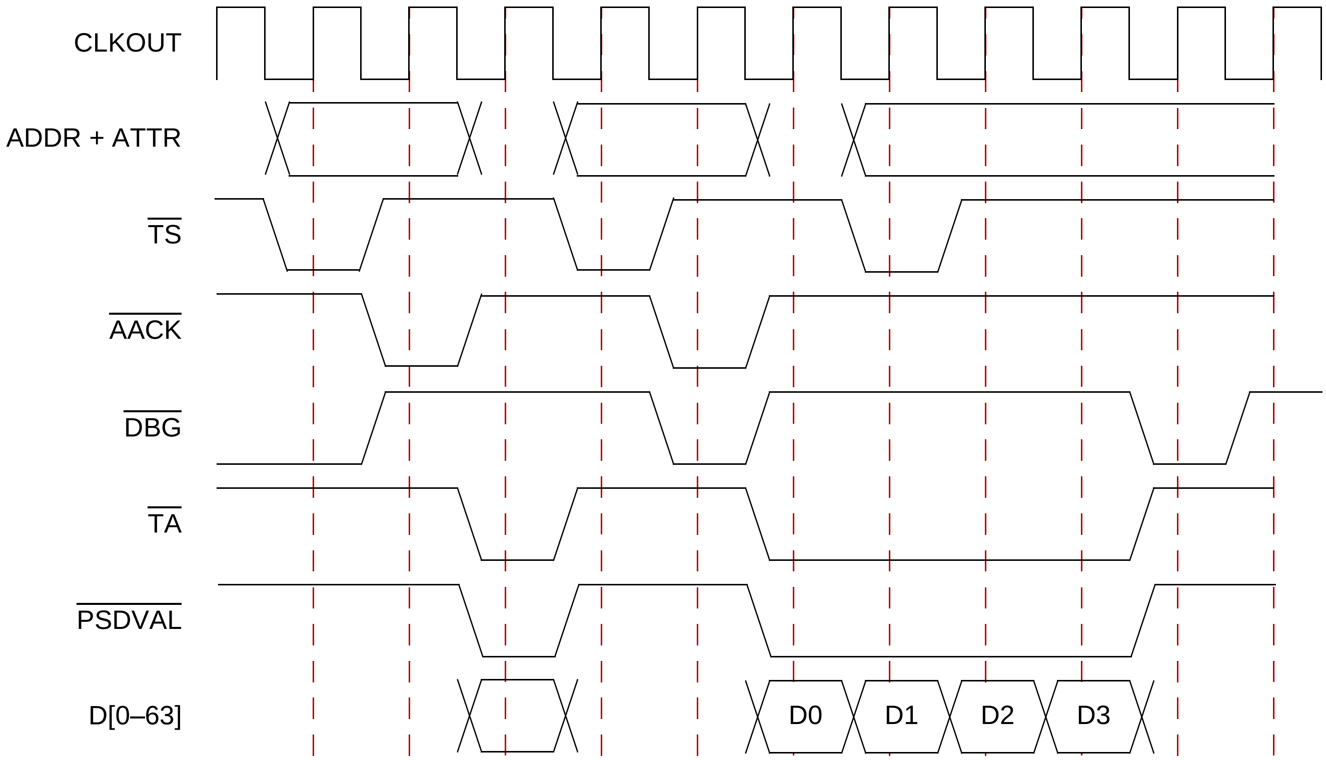


Figure 13-9. Single-Beat and Burst Data Transfers

13.2.4.5 Port Size Data Bus Transfers and PSDVAL Termination

The MSC8113 device transfers data via data ports of 8, 16, 32 and 64 bits, as shown in **Section 13.2.3.3**, *Address Transfer Attribute Signals*, on page 13-22. Single-beat transaction sizes can be 8, 16, 32, 64, 128, and 192 bits; burst transactions are 256 bits. Single-beat and burst transactions are divided into a number of intermediate beats depending on the port size. The MSC8113 asserts PSDVAL to mark the cycle in which data is accepted. Assertion of PSDVAL in conjunction with TA marks the end of the transfer in single-beat mode. The eight assertions of PSDVAL in conjunction with TA signals the end of a burst transfer. **Figure 13-10** shows an extended transaction of four 8-byte sets to a port size of 32 bits. The single-beat transaction is translated to four port-sized beats.

Figure 13-11 shows a burst transfer to a 32-bit port. Each 8-byte burst beat is divided into two port-sized beats so that the four 8-byte sets are transferred in eight beats.

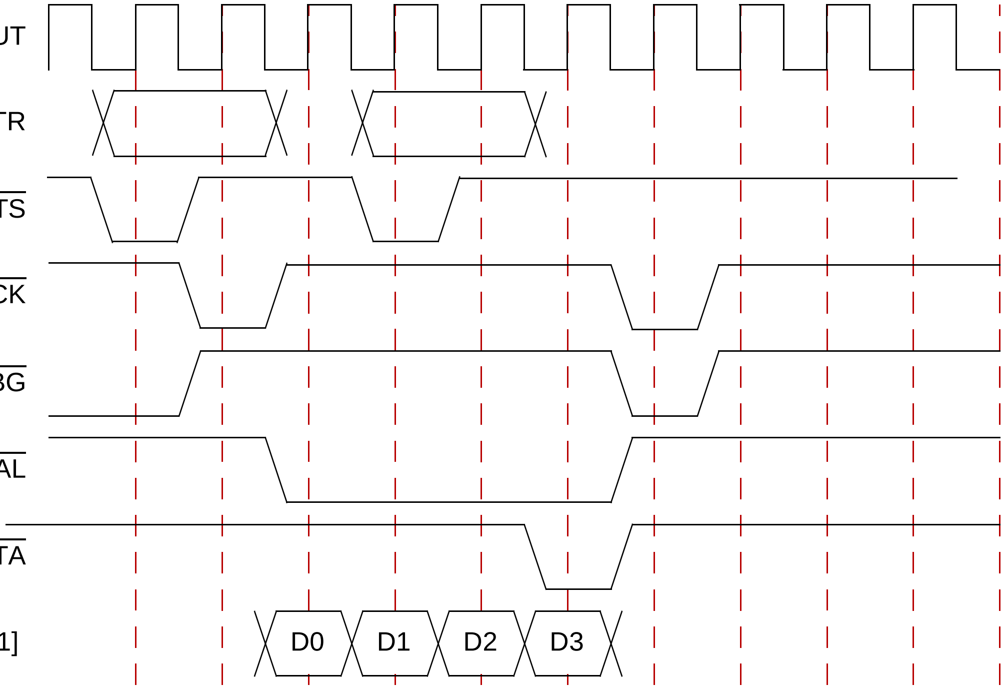


Figure 13-10. 128-Bit Extended Transfer to 32-Bit Port Size

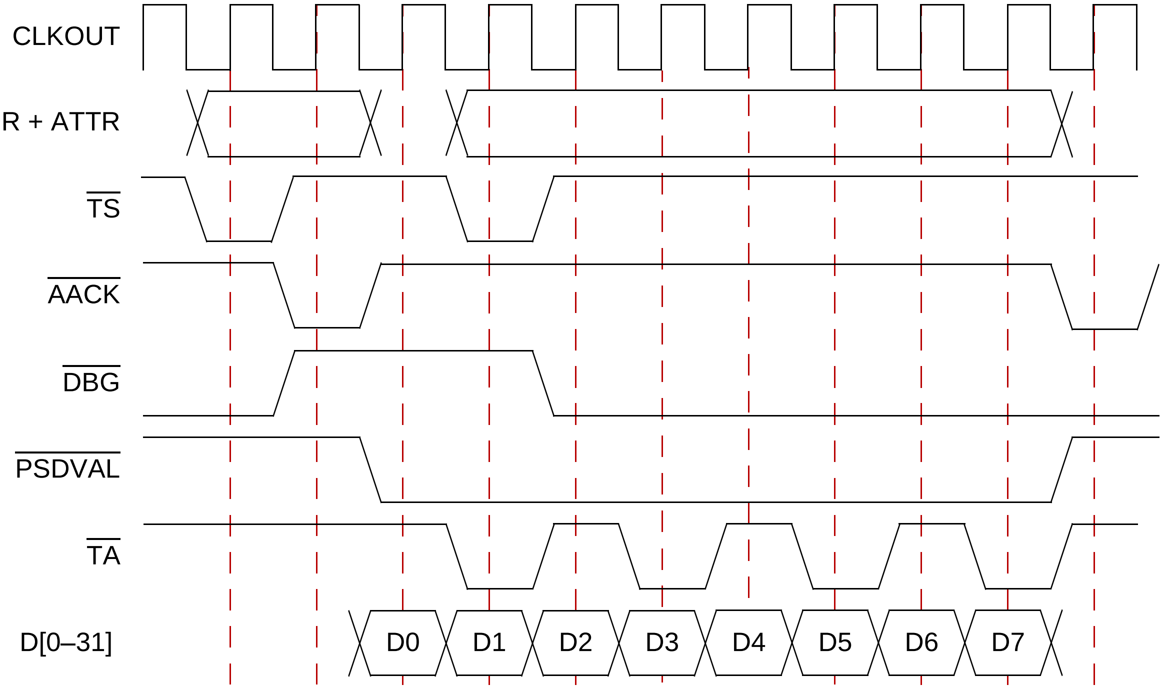


Figure 13-11. Burst Transfer to 32-Bit Port Size

13.2.4.6 Data Bus Termination by Assertion of $\overline{\text{TEA}}$ Signal

If a device initiates an unsupported transaction, the MSC8113 device signals an error by asserting $\overline{\text{TEA}}$. This occurs because the assertion of $\overline{\text{TEA}}$ is sampled by the device only during the data tenure of the bus transaction. The MSC8113 ensures that the device master receives a qualified data bus grant by asserting $\overline{\text{DBG}}$ before asserting $\overline{\text{TEA}}$. The data tenure is terminated by a single assertion of $\overline{\text{TEA}}$, regardless of the port size or whether the data tenure is a single-beat or burst transaction. This sequence is shown in **Figure 13-12**, where the data bus is busy at the beginning of the transaction and thus delays the assertion of $\overline{\text{DBG}}$.

Note: Data errors (parity and ECC) are reported by assertion of $\overline{\text{MCP}}$ rather than by assertion of $\overline{\text{TEA}}$.

The MSC8113 device interprets bus transactions as bus errors when bus errors are asserted by slaves (internal or external).

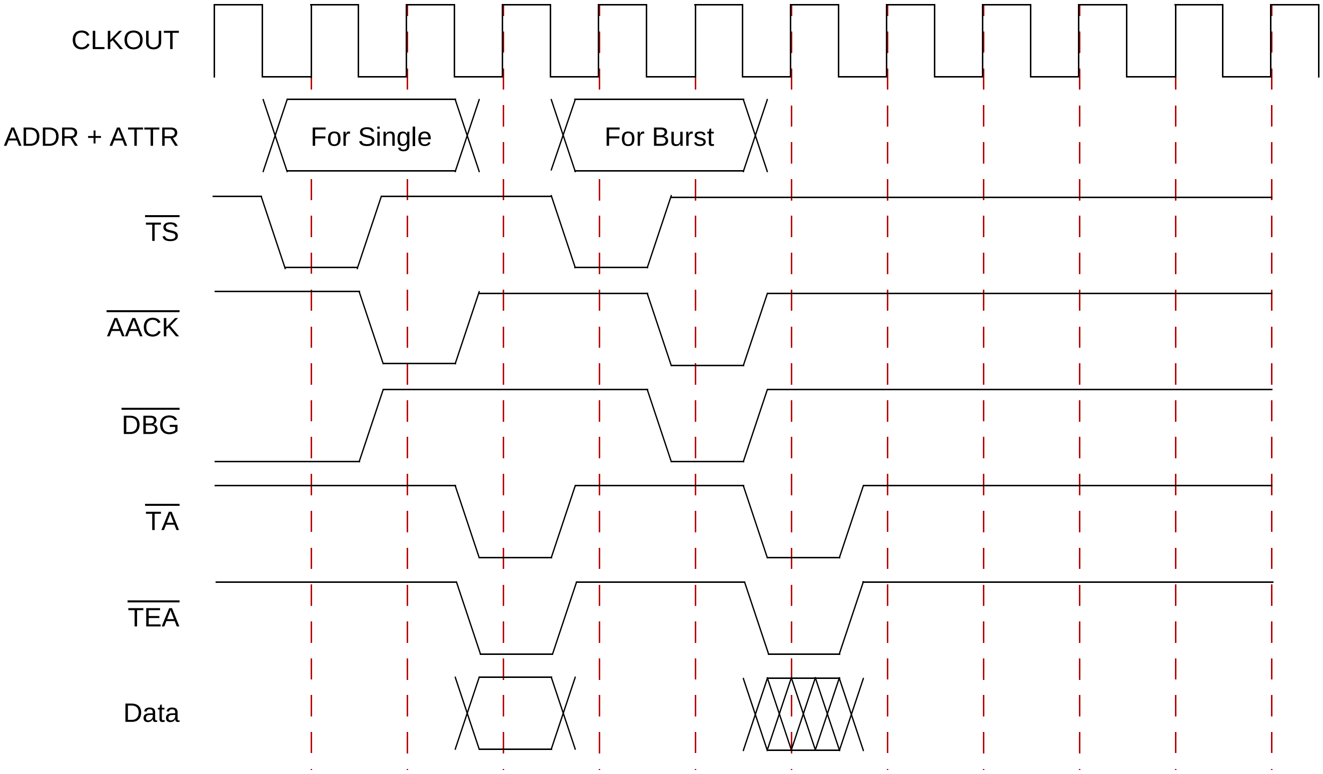


Figure 13-12. Data Tenure Terminated by Assertion of $\overline{\text{TEA}}$ Signal

Direct Slave Interface (DSI) 14

The direct slave interface (DSI) gives an external host direct access to the MSC8113 device and external memory. It provides the following slave interfaces to an external host:

- Asynchronous interface giving the host single accesses (with no external clock).
- Synchronous interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8113 internal bus clock.

A write buffer stores the address and the data of write accesses until they are performed, so the external host can perform multiple writes without wait states. The DSI write FIFO is 8 × 64 bits (512 bits) and can write up to two bursts. A read buffer stores prefetched data, so the external host can perform successive read accesses from consecutive addresses without wait states. The DSI read FIFO is 16 × 64 bits (1 K bits) and can read up to four bursts. A Host Transfer Acknowledge ($\overline{\text{HTA}}$) extends accesses that the DSI is not ready to complete. DSI read accesses from the MSC8113 internal or external address space are performed through buses shared by other internal clients. For example, to access one of the M1 memories, the DSI must gain ownership of the internal local bus that it shares with the TDM, the DMA controller, and the local-to-system bus bridge. The DSI may need to wait until it is granted the bus, which adds to the access latency. Also, clock separation on the DSI interface adds synchronization delay to the access latency.

To reduce read access latency, the read data path contains an internal read buffer and a prefetch mechanism. When the DSI reads a data from the internal or external memory (M2 memory, four M1 memories or external memory but not registers) and when the RPE bit in the DSI Control Register (DCR) is set, the DSI continues prefetching the next data into its read buffer. Typically, the latency of the next accesses is smaller because they are from consecutive addresses and are already stored in the read buffer.

The DSI stops prefetching data and flushes the read buffer when one of the following conditions is met:

- The host read access address is not consecutive to previous access address. A consecutive address is the last address + 4 bytes in 32-bit bus mode and the last address + 8 bytes in 64-bit bus mode. For burst read accesses, the consecutive address is the last address (the address of the first beat of the burst) + 32 bytes for both 32-bit and 64-bit bus mode.
- A host write access has begun (to ensure data coherency).
- In Synchronous mode, a burst read access starts after a single read access, or a single read access starts after a read burst access.

Note: Using the prefetch mechanism, the host reads the data from the read buffer instead of directly from memory. It can therefore read data that is not up-to-date.

To reduce latency for accesses to the MSC8113 internal or external address space, writes by the DSI are done via the write buffer. Write accesses to the DSI registers bypass the write buffer; therefore, you can change the DSI control registers and immediately perform the next access with the guarantee that this access is controlled by the new setting programmed in the DSI control registers. Overflow can occur only during broadcast accesses because there is no $\overline{\text{HTA}}$ signal to validate the write access (See **Section 14.3.5**). To preserve data coherency, a read access to the DSI stalls until all previous write accesses in the write buffer are complete.

When a host writes memory buffers to the device internal or external memory from the DSI port and then generates a Buffer Ready interrupt into one of the device $\overline{\text{IRQ}}$s, the interrupt may reach its destination before the buffer contents are correctly placed inside the memory because the DSI first writes to its internal write buffer and not directly to the memory. To avoid this situation, the host should perform the write accesses, then a read access (for example, a read of the DSI status register), and only then issue the Buffer Ready interrupt. Because the read access flushes the write buffer into the memory, the interrupt cannot arrive before the end of the last write (the interrupt occurs only at the end of the read access).

Note: For the description of the DSI external signals, see **Chapter 3**, *External Signals*.

Note: In systems that include a host device that connects to the MSC8113 through the DSI and the MSC8113 connects to other MSC8113, MSC8122, or MSC8126 devices through the system bus, all devices on the system bus must allow 64-bit read access for reads initiated by the host device. This is because any read access by the host through the DSI is translated by the MSC8113 DSI block as a 64-bit access.

14.1 Data Bus

The DSI data bus is organized from left to right in ascending order, so Bit 0 is the msb. The bytes are organized the same way, with byte lane 0 for the MSB. The width of the data transfer does not imply the operand's data structure. For example, a 16-bit data transfer on a 32-bit bus contains one data structure of 16 bits or two data structures of 8 bits. The operand data structure determines the order of the bytes in the data transfer, depending on the Little- or Big-Endian mode (see **Section 14.2.4**).

14.1.1 Data Bus Width

The DSI64 signal is sampled during $\overline{\text{PORESET}}$ signal deassertion and defines the data bus width as 64-bit or 32-bit (see the description of DSR[DSI64] on **page 14-35**). **Table 14-1** defines the possible data transfers in the 64-bit data bus mode.

Table 14-1. Data Transfers for Single Accesses in 64-bit Data Bus Mode

Transfer Size	Byte Enable [0–7]	Data Bus Byte Lanes							
		HD[0–7]	HD[8–15]	HD[16–23]	HD[24–31]	HD[32–39]	HD[40–47]	HD[48–55]	HD[56–63]
		Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
8-bit	0 1 1 1 1 1 1 1	8-bit data	—	—	—	—	—	—	—
	1 0 1 1 1 1 1 1	—	8-bit data	—	—	—	—	—	—
	1 1 0 1 1 1 1 1	—	—	8-bit data	—	—	—	—	—
	1 1 1 0 1 1 1 1	—	—	—	8-bit data	—	—	—	—
	1 1 1 1 0 1 1 1	—	—	—	—	8-bit data	—	—	—
	1 1 1 1 1 0 1 1	—	—	—	—	—	8-bit data	—	—
	1 1 1 1 1 1 0 1	—	—	—	—	—	—	8-bit data	—
	1 1 1 1 1 1 1 0	—	—	—	—	—	—	—	8-bit data
16-bit	0 0 1 1 1 1 1 1	16-bit data		—	—	—	—	—	—
	1 1 0 0 1 1 1 1	—	—	16-bit data		—	—	—	—
	1 1 1 1 0 0 1 1	—	—	—	—	16-bit data		—	—
	1 1 1 1 1 1 0 0	—	—	—	—	—	—	16-bit data	
32-bit	0 0 0 0 1 1 1 1	32-bit data				—	—	—	—
	1 1 1 1 0 0 0 0	—	—	—	—	32-bit data			
64-bit	0 0 0 0 0 0 0 0	64-bit data							

Notes:
1. Lanes specified as 8-, 16-, 32-, or 64-bit data is read or written during the bus transaction.
2. Lanes specified with "—" are ignored during writes and driven with undefined data during reads.
3. If the DCR[BEM] bit (see **page 14-29**) is cleared (0), only the $\overline{\text{HWBS0}}$, $\overline{\text{HDBS0}}$, $\overline{\text{HWBE0}}$, or $\overline{\text{HDBE0}}$ signal is used. The specific signal used depends on the configured access mode (that is, Asynchronous Dual Strobe mode write, Asynchronous Single Strobe mode read or write, Synchronous Dual Strobe mode write, or Synchronous Single Strobe mode read or write, respectively). For details, see **Section 14.3.1** and **Section 14.3.2**.

Table 14-2 defines the possible data transfers in 32-bit mode.

Table 14-2. Data Transfers for Single Accesses in 32-bit Data Bus Mode

Transfer Size	**Byte Enable [0–3]**	**Data Bus Byte Lanes**			
		HD[0–7]	**HD[8–15]**	**HD[16–23]**	**HD[24–31]**
		Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
8-bit	0 1 1 1	8-bit data	—	—	—
	1 0 1 1	—	8-bit data	—	—
	1 1 0 1	—	—	8-bit data	—
	1 1 1 0	—	—	—	8-bit data
16-bit	0 0 1 1	16-bit data		—	—
	1 1 0 0	—	—	16-bit data	
32-bit	0 0 0 0	32-bit data			

Notes:
1. Lanes specified as 8-, 16-, or 32-bit data is read or written during the bus transaction.
2. Lanes specified with “—” are ignored during writes and driven with undefined data during reads.
3. If the DCR[BEM] bit (see **page 14-29**) is cleared (0), only the $\overline{\text{HWBS0}}$, $\overline{\text{HDBS0}}$, $\overline{\text{HWBE0}}$, or $\overline{\text{HDBE0}}$ signal is used. The specific signal used depends on the configured access mode (that is, Asynchronous Dual Strobe mode write, Asynchronous Single Strobe mode read or write, Synchronous Dual Strobe mode write, or Synchronous Single Strobe mode read or write, respectively). For details, see **Section 14.3.1** and **Section 14.3.2**.

During a burst, 256 bits are transferred. Eight beats of full 32-bit data is transferred in 32-bit mode, and four beats of full 64-bit data is transferred in 64-bit mode.

14.1.2 DCR[BEM] Bit Access Considerations

For 32-bit mode, you must consider the following cases:

- *DCR[BEM] bit is set*. When the MSC8113 memory space (Bank 11), external memory, or a register space (Bank 9 or the system registers) is accessed, address bit HA29 is decoded.
  - — During write accesses, the data lanes are written according to the Byte Strobe/Enable signals (see **Table 14-2**).

Note: Write accesses to the DSI registers can only be performed using a 32-bit data bus width and not less, so all four byte enables [0–3] must be asserted during a write access to these registers.

  - — During read accesses, all four data lanes are driven. In Dual Strobe mode, all lanes are valid for reads. In Single Strobe mode, only the enabled lanes have valid data.
- *DCR[BEM] bit is cleared*. When the MSC8113 memory space (Bank 11) or external memory or one of the register spaces (Bank9 or the system registers) is accessed, address bit HA29 is decoded.
  - — During write accesses, all four data lanes are written and are valid.
  - — During read accesses, all four data lanes are driven, are valid, and are read.

For 64-bit mode, you must consider the following cases:

- *DCR[BEM] bit is set*. When the MSC8113 memory space (Bank 11)or external memory is accessed, address bit HA29 is ignored.
  - — During write accesses, the data lanes are written according to the Byte Strobe/Enable signals (as in **Table 14-1**).
  - — During read accesses, all eight data lanes are driven. In Dual Strobe mode, all lanes are valid for reads. In Single Strobe mode, only the enabled lanes have valid data. When the MSC8113 registers are accessed (Bank 9 or the system registers), address bit HA29 is decoded.
  - — A write access to the MSC8113 registers writes up to 32 bits, indicated by HA29. This applies even if all eight byte enables are asserted
    - If HA29 = 0, then data is written from HD[0–31], according to the state of Byte Enables [0–3] (in Little-Endian mode, HD[32–63], and Byte Enables [4–7]. See **Section 14.2.4**, *DSI Endian Modes*, on page 14-10). Write access to the DSI registers can only occur in 32-bit widths, so all four Byte Enables [0-3] ([4-7] in Little-Endian mode) must be asserted during the access.
    - If HA29 = 1, data is written from HD[32–63], according to the state of Byte Enables [4–7] (in Little-Endian mode, HD[0–31] and Byte Enables [0–3]).
  - — A read access to the MSC8113 registers reads the 32 bits indicated by HA29. The DSI drives all 64-bits (HD[0–64]), but only the 32 bits indicated by HA29 carry meaningful data:
    - If HA29 = 0, data is read to HD[0–31] (in Little-Endian mode, HD[32–63]).
    - If HA29 = 1, data is read to HD[32–63] (in Little-Endian mode, HD[0–31]).
- *DCR[BEM] bit is cleared*. When the MSC8113 memory space (Bank 11)or external memory is accessed, address bit HA29 is ignored.
  - — During write accesses, all eight data lanes are written and valid.
  - — During read accesses, all eight data lanes are driven and valid for reads.

  When one of the MSC8113 register spaces is accessed (Bank 9 or the system registers), address bit HA29 is decoded.
  - — A write access to the MSC8113 registers writes the 32 msbs or 32 lsbs, indicated by the HA29 value.
    - If HA29 = 0, data is written from HD[0–31](in Little-Endian mode, HD[32–63]).
    - If HA29 = 1, data is written from HD[32–63] (in Little-Endian mode, HD[0–31]).
  - — A read access to the MSC8113 registers reads 32 bits, according to the HA29 value. The DSI drives all 64 bits (HD[0–64]), but only the 32 bits indicated by HA29 carry meaningful data.
    - If HA29 = 0, data is read to HD[0–31] (in Little-Endian mode, HD[32–63]).
    - If HA29 = 1, data is read to HD[32–63] (in Little-Endian mode, HD[0–31]).

14.2 Address Bus

One of two DSI main addressing modes is determined during the MSC8113 boot sequence. Both modes allow two address space access Internal and External. The DCR[SLDWA] bit (see **page 14-29**) defines the addressing mode:

- Full Address Bus mode (SLDWA = 0) with HA[11–29] as mandatory bits and HA[7–10] which can be enabled by DCR[ADREN] control bits (see **page 14-29**).
- Sliding Window mode (SLDWA = 1) with HA[14–29] as address bits.

Following figures describe the address decoding and final internal address construction according to the main and sub addressing modes in use.

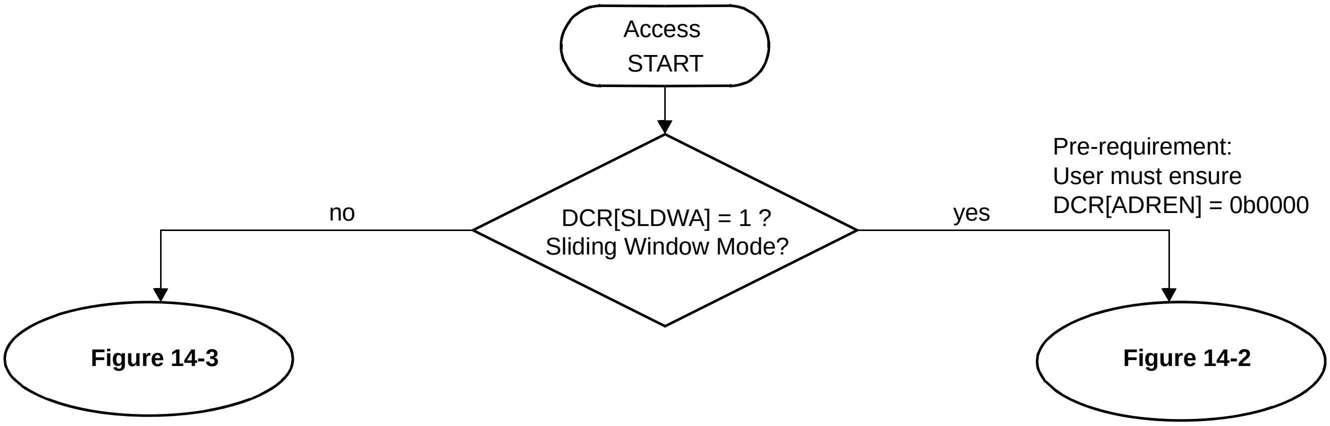


Figure 14-1. DSI Main Addressing Mode Selection

14.2.1 Sliding Window Addressing Mode

Setting the DCR[SLDWA] bit activates Sliding Window Addressing mode, which interfaces with hosts that do not have enough address lines to map the whole internal and external available MSC8113 address space. In Sliding Window Addressing mode, the addresses used in the accesses to the memory space are formed by concatenating a window addressed by the 15 least significant DSI address signals (HA[15–29]) up to 15 base address bits (DSWBAR[BAVAL_H, BAVAL_L]]). The window size is 128 KB while the whole Internal address space is 2 MB and the External is of 2 GB. When the HA14 signal is set, the base address is fixed and equals 0b1101. When the HA14 signal is cleared, the base address value to be used is determined by DSWBAR[EXTACC] control bit which selects between access to External Memory Space or Internal (**Figure 14-2**, *Sliding Window Mode Address Construction*, on page 14-7). In Sliding Window Addressing mode, chip ID decoding is used, and you must ensure that the HCID[0–3] signals carry the same value as the DCIR[CHIPID] value.

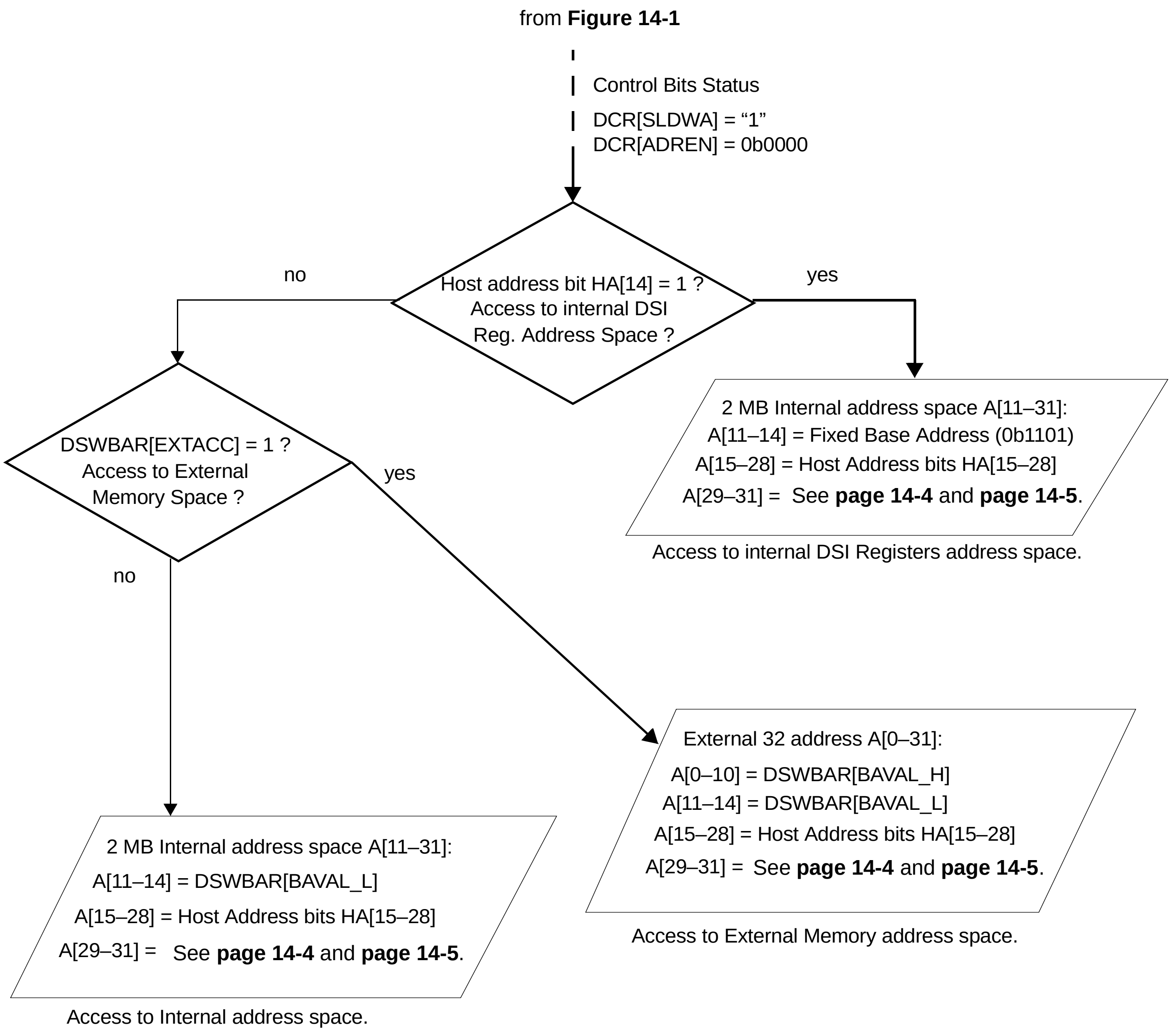
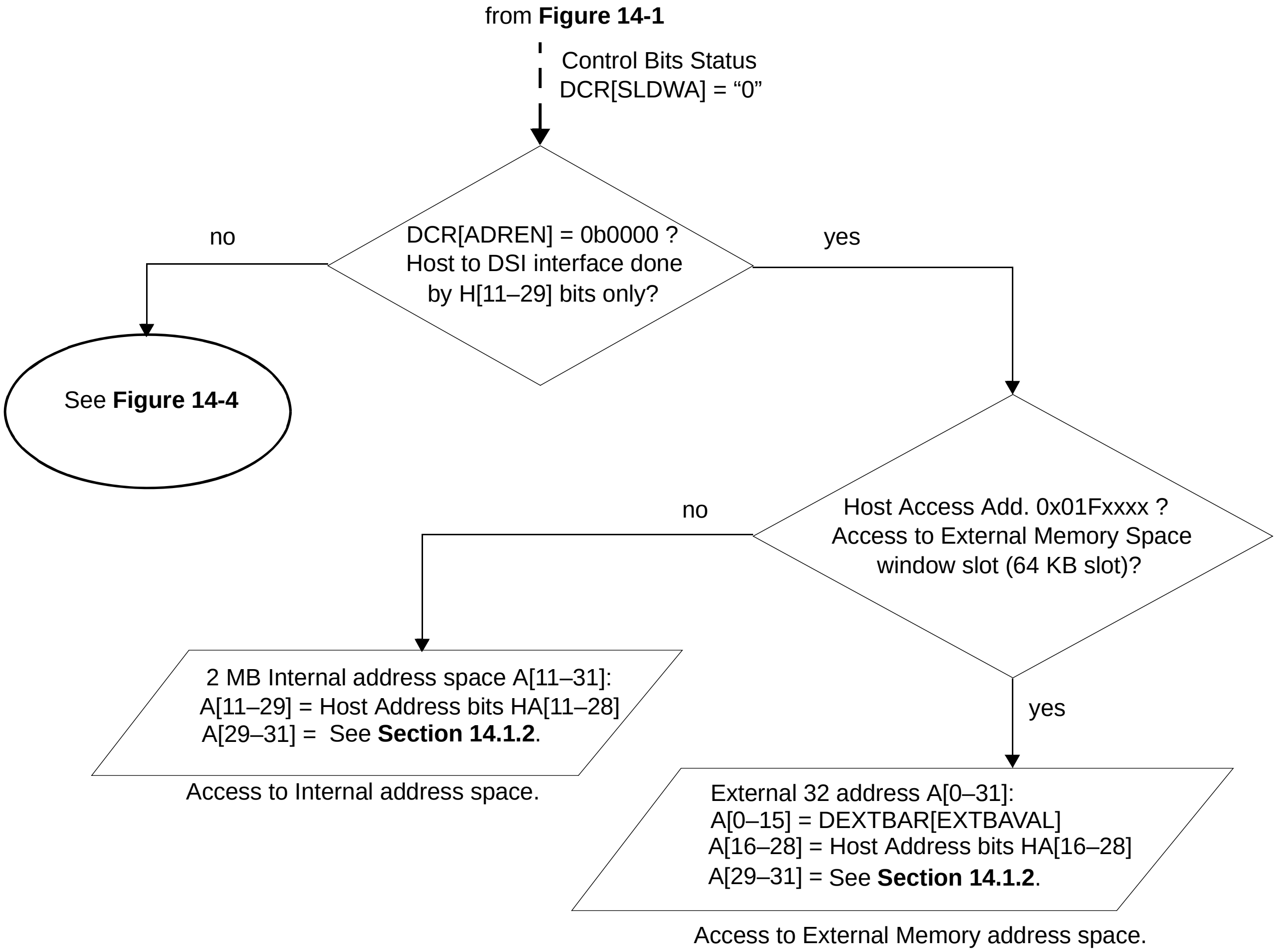

from **Figure 14-1**
Control Bits Status
DCR[SLDWA] = “1”
DCR[ADREN] = 0b0000
Host address bit HA[14] = 1 ?
Access to internal DSI
Reg. Address Space ?
no
yes
2 MB Internal address space A[11–31]:
A[11–14] = Fixed Base Address (0b1101)
A[15–28] = Host Address bits HA[15–28]
A[29–31] = See **page 14-4** and **page 14-5**.
Access to internal DSI Registers address space.
DSWBAR[EXTACC] = 1 ?
Access to External
Memory Space ?
yes
no
External 32 address A[0–31]:
A[0–10] = DSWBAR[BAVAL_H]
A[11–14] = DSWBAR[BAVAL_L]
A[15–28] = Host Address bits HA[15–28]
A[29–31] = See **page 14-4** and **page 14-5**.
Access to External Memory address space.
2 MB Internal address space A[11–31]:
A[11–14] = DSWBAR[BAVAL_L]
A[15–28] = Host Address bits HA[15–28]
A[29–31] = See **page 14-4** and **page 14-5**.
Access to Internal address space.

Figure 14-2. Sliding Window Mode Address Construction

14.2.2 Full Address Addressing Mode

In Full Address mode, the host can access the DSI with HA[11–29] as mandatory bits and HA[7–10], which can be enabled by the DCR[ADREN] control bits (see **page 14-29**). If only HA[11–29] uses (DCR[ADREN] = 0b0000), then the external address space is accessed through a slot (window), in the internal 2 MB address space (see **Figure 14-3**).

If at least one of the bits in the register field DCR[ADREN] has a value of 1, then the host uses the address MSB (see **Table 14-3**) to differentiate between external and internal access. If the address MSB is 0, then the access is to the internal chip address space; if the address MSB is 1, the access is to the external address space. See **Figure 14-4** and **Table 14-3** for more information.



from Figure 14-1
Control Bits Status
DCR[SLDWA] = "0"
DCR[ADREN] = 0b0000 ?
Host to DSI interface done
by H[11–29] bits only?
no
yes
See Figure 14-4
Host Access Add. 0x01Fxxxx ?
Access to External Memory Space
window slot (64 KB slot)?
no
yes
2 MB Internal address space A[11–31]:
A[11–29] = Host Address bits HA[11–28]
A[29–31] = See Section 14.1.2.
Access to Internal address space.
External 32 address A[0–31]:
A[0–15] = DEXTBAR[EXTBAVAL]
A[16–28] = Host Address bits HA[16–28]
A[29–31] = See Section 14.1.2.
Access to External Memory address space.

Figure 14-3. Full Address Mode Address Construction

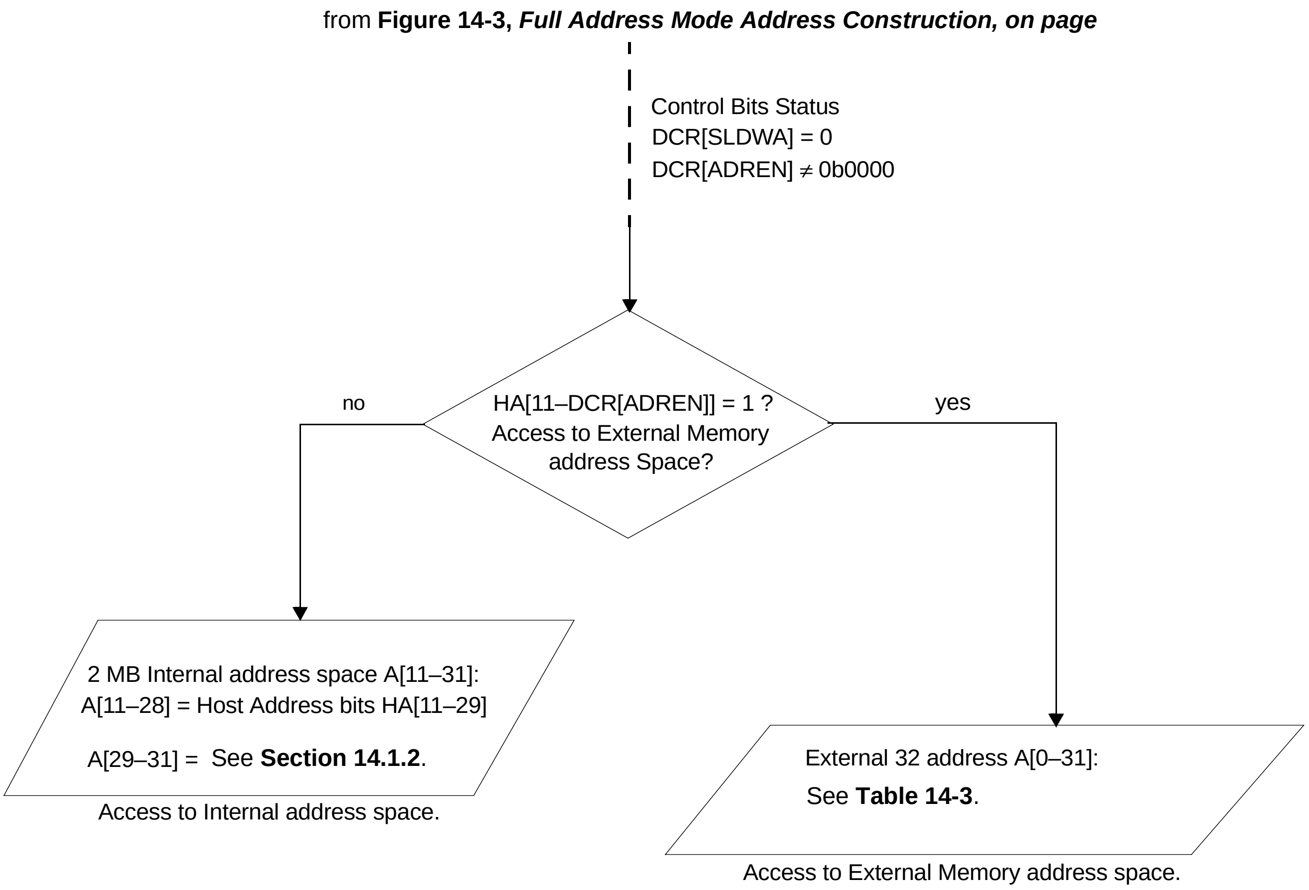

from Figure 14-3, Full Address Mode Address Construction, on page
Control Bits Status
DCR[SLDWA] = 0
DCR[ADREN] ≠ 0b0000
HA[11–DCR[ADREN]] = 1 ?
Access to External Memory
address Space?
no
yes
2 MB Internal address space A[11–31]:
A[11–28] = Host Address bits HA[11–29]
A[29–31] = See Section 14.1.2.
Access to Internal address space.
External 32 address A[0–31]:
See Table 14-3.
Access to External Memory address space.

Figure 14-4. Full Address Mode With High Address Bits Address Construction

Table 14-3. DCR[ADREN] Description of Decoding

DCR [ADREN]	External Window Size	A[0–7]	A[8]	A[9]	A[10]	A[11–28]	A[29–31]
0b0001	2 MB	DEXTBAR[EXTBAVAL[0–7]]	DEXTBAR[EXTBAVAL[8]]	DEXTBAR[EXTBAVAL[9]]	DEXTBAR[EXTBAVAL[10]]	HA[11–28]	See **Section 14.1.2**.
0b0010	4 MB				HA[10]		
0b0011	8 MB			HA[9]			
0b0100	16 MB		HA[8]				

14.2.3 Host Chip ID Signals (HCID[0–3])

The CHIPID field in the DSI Chip ID Register (DCIR) (see **page 14-34**) contains the value of the CHIP_ID[0–3] signals that is sampled during the PORESET sequence. For each host access to the DSI, the HCID[0–3] signals are compared to the DCIR[CHIPID] value. This decoding enables the host to use one chip select signal to access each of up to sixteen MSC8113 devices. You can write a new value to the DCIR after the reset sequence ends. HCID[3] is multiplexed with HA[8] which means that it is used as address bit and not a Chip ID bit, if DCR[ADREN] equals 0b0011 or 0b0100 (see **page 14-34**). In this case, you should ensure that CHIPID[3] is sampled low during the PORESET sequence.

14.2.4 DSI Endian Modes

The DSI supports hosts that use big-endian, little-endian, or munged little-endian byte ordering. The LTLEND bit in the Hard Reset Configuration Word (HRCW) is set for host accesses in Little-Endian mode. A host working in munged Little-Endian mode must also set the PPCLE bit in the HRCW (see **Section 5.6.1**, *Hard Reset Configuration Word*, on page 5-13 and **Section 3.1.4** in *The Programming Environments for 32-Bit Processors that Implement the PowerPC Architecture* (MPCFPE32B/AD)).

MSC8113 internal memory is structured as big-endian, so the DSI reorganizes data structures written by little-endian hosts. When bit LTLEND is set, the DSI translates all host accesses to the big-endian structure before placing them in the internal memory space. The translation of a little-endian host access to a big-endian structure occurs according to the type of the data structure in the access; that is, according to whether the data structure is 8-bit, 16-bit, 32-bit, or 64-bit. The DCR[DSRFA] bit defines how the host declares the type of the data structure in the access:

- Using the HDST[0–1] signals (see **Table 14-4**)
- Using the DCR[LEDS] field

Table 14-4. HDST[0–1] Signal Decoding

HDST[0–1]	00	01	10	11
Data Structure	8 bit	16 bit	32 bit	64 bit

Note: The DSI refers to host accesses to MSC8113 registers as 32-bit data structure accesses, overriding any value defined by the HDST[0–1] signals or by the DCR[LEDS] field.In munged little-endian mode, when working with 32 bit data bus width, prefetch mechanism is not supported. In this combination of modes bit DCR[RPE] should not be set.

14.3 Host Access Modes and Timings

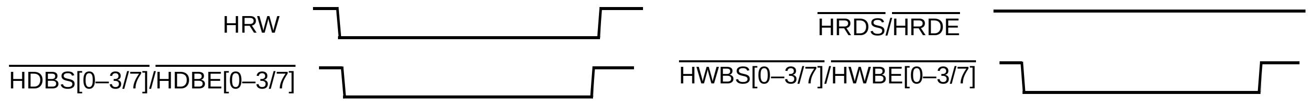
This section covers the DSI accessing modes, asynchronous mode for reads and writes, synchronous reads and writes, and broadcast accesses.

14.3.1 Single Strobe Versus Dual Strobe Access Modes

The DCR[SNGLM] bit defines the DSI accessing mode (see **page 14-29**):

- Single Strobe mode (DCR[SNGLM] = 1):
  - — The HRW signal selects the direction of the access, read or write.
  - — The $\overline{\text{HDBS[0–3/7]}}$/$\overline{\text{HDBE[0–3/7]}}$ signals are data byte strobes/enables.
- Dual Strobe mode (DCR[SNGLM] = 0):
  - — The $\overline{\text{HRDS}}$/$\overline{\text{HRDE}}$ signal is a read strobe/enable.
  - — The $\overline{\text{HWBS[0–3/7]}}$/$\overline{\text{HWBE[0–3/7]}}$ signals are write byte strobes/enables.

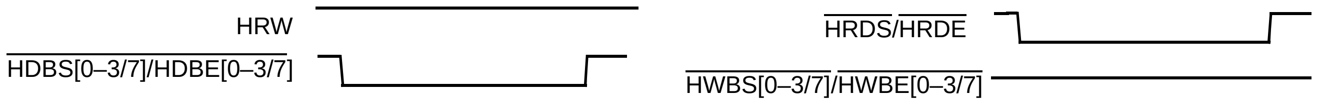
Figure 14-5 and **Figure 14-6** demonstrate the different access modes during read/write cycles.


HRW
$\overline{\text{HDBS[0–3/7]}}$/$\overline{\text{HDBE[0–3/7]}}$
$\overline{\text{HRDS}}$/$\overline{\text{HRDE}}$
$\overline{\text{HWBS[0–3/7]}}$/$\overline{\text{HWBE[0–3/7]}}$

Single Strobe Mode

Dual Strobe Mode

Figure 14-5. DSI Access Modes, Write Access


HRW
$\overline{\text{HDBS[0–3/7]}}$/$\overline{\text{HDBE[0–3/7]}}$
$\overline{\text{HRDS}}$/$\overline{\text{HRDE}}$
$\overline{\text{HWBS[0–3/7]}}$/$\overline{\text{HWBE[0–3/7]}}$

Single Strobe Mode

Dual Strobe Mode:

Figure 14-6. DSI Accessing Modes, Read Access

14.3.2 Synchronous Versus Asynchronous Access Mode

The DSI operates in two modes that are determined during the $\overline{\text{PORESET}}$ flow, Asynchronous and Synchronous. In Asynchronous mode, the DSI functions as an SRAM with the addition of an acknowledge signal ($\overline{\text{HTA}}$). In Synchronous mode, the DSI functions as a synchronous SRAM (SSRAM) with the addition of an acknowledge signal ($\overline{\text{HTA}}$). In Asynchronous mode, the DSI Control Register (DCR) enables you to define logic level to which $\overline{\text{HTA}}$ is driven at the end of an access and the drive time after the end of access. In Synchronous mode, the DSI is accessed in either single accesses or bursts, and the host clock need not be synchronized to the internal local bus clock. The DSI has two clock regions, the host clock region and the local bus clock region. Synchronizers on the signals pass from one region to another.

14.3.2.1 Burst Transfers

Bursts have a fixed number of beats according to the data bus width, four beats on a 64-bit data bus and eight beats on a 32-bit data bus. Burst transfers must always start at an address aligned to 64-bit data according to the internal memory map. Burst transfers are *linear* (see **Table 14-5**). The local bus is 64 bits wide, so four beats are required for the internal burst. In 32-bit mode, eight external beats are required. Burst accesses are identified via the $\overline{\text{HBRST}}$ signal. The DSI samples $\overline{\text{HBRST}}$ only at the beginning of an access and determines whether it is a burst access or a single access. For a burst access, the DSI ignores $\overline{\text{HBRST}}$ during the burst. A host that uses LAST can tie it to the $\overline{\text{HBRST}}$ signal. In a single access, LAST is asserted on the first access; in a burst access, LAST is deasserted on the first access. If the host performs a burst access, the assertion of LAST at the end of the access is ignored because the DSI performs a fixed number of beats per burst. $\overline{\text{HBRST}}$ signal polarity is defined by the DCR[BRSTP] bit (see **page 14-29**), which is typically used with an active low $\overline{\text{LAST}}$. Burst accesses are allowed only to memory regions (Bank 11and external memory, see **Figure 8-3**) and not to control/status register regions.

Note: Burst accesses are not allowed to overflow from the boundaries of the memory regions.

Table 14-5. Linear Burst Address Table

First Address (External)	Second Address (Internal)	Third Address (Internal)	Fourth Address (Internal)
HA[11–29]	**A[11–31]**	**A[11–31]**	**A[11–31]**
0bxxxxxxxxxxxxxxxx000	0bxxxxxxxxxxxxxxxx01000	0bxxxxxxxxxxxxxxxx10000	0bxxxxxxxxxxxxxxxx11000
0bxxxxxxxxxxxxxxxx010	0bxxxxxxxxxxxxxxxx10000	0bxxxxxxxxxxxxxxxx11000	0bxxxxxxxxxxxxxxxx00000
0bxxxxxxxxxxxxxxxx100	0bxxxxxxxxxxxxxxxx11000	0bxxxxxxxxxxxxxxxx00000	0bxxxxxxxxxxxxxxxx01000
0bxxxxxxxxxxxxxxxx110	0bxxxxxxxxxxxxxxxx00000	0bxxxxxxxxxxxxxxxx01000	0bxxxxxxxxxxxxxxxx10000

Note: Bits shown as x can be 1 or 0. Addresses 29–31 have a value of 0 because bursts should be 64-bit aligned. However, in munged Little-Endian mode, in 32-bit mode, HA29 must be equal to 1, but is interpreted internally as 0.

14.3.2.2 DSI Access Modes

The four DSI access modes (Dual Strobe/Single Strobe, Asynchronous/Synchronous) share the same DSI external signals with different naming conventions, as listed in **Table 14-6**.

Table 14-6. DSI Access Mode Signals

Access Mode	Strobe Mode	32-Bit Data Bus		64-Bit Data Bus	
		Read Signals	Write Signals	Read Signals	Write Signals
Asynchronous	Dual	$\overline{\text{HRDS}}$	$\overline{\text{HWBS[0–3]}}$	$\overline{\text{HRDS}}$	$\overline{\text{HWBS[0–7]}}$
Asynchronous	Single	$\overline{\text{HDBS[0–3]}}$ and HRW	$\overline{\text{HDBS[0–3]}}$ and HRW	$\overline{\text{HDBS[0–7]}}$ and HRW	$\overline{\text{HDBS[0–7]}}$ and HRW
Synchronous	Dual	$\overline{\text{HRDE}}$	$\overline{\text{HWBE[0–3]}}$	$\overline{\text{HRDE}}$	$\overline{\text{HWBE[0–7]}}$
Synchronous	Single	$\overline{\text{HDBE[0–3]}}$ and HRW	$\overline{\text{HDBE[0–3]}}$ and HRW	$\overline{\text{HDBE[0–7]}}$ and HRW	$\overline{\text{HDBE[0–7]}}$ and HRW

14.3.3 Asynchronous Mode Operation

This section discusses both asynchronous writes and reads. These examples use 64-bit mode. For 32-bit mode refer to the enable and strobe signal definitions listed in **Table 14-6**.

14.3.3.1 Asynchronous Write Using Dual Strobe Mode

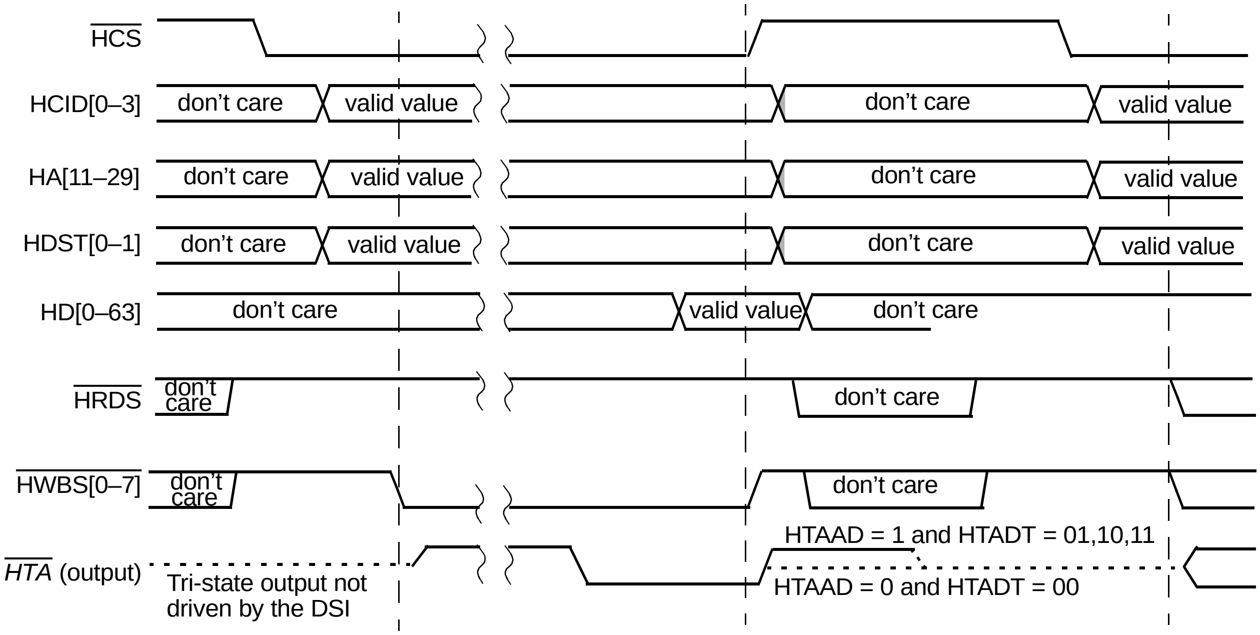
Figure 14-7 shows an asynchronous write access using Dual Strobe mode. The DSI samples the Host Chip ID signals (HCID[0–3]) on the first falling edge of the Host Write Byte Strobe signals ($\overline{\text{HWBS}}$) on which the Host Chip Select signal ($\overline{\text{HCS}}$) is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. Assertion of the Host Transfer Acknowledge ($\overline{\text{HTA}}$) signal indicates that the DSI is ready to sample the host data bus (HD[0–63]), and the host can terminate the access by deasserting $\overline{\text{HWBS}}$.

The DCR[HTAAD] and DCR[HTADT] fields determine which of the following actions the DSI takes at end of an access (the rising edge of $\overline{\text{HWBS}}$):

- *Stop driving* $\overline{\text{HTA}}$. DCR[HTAAD] = 0 and DCR[HTADT] = 00 (no drive time). This mode requires a pull-down resistor on $\overline{\text{HTA}}$.
- *Drive* $\overline{\text{HTA}}$ *high*. DCR[HTAAD] = 1 and DCR[HTADT] ≠ 00. The DCR[HTADT] value indicates the amount of time to drive $\overline{\text{HTA}}$. This mode requires a pull-up resistor on $\overline{\text{HTA}}$.

In both cases, the host can start its next access (back-to-back accesses) without deasserting HCS between accesses. When the DCR[HTAAD] and DCR[HTADT] are both cleared, the host must ignore the HTA value from the start of the access until the DSI drives it to its correct value. The required delay is defined in the AC characteristics section of the *MSC8113 Technical Data* sheet.

When the DCR[HTAAD] bit is set and the DCR[HTADT] bits do not equal 00 and if the next access is not to the same MSC8113, then to prevent contention on HTA, the host must wait until the previous DSI stops driving the HTA signal before it accesses the next device. When the DCR[HTAAD] bit is set and the next access is to the same MSC8113 device, the host must *not* start a consecutive access before the HTA signal is actively driven high by the previous access.



Note: The signal timing shown for HWBS[0–7] is for signals that are asserted. Unused signals remain high (deasserted).

Figure 14-7. Asynchronous Write Using Dual Strobe Mode

14.3.3.2 Asynchronous Write Using Single Strobe Mode

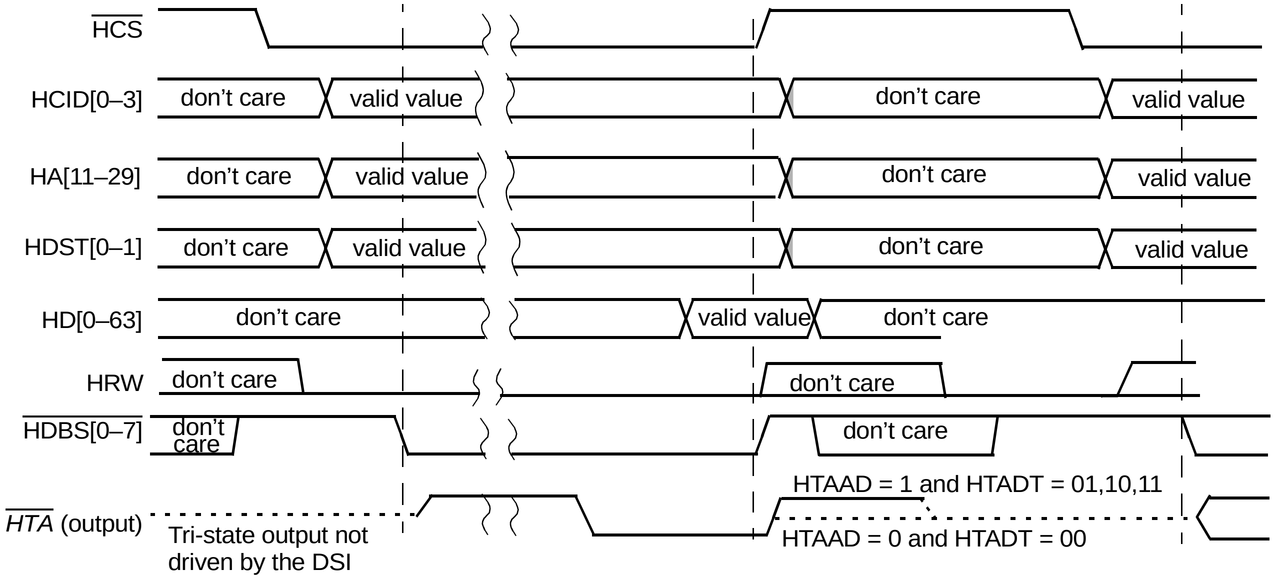
Figure 14-8 shows an asynchronous write access using Single Strobe mode. The DSI samples the Host Chip ID signals (HCID[0–3]) on the first falling edge of the Host Data Byte Strobe (HDBS) on which the Host Chip Select signal (HCS) is asserted and HRW is low. If HCID[0–3] matches the CHIPID value, the DSI is accessed. Assertion of the Host Transfer Acknowledge (HTA) signal indicates that the DSI is ready to sample the host data bus (HD). The host can terminate the access by deasserting HDBS.

The DCR[HTAAD] and DCR[HTADT] fields determine which of the following actions the DSI takes at end of an access (the rising edge of HWBS):

- *Stop driving HTA*. DCR[HTAAD] = 0 and DCR[HTADT] = 00 (no drive time). This mode requires a pull-down resistor on HTA.
- *Drive HTA high*. DCR[HTAAD] = 1 and DCR[HTADT] ≠ 00. The DCR[HTADT] value indicates the amount of time to drive HTA. This mode requires a pull-up resistor on HTA.

In both cases, the host can start its next access (back-to-back accesses) without deasserting HCS between accesses. When the DCR[HTAAD] and DCR[HTADT] are both cleared, the host must ignore the HTA value from the start of the access until the DSI drives it to its correct value. The required delay is defined in the AC characteristics section of the *MSC8113 Technical Data* sheet.

When the DCR[HTAAD] bit is set and the DCR[HTADT] bits do not equal 00 and if the next access is not to the same MSC8113, to prevent contention on HTA, the host must wait until the previous DSI stops driving the HTA signal before it accesses the next device. When the DCR[HTAAD] bit is set and the next access is to the same MSC8113 device, the host must *not* start a consecutive access before the HTA signal is actively driven high by the previous access.



Note: The signal timing shown for HDBS[0–7] is for signals that are asserted. Unused signals remain high (deasserted).

Figure 14-8. Asynchronous Write Using Single Strobe Mode

14.3.3.3 Asynchronous Read Using Dual Strobe Mode

Figure 14-9 shows an asynchronous read access using Dual Strobe mode. The DSI samples the host address bus (HA[11–29]) and the HCID on the first falling edge of the Host Read Strobe signal (HRDS) on which the HCS is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. When the DCR[RPE] bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. Assertion of HTA indicates that data is valid, and the host can sample the host data bus (HD) and terminate the access by deasserting HRDS. If the data for this access is already in the read buffer due to the prefetch mechanism, the assertion time of HTA is improved.

The DCR[HTAAD] and DCR[HTADT] bits determine which of the following actions the DSI takes at the rising edge of HRDS:

- *Stop driving HTA*. DCR[HTAAD] = 0 and DCR[HTADT] = 00. This mode requires a pull-down resistor on HTA.
- *Drive HTA high.* DCR[HTAAD] = 1 and DCR[HTADT] ≠ 00. The DCR[HTADT] value indicates the amount of time to drive HTA. This mode requires a pull-up resistor on HTA.

In either case, the host can start a back-to-back access without deasserting HCS between accesses. When DCR[HTAAD] and DCR[HTADT] are both cleared, the host must ignore the HTA value from the start of the access until the DSI drives it to its correct value. The required delay period is defined in the AC characteristics section of the *MSC8113 Technical Data* sheet. If DCR[HTAAD] is set and DCR[HTADT] does not equal 00 and the next access is not to the same MSC8113 device, then to prevent contention on the HTA signal, the host must wait until the previous DSI stops driving the HTA signal before it accesses the next device. When the DCR[HTAAD] bit is set and the next access is to the same MSC8113, the host must *not* start consecutive access before the previous access deasserts HTA.

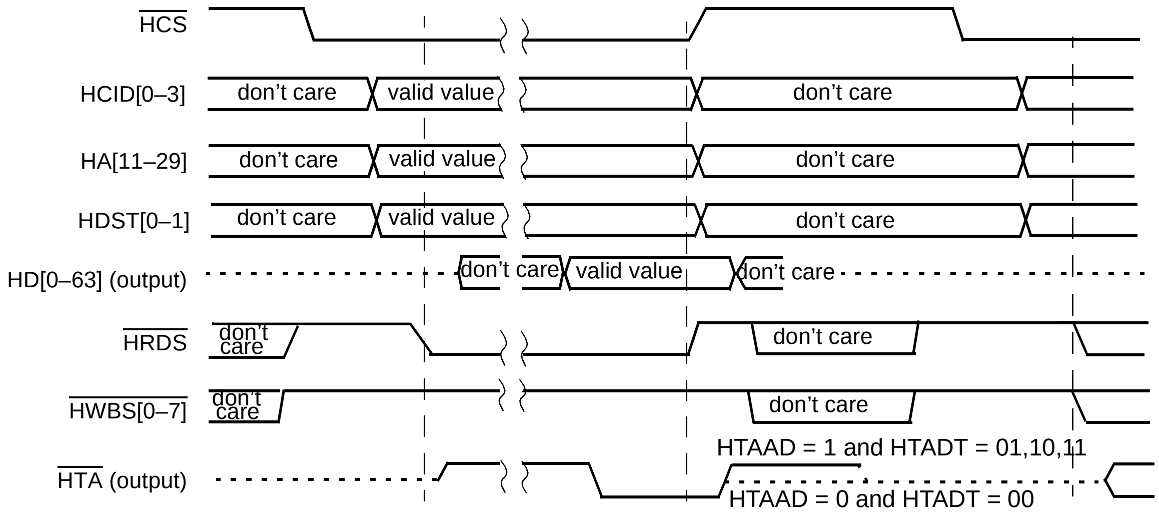


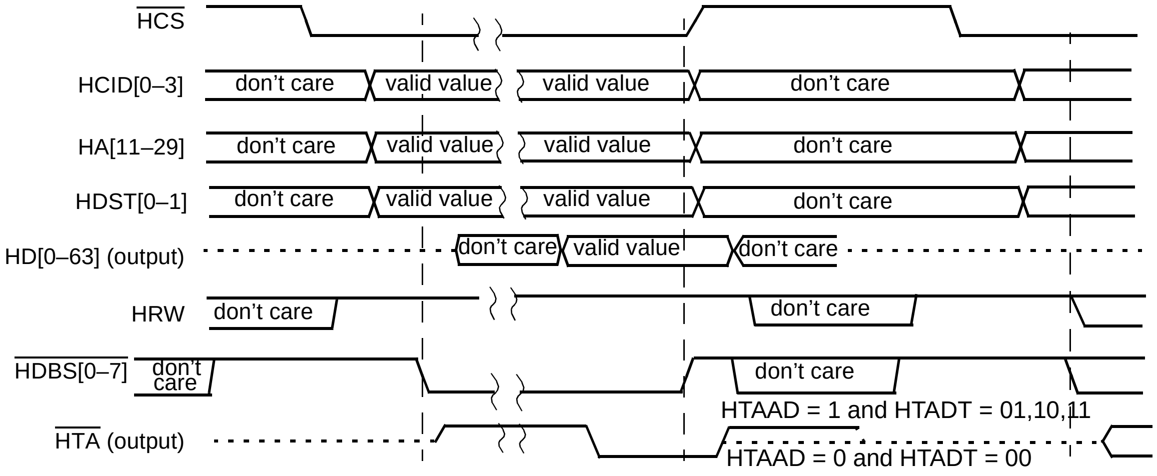
Figure 14-9. Asynchronous Read Using Dual Strobe Mode

14.3.3.4 Asynchronous Read Using Single Strobe Mode

Figure 14-10 shows an asynchronous read access using Single Strobe mode. The DSI samples the host address bus (HA[11–29]) and the HCID on the first falling edge of the Host Data Byte Strobe signals ($\overline{\text{HDBS}}$) on which the $\overline{\text{HCS}}$ is asserted with HRW driven high. If HCID[0–3] match the CHIPID value, the DSI is accessed. When the DCR[RPE] bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid, and the host can sample the host data bus (HD) and terminate the access by deasserting $\overline{\text{HDBS}}$. If the data for this access is already in the read buffer due to the prefetch mechanism, the assertion time of $\overline{\text{HTA}}$ is improved. The DCR[HTAAD] and DCR[HTADT] bits determine which of the following actions the DSI takes at the rising edge of $\overline{\text{HRDS}}$:

- *Stop driving $\overline{\text{HTA}}$*. DCR[HTAAD] = 0 and DCR[HTADT] = 00. This mode requires a pull-down resistor on $\overline{\text{HTA}}$.
- *Drive $\overline{\text{HTA}}$ high*. DCR[HTAAD] = 1 and DCR[HTADT] ≠ 00. The DCR[HTADT] value indicates the amount of time to drive $\overline{\text{HTA}}$. This mode requires a pull-up resistor on $\overline{\text{HTA}}$.

In either case, the host can start a back-to-back access without deasserting $\overline{\text{HCS}}$ between accesses. When DCR[HTAAD] and DCR[HTADT] are both cleared, the host must ignore the $\overline{\text{HTA}}$ value from the start of the access until the DSI drives it to its correct value. The required delay period is defined in the AC characteristics section of the *MSC8113 Technical Data* sheet. If DCR[HTAAD] is set and DCR[HTADT] does not equal 00 and the next access is not to the same MSC8113, then to prevent contention on the $\overline{\text{HTA}}$ signal, the host must wait until the previous DSI stops driving the $\overline{\text{HTA}}$ signal before it accesses the next device. When the DCR[HTAAD] bit is set and the next access is to the same MSC8113, the host must *not* start consecutive access before the previous access deasserts $\overline{\text{HTA}}$.



Note: The signal timing shown for $\overline{\text{HDBS[0–7]}}$ is for signals that are asserted. Unused signals remain high (deasserted).

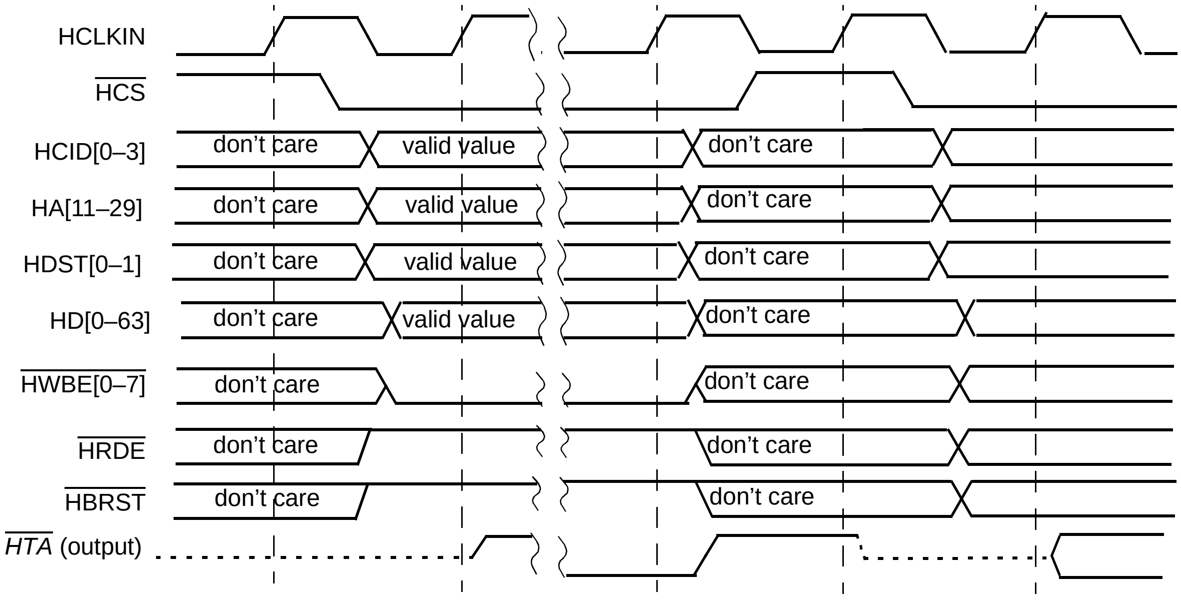
Figure 14-10. Asynchronous Read Using Single Strobe Mode

14.3.4 Synchronous Mode Operation

This section covers synchronous single write and read and synchronous burst write and read. These examples use 64-bit mode. For 32-bit mode refer to the enable and strobe signal definitions listed in **Table 14-6**.

14.3.4.1 Synchronous Single Write Using Dual Strobe Mode

Figure 14-11 shows a synchronous single write access using Dual Strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], HD[0–63], HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. At least one HWBE signal must be asserted with HRDE and HBRST deasserted. Assertion of HTA indicates that the DSI is ready to complete the current access and the host must terminate this access. Typically, HTA is asserted immediately. If the write buffer is full, HTA assertion is delayed. HTA is asserted for one HCLKIN cycle, deasserted in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 immediately in the next HCLKIN rising edge without deasserting HCS between accesses. If the next access is not to the same MSC8113, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. There is no error condition to prevent this contention. It is your responsibility to ensure that no contention occurs.

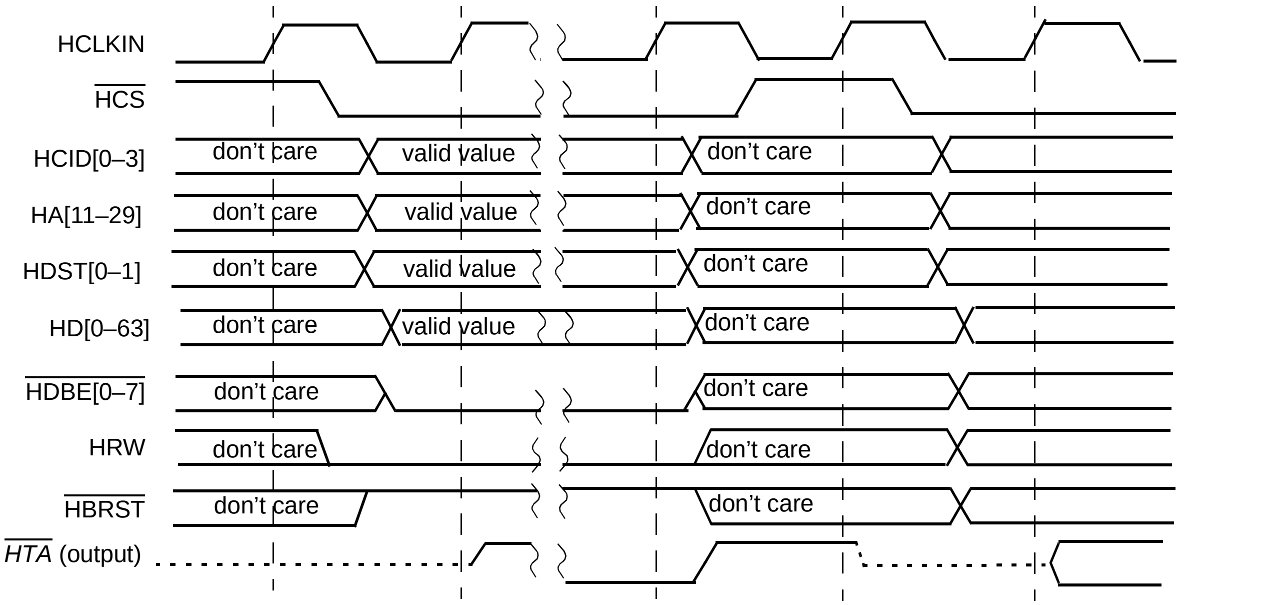


Note: The signal timing shown for HWBE[0–7] is for signals that are asserted. Unused signals remain high (deasserted).

Figure 14-11. Synchronous Single Write Using Dual Strobe Mode

14.3.4.2 Synchronous Single Write Using Single Strobe Mode

Figure 14-12 shows a synchronous single write access using Single Strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], HD[0–63], $\overline{\text{HDBE[0–7]}}$, HRW, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. At least one $\overline{\text{HDBE}}$ signal must be asserted, HRW must be low, and $\overline{\text{HBRST}}$ must be deasserted. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current access and the host must terminate this access. Typically, $\overline{\text{HTA}}$ is asserted immediately. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. $\overline{\text{HTA}}$ is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 immediately in the next HCLKIN rising edge without deasserting $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8113, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. There is no error condition to prevent this contention. It is your responsibility to ensure that no contention occurs.



Note: The signal timing shown for HDBE[0–7] is for signals that are asserted. Unused signals remain high (deasserted).

Figure 14-12. Synchronous Single Write Using Single Strobe Mode

14.3.4.3 Synchronous Single Read Using Dual Strobe Mode

Figure 14-13 shows a synchronous single read access using Dual Strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], HWBE[0–7], HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If the HCID[0–3] signals match the CHIPID value, the DSI is accessed. HRDE is asserted with HWBE and HBRST deasserted. When the DCR[RPE] bit is set (see **page 14-29**), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of HTA indicates that data is valid and that the host must sample the host data lines and terminate the access. The HTA is asserted earlier when the data for this access is already prefetched to the read buffer. HTA is asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 device immediately in the next HCLKIN rising edge without deasserting HCS between accesses. If the next access is not to the same MSC8113 device, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA.

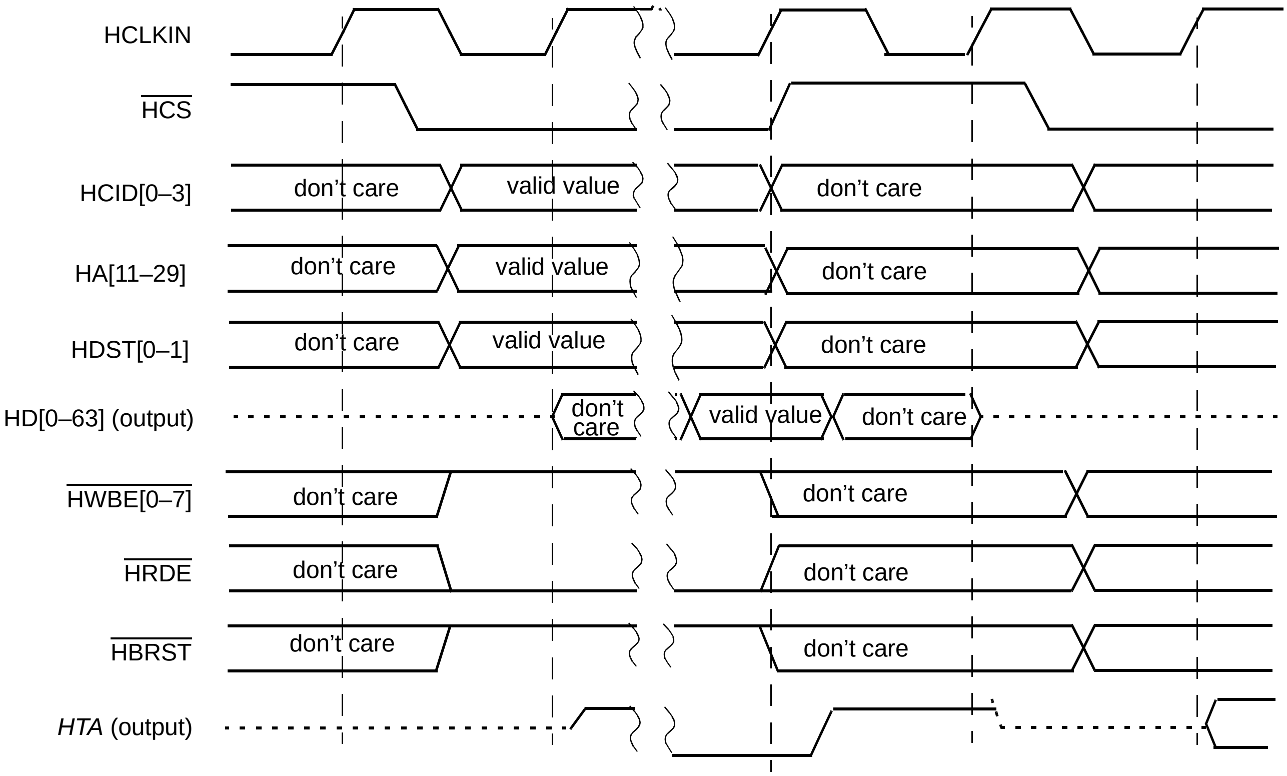
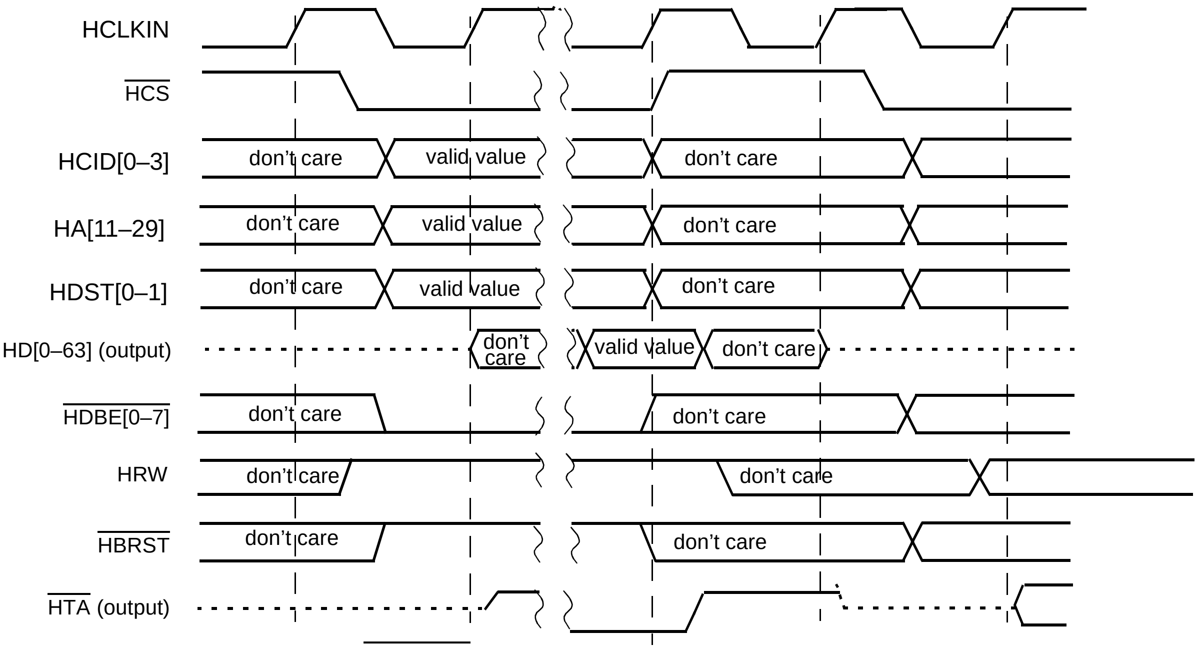


Figure 14-13. Synchronous Single Read Using Dual Strobe Mode

14.3.4.4 Synchronous Single Read Using Single Strobe Mode

Figure 14-14 shows a synchronous single read access using Single Strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], $\overline{\text{HDBE[0–7]}}$, HRW, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If the HCID[0–3] signals match the CHIPID value, the DSI is accessed. At least one $\overline{\text{HDBE}}$ signal must be asserted, HRW must be high, and $\overline{\text{HBRST}}$ must be deasserted. When the DCR[RPE] bit is set (see **page 14-29**), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid and that the host must sample the host data lines and terminate the access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. $\overline{\text{HTA}}$ is asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 immediately in the next HCLKIN rising edge without deasserting $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8113, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$.

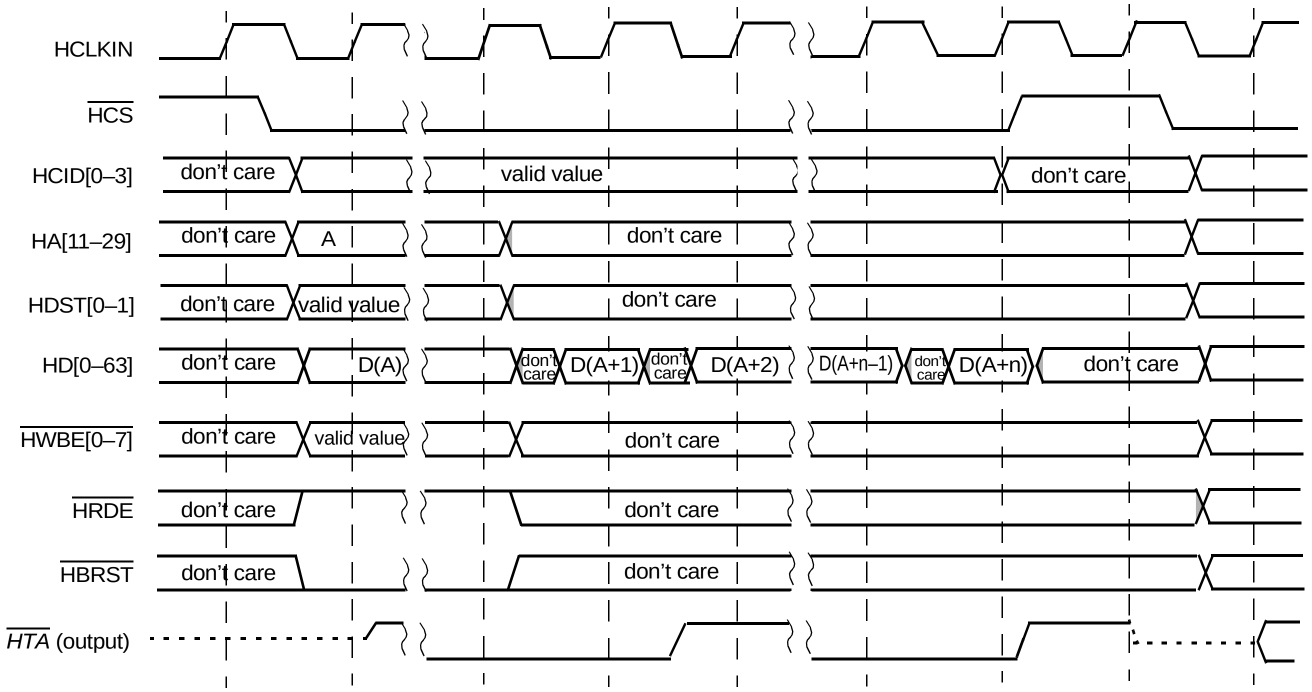


Note: The signal timing shown for $\overline{\text{HDBE[0–7]}}$ is for signals that are asserted. Unused signals remain high (deasserted).

Figure 14-14. Synchronous Single Read Using Single Strobe Mode

14.3.4.5 Synchronous Burst Write Using Dual Strobe Mode

Figure 14-15 shows a synchronous burst write access using dual strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], HD[0–63], $\overline{\text{HWBE}}$, $\overline{\text{HRDE[0–7]}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. $\overline{\text{HWBE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HRDE}}$ is deasserted. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically $\overline{\text{HTA}}$ is asserted immediately for each beat of the access. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. After the last beat of the access, $\overline{\text{HTA}}$ is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 immediately in the next HCLKIN rising edge without deasserting $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8113, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$.

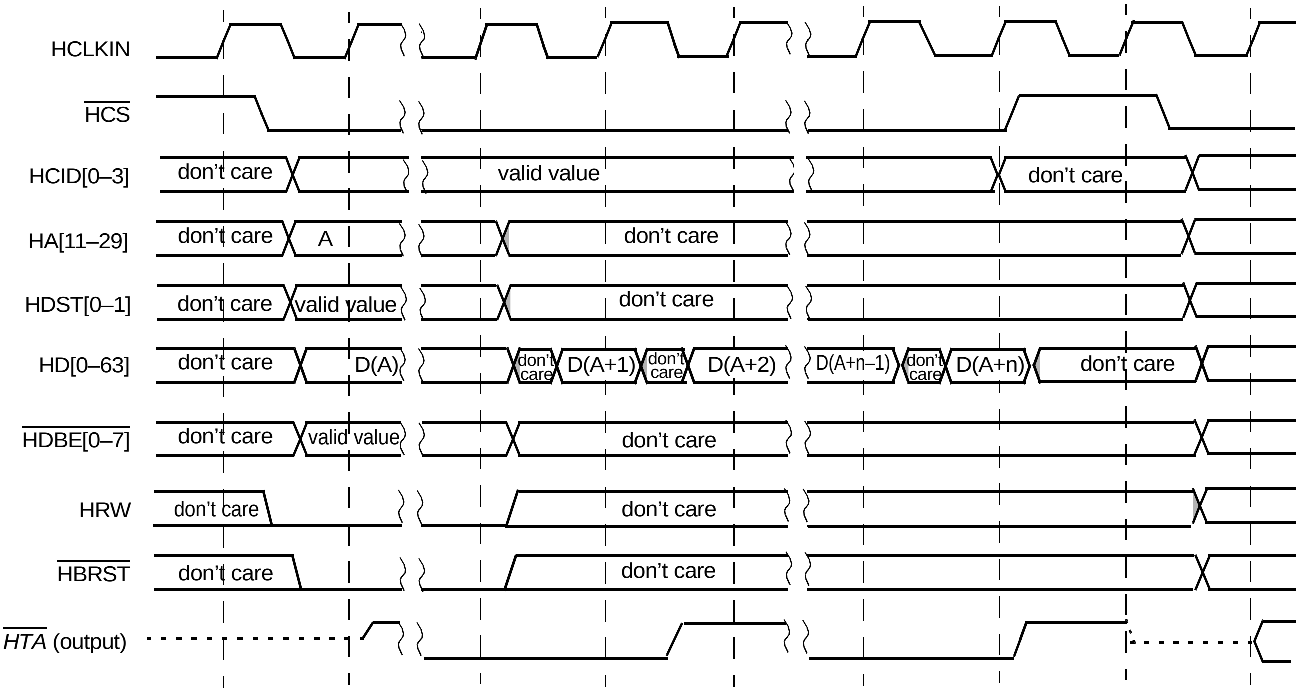


Note: For the byte transfers on HD[0–63], n = 3 for a 64-bit data bus interface and n = 7 for a 32-bit data bus interface

Figure 14-15. Synchronous Burst Write Using Dual Strobe Mode

14.3.4.6 Synchronous Burst Write Using Single Strobe Mode

Figure 14-16 shows a synchronous burst write access using single strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], HD[0–63], HRW, HDBE[0–7], and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. HDBE and HBRST are asserted and HRW is deasserted. Assertion of HTA indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically HTA is asserted immediately for each beat of the access. If the write buffer is full, HTA assertion is delayed. After the last beat of the access, HTA is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 immediately in the next HCLKIN rising edge without deasserting HCS between accesses. If the next access is not to the same MSC8113, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA.

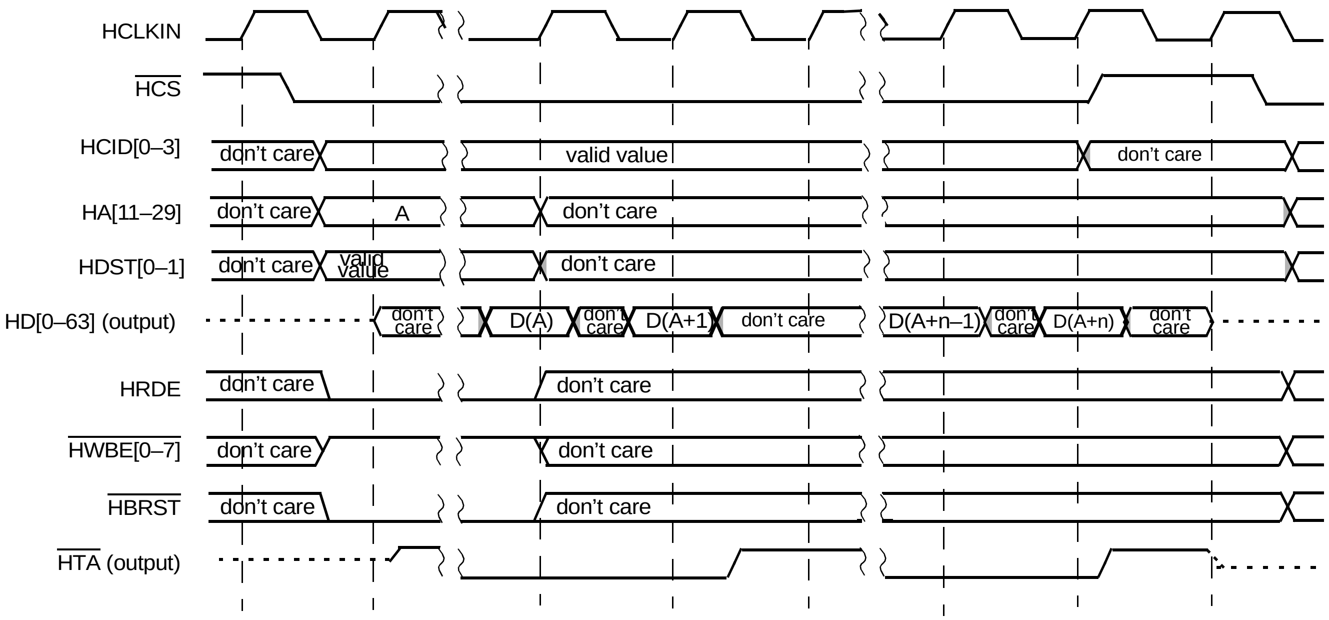


Note: For the byte transfers on HD[0–63], n = 3 for a 64-bit data bus interface and n = 7 for a 32-bit data bus interface

Figure 14-16. Synchronous Burst Write Using Single Strobe Mode

14.3.4.7 Synchronous Burst Read Using Dual Strobe Mode

Figure 14-17 shows a synchronous burst read access using dual strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], $\overline{\text{HWBE[0–7]}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. $\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE[0–7]}}$ are deasserted. When the DCR[RPE] bit (see **page 14-29**) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 device immediately in the next HCLKIN rising edge without deasserting $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8113, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal.

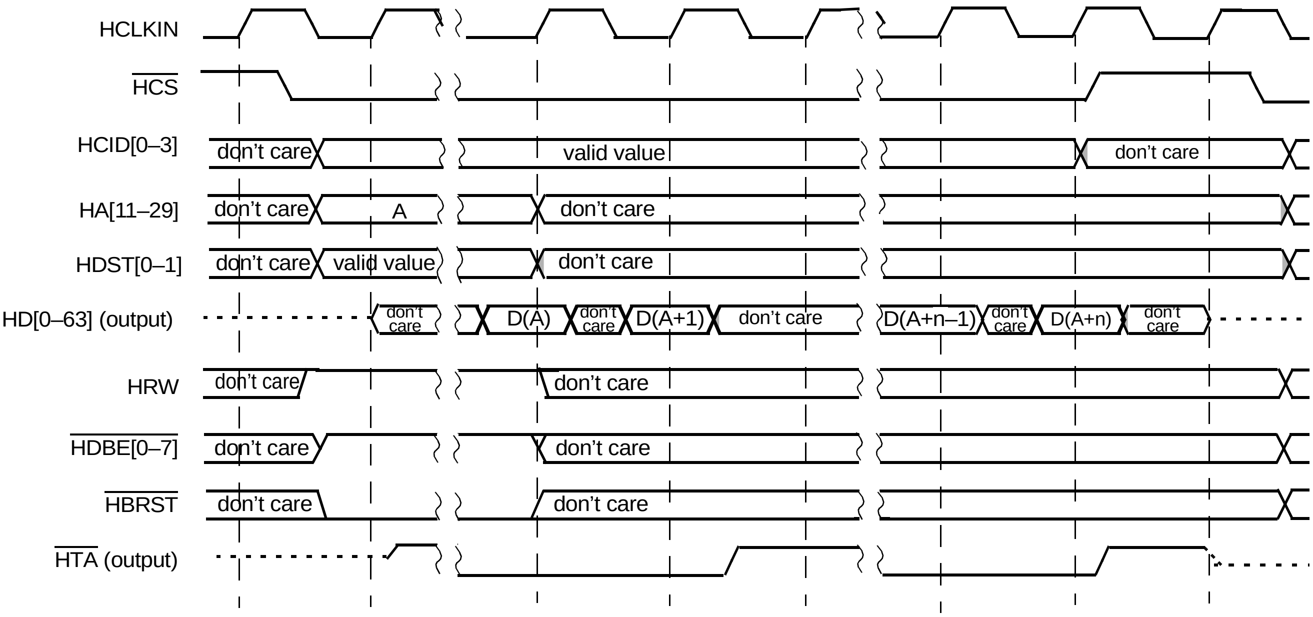


Note: For the byte transfers on HD[0–63], n = 3 for a 64-bit data bus interface and n = 7 for a 32-bit data bus interface.

Figure 14-17. Synchronous Burst Read Using Dual Strobe Mode

14.3.4.8 Synchronous Burst Read Using Single Strobe Mode

Figure 14-18 shows a synchronous burst read access using single strobe mode. The DSI samples HA[11–29], HDST[0–1], HCID[0–3], $\overline{\text{HDBE[0–7]}}$, HRW, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. HRW and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HDBE[0–7]}}$ are deasserted. When the DCR[RPE] bit (see **page 14-29**) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8113 device immediately in the next HCLKIN rising edge without deasserting $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8113, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal.



Note: For the byte transfers on HD[0–63], n = 3 for a 64-bit data bus interface and n = 7 for a 32-bit data bus interface.

Figure 14-18. Synchronous Burst Read Using Single Strobe Mode

14.3.5 Broadcast Accesses

Using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8113 devices for broadcasting write accesses. In Broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode HCID[0–3]. Broadcasting is allowed *only* for write accesses.

The DSI sets the DSI Error Register (DER) OVF bit (see **page 14-36**) if there is an overflow during broadcast accesses. You can reset this bit by writing a value of 1 to it. To avoid overflow when DSI registers are accessed during broadcast accesses, wait at least ten host clock cycles (in Synchronous mode, if the HCLKIN frequency is less than the internal clock frequency), ten internal clock cycles (in Synchronous mode, if the HCLKIN frequency is greater than the internal clock frequency), or eight internal clock cycles (in Asynchronous mode) after each DSI register access. In Asynchronous mode, to minimize the possibility of overflow when an internal address (other than DSI registers) is accessed during a broadcast access, wait at least three internal clock cycles before each broadcast access.

To avoid data corruption, when the DER[OVF] bit is set, any broadcast access is not written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, you must first read the DER[OVF] bit and reset it if it is set.

Note: In Asynchronous mode, write data from a previous access (even if it is from a previous normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers. Depending on the type of previous access, perform a read access prior to the first broadcast access. In Asynchronous mode, during a write to the DER to reset the overflow (OVF) bit, the host must allow eight internal clock cycles between the end of this write access and the start of a new broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In Asynchronous mode, the $\overline{\text{HWBS[0–7]}}$/$\overline{\text{HDBS[0–7]}}$ signal assertion time must be at least the minimum length of time, as defined in the *MSC8113 Technical Data* sheet in **Section 2.7**, *AC Timings*.
- In Synchronous mode single access, the host must wait one cycle before terminating the access. The access signals must be in the same valid state during two positive edges of the host clock cycles. The access duration is two clock cycles (The DSI may translate accesses lasting more than two clock cycles as two or more back-to-back accesses).
- In Synchronous mode burst accesses, broadcast accesses are not allowed.

14.4 DSI Configuration

The host can write the HRCW via the DSI during the PORESET flow. To simplify operation, this write access works as follows:

- The DSI uses only the HWBS0, HDBS0, HWBE0, or HDBE0 signal.
- Only one write cycle is allowed.
- The address for the write of the HRCW via the DSI is 0x1BE050. Address decoding for the write of the HRCW via the DSI is also performed on HCID[0–3]. Decoding for Host Chip-ID signals is not performed if the write access occurs in broadcast access.
- Only HD[0–31] are sampled with HD0 sampled as the most significant bit of the register.
- The host can work in Single Strobe or Dual Strobe mode even though the DSI is configured to Dual Strobe mode immediately after reset.

Note: The next access to the DSI must not be performed before the MSC8113 device is out of reset (see **Chapter 5**, *Reset*).

The following DSI operating modes are configured during reset:

- Synchronous/Asynchronous DSI mode is defined during the PORESET sequence by sampling the DSISYNC signal when PORESET is deasserted (see **Chapter 5**, *Reset*).
- DSI 32/64 bit data bus mode is defined during the PORESET sequence by sampling the DSI64 signal when PORESET is deasserted (see **Chapter 5**, *Reset*).
- The HRCW source is defined during the PORESET sequence by sampling the CNFGS and RSTCONF signals when PORESET is deasserted (see **Chapter 5**, *Reset*).
- DSI endian modes are defined in the HRCW during the PORESET sequence (see **Chapter 5**, *Reset*).

Note: All other DSI configuration settings are controlled by writing to the DSI control registers only after the MSC8113 is out of RESET.

After the reset sequence ends, the default value of some bits in the DSI control registers must be changed so that the host can work normally. All these bits are described on **page 14-29**. When changing these bit values, remember the following points about host operation:

- The default value of the DCR[SNGLM] bit after the reset sequence ends is logic 0, which is Dual Strobe access mode. If the host works in Single Strobe access mode, it can perform only write accesses to the DSI registers until this bit is set.
- The default value of the DCR[BEM] bit after the reset is logic 0, which specifies a single byte enable signal. If the host works with multiple byte enable signals, the host must assert HWBS0, HDBS0, HWBE0, or HDBE0 to perform a write. All other byte enable/strobe signals (HWBS[1–3/7], HDBS[1–3/7], HWBE[1–3/7], and HDBE[1–3/7]) are ignored.

- The default value of the DCR[HTAAD] bit after the reset sequence ends is logic 0, which specifies that the $\overline{\text{HTA}}$ signal released in logic 0. If the host expects the $\overline{\text{HTA}}$ signal to be released in logic 1, all the accesses should be spaced to allow enough time for the pull-up resistor to pull the signal value up to logic 1 before the HTAAD bit is set.
- The default value of the DCR[BRSTP] bit after the reset sequence ends is logic 0, which is the $\overline{\text{HBRST}}$ signal is active low. If the host works with active high signal connected to the DSI $\overline{\text{HBRST}}$ signal, it can perform only write accesses to the DSI registers until this bit is set. All write accesses to the DSI registers are treated as single accesses without decoding the value of $\overline{\text{HBRST}}$.
- The default value of the DCR[SLDWA] bit after the reset sequence ends is logic 0, which is sliding window non-active. A host in sliding window mode must connect the DSI address signals HA[11–13] as follows: HA[11–13] = [V_{CC}, V_{CC}, GND]. For accesses with HA14 = V_{CC}, this value equals the sliding window fixed base address so the host can reach the DCR to set SLDWA.The default value of the DCR[ADREN] bits after the reset sequence ends is logic 0, which specifies that access to external memory space can be done only by the “internal” slot or sliding window (DCR[SLDWA]) bit is set.

If you do not use the DSI, you can leave all the DSI pads floating, except for the $\overline{\text{HCS}}$ and $\overline{\text{HBCS}}$ pads, which must connect to the Vcc value. In this case, you must set the DDR[DSIDIS] bit as soon as possible after the MSC8113 reset sequence ends. Setting this bit disables the DSI input and output pads.

14.4.1 Stop Mode

The DSI internal clock is stopped under three conditions. See **Chapter 19**, *Internal Peripheral Bus (IPBus)*:

- The DDR[DSISTP] bit is set.
- There is no activity inside the DSI (all pending accesses have ended).
- There is a stop request from the IPBus.

When all of these conditions are simultaneously fulfilled, the DSI sends a stop acknowledge to the IPBus master, and the internal clock to the DSI can be stopped. To reset DDR[DSISTP], the internal clock must be activated.

14.4.2 DSI Reset During Host Access

If the MSC8113 completes a reset sequence during a host access to the DSI, then the DSI operation is undefined. It is therefore advisable that the host abort this access or at least use the MSC8113 $\overline{\text{SRESET}}$ to be aware that such a reset occurred.

14.5 DSI Programming Model

This section describes the programmable DSI registers mapped on the IPBus. For the complete list of registers and their addresses refer to **Table 8-6**, **Table 8-8**, and **Table 8-10**.

Note: The DSI registers can be accessed only with a 32-bit data structure. Smaller partitions of 8 or 16 bits are not supported.

The DSI registers discussed in this section are as follows:

- DSI Control Register (DCR), **page 14-29**
- DSI Sliding Window Base Address Register (DSWBAR), **page 14-31**
- DSI External Base Address Register (DEXTBAR), **page 14-32**
- DSI Internal Base Address Register (DIBAR[9, 11]), **page 14-32**
- DSI Internal Address Mask Register (DIAMR[9, 11]), **page 14-33**
- DSI Chip ID Register (DCIR), **page 14-34**
- DSI Disable Register (DDR), **page 14-34**
- DSI Status Register (DSR), **page 14-35**
- DSI Error Register (DER), **page 14-36**

14.5.1 Control Registers

The control registers control the DSI operation and can be read and written during operation.

DCR DSI Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SLDWA	BRSTP	BEM	SNGLM	HTAAD	LEDS		DSRFA	RPE	HTADT		—	ADREN			
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
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: In asynchronous mode, the first host access to the DSI after $\overline{\text{PORESET}}$ flow end must be a write access to the DCR to set the DSI to the correct mode of operation according to the host. This action must be taken even if the default value of the DCR is appropriate for the host operating mode. In Asynchronous mode, after any host write access to the DCR, the host must allow five internal clock cycles between the end of the write access and any other DSI access. During that time the $\overline{\text{HWBS[0–3/7]}}$/$\overline{\text{HDBS[0–3/7]}}$/$\overline{\text{HWBE[0–3/7]}}$/$\overline{\text{HDBE[0–3/7]}}$ and $\overline{\text{HRDS}}$/HRW/$\overline{\text{HRDE}}$ signals must be deasserted (held high).

Table 14-7. DCR Bit Descriptions

Name	Reset	Description	Settings
SLDWA 0	0	**Sliding Window Active** Defines the DSI Sliding Window mode of operation. See **Section 14.2.1** for details.	0 Sliding Window is non-active. 1 Sliding Window is active.
BRSTP 1	0	**Burst Signal Polarity** Defines the HBRST signal polarity. See **Section 14.3.2** for details.	0 HBRST signal is active low. 1 HBRST signal is active high.
BEM 2	0	**Byte Enable Multiple** Indicates whether the host uses all the byte enable signals. See **Section 14.1** for details.	0 Single byte enable signal is used. 1 Multiple byte enable signals are used.
SNGLM 3	0	**Single Strobe Mode** Indicates whether the host operates in single strobe mode. For details, see **Section 14.3.1**.	0 Dual strobe mode. 1 Single strobe mode.
HTAAD 4	0	**HTA Actively Driven** Indicates whether, at the end of the access, the HTA signal stops driving only after it is driven to a logic 1 at the end of the access. HTAAD is valid only for asynchronous accesses. See **Section 14.3.3** for details. **Note:** The reset value is 0—the device does not drive HTA high after a host access. If the system uses a pull-up resistor, you must set the HTAAD bit. Until HTAAD is set, the rising slope of HTA at the end of the access depends on the strength of the pull-up resistor; this results in a long delay for access termination.	0 HTA is released in logic 0.[1] 1 HTA is released in logic 1.[2]
LEDS 5–6	0	**Little-Endian Data Structure** When bit LTLEND in the HRCW (see **Chapter 5**, *Reset*) and bit DSRFA are set, LEDS defines the data structure of the host DSI accesses. For details, see **Section 14.2.4**.	00 8-bit data structure. 01 16-bit data structure. 10 32-bit data structure. 11 64-bit data structure.
DSRFA 7	0	**Data Structure Register Field Active** When bit LTLEND in the HRCW is set (see **Chapter 5**, *Reset*), DSRFA defines whether the data structure of the host DSI access is the value in the LEDS field or the value of the HDST signals. For details, see **Section 14.2.4**.	0 Data structure is from the HDST signals. 1 Data structure is from the LEDS field.
RPE 8	0	**Read Prefetch Enable**[3] Indicates whether the read prefetch mechanism is enabled.	0 Read prefetch mechanism is disabled. 1 Read prefetch mechanism is enabled.
HTADT 9-10	0	**HTA Drive Time** Defines the driving time of the HTA signal after the end of the access. It enables active drive HTA to logic “1” at the end of the access on PCB’s in which a pull-up is implemented on the HTA. To compensate for different available loads on HTA, three drive time options are provided. HTADT is valid only for asynchronous accesses. See **Section 14.3.3**.	00 No drive time. 01 HTA is driven for 0.5–1 internal bus clock cycles. 10 HTA is driven for 1–1.5 internal bus clock cycles. 11 HTA is driven for 1.5–2.5 internal bus clock cycles.
— 11	0	Reserved. Write to zero for future compatibility.	

Table 14-7. DCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
ADREN 12–15	0	**Address Enable** Specifies the address pins used for accessing the DSI.	0000 HA[11–29] address bits used.[5] 0001 HA[10–29] address bits used. 0010 HA[9–29] address bits used. 0011 HA[8–29] address bits used. 0100 HA[7–29] address bits used. 0101–1111 reserved
— 16–31	0	Reserved. Write to zero for future compatibility.	

Notes:
1. Should be used with HTADT = 00.
2. Should be used with HTADT = 01, 10, 11.
3. In munged little-endian mode, when working with 32 bit data bus width, this bit should be 0.
4. ADREN bits must be cleared (0) for the Sliding Window Mode.
5. Used at external region access by the “internal” slot or at sliding window mode.

DSWBAR DSI Sliding Window Base Address Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	BAVAL_H											BAVAL_L				—
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**
	—															EXTACC
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSWBAR is a control register that holds the DSI sliding window base address value.

Table 14-8. DSWBAR Bit Descriptions

Name	Reset	Description	Settings
BAVAL_H 0–10	0	**Base Address Value High** Stores the High bits of the External Access sliding window base address value. See **Figure 14-2**, *Sliding Window Mode Address Construction,* on page 14-7.	Any value from 0b0000 to 0b1111.
BAVAL_L 11–14	0	**Base Address Value Low** Stores the Low bits of the sliding window base address value (for External and Internal access). See **Figure 14-2**, *Sliding Window Mode Address Construction,* on page 14-7.	Any value from 0b0000 to 0b1111.
15–30	0	Reserved. Write to zero for future compatibility.	
EXTACC 31	0	External Access Used at *Sliding Window Mode*, the value written in EXTACC represent internal/external access. See **Figure 14-2**, *Sliding Window Mode Address Construction,* on page 14-7	0 Access is to the chip internal address space. 1 Access is to the external address space.

DEXTBAR DSI External Sliding Window Base Address Register

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

DEXTBAR is a control register that holds the DSI External sliding window base address value.

Table 14-9. DEXTBAR Bit Descriptions

Name	Reset	Description	Settings
EXTBAVAL 0–15	0	**Base Address Value** Stores the External Memory base address. Used for address construction at Full Address Mode (see **Figure 14-1**, *DSI Main Addressing Mode Selection,* on page 14-6).	Any value.
15–31	0	Reserved. Write to zero for future compatibility.	

DIBAR9
DIBAR11 DSI Internal Base Address Registers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	BA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	See **Table 8-7*Banks 9 and 11 Address Space,*** on page 8>-28															

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

Table 14-10. DIBAR Bit Descriptions

Name	Reset	Description
BA 0–16	0	**Base Address** Must contain the same value as the BA field in the BR9 and BR11. See **Section 12.8**, *Memory Controller Programming Model*.
— 17–31	0	Reserved. Write to zero for future compatibility.
Note:	There must be no change in the DSI DIBARx registers or in the SIU BR9 or BR11 if accesses are pending in the DSI write FIFO. To avoid this situation, you should make any changes in these registers immediately after setting the DSI configuration registers or perform a read access to the DSI to empty the write FIFO. If a host on the DSI updates these registers, all registers must be written one after the other, without any access to a different memory region in between.	

DIAMR9 DSI Internal Address Mask Register 9

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	AM															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	AM	—														
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-11. DIAMR9 Bit Descriptions

Name	Reset	Description
AM 0–16	0	**Address Mask** Must contain the same value as the AM field in OR9 in GPCM mode. See **Section 12.8**, *Memory Controller Programming Model*.
— 17–31	0	Reserved. Write to zero for future compatibility.
Note: There must be no change in DSI DIAMR9 or in SIU OR9 if there are pending accesses in the DSI write FIFO. To avoid this situation, you should make any changes in these registers immediately after setting the DSI configuration registers or perform a read access to the DSI to empty the write FIFO. If a host on the DSI updates these registers, all registers must be written one after the other, without any access to a different memory region in between.		

DIAMR11 DSI Internal Address Mask Register 11

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	AM															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	AM	—														
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-12. DIAMR11 Bit Descriptions

Name	Reset	Description
AM 0–16	0	**Address Mask** Must contain the same value as the AM field in the OR11 register, UPM mode. Refer to **Section 12.8**, *Memory Controller Programming Model*.
— 17–31	0	Reserved. Write to zero for future compatibility.
Note: There must be no change in DSI DIAMR11 or in SIU OR11 if accesses are pending in the DSI write FIFO. Always make any changes to these registers immediately after setting the DSI configuration registers or perform a read access to the DSI to empty the write FIFO. If a host on the DSI updates these registers, all the registers must be written one after the other, with no access to a different memory region in between.		

DCIR DSI Chip ID Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	CHIPID				—											
Type	R/W															
Reset	X	X	X	X	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
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-13. DCIR Bit Descriptions

Name	Reset	Description	Settings
CHIPID 0–3	x[1]	**Chip ID Value** Stores the Chip ID value sampled during PORESET signal deassertion. Writing to this register overrides the reset value. For details, see **Chapter 5**, *Reset*. CHIPID[3] unused in case of DCR[ADREN] set to 0b0011 or 0b0100.	Any value between 0b0000 to 0b1111
— 4–31	0	Reserved. Write to zero for future compatibility.	

Notes: **1.** The reset value depends on the value of CHIP_ID[0–3] during PORESET signal deassertion.

2. In asynchronous mode, after a host write access to the DCIR, the host must allow a period of 5 internal clock cycles between the end of the write access and the beginning of any other access on the bus (during which time, HCS signal must be deasserted.

DDR DSI Disable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	DSISTP	DSIDIS	—													
Type	R/W from IPBus. Read-only from host side															
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
	—															
Type	R/W from IPBus. Read-only from host side															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-14. DDR Bit Descriptions

Name	Reset	Description	Settings
DSISTP 0	0	**DSI Stop** Setting this bit is one of the conditions that can freeze the DSI inner clock. This register is writable only from the IPBus; an external host can only read this register. See **Section 14.4.1**.	0 DSI is not allowed to access into stop mode. 1 DSI is allowed to access into stop mode.
DSIDIS 1	0	**DSI Disable** Setting this bit disables the input and output pads of the DSI so you can leave the DSI pads not connected (except for the $\overline{\text{HCS}}$ and $\overline{\text{HBCS}}$ pads). This register is writable only from the IPBus; an external host can only read this register. **Note:** If you are using the DSI and want to disable its pads, make sure that you do not set the DSIDIS bit in the middle of a host-to-DSI transaction. When this bit is set, host transactions to the DSI are not allowed.	0 DSI pads are enabled. 1 DSI pads are disabled.
— 2–31	0	Reserved. Write to zero for future compatibility.	

14.5.2 Status Registers

DSR DSI Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	DSI64	DSISYNC	LTLEND	PPCLE	RCWSRC	—										
Type	R															
Reset	X	X	X	X	X	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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-15. DSR Bit Descriptions

Name	Reset	Description	Settings
DSI64 0	X[1]	**32/64 bit Data Bus** Stores the value of the DSI64 signal that is sampled during $\overline{\text{PORESET}}$ signal deassertion. For details, see **Chapter 5**, *Reset*.	0 32-bit data bus. 1 64-bit data bus.
DSISYNC 1	X[2]	**Asynchronous/Synchronous mode** Stores the value of the DSISYNC signal, which is sampled during $\overline{\text{PORESET}}$ signal deassertion. For details, see **Chapter 5**, *Reset*.	0 Asynchronous mode. 1 Synchronous mode.
LTLEND 2	X[3]	**Little-Endian mode** Stores the value of the LTLEND bit in the HCRW. For details, see **Chapter 5**, *Reset*.	0 Big Endian mode. 1 Little Endian mode.
PPCLE 3	X[4]	**Munged Little-Endian mode** Stores the value of the PPCLE bit in the HRCW. For details, see **Chapter 5**, *Reset*.	0 True Little Endian host. 1 PowerPC Little Endian host.

Table 14-15. DSR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RCWSRC 4	X[5]	**Hard Reset Configuration Word Source** Stores a value which indicates if the HRCW source is the DSI. This value is taken from the combination of the CNFGS and $\overline{\text{RSTCONF}}$ signals, which are sampled during $\overline{\text{PORESET}}$ signal deassertion. For details, see **Chapter 5**, *Reset*.	0 DSI is not the hard reset configuration word source. 1 DSI is the hard reset configuration word source.
— 5–31	0	Reserved. Write to zero for future compatibility.	

Notes:
1. The reset value depends on the value of DSI64 during $\overline{\text{PORESET}}$ signal deassertion.
2. The reset value depends on the value of DSISYNC during $\overline{\text{PORESET}}$ signal deassertion.
3. The reset value depends on the value of bit LTLEND in the HRCW.
4. The reset value depends on the value of bit PPCLE in the HRCW.
5. The reset value depends on the value of CNFGS and $\overline{\text{RSTCONF}}$ during $\overline{\text{PORESET}}$ signal deassertion.

DER DSI Error Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	OVF	—														
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**
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DER is the DSI error register. Its bits are sticky bits set by the DSI when an error occurs. You can reset the sticky bit by writing to this register with the bit location set.

Table 14-16. DER Bit Descriptions

Name	Reset	Description	Settings
OVF 0	0	**Overflow** Indicates whether an overflow has occurred. Writing 1 to this bit clears it. See **Section 14.3.5**, *Broadcast Accesses,* on page 14-26.	0 Overflow did not occur. 1 Overflow occurred.
— 1–31	0	Reserved. Write to zero for future compatibility.	

Hardware Semaphores 15

The MSC8113 hardware semaphores (HS) hold eight coded hardware semaphores. A coded hardware semaphore provides a simple way to achieve a “lock” operation via a single write access, eliminating the need for such sophisticated read-modify-write mechanisms as the reservation. Using the hardware semaphores, external hosts such as DSI external masters can protect internal and external shared resources and ensure coherency in any sequence of operations on these resources. Each coded hardware semaphore is an eight-bit register with a selective write mechanism, as follows (see **Figure 15-1**):

- The semaphore is *free* if its value is zero.
- The semaphore is *locked* if its value is non-zero and its value is the *lock code*. Each processor/task that needs the lock capability of the semaphore must have a unique non-zero eight-bit code for locking the semaphore for correct protocol operation.
- A write of a non-zero value (lock code) is successful only if the current value of the semaphore is zero (free). This write is defined as a successful *lock* operation, and the written value is the *lock code*.
- A write of a non-zero value (lock code) is ignored if the current value of the semaphore is non-zero (locked). This write is defined as a failed *lock* operation, since the coded semaphore is considered locked with the non-zero code it holds.
- To maintain the protocol coherency, only the master that successfully locked the semaphore is allowed (and obligated) to free it. A write of zero is always successful and is defined as a *free* operation.

When a processor/task must lock an unlocked semaphore to its unique code, it writes its unique lock code to the semaphore register. It then reads back the semaphore value, and if it matches its lock code, the lock operation is successful. A value mismatch (a different non-zero code or zero) indicates to the master that the lock operation failed and must be retried. After a successful lock operation, the master can do any coherent operation associated with this semaphore and then it must free the semaphore (write zero).

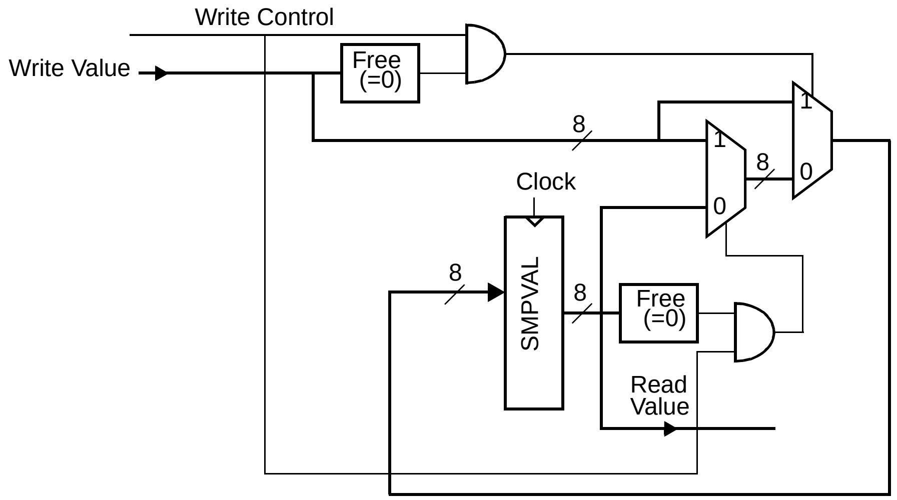


Figure 15-1. Hardware Semaphore Block Diagram

The hardware semaphore registers reside in the 256 KB address space of the IPBus and have a constant offset. They are accessed through the SQBus, the system bus, and the DSI. The addresses of the hardware semaphore registers for accesses through the SQBus are presented in **Section 8.5**, *IPBus Address Space*, on page 8-12. The addresses of the hardware semaphore registers for accesses through the system bus are presented in **Section 8.7**, *System Bus Address Space*, on page 8-55. The addresses of the hardware semaphore registers for accesses through the DSI are presented in **Section 8.8**, *DSI Address Map*, on page 8-61.

HSMPR[0–7] Hardware Semaphore Register 0–7

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—								SMPVAL							
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-1. HSMPRx Bit Descriptions

Name	Reset	Description	Settings
— 0–23	0	Reserved. Write to zero for future compatibility.	
SMPVAL 24–31	0	**Semaphore Value** The eight-bit coded semaphore value. It holds the current semaphore value. A non-zero value indicates the current lock code.	0 Semaphore is free. It is writable to any value. ≠ 0 Semaphore is locked with lock code indicated by its current value. It is writable only to zero.

Direct Memory Access (DMA) Controller 16

The MSC8113 multi-channel DMA system supports up to sixteen time-multiplexed channels and buffer alignment by hardware. The DMA controller connects to both the system bus and local bus and functions as a bridge between both buses. Transactions run simultaneously on both buses. Flyby transactions (also known as *single access transactions*) can occur on either bus. The DMA controller enables hot swap between time-multiplexed channels with no cost in clock cycles. Synchronous and asynchronous transfers occur using sixteen priority levels or round-robin priority on the bus and give a varying bus bandwidth per channel. The DMA controller services up to thirty-two different requestors. A requestor can be any one of four external peripherals or sixteen internal requests generated by the DMA FIFO itself, plus eight M1 flyby counters.

Using all the bus features, the DMA controller accommodates a total of seven different issued bus transactions on both the system bus and the local bus. For example, each bus handles one transaction in the data phase, one transaction in the address phase, and one pending transaction. Each DMA channel handles a single unidirectional transaction at one time. The transaction can be any one of the following (see **Figure 16-1**):

- Memory to DMA FIFO
- DMA FIFO to memory
- Peripheral to DMA FIFO
- DMA FIFO to peripheral
- Memory to peripheral, in Flyby mode
- Peripheral to memory, in Flyby mode
- Internal memory to internal memory, in Flyby mode using flyby address counters

You can modify channel attributes. Each request activates a buffer according to its type, as programmed in the dedicated parameter RAM.

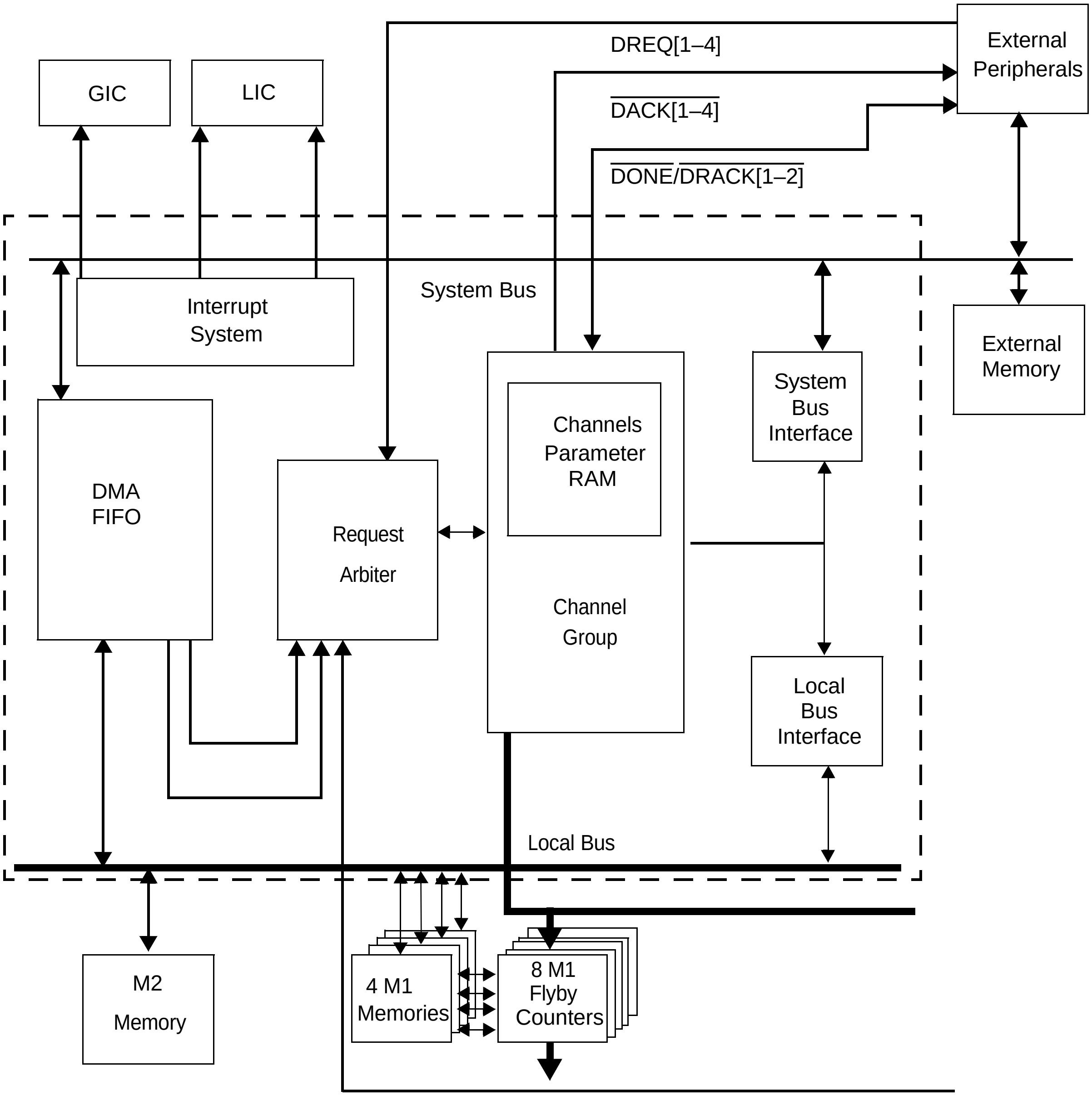


Figure 16-1. DMA System Diagram

16.1 DMA Signals: Requestor Interface

A requestor is an external peripheral or an internal request generated by the DMA FIFO. A peripheral interfaces with the DMA controller by placing a request for service. The request can be external or internal, depending on its origin.

16.1.1 Signal Functionality

The DMA signals are as follows:

- *DMA Peripherals Handshake*. The DMA handshake to a peripheral uses four peer-to-peer signals. Normally, peripherals use only a subset of these signals.
  - — DMA request (DREQ). Sent to indicate that the peripheral is requesting DMA service. When the DMA controller samples it, the DMA controller initiates a transaction on the bus that services the requesting peripheral. DREQ can be configured to work in high or low level-triggered mode or in rising or falling edge-triggered mode. This signal is synchronized by the DMA controller and can be driven asynchronously by a peripheral.
  - — DMA data acknowledge ($\overline{\text{DACK}}$). Asserted by the DMA controller. Qualification with the $\overline{\text{PSDVAL}}$ signal means that the data phase on the bus is generated by the DMA controller for the peripheral connected to this $\overline{\text{DACK}}$. It is asserted during the entire data phase. The transaction is as follows:
    - *Write transaction*. The peripheral samples the data during $\overline{\text{DACK}}$ with the qualifying $\overline{\text{PSDVAL}}$ signal.
    - *Read transaction*. The peripheral drives the data during $\overline{\text{DACK}}$ with the qualifying $\overline{\text{PSDVAL}}$ signal.
  - — $\overline{\text{DONE}}$. This bidirectional signal signifies that the channel must be terminated.
    - If the DMA controller generates $\overline{\text{DONE}}$, the channel handling this peripheral is inactive and the peripheral is not serviced further.
    - As an input to the DMA controller, $\overline{\text{DONE}}$ closes the channel, as in normal channel closing, when the channel size reaches zero. During closing, the FIFOs are emptied and the ACTV bit in the channel's configuration register (DCHCR[ACTV]) is deasserted.
  - — Data request acknowledge ($\overline{\text{DRACK}}$). The DMA controller asserts this signal to indicate that it has sampled the peripheral request. The peripheral can deassert its current request and assert a new request if needed. When a peripheral is using the $\overline{\text{DRACK}}$ signal option, it should not assert $\overline{\text{DONE}}$.

With a regular DMA protocol, a peripheral cannot issue a new request until the DACK signal to this peripheral is asserted. When the FIFO of a peripheral is larger than the port size, the efficiency of the DMA operation can be improved by allowing this peripheral to issue a second request before the current request is serviced. For details on the DMA to peripheral access size, see **Section 16.2.1**. When the DMA controller samples the DREQ of a device, it generates a bus command within a clock, but it can take a number of clocks until the transaction generated by the DMA controller appears on the system bus with the DACK to the peripheral. This delay can occur since both the system bus and the local bus function as multi-master buses. Additional delays can occur because the DMA controller and the buses are pipelined, causing a newly generated command to wait up to three pipe stages. With the DRACK protocol, the DRACK signal is asserted whenever the DMA controller samples a peripheral request, enabling an advanced peripheral to assert a new edge-triggered request if needed.

- *DMA-FIFO Handshake*. The internal FIFO signals the DMA logic that service is needed in the same way that a peripheral requests DMA service. When a channel is enabled and the FIFO has space for at least one burst, a hungry request is generated. If at least one burst is valid in the FIFO, a watermark request is generated. This enables memory-to-memory transfer where no external request is available. If the FIFO cannot handle the request from the internal FIFO, the request is masked by the FIFO.

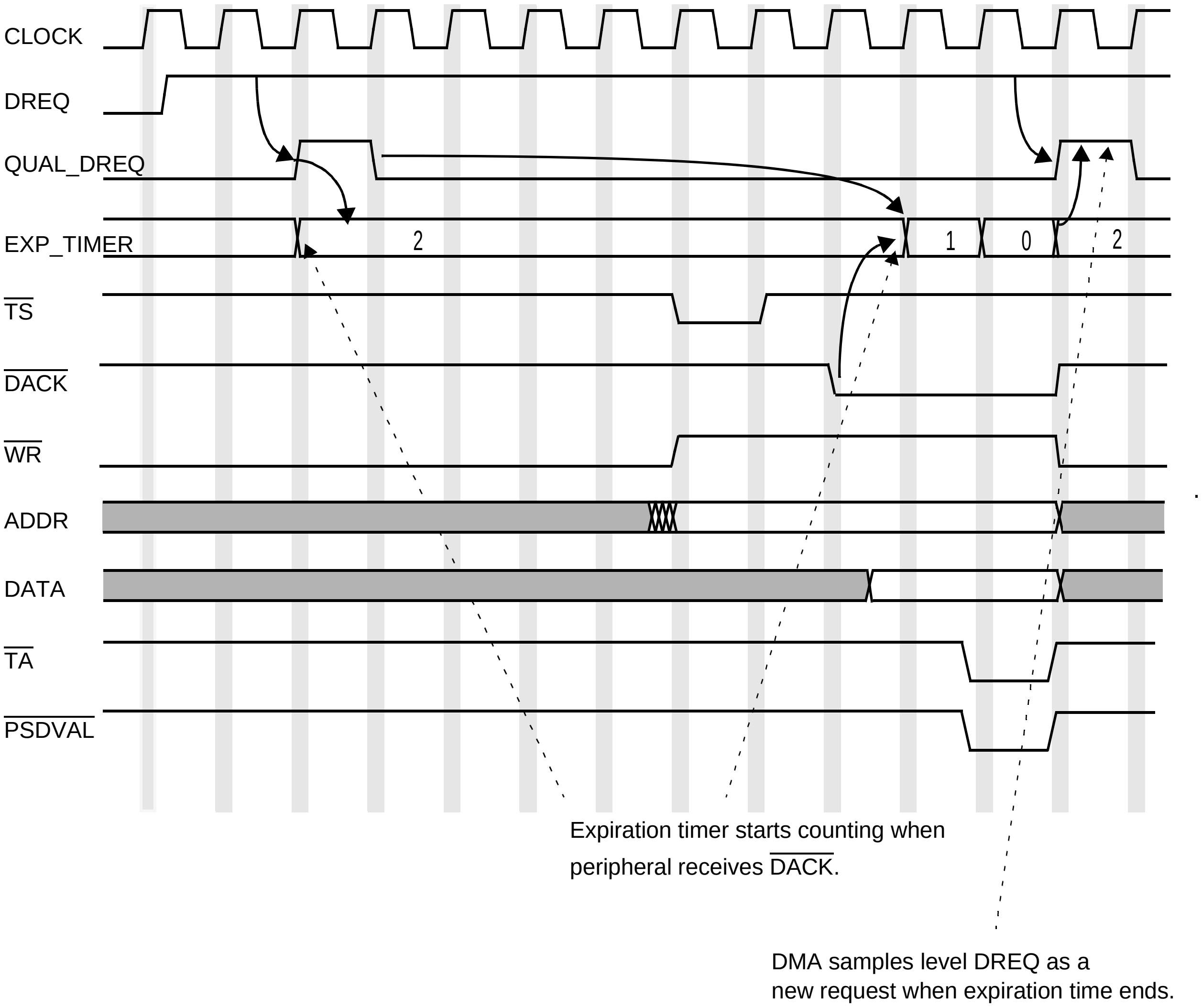
16.1.2 Peripheral Access Timing

The timing of DMA operations depends on the following factors:

- Type of request: edge-triggered or level-triggered
- Protocol used: regular or DRACK
- Expiration timer value for level-triggered requests

Note: When a peripheral drives the DREQ signal asynchronously, DREQ must remain at the same level for at least 2.5 bus clocks. Otherwise, the DMA controller can “miss” requests. When asynchronous DREQ is used, the DCHCR[EXP] field (see **page 16-35**) must be programmed to the response time of the peripheral plus two clocks.

Figure 16-2 illustrates the timing for a peripheral with a level-triggered request and expiration timer, without the use of the DRACK protocol.



Expiration timer starts counting when peripheral receives DACK.

DMA samples level DREQ as a new request when expiration time ends.

Figure 16-2. Level-Triggered Request and Expiration Timer

Figure 16-3 shows the timing for a peripheral with an edge-triggered request, synchronized DREQ input, and the use of the $\overline{\text{DRACK}}$ protocol.

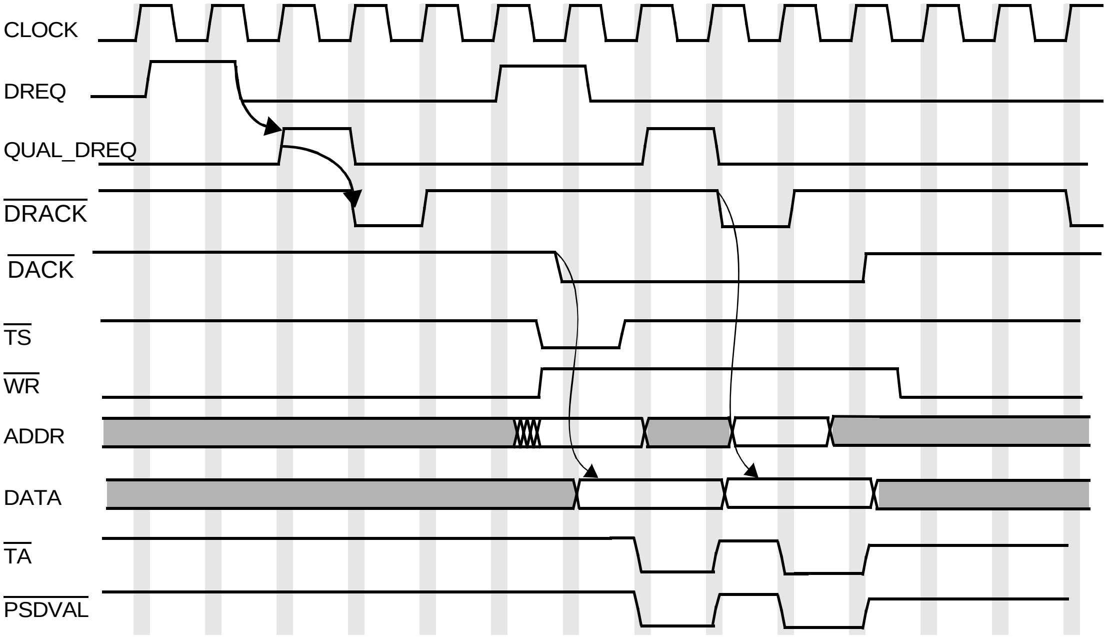


Figure 16-3. Edge-Triggered Request, Synchronous DREQ, $\overline{\text{DRACK}}$ Signals

Figure 16-4 shows the timing for a peripheral with a level-triggered request, asynchronous DREQ input, and the use of the $\overline{\text{DRACK}}$ protocol.

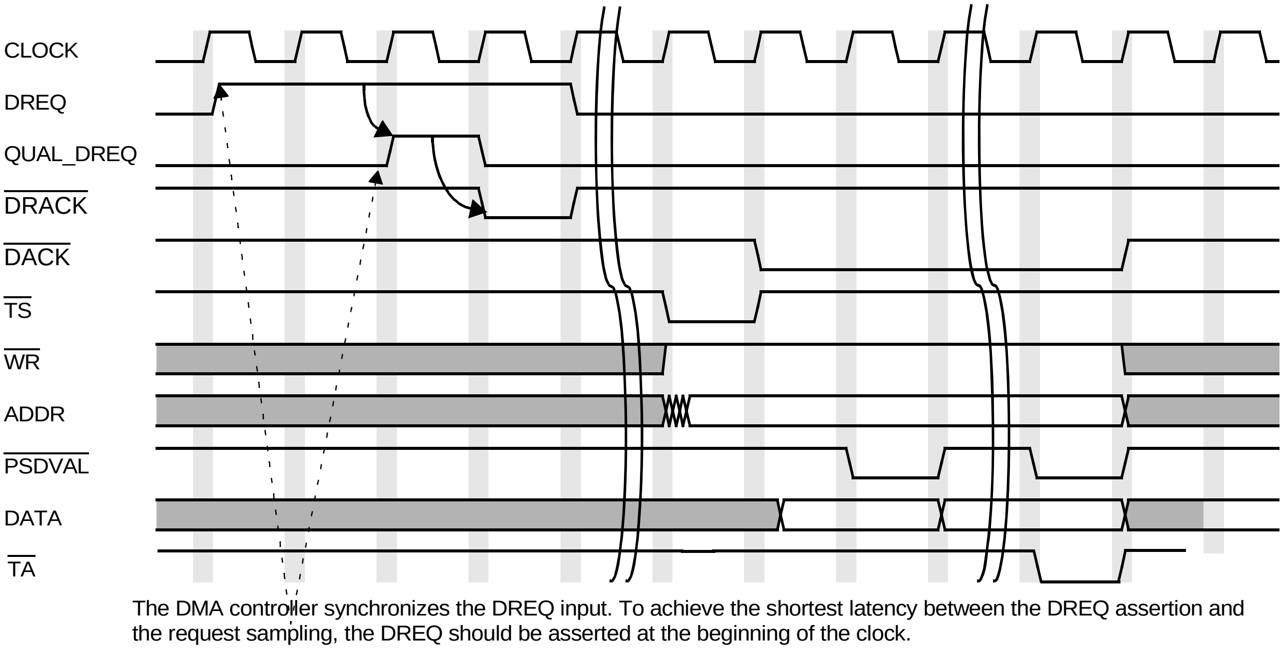
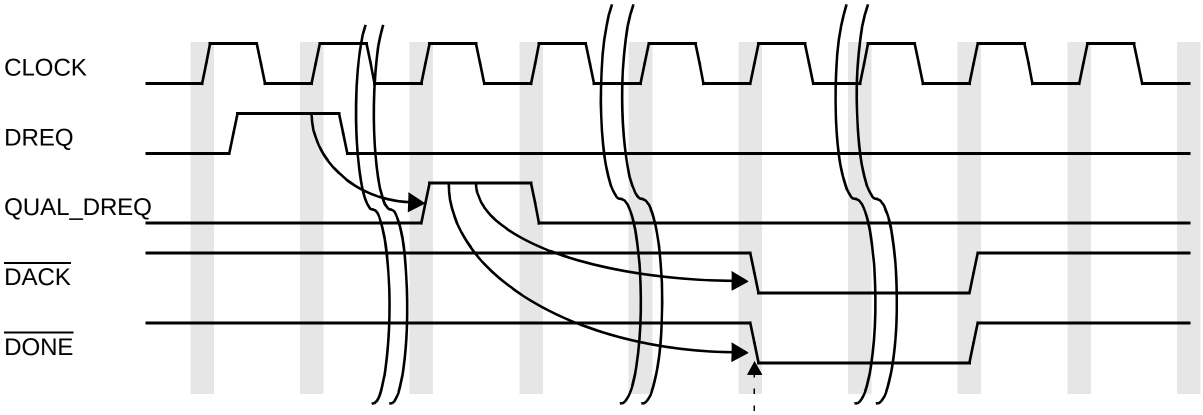

CLOCK
DREQ
QUAL_DREQ
$\overline{\text{DRACK}}$
$\overline{\text{DACK}}$
$\overline{\text{TS}}$
$\overline{\text{WR}}$
ADDR
$\overline{\text{PSDVAL}}$
DATA
$\overline{\text{TA}}$
The DMA controller synchronizes the DREQ input. To achieve the shortest latency between the DREQ assertion and the request sampling, the DREQ should be asserted at the beginning of the clock.

Figure 16-4. Level-Triggered Request, Asynchronous DREQ, $\overline{\text{DRACK}}$ Signals

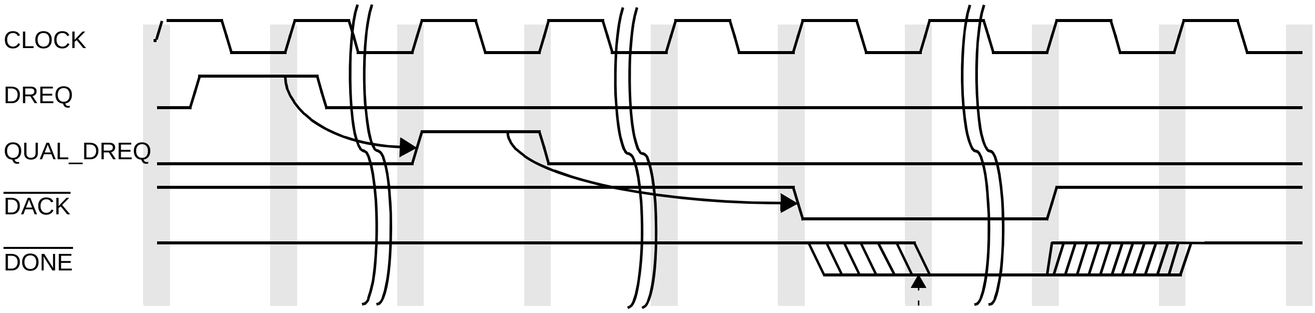
Figure 16-5 shows the timing for a peripheral with a request, and simultaneous assertion of DACK and DONE by the DMA controller.



DMA controller asserts DONE at
the same time as DACK.

Figure 16-5. Simultaneous Assertion of DACK and DONE Signals

Figure 16-6 shows the timing for a peripheral with a request, and sequential assertion by the peripheral of DACK and DONE.



A peripheral should assert the DONE
signal only after the DACK signal is asserted
for this peripheral’s last request. It should deassert
the DONE signal not later than one cycle
after the DACK signal is deasserted. DONE
should be asserted for at least one clock.

Figure 16-6. Sequential Assertion of DACK and DONE Signals

16.2 DMA Operating Modes: Transfer Types

The MSC8113 DMA controller supports all combinations of data transfers between external memory, internal memories, and external peripherals. Typical transfers are as follows:

- External memory to or from an external peripheral in normal mode
- External peripheral to or from internal memories in normal mode
- External memory to external memory in normal mode
- External memory to or from any of the internal memories in normal mode
- Internal memory to any of the internal memories in normal mode
- External peripheral to or from external memory (flyby mode)
- Internal memory to internal memory (flyby mode)

This section describes these transfer types and illustrates them with figures that show the routing for each transfers. However, we must first consider DMA transfer and peripheral port size and the two DMA access modes.

16.2.1 DMA Transfer Size and Peripheral Port Size

Accesses to peripherals usually occur with fixed-size transfers known as peripheral port size (PS). The memory controller adjusts the port size. For example, if the peripheral PS is 32 bits and the DMA controller issues a 64-bit transaction, the memory controller divides the 64-bit transaction into two 32-bit transactions. See **Section 12.8**, *Memory Controller Programming Model*, on page 12-95. The transfer size parameter, TSZ, determines the size of the transaction issued by the DMA controller after each peripheral request DREQ. The size of the TSZ parameter ranges from one byte to a full burst. The transfer size is related to the size of the peripheral FIFO. See the description of the DMA Channel Parameter RAM (DCPRAM) on **page 16-38**.

16.2.2 DMA Access Modes

The DMA controller generates two types of transactions: Normal and flyby:

- *Normal or dual access transaction*. In Normal mode, the data path is as follows:
  - — Peripheral to DMA FIFO in a read transaction
  - — DMA FIFO to peripheral in a write transaction

  The peripheral behaves as a memory-mapped area.

  The DMA controller uses a bus access (read or write) to access a peripheral. The peripheral detects the access in one of two possible ways:

  - — $\overline{\text{DACK}}$ assertion during the data phase. The access is into a virtual address
  - — Bus address and control decoding. The access is into the peripheral’s internal register.

  A peripheral can be accessed at a fixed address location, at incremental addresses, or at cyclic incremental addresses. For dual access transfers, the DMA controller uses two consecutive channels. The even channel performs the read (from requestor/memory into the DMA FIFO), and the odd channel performs the write (DMA FIFO to the other memory/requestor).

- *Flyby or single access transaction*. In Flyby mode, the data path is between a peripheral and memory with the same port size, located on the same bus. Flyby operations do not require access to the DMA FIFO:

  — A *read* from peripheral is a *write* transaction to memory.
  — A *write* to peripheral is a *read* transaction from memory.

  The advantage of a flyby transaction is that the data is transferred in a single cycle and not in two, as in a normal transaction. When the transaction source drives data on the bus, the transaction destination samples it. Flyby transactions occur between external peripherals and external memories on the system bus. They also occur between internal memories and internal memories (M1 of one SC140 core to M1 of another SC140 core, M2 to M1, M1 to M2). When a flyby transaction between two internal memories is requested, one of the M1 memories operates as a peripheral. This memory ignores the address phase. It has an associated flyby counter, which receives a $\overline{\text{DACK}}$ signal from the DMA controller. The flyby counter should be programmed with the initial M1 memory address. When the counter receives the asserted $\overline{\text{DACK}}$ qualified with $\overline{\text{PSDVAL}}$, it replaces the local bus address to the M1 memory by its own value. The other memory operates as a “normal” memory responding to the address phase. To change the flyby counter A value, program the FlyBy Address Control Registers (FLBACRA) in the EQBS with the M1 memory address (according to the local bus address space). The address should be divided by 8 and written to the FlyBy Start Address, FLBSA in the FlyBy Address Control Register. Same procedure can be done also on FlyBy Address Control Register B (FLBACRB). These two Flyby Address counters can work in parallel. See the FLBSA, FLBSB description in **Section 9.3.9**, *EQBS Programming Model*.

Note: When programming the DMA controller for a flyby transaction between two on-device memories, use the $\overline{\text{DRACK}}$ protocol for higher performance. See **Section 16.1**, *DMA Signals: Requestor Interface*.

Note: DMA channels are coupled in pairs (0 and 1, 2 and 3, up to 14 and 15). Do not use two coupled channels simultanously for flyby or single access transactions.

16.2.3 Application Examples

16.2.3.1 External Memory and an External Peripheral on the System Bus

Figure 16-7 shows a transfer of data from an external memory to an external peripheral on the system bus. Two accesses are executed: external memory to DMA FIFO and DMA FIFO to external peripheral.

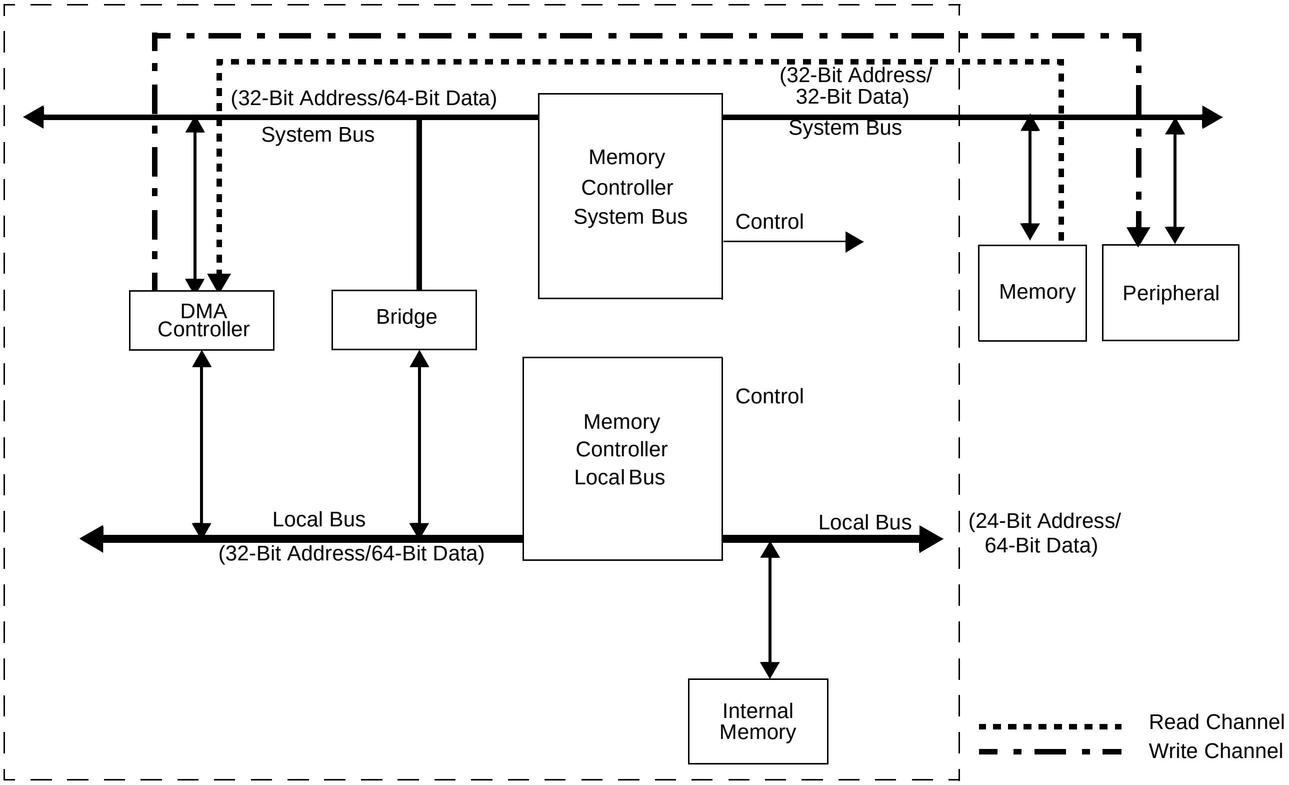


Figure 16-7. External Memory to an External Peripheral on the System Bus

16.2.3.2 External Peripheral to Internal Memory

Figure 16-8 shows a transfer of data from an external peripheral to an internal memory. Two accesses are executed: external peripheral to DMA FIFO and DMA FIFO to internal memory.

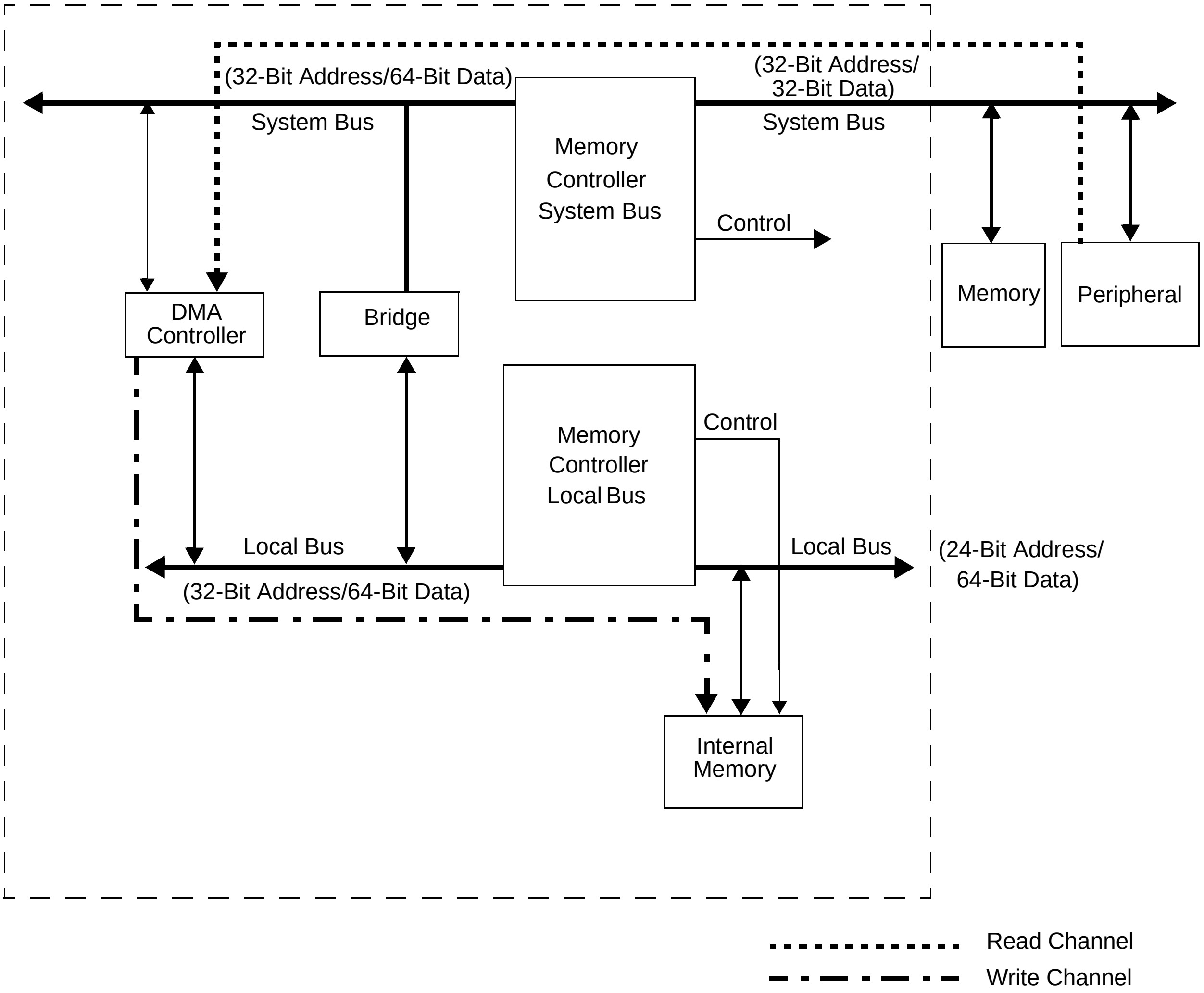


Figure 16-8. External Peripheral to Internal Memory

16.2.3.3 External Peripheral to External Peripheral

Figure 16-9 shows a transfer of data from an external peripheral to an external peripheral. Two accesses are executed: external peripheral to DMA FIFO and DMA FIFO to external peripheral.

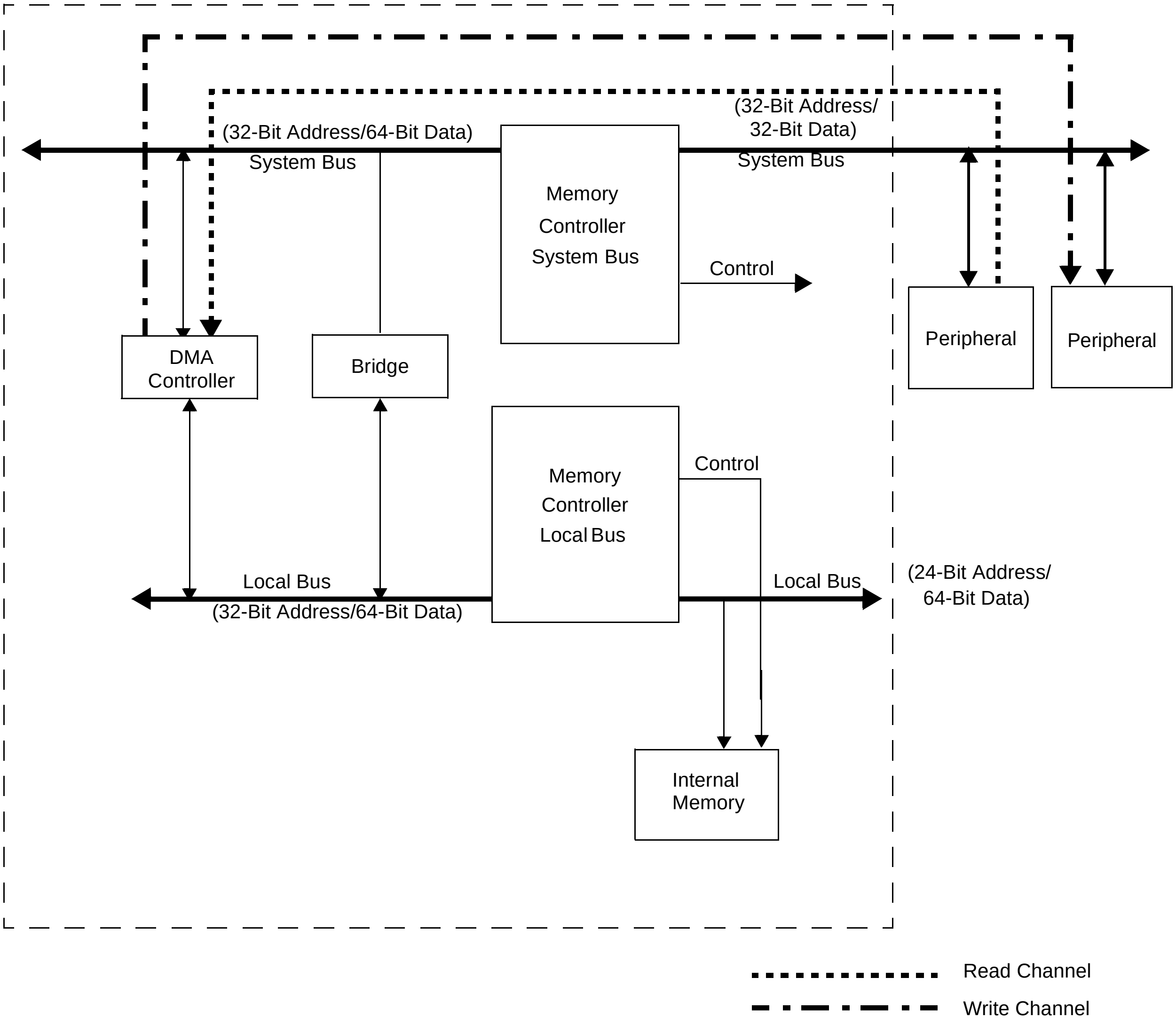


Figure 16-9. External Peripheral to an External Peripheral

16.2.3.4 External Memory and External Memory on the System Bus

Figure 16-10 shows a transfer of data from an external memory to an external memory on the system bus. Two accesses are executed: external memory to DMA FIFO and DMA FIFO to external memory.

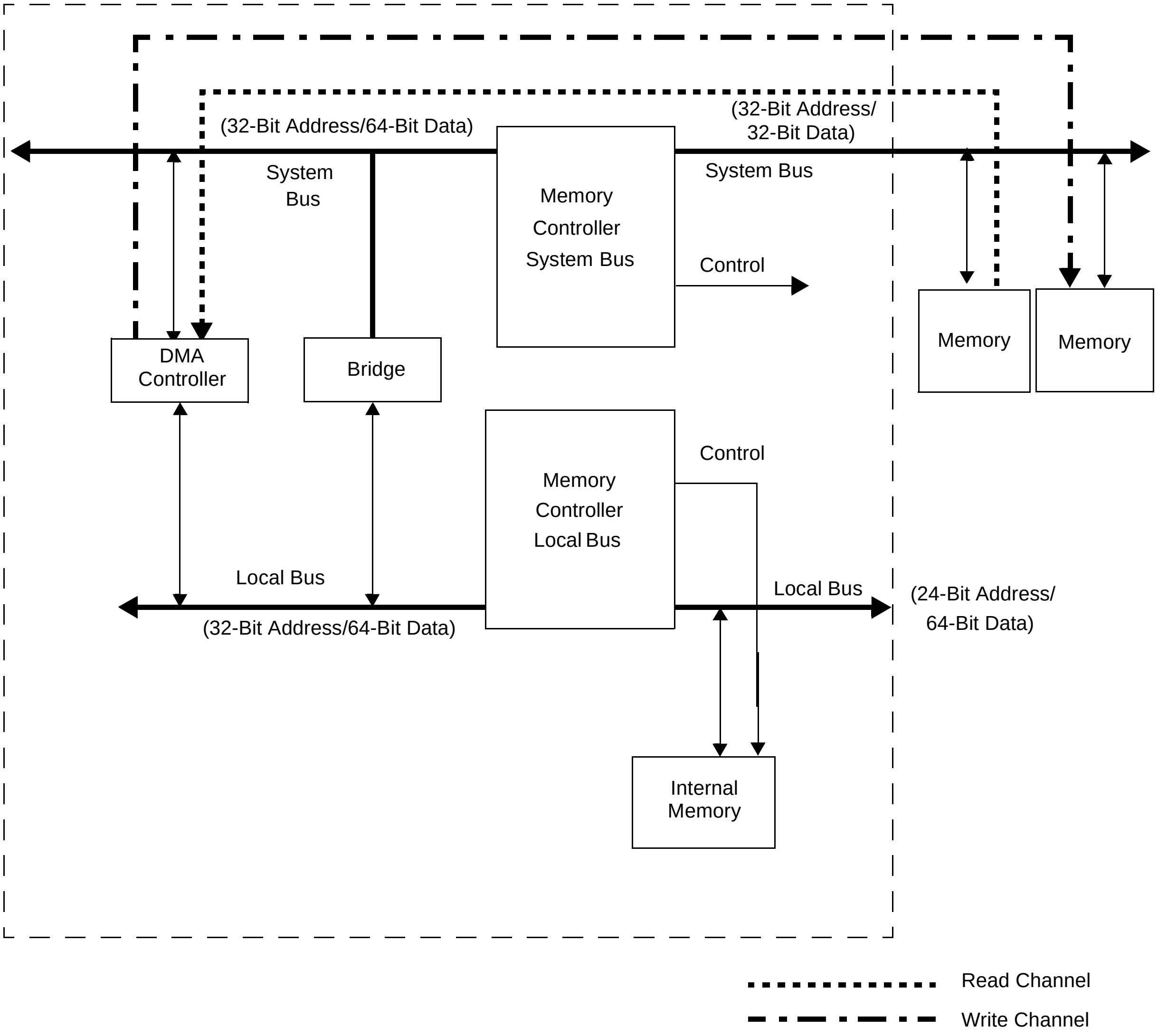


Figure 16-10. External Memory to External Memory

16.2.3.5 External Memory to Internal Memory on the System Bus

Figure 16-11 shows a transfer of data from an external memory on the system bus to internal memory. Two accesses are executed: external memory to DMA FIFO and DMA FIFO to internal memory.

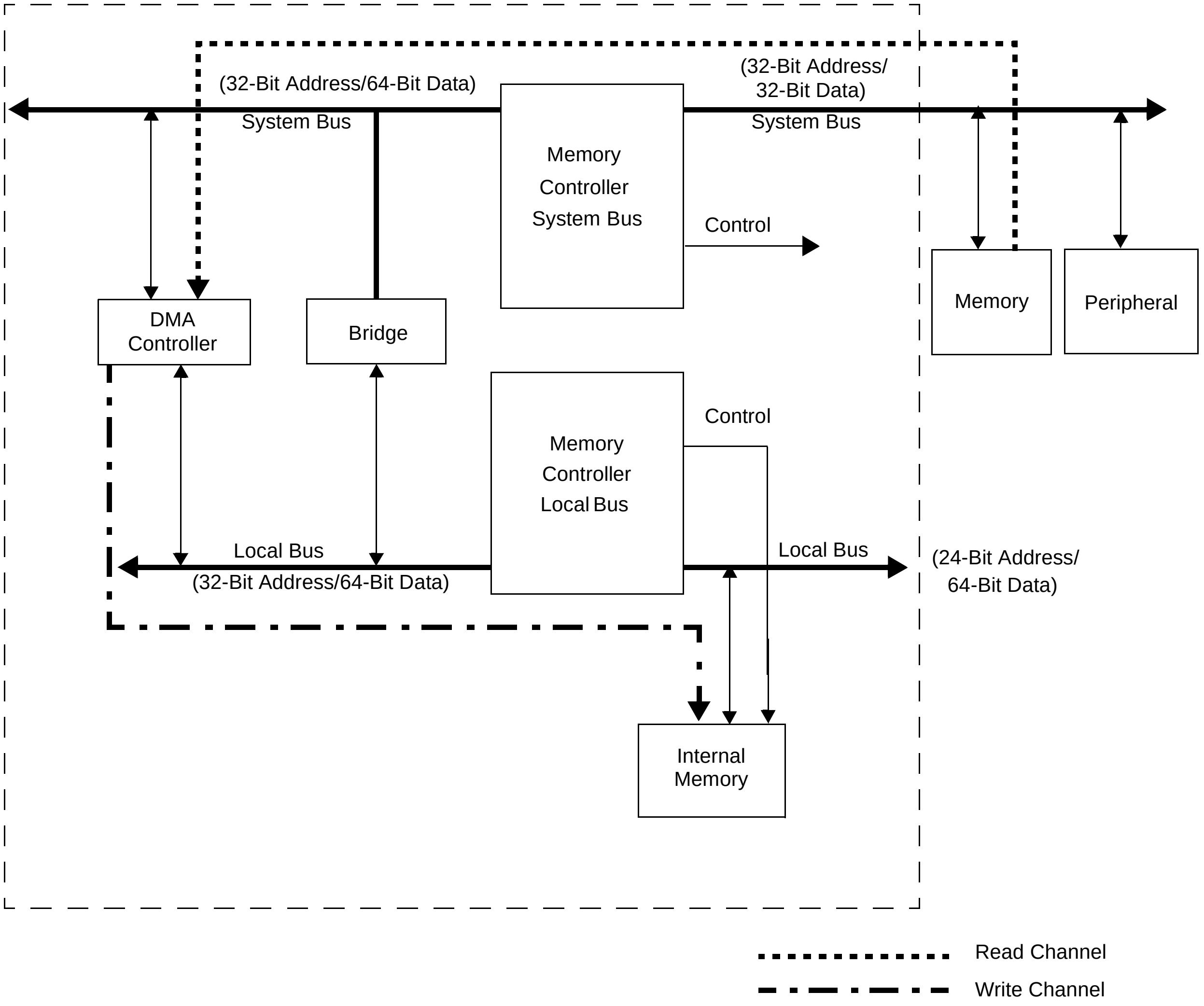


Figure 16-11. External Memory to Internal Memory

16.2.3.6 Internal Memory to Internal Memory

Figure 16-12 shows a transfer from an internal memory to an internal memory. Two accesses are executed: internal memory to DMA FIFO and DMA FIFO to internal memory.

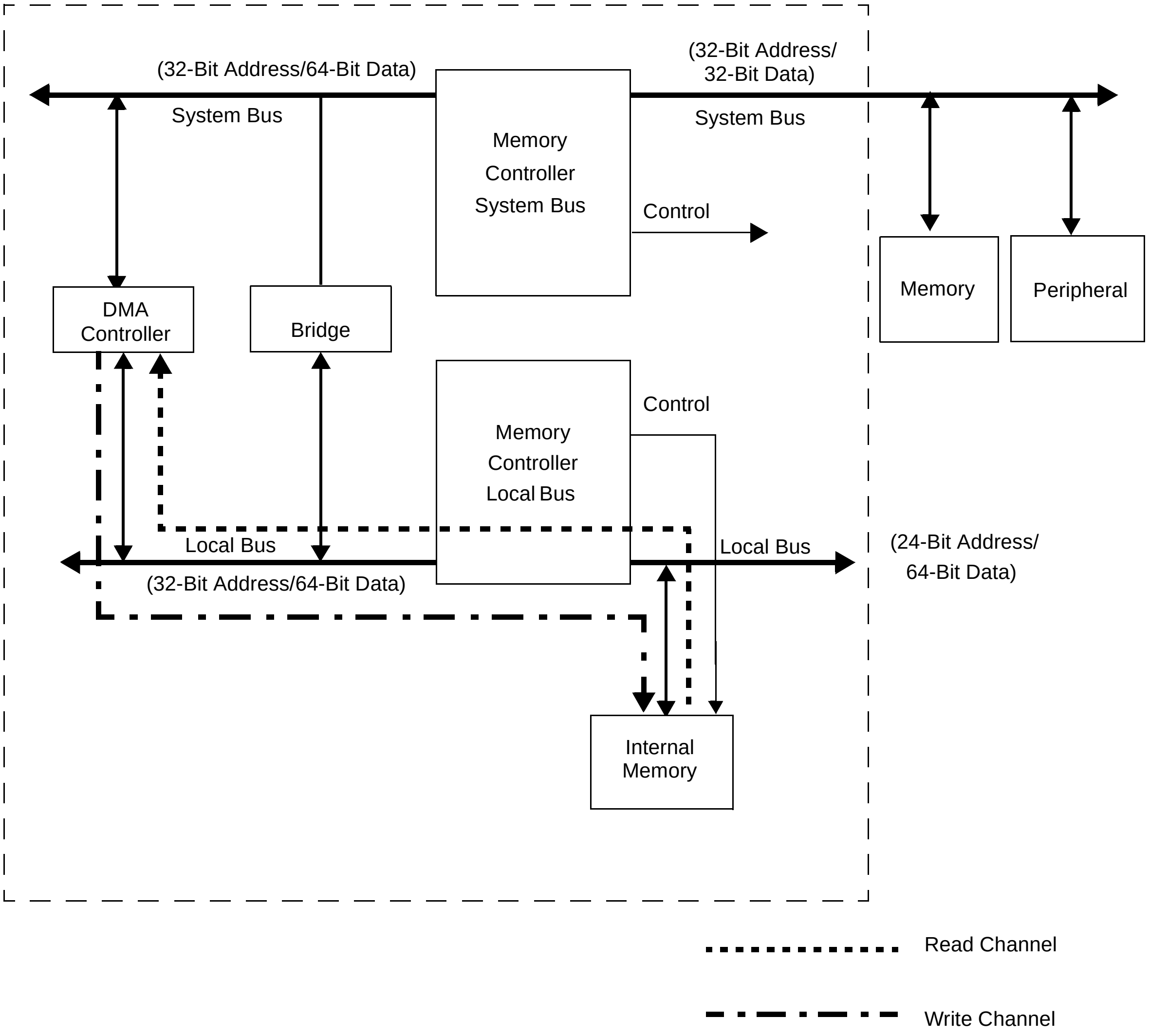


Figure 16-12. Internal Memory to Internal Memory

16.2.3.7 Flyby Transfer from External Peripheral to External Memory

Figure 16-13 shows a transfer between an external memory and an external peripheral executed in Flyby mode—that is, in a single access transaction. The DMA controller only signals the peripheral, and the transfer is performed directly between the external memory and the external peripheral.

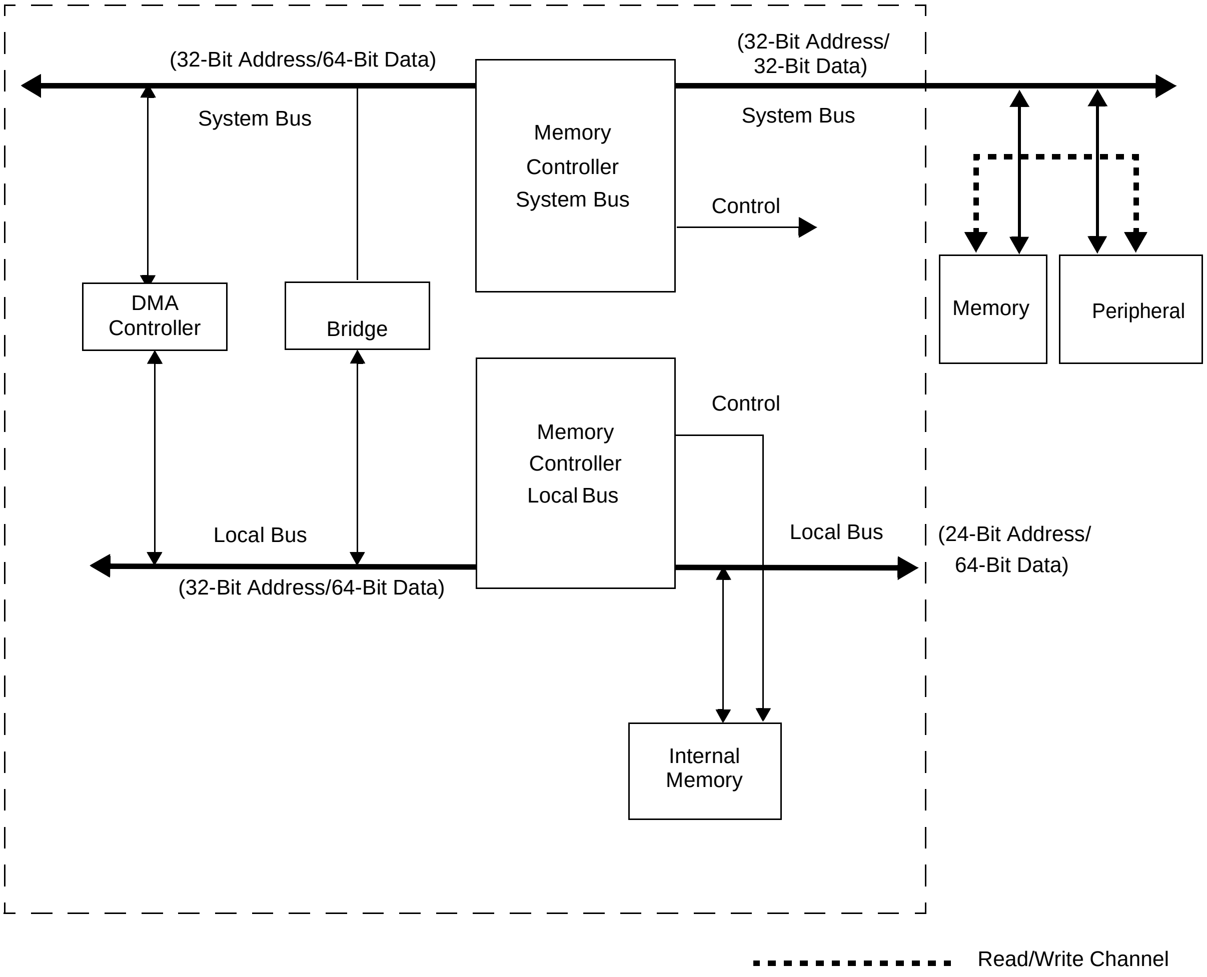


Figure 16-13. Flyby Transfer From External Peripheral to External Memory

16.2.3.8 Flyby Transfers Between Internal Memories, M2 and M1

Figure 16-14 shows a transfer between an internal memory (M2) and an internal memory (M1) executed in Flyby mode. The DMA controller sends a $\overline{\text{DACK}}$ signal to the M1 flyby counter while M2 receives a real address. The transfer executes directly between M1 and M2.

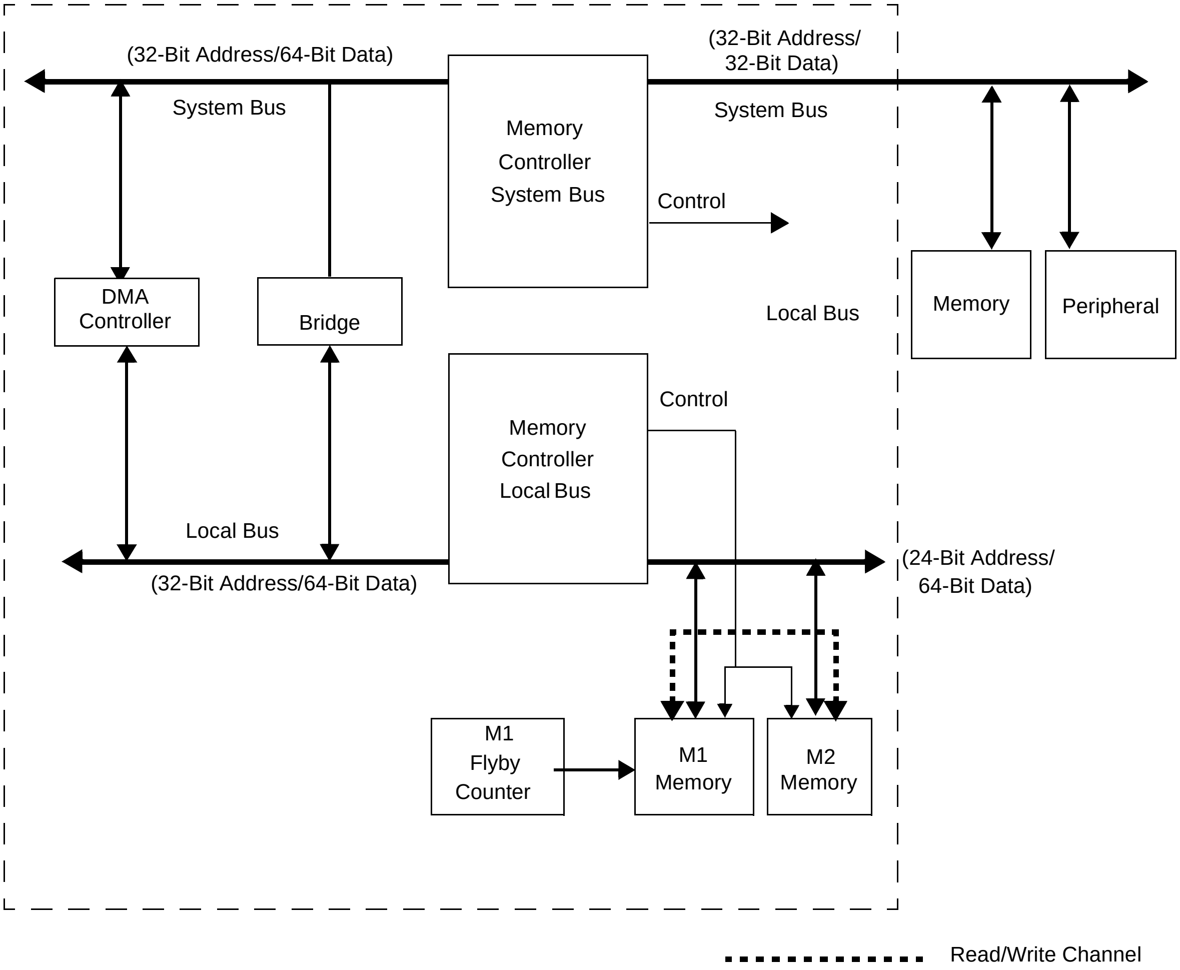


Figure 16-14. Flyby Transfer Between Internal Memories, M2 and M1

16.2.3.9 Transfers Between Internal Memories M1 and M1 (Flyby Mode)

Figure 16-15 shows a transfer between an internal memory block (M1) from one SC140 core and an internal memory block (M1) from another SC140 core executed in Flyby mode. The DMA controller sends a $\overline{\text{DACK}}$ signal to the M1 counter of one of the SC140 cores, and the other memory is memory-mapped. The transfer executes directly between core 0 M1 and core 1 M1.

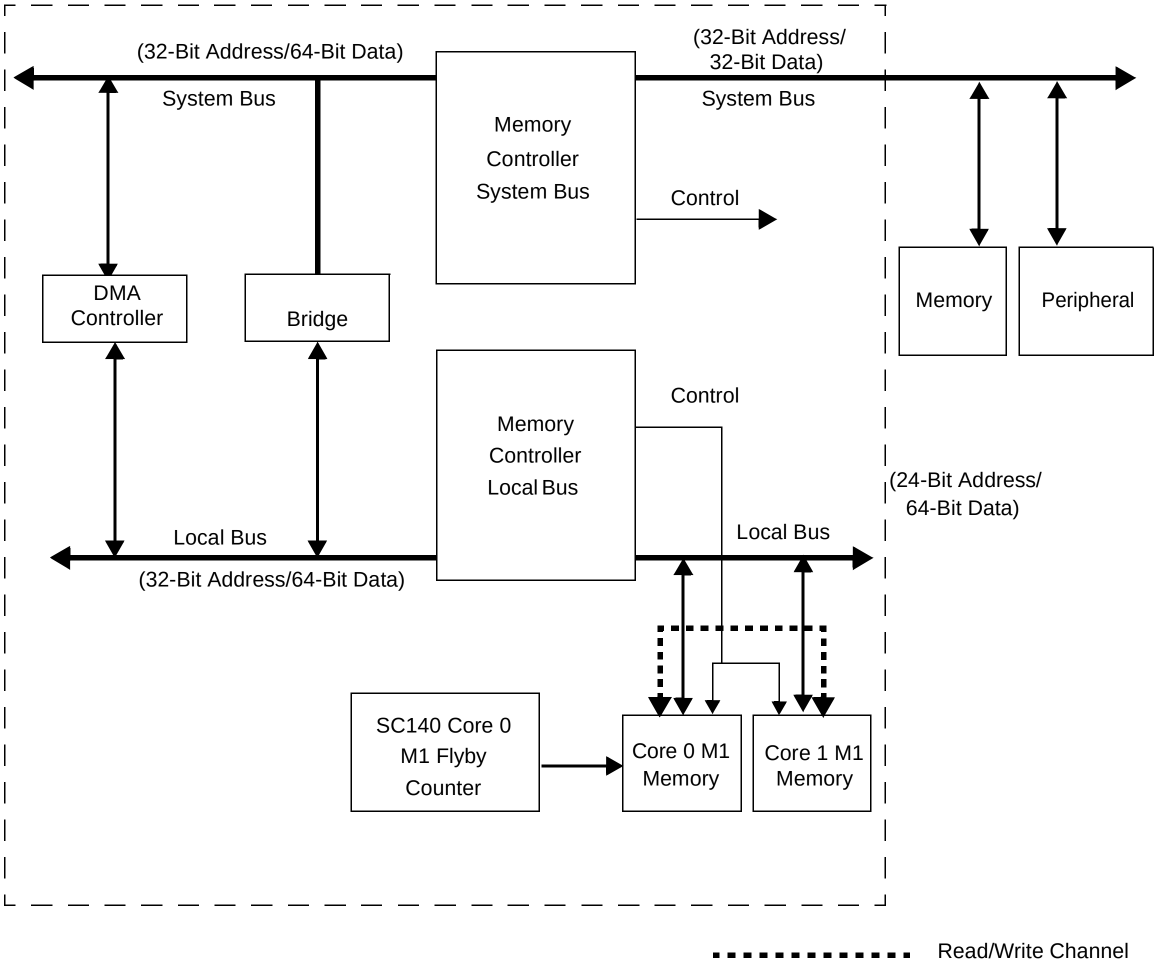

(32-Bit Address/64-Bit Data)
System Bus
Memory Controller System Bus
(32-Bit Address/ 32-Bit Data)
System Bus
Control
Memory
Peripheral
DMA Controller
Bridge
Memory Controller Local Bus
Control
(24-Bit Address/ 64-Bit Data)
Local Bus
(32-Bit Address/64-Bit Data)
Local Bus
SC140 Core 0 M1 Flyby Counter
Core 0 M1 Memory
Core 1 M1 Memory
Read/Write Channel

Figure 16-15. Flyby Transfer Between Internal Memories, M1 and M1

16.2.4 DMA Operating Modes: Buffer Types

The DMA controller employs several types of buffers. A channel buffer is described in the DMA Channel Parameter RAM (DCPRAM) space as a one-line buffer descriptor (BD) composed of four 32-bit parameters. The buffer is activated by a DMA channel when the channel wins DMA internal arbitration. When the buffer is activated, the DMA generates a bus transaction with the size described in the Buffer Descriptor Attributes Transfer Size field (BD_ATTR[TSZ]) and decrements the Buffer Descriptor Size (DCPRAM[BD_SIZE]), accordingly. The address can be incremented or frozen. When the BD_SIZE reaches zero, the channel does one of the following:

- Shuts down (simple buffer)
- Reinitializes itself (cyclic buffer)
- Reinitializes its size (incremental buffers)
- Switches to another buffer (chained-buffers)
- Any combination of the above (dual cyclic buffers)

See the description of the DCPRAM on **page 16-40** for details on the different fields of the buffer descriptor attributes. The following example code shows the DCPRAM parameter functionality. The sections that follow provide examples of several types of buffers. The DCPRAM[BD_ATTR] fields listed for each example include only the fields that do not contain zero values.

Example 16-1. Behavior of DCPRAM Parameters

```
if (handle channel[bdptr]) {
/* Any buffer type in steady state */
   if (BD_SIZE[bdptr] > transfer_size)
     BD_SIZE[bdptr] = BD_SIZE[bdptr] - transfer_size;
       if (NO_INC == increment address mode)
       BD_ADDR[bdptr] = BD_ADDR[bdptr] + transfer_size;
   }
/* Any buffer type, before last transaction */
   elsif (BD_SIZE[bdptr] != 0) {
     if (NO_INC == increment address mode) BD_ADDR[bdptr] = BD_ADDR[bdptr]+
       BD_SIZE[bdptr];
   }
/* Any buffer type, in last transaction */
   elsif (BD_SIZE[bdptr] == 0) {
       if (CYC[bdptr] == 1) & (NO_INC[bdptr] == 0) { /* cyclic buffer with
                      address increment */
       BD_ADDR[bdptr] = BD_ADDR[bdptr] - BD_BSIZE[bdptr];
       }
       BD_SIZE[bdptr] == BD_BSIZE[bdptr];/* initial BD_size */
/* Not continuous buffer */
   if (CONT[bdptr] == 1){
/* Chained buffer, size and address are determined according to new channel */
       if (NBD[bdptr] != bdptr) bdptr=NBD;
   }
   else Close_buffer, gen_interrupt;
}
```

When programming code that uses DMA buffers, you must consider how the DMA controller uses buffers and channels and the order in which related events occur.

Example 16-2. Potential Race Conditions

Scenario: If the DMA controller writes to a destination that is in memory and the optional interrupt function is enabled, the interrupt is generated when the controller completes the channel or buffer operation. The interrupt only indicates that the write operation was initiated by the DMA controller. Due to internal bus arbitration, the actual data transfer may not occur until several core clocks later. If the core processes the interrupt handler immediately, the resulting data operations may be erroneous due to a race condition.

Solution: There are at least two possible options.

1. In the interrupt handler, include code that polls the end of the destination buffer until the contents change before further data transfer or processing.
2. Program the DMA controller to use an additional small buffer and trigger the interrupt at the end of that dummy buffer. The larger buffer transfer should be completed by the time the interrupt occurs.

Example 16-3. Double Interrupt Generation

Scenario: If the INTRPT bit is set in the BD_ATTR of the last buffer in a non-continuous DMA transfer, the DMA channel issues two interrupts: one when the last buffer completes and one because the BD_SIZE value reaches zero.

Solution: Make sure that the INTRPT bit in the BD_ATTR of the last buffer in a non-continuous DMA transfer is cleared.

Example 16-4. Missing DMA Interrupt in Chained Buffer

Scenario: The DMA controller is activated with a chained buffer. Every buffer should generate an interrupt when it is done. The interrupt service routine (ISR) performed by the core clears the relevant status bit in the DSTR. If an additional buffer completes before the core clears the DSTR bit, the core does not receive the appropriate interrupt because it is a level interrupt.

Solution: Make sure that the ISR clears the DSTR as soon as possible. You can read the DCHCR after the interrupt is processed to determine whether another buffer was completed or if the DMA channel is still active.

16.2.4.1 Simple Buffer

A simple buffer closes when DCPRAM[BD_SIZE] reaches zero. It is defined by setting the BD_ATTR[CONT] field to zero. **Figure 16-16** shows an example of a simple buffer.

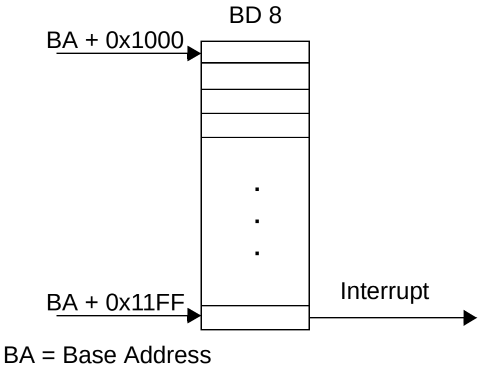


Figure 16-16. Simple Buffer

Table 16-1 lists the configuration of a simple buffer, designated as channel BD 8 in this example. A 0x200 byte block is read from address 0x1000, the channel closes when the transfer completes, and an interrupt is generated. Burst transactions are used on the bus.

Table 16-1. DCPRAM Values for a Simple Buffer

BD	DCPRAM Parameters		Value	Description
8	BD_ADDR		BA + 0x1000	External memory buffer current address
	BD_SIZE		0x200	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate an interrupt when buffer ends
		CYC	0x0	Increment BD_ADDR when the size reaches zero
		CONT	0x0	Non-continuous mode: the channel is closed when the size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		—	Buffer base size of cyclic buffer

16.2.4.2 Cyclic Buffer

A cyclic buffer is a continuous buffer. When the buffer at the current address reaches zero, the pointer jumps back to the base address and the buffer is executed again. An example cyclic buffer is shown in **Figure 16-17**.

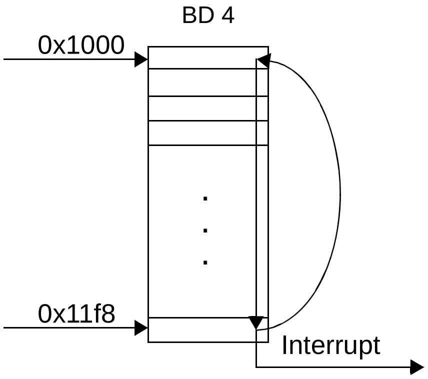


Figure 16-17. Cyclic Buffer

Table 16-2 lists the DCPRAM values for channel BD 4. A 0x200 byte block is read from address 0x1000, an interrupt is generated when the buffer size reaches zero, and the transfer restarts from base address 0x1000

Table 16-2. DCPRAM Values for a Cyclic Buffer

BD	DCPRAM Parameters		Value	Description
4	BD_ADDR		0x1000	External memory buffer current address
	BD_SIZE		0x200	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x1	Generate an interrupt when the buffer ends
		CYC	0x1	Reinitialize BD_ADDR to original value when the size reaches zero
		CONT	0x1	Continuous mode: the channel is not closed when the size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x4	When the size reaches zero, the next request calls buffer 4
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		0x200	Buffer base size of cyclic buffer

16.2.4.3 Incremental Buffer

A buffer is incremental when the data transfer starts at the buffer base address and continues until all the data is transferred. An interrupt is generated each time DCPRAM[BD_SIZE] reaches zero. However, BD_ATTR[CONT] is set, and the channel does not close when BD_SIZE reaches zero. The BD_ATTR[CYC] bit is cleared, signifying sequential addressing. BD_ATTR[NBD] points to the buffer itself. **Figure 16-18** shows an example incremental buffer.

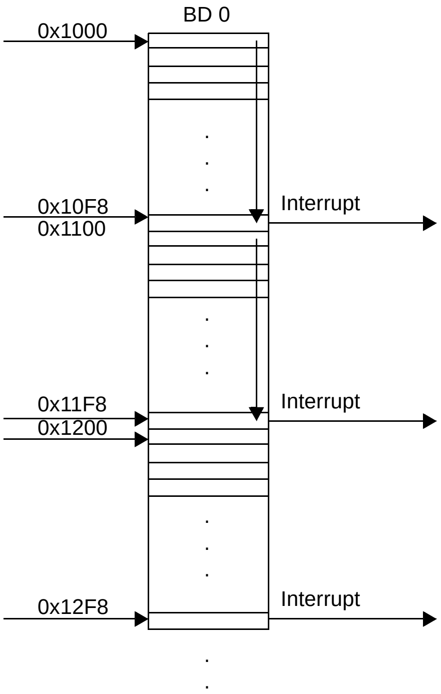


Figure 16-18. Incremental Buffer

Table 16-3 lists the DCPRAM values for an incremental buffer (BD 0). Blocks of 0x100 bytes are read, starting at address 0x1000, and an interrupt is generated every 0x100 bytes. The mode is continuous and addressing is sequential.

Note: With an incremental buffer, memory can be corrupted because of overwriting

Table 16-3. DCPRAM Values for an Incremental Buffer

BD	DCPRAM Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address
	BD_SIZE		0x100	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x1	Generate interrupt when buffer ends
		CYC	0x0	Increment BD_ADDR when size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x0	Next request calls Buffer 0 when size reaches zero
		RD	0x1	Read buffer
	BD_BSIZE		—	Buffer base size of cyclic buffer

16.2.4.4 Chained Buffer

A chained buffer scheme involves two or more buffers. When the size of the first buffer reaches zero, the read jumps to the address of the next buffer, which can be another chained buffer or one of the other buffer types (simple, cyclic or incremental). **Figure 16-19** shows a buffer chained to a simple buffer. In this example, BD 0 does not create an interrupt at the end of the buffer while BD 1 creates an interrupt when closing the buffer.

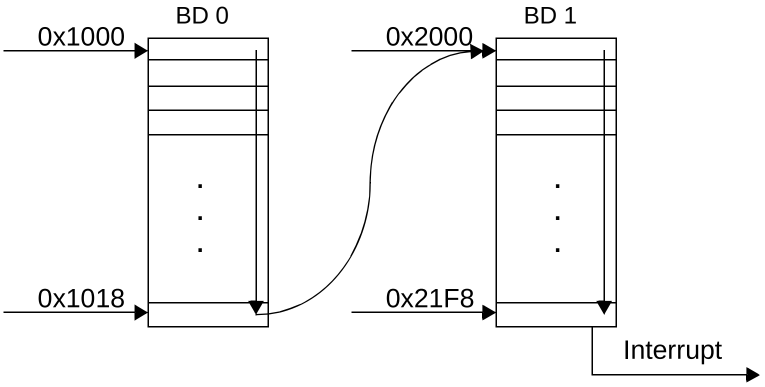


Figure 16-19. Chained Buffer

Table 16-4 lists the DCPRAM values for a chained buffer (BD 0) and a simple buffer (BD 1). A 0x20 byte block is read starting from address 0x1000 (Buffer 0). A jump to address 2000 (Buffer 1) occurs when the buffer size reaches zero, and an interrupt is generated when 0x200-byte blocks are read.

Table 16-4. DCPRAM Values for a Chained Buffer and a Simple Buffer

BD	DCPRAM Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address
	BD_ATTR	INTRPT	0x0	Do not generate an interrupt when buffer ends
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x1	When size reaches zero, next request calls buffer1
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		0x20	Buffer base size of cyclic buffer
1	BD_ADDR		0x2000	External memory buffer current address
	BD_ATTR	INTRPT	0x0	Do not generate interrupt when the buffer ends. The channel generates the interrupt.
		CONT	0	Non-continuous mode. The channel is closed when the size reaches zero
		NO_INC	0x0	Increment address after request is serviced

Table 16-4. DCPRAM Values for a Chained Buffer and a Simple Buffer (Continued)

BD	DCPRAM Parameters		Value	Description
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		0x200	Buffer base size of cyclic buffer

While there is no specific system requirement that all chained buffers be on the same bus, and some devices recommend using the flush option by setting the BD_ATTR[FLS] bit in the source buffer when switching between buffers on different buses, there is a possibility that such operations can cause the DMA FIFO to reset unexpectedly causing data to be shifted. Use the following steps to maintain data consistency when using chained buffers:

1. Program the chained buffers without bus switching in the local SRAM (M1 or M2).
2. Program the DCHCR in the local SRAM.
3. Activate a channel to load the buffer to the parameter RAM and activate another DCHCR.
4. The source BDs must all be on the same bus.
5. The destination BDs must all be on the same bus.

Using these steps prevents the need for switching between buses and using flushing.

16.2.4.5 Complex Buffers—Dual Cyclic Buffers

A dual cyclic buffer scheme uses two buffers, which can be any combination of the previously described buffers. Two areas in memory are used to store data. While one area is processed, the other memory area receives new data (see **Figure 16-20**). The scheme uses two buffers: BD 0 and BD 1. Buffer 0 starts at address 0x1000 and transfers 0x200-byte blocks. Buffer 1 starts at address 0x2000 and also transfers 0x200-byte blocks.

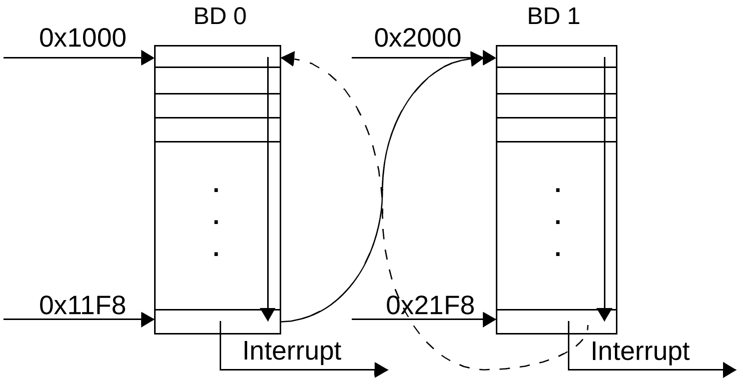


Figure 16-20. Dual Cyclic Buffers

Table 16-5 lists the DCPRAM values for dual cyclic buffers.,

Table 16-5. DCPRAM Values for Dual Cyclic Buffers

BD	DCPRAM Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address
	BD_SIZE		0x200	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x1	Generate interrupt when buffer ends
		CYC	0x1	Reinitialize BD_ADDRESS to original value when size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x1	When size reaches zero, next request calls Buffer 1
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		0x200	Buffer base size of cyclic buffer
1	BD_ADDR		0x2000	External memory buffer current address
	BD_SIZE		0x200	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x1	Generate interrupt when buffer ends
		CYC	0x1	Reinitialize BD_ADDRESS to original value when size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x0	When size reaches zero, next request calls Buffer 0
		TSZ	0x4	Maximum transfer size is one burst
		RD	0x1	Read buffer
	BD_BSIZE		0x200	Buffer base size of cyclic buffer

16.2.5 2D Data Transfers

2D data transfers are performed using chained buffers for the data transfer into one FIFO and one simple buffer for the data transfer from the FIFO. The chained buffers are each programmed to transfer 16 bits at a time. The buffers are programmed in to be cyclic so that the last buffer points to the first buffer. The buffers themselves are incremental with 0x2 as their base size. The simple buffer determines when the transfer ends by generating an interrupt when the buffer size reaches zero. In **Figure 16-21**, BD 1-BD 4 are chained buffers belonging to the read channel while BD 0 is a simple buffer belonging to the write channel, generating an interrupt at the end of the transfer.

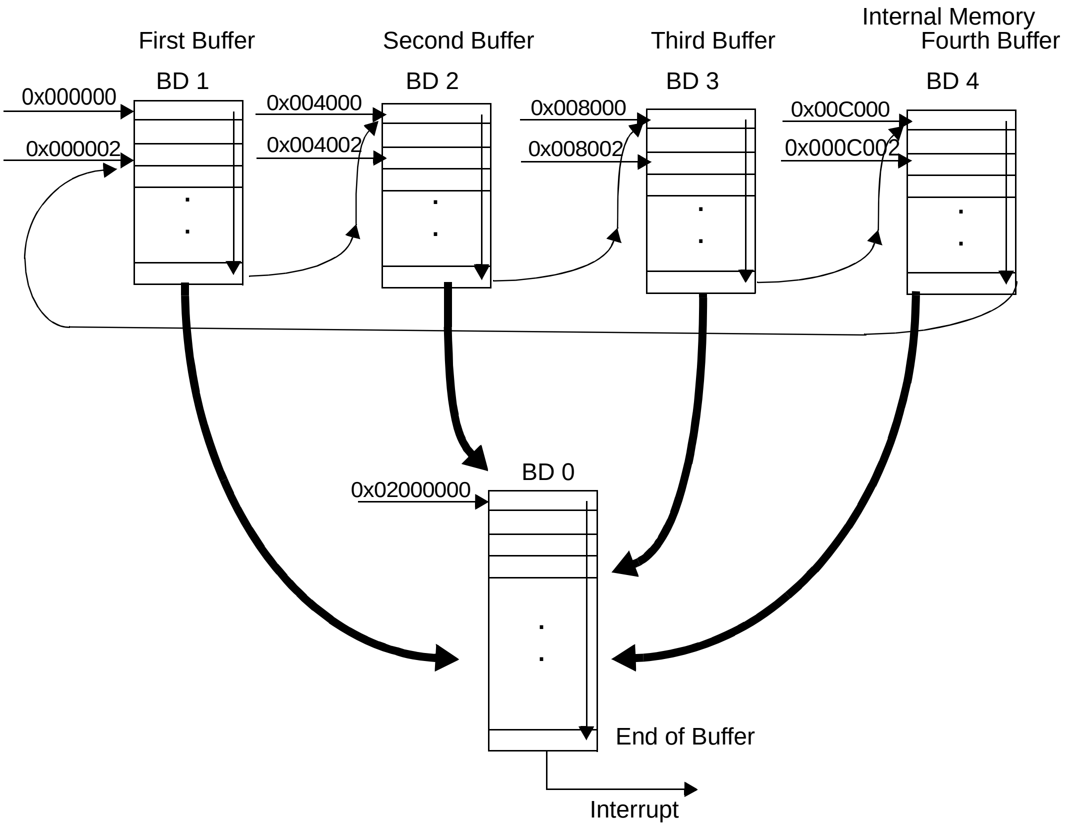


Figure 16-21. 2D Buffers

Table 16-6 lists the DCPRAM values for 2D transfers.

Table 16-6. DCPRAM Values for 2D Transfers

BD	DCPRAM Parameters		Value	Description
0	BD_ADDR		0x02000000	Internal memory current address
	BD_SIZE		0x10000	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate an interrupt when the buffer ends
		CYC	0x0	Sequential address. Increment BD_ADDR when the size reaches zero
		CONT	0x0	Non-continuous mode: the channel is closed when the size reaches zero
		NO_INC	0x1	Do not increment address after request is serviced
		TSZ	0x2	Maximum transfer size is two bytes
		RD	0x0	Write buffer
	BD_BSIZE		—	Buffer base size of cyclic buffer

Table 16-6. DCPRAM Values for 2D Transfers (Continued)

BD	DCPRAM Parameters		Value	Description
1	BD_ADDR		0x00000	External memory buffer current address
	BD_SIZE		0x2	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate interrupt when buffer ends
		CYC	0x0	Sequential address - Increment BD_ADDR when the size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x2	When size reaches zero, next request calls Buffer 2
		TSZ	0x2	Maximum transfer size is two bytes
		RD	0x1	Read buffer
	BD_BSIZE		0x2	Buffer base size of cyclic buffer
2	BD_ADDR		0x04000	External memory buffer current address
	BD_SIZE		0x2	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate interrupt when buffer ends
		CYC	0x0	Sequential address - Increment BD_ADDR when the size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x3	When size reaches zero, the next request calls Buffer 3
		TSZ	0x2	Maximum transfer is two bytes
		RD	0x1	Read buffer
	BD_BSIZE		0x2	Buffer base size of cyclic buffer
3	BD_ADDR		0x08000	External memory buffer current address
	BD_SIZE		0x2	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate interrupt when buffer ends
		CYC	0x0	Sequential address. Increment BD_ADDR when the size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x4	When size reaches zero, the next request calls Buffer 4
		TSZ	0x2	Maximum transfer size is two bytes
		RD	0x1	Read buffer
	BD_BSIZE		0x2	Buffer base size of cyclic buffer
4	BD_ADDR		0x0c000	External memory buffer current address
	BD_SIZE		0x2	Size of transfer left for this buffer
	BD_ATTR	INTRPT	0x0	Do not generate an interrupt when buffer ends
		CYC	0x0	Sequential address. Increment BD_ADDR when the size reaches zero
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		NO_INC	0x0	Increment address after request is serviced
		NBD	0x1	When size reaches zero, next request calls Buffer 1
		TSZ	0x2	Maximum transfer size is two bytes
		RD	0x1	Read buffer
	BD_BSIZE		0x2	Buffer base size of cyclic buffer

16.3 DMA Transfer Programming

Figure 16-22 illustrates the process of normal DMA programming. To support dual access transfers, the DMA uses two consecutive channels. The even channel must be configured to perform the read (from requestor/memory into the DMA FIFO), and the odd channel performs the write (DMA FIFO to the other memory/requestor).

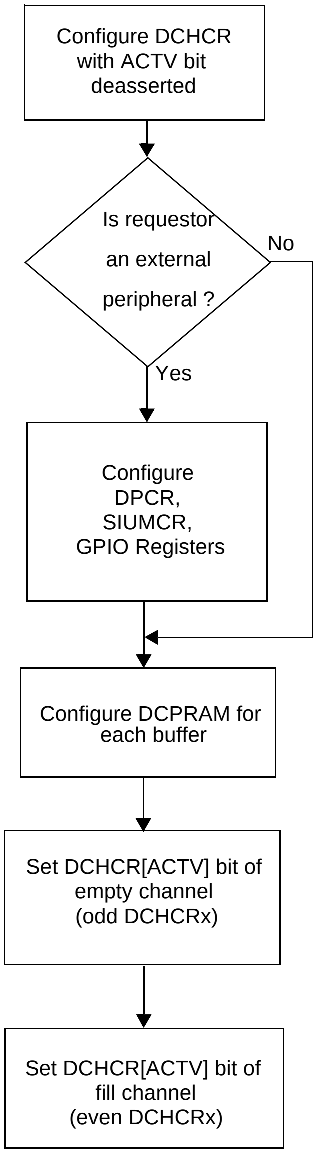

Configure DCHCR with ACTV bit deasserted
Is requestor an external peripheral ?
No
Yes
Configure DPCR, SIUMCR, GPIO Registers
Configure DCPRAM for each buffer
Set DCHCR[ACTV] bit of empty channel (odd DCHCRx)
Set DCHCR[ACTV] bit of fill channel (even DCHCRx)

Figure 16-22. DMA Configuration Flow

16.3.1 DMA Priority Type

The DMA controller supports both fixed-priority and round-robin algorithms. The DPCRx[AM] bit controls the priority scheme mode of the DMA controller:

- When cleared (RESET value), the bit selects Fixed-Priority priority mode (MSC8102 mode). The DCHCRx[PRIO] bits are active
- When set, the bit selects Round-Robin mode. The DCHCRx[PRIO] bit must be cleared (all bits = 0). Failure to clear these bits causes unpredictable behavior.

Writing to DCPR[AM] to change the priority mode is permitted only when all DMA channels are inactive.

16.3.1.1 Fixed-Priority Mode

In Fixed-Priority mode, every channel has a priority set by the user in the DCHCRx[PRIO] register. This priority is not changed by the DMA. If two channels have the same priority, the channel with the lower channel number has the higher priority. **Figure 16-23** illustrates the fixed priority flow.

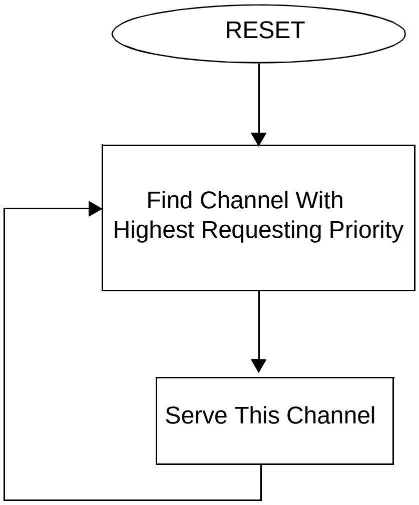


Figure 16-23. Fixed-Priority Flow Diagram

16.3.1.2 Round-Robin Priority Mode

The round-robin algorithm can be described as two analog clocks. One clock is for the channels associated with the system bus, and the other clock is for the channels associated with the local bus. Each clock has sixteen markers, one for each of the 16 channels, and one big hand, which rotates from 0 to 15 clockwise. **Figure 16-24** illustrates the system and local bus round robin clocks. The pointer hand indicates the current active channel (the markers on the clock) used by the local bus and the system bus. When the transfer is complete for a selected channel, the clock hand moves on to the next channel.

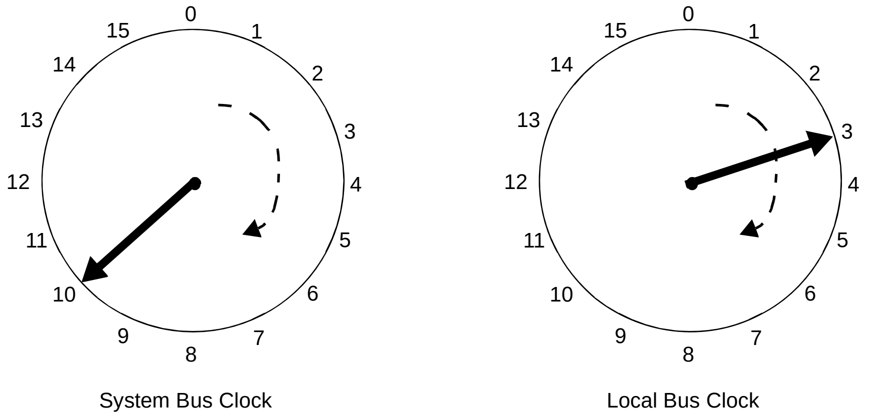


Figure 16-24. Round-Robin Clock Examples

The round-robin operation continues from one channel to the next in sequence for all 16 channels, regardless of the source and destination. **Figure 16-25** shows the round-robin flow for either of the buses. There are two parallel flows, one for each bus.The variable *N* represents the channels numbers.

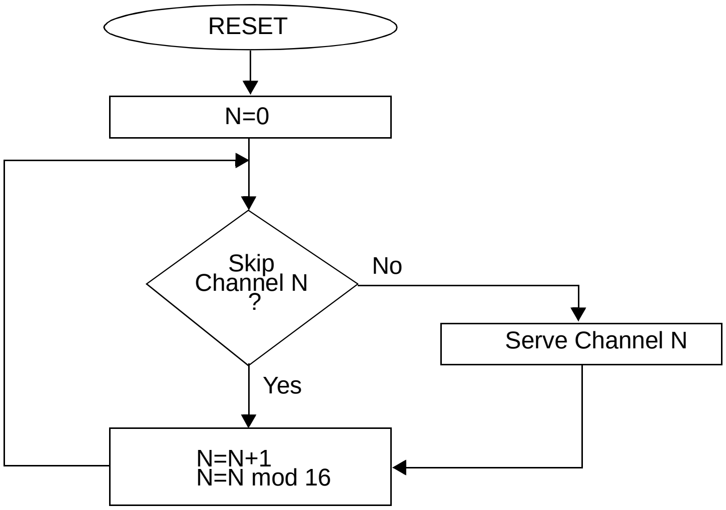


Figure 16-25. Round-Robin Flow Diagram

- The hand skips a channel in the following cases:
  - — The channel is non-active.
  - — The channel is not requesting service when the hand selects the channel.
  - — The channel is not associated with the specific bus, that is, if a channel is associated with the system bus, the local bus round robin clock always skips that channel.
- If a channel requests service, but the hand already passed it in the current round, it is served during the next round.

After reset, the channel value is set to 0. The DMA determines whether to process or skip the channel. If it skips the channel, the DMA controller continues to test and increment the channel value until it reaches a channel that does not match the skip parameters. The channel is then served and the channel number is incremented. After processing or skipping channel 15, the DMA controller rolls the channel number back to 0 and begins the next round.

16.3.1.3 DMA Arbitration Device Level Considerations

Any access issued by the DMA controller must pass two arbitration layers: the DMA arbitration and the bus controller arbitration. The first arbitration selects the DMA channel that generates the bus access. The second arbitration is the selection of the master by the bus arbitrator (DMA controller, TDM interface, Ethernet controller, DSI, and so on). The DMA arbiter selects the channel priority based on the values of DCHCRx[PRIO] and DPCR[AM]. The bus arbiter uses BD_ATTRx[BP] to select the channel priority on the bus.

You must assign correlating priorities to ensure correct operation of the transfers. The hierarcical arbitration may cause a high priority task to delay a low priority task pending in the bus arbiter.

Example 16-5. Multiple Device Arbitration

Scenario: Two DMA tasks (DMA_1 and DMA_2) are activated to use the device local bus. In addition, the TDM interface uses the local bus. The DMA controller uses fixed priority mode with the following setting:

— DMA_1: DCHCR[PRIO] = 0 (high), BD_ATTRx[BP] = 2 (high)

— DMA_2: DCHCR[PRIO] = F (low), BD_ATTRx[BP] = 0 (low)

The local bus arbiter uses the following priorities, from high to low:

— DMA_1, TDM, DMA_2

At some point during the operation, the DMA controller has no available data from DMA_1 and DMA_2 wins the arbitration. The DMA controller generates a low priority bus access based on the DMA_2 settings. At the same time, the TDM interface also attempts to access the local bus. The local bus arbitrater grants the bus to the TDM interface because its priority is higher than DMA_2. The TDM uses the bus. If, during the TDM transfers, DMA_1 has data to transfer, it tries to generate a bus access, but the DMA_2 access is still pending. This scenario causes the high priority DMA_1 activity to wait due to the pending low priority access by DMA_2. and the ongoing TDM access.

Solution: Assign the TDM a lower priority than DMA_2 to prevent the situation in the scenario from occurring.

16.3.2 DMA Data Transfer Examples

Typical DMA programming schemes include the following:

- Simple buffer transfer from a system bus external peripheral to internal memory on the local bus
- Cyclic block transfer from system bus external memory to internal memory on the local bus
- Continuous block transfer from an external peripheral to internal memory on the local bus
- Simple buffer transfer from M2 to M1 on the local bus in Flyby mode

Note: Refer to **Section 16.4**, *DMA Programming Model*, on page 16-34 for DMA controller register descriptions. For details on programming a base address, see **Section 12.8**, *Memory Controller Programming Model*, on page 12-95.

16.3.3 Terminating a DMA Transfer

To stop the DMA transfer temporarily, set the DCHCR[FRZ] bit. When set, this bit masks requests assigned to the specific channel and the requestor is not serviced. All other internal states are unaffected, so clearing this bit resumes DMA transfer without loss of data or requests. Because the DMA controller uses pipelining, up to 96 bytes can remain in the channel FIFO and not be delivered to the destination. This data is transferred when the channel is enabled. A DMA channel is terminated externally either when the peripheral asserts the $\overline{\text{DONE}}$ signal or you clear the DCHCR[ACTV] bit.

Either source channel or destination channel can be terminated. Termination of the source (read) channel proceeds as follows:

1. The DMA controller ignores any further requests from the peripheral.
2. All the data in the FIFO is transferred to the destination channel (flushed).
3. If transfer size is bigger than the data stored in the FIFO, additional data is flushed.
4. DCPRAM[BD_SIZE] and DCPRAM[BD_ADDR] are updated to the correct size and address of the buffer serviced by the DMA until the source channel is terminated.
5. If enabled, an interrupt is generated after the last data is written to the destination channel.
6. The DCHCR[ACTV] bit of the destination is not cleared. You must clear the channel before reusing it.

Termination of the destination (write) channel proceeds as follows:

1. The DMA controller ignores any further requests from the peripheral.
2. Tasks in progress — bus data phase, bus address phase, and pending phase — are flushed.

3. If $\overline{\text{DONE}}$ termination is used, no additional transfers occur since peripheral protocol enforces the non-pipeline mode of DMA.
4. DCPRAM[BD_SIZE] and DCPRAM[BD_ADDR] are updated to the correct size and address of the buffer serviced by the DMA until the destination channel is terminated.
5. If enabled, an interrupt is generated after the last data is written to the destination channel.
6. The DCHCR[ACTV] of the source is not cleared. You must clear the channel before reusing it.

Because the DMA controller uses pipelining, up to 96 bytes can be transferred to the destination channel after the source channel is terminated. Bus error also causes the DMA controller to terminate all channels associated with the bus on which a transfer error is detected. The DCHCR[ACTV] bit is immediately cleared, and the bus identification, the address, and the number (RQNUM) of the channel that caused the error are captured in registers DTEAR, PDMTEA/LDMTEA, and PDMTER/LDMTER, respectively. All other parameters, such as DCPRAM[BD_ADDR] and DCPRAM[BD_SIZE], are undefined. Channels on the other bus close normally, as if you deasserted their corresponding DCHCR[ACTV] bit.

Note: If the FLS bit in the BD_ATTR is set, the DMA controller flushes the FIFO at the end of a transfer and issues an interrupt. See the description of the FLS bit in **Table 16-10** for details.

16.4 DMA Programming Model

Each DMA channel is triggered by a requestor, which can be any one of the following:

- One of four external peripherals, selected by using the corresponding DREQ[1–4]
- A hardwired request associated with one of the M1 flyby counters
- One of the internal requests used in memory-to-memory transactions.

Any request initiates one transaction for the channel. If the requestor is external, four request modes are available:

- Active high level-triggered mode
- Active low level-triggered mode
- Rising edge-triggered mode
- Falling edge-triggered mode

The channels involved in the transfer must be configured for the given task via the DMA registers. This section describes the DMA registers in detail.

- DMA Channel Configuration Registers (DCHCR[0–15]), **page 16-35**
- DMA Pin Configuration Register (DPCR), **page 16-38**

- DMA Channel Parameters RAM (DCPRAM), **page 16-38**
- DMA Status Register (DSTR), **page 16-42**
- DMA Internal Mask Register (DIMR), **page 16-43**
- DMA External Mask Register (DEMR), **page 16-43**
- DMA Transfer Error Address Status Register (DTEAR), **page 16-44**
- DMA Transfer Error Requestor Number Register (System bus) (PDMTER), **page 16-44**
- DMA Transfer Error Requestor Number Register (Local bus) (LDMTER), **page 16-44**
- DMA Transfer Error Address Register (System bus) (PDMTEA), **page 16-45**
- DMA Transfer Error Address Register (Local bus) (LDMTEA), **page 16-45**

16.4.1 Configuration Registers

DCHCR[0–15] DMA Channel Configuration Registers

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	ACTV	PPC	—			EXP			DRS	DPL	BDPTR					
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**
	DRACK	FLY	—	RQNUM					FRZ	INT	—		PRIO			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each of the 16 DCHCR*x* registers configures the connection between a DMA requestor and the corresponding DMA channel. You should program all the channel properties, including the relevant line in DCPRAM, before enabling the channel by asserting the ACTV bit. The DMA logic can modify some fields in this register while the channel is active.

Note: You can change the INT, PRIO, FRZ, PPC, and ACTV bits in the DCHCR while the channel is active. The DMA controller can also modify the BDPTR and ACTV fields. To avoid a conflict with the DMA logic and to avoid overwriting the DMA modifications, use byte access to these fields while the channel is active.

Table 16-7. DCHCRx Bit Descriptions

Name	Reset	Changed By	Description	Settings
ACTV 0	0	User DMA	**Active DMA Channel x** While channel x is disabled, all requests are ignored and any non-serviced request is lost. The ACTV bit is reset by the DMA logic upon completion of the channel task.	0 Channel is disabled. 1 Channel is enabled.
PPC 1	0	User DMA	**System Bus** Selects the system bus associated with this channel.	0 Channel is assigned to the local bus. 1 Channel is assigned to the system bus.
— 2–4	0	—	Reserved. Write to zero for future compatibility.	
EXP 5–7	0	User	**Expiration Timer** Ignored in edge-triggered request mode. The channel ignores level request to “EXP+1” bus cycles after assertion of the DRACK or DACK signal, as defined by the DRACK bit.	
DRS 8	0	User	**DREQ Sensitivity Mode**	0 DREQ is edge-triggered. 1 DREQ is level-triggered.
DPL 9	0	User	**DREQ Polarity** Indicates the polarity of the DREQ signal.	0 DREQ is active high or rising edge-triggered, according to the DRS value. 1 DREQ is active low or falling edge-triggered, according to the DRS value.
BDPTR 10–15	0	User DMA	**Buffer Pointer** Value can be changed by the DMA logic in case of multi-buffer channel.	Pointer to the line in DCPRAM which is assigned to this channel.
DRACK 16	0	User	**DRACK Protocol** Indicates whether the peripheral supports DRACK protocol. See the DMA Request Acknowledge (DRACK) bullet in **Section 16.1**, *DMA Signals: Requestor Interface,* on page 16-3.	0 Channel does not use DRACK. Expiration timer starts counting after DACK assertion. 1 Channel uses DRACK. Expiration timer starts counting after DRACK assertion.
FLY 17	0	User	**Flyby Transaction** Indicates whether a single address transaction can be used. If dual access is selected, a complementary channel should be configured. When the DMA channel handles an internal request, the FLY bit must be cleared.	0 Dual-access transaction. 1 Flyby mode. Single-access transaction.
— 18	0	—	Reserved. Write to zero for future compatibility.	

Table 16-7. DCHCRx Bit Descriptions (Continued)

Name	Reset	Changed By	Description	Settings
RQNUM 19–23	0	User DMA	**Requestor Number** The channel is triggered by the requestor identified by RQNUM. This field is valid only when INT is deasserted—external request.	000xxReserved 00100 Core 0 flyby counter-a request. 00101 Core 1 flyby counter-a request 00110 Core 2 flyby counter-a request 00111 Reserved 01000 External request 1, DREQ1. 01001 External request 2, DREQ2. 01010 External request 3, DREQ3. 01011 External request 4, DREQ4. 01100 Core 0 flyby counter-b request. 01101 Core 1 flyby counter-b request. 01110Core 2 flyby counter-b request. 01111 Reserved. 1xxxx Reserved.
FRZ 24	0	User	**Freezes Channel** All channel settings are valid, and new DREQ requests are considered, but the DMA controller does not issue any transactions to this channel. Data can be left in the FIFO store. However, upon unfreezing the channel, no data or requests are lost.	0 Channel operates normally. 1 Channel is frozen.
INT 25	0	User	**Internal Requestor** Indicates if requestor assigned to this channel is a peripheral.	0 External request. Transaction is initiated by a peripheral 1 Internal request. Transaction between memory and DMA is initiated by the DMA controller.
— 26–27	0	—	Reserved. Write to zero for future compatibility.	
PRIO 28–31	0	User	**Channel Priority** DMA internal request priority for this channel. For channels assigned with the same priority (that is, the same PRIO value), the relative priority is determined by the channel number. The lowest channel has the highest priority. **Note:** When working with Round-Robin priority (DPCR[AM] = 1), these bits must be cleared for all channels.	0000 Highest priority. 1111 Lowest priority.

DPCR DMA Pin Configuration Register

Bit	0	1	2	3	4	5	6	7
	—			AM	SDN0	SDN1	—	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

DPCR selects the functionality of DONE/DRACK. The DONE/DRACK protocol is supported only by the following requests:

- External request 1—RQNUM = 01000
- External request 2—RQNUM = 01001

Table 16-8. DPCR Bit Descriptions

Name	Reset	Description	Settings
— 0–2	0	Reserved. Write to zero for future compatibility.	
AM 3	0	**Arbitration Mode** Selects the arbitration mode to use. **Note:** If Round-Robin Priority mode is selected (AM = 1), then DCHCRx[PRIO] must be cleared for all channels.	0 Fixed priority. 1 Round-Robin priority.
SDN0 4	0	**Select DONE0** Controls the functionality of DONE1/DRACK1. The functionality is determined by the SDN0 value and the GPIO registers configuration. For details, see **Section 23.5**.	0 Functionality is DONE1. 1 Functionality is DRACK1.
SDN1 5	0	**Select DONE1** Controls the functionality of DONE2/DRACK2. The functionality is determined by the SDN1 value and the GPIO register configuration. For details, see **Section 23.5**.	0 Functionality is DONE2. 1 Functionality is DRACK2.
— 6–7	0	Reserved. Write to zero for future compatibility.	

DCPRAM DMA Channel Parameters RAM

DCPRAM holds the parameters of all the channels. Each buffer descriptor uses 128 bits (16 bytes) to contain its status and parameters. DCPRAM is memory-mapped and can be accessed with a read/write transaction from the system bus. If the DMA access and an external access occur at the same time, the DMA hardware waits one clock until the external transaction terminates. **Figure 16-26** depicts the structure of the DCPRAM.

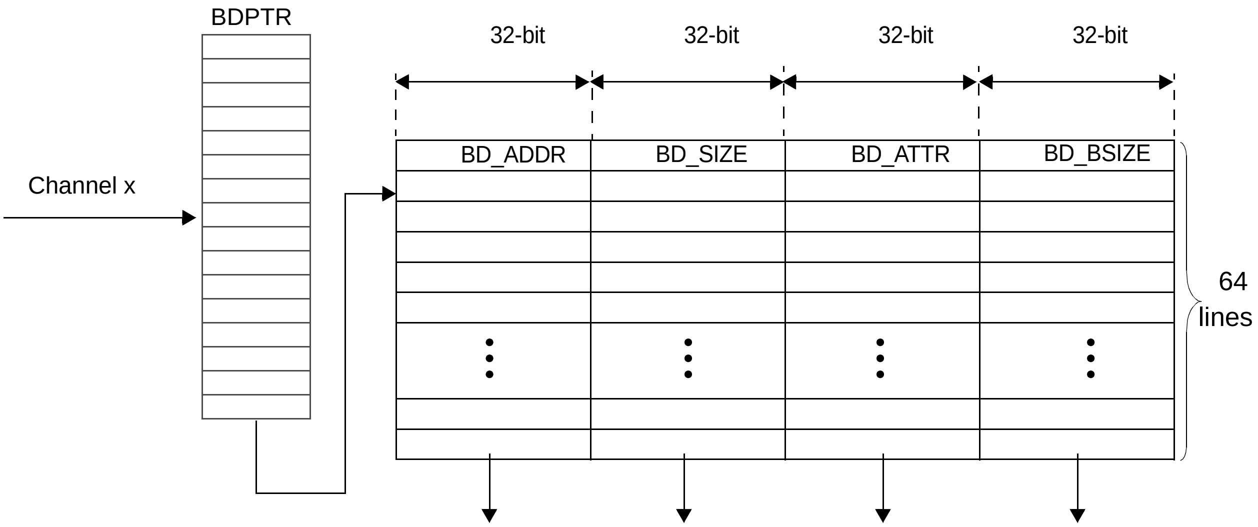


Figure 16-26. DCPRAM Structure

The address of the parameter space for a given channel x is indicated by the contents of the BDPTR field in the DCHCRx register. The DCPRAM space is at address 0x10800–0x10BFF. The DCPRAM fields of a given buffer are described in **Table 16-9**.

Table 16-9. DCPRAM Bit Descriptions

Bits	Name	Description
0–31	BD_ADDR	**Buffer current address** Holds the buffer address pointer. If the buffer is cyclic, the original address value is restored when the BD_SIZE value reaches zero by decrementing BD_BSIZE from BD_ADDR. See **Section 16.2.4.2**, *Cyclic Buffer,* on page 16-22.
32–63	BD_SIZE	**Size of transfer left for the current buffer** Contains the remaining size of the buffer. This value decrements by the transfer block size each time the DMA controller issues a transaction, until it reaches zero. When BD_SIZE reaches zero, the original value is restored to the value of BD_BSIZE. Program BD_SIZE with a value larger than 0.
64–95	BD_ATTR	**Buffer attributes and temporary data** A 32-bit parameter that describes the attributes of the channel handling this buffer. See **Table 16-10**.
96–127	BD_BSIZE	**Buffer base size** Holds the base size of the buffer. if used, program BD_SIZE with a value greater than 0.

BD_ATTR Buffer Attributes Parameter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	INTRPT	CYC	CONT	—	NO_INC	BP		—		NBUS	NBD					
Type	R/W															
Reset	Undefined															

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	—						TSZ			—	FLS	RD	—	TC	—	GBL
Type	R/W															
Reset	Undefined															

Table 16-10. BD_ATTR Bit Descriptions

Name	Reset	Description	Settings
INTRPT 0	Undefined	**Interrupt** Indicates whether to issue an interrupt when size reaches zero.	0 Do not issue interrupt. 1 Issue interrupt when size reaches zero.
CYC 1	Undefined	**Cyclic Address** Indicates the behavior of BD_ADDR in continuous buffer mode when BD_SIZE reaches zero. For details, see **Section 16.2.4.2**, *Cyclic Buffer,* on page 16-22.	0 Sequential address. BD_ADDR is incremented. 1 Cyclic address. BD_ADDR is restored to its original value by decrementing BD_BSIZE from BD_ADDR.
CONT 2	Undefined	**Continuous Buffer Mode** Indicates whether the buffer is to be closed when BD_SIZE reaches zero.	0 Buffer closes when BD_SIZE reaches zero. 1 Buffer continues operating when BD_SIZE reaches zero.
— 3	Undefined	Reserved. Write to zero for future compatibility.	
NO_INC 4	Undefined	**Increments Address** Indicates the behavior of the buffer address.	0 Increment address after request is serviced. 1 Do not increment address after request is serviced.
BP 5–6	Undefined	**Bus Priority** Indicates the bus mastership request to be initiated with this channel. For details, see the description of the System Bus Arbiter Configuration Register (PPC_ACR) on **page 4-13** and the PPC_ALRL System Bus Arbitration-Level Register (PPC_ALRL) on **page 4-15**.	00 Arbitrate for bus mastership with DMA low priority request (bus requestor number 12). 01 Arbitrate for bus mastership with DMA middle priority request (bus requestor number 11). 10 Arbitrate for bus mastership with DMA high priority request (bus requestor 10). 11 Reserved.
— 7–8	Undefined	Reserved. Write to zero for future compatibility.	

Table 16-10. BD_ATTR Bit Descriptions (Continued)

Name	Reset	Description	Settings
NBUS 9	Undefined	**Next Bus** When size reaches zero and CONT is set, the PPC field in the DCHCR is updated according to the NBUS field.	0 Local bus. 1 System bus.
NBD 10–15	Undefined	**Next Buffer** When size reaches zero and CONT is set, the next request calls the buffer to which NBD points.	
— 16–21	Undefined	Reserved. Write to zero for future compatibility.	
TSZ 22–24	Undefined	**Transfer Size** Indicates the maximum transaction size that the DMA controller issues when a request is detected.	001Maximum transfer size is 8 bits. 010Maximum transfer size is 16 bits. 011 Maximum transfer size is 32 bits. 000Maximum transfer size is 64 bits. 100Maximum transfer size is one burst. 101Reserved. 11x Reserved.
— 25	Undefined	Reserved. Write to zero for future compatibility.	
FLS 26	Undefined	**Flush FIFO** Indicates the behavior of the FIFO when BD_SIZE reaches zero. Typically, in continuous buffers, the FIFO is not flushed. **Note:** FLS is useful when buses change from one BD to the next. FLS is also useful for continuous buffers in which the data must be flushed after the end of each buffer. Whenever a flush occurs, the DMA controller issues a flush interrupt. The interrupt is necessary because it is the only indication that a flush occurs. However, this feature may not be desirable for some applications. In such cases, options include: • Do not use flush (clear the FLS bit). • Use polling to determine the FIFO status. • Test the DCHCRx[ACTV] bit when handling a flush interrupt.	0 Do not flush the FIFO. 1 Flush the FIFO.
RD 27	Undefined	**Read Channel** Indicates the type of transaction to be initiated by the DMA channel.	0 Write transaction. 1 Read transaction.
— 28	Undefined	Reserved. Write to zero for future compatibility.	
TC 29	Undefined	**Transfer Code** Indicates the TC code to be associated with the transaction generated by the DMA controller. Refer to **Table 13-11*Transfer Code Encoding,*** on page 13>-22.	0 TC[0–2] value is 110. 1 TC[0–2] value is 111.

Table 16-10. BD_ATTR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 30	Undefined	Reserved. Write to zero for future compatibility.	
GBL 31	Undefined	**Global Transaction** Indicates whether the bus transaction is global. Global transactions are used mainly for memory coherency. Refer to **Section 13.2.2.3**, *Memory Coherency*.	0 Non-global transaction. 1 Global transaction.

16.4.2 DMA Status and Interrupt Registers

The DMA controller reports status and events to the host and generates a maskable interrupt for each channel. The maskable interrupts are routed to the local interrupt controllers (LIC) or global interrupt controller (GIC), according to the setting of the M bit of the relevant channel in DIMR or DEMR. The LICs can receive any of the 16 interrupt lines (one for each channel) while the GIC receives one interrupt line which is an OR of interrupts from the channels. The interrupt request is level triggered, active high. Refer to **Chapter 17**, *Interrupt Processing* for details. Interrupt sources are as follows:

- End of buffer
- DONE indication
- Zero BD_SIZE in a continuous buffer

A non-maskable interrupt is generated when the transfer error address (TEA) is indicated on one of the buses.

DSTR DMA Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15
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**
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In DSTR, each of the 16 MSBs corresponds to an interrupt request from the corresponding channel. If set, a bit associated with a channel indicates that interrupt service is required. A bit is cleared by writing a one to it. Writing zero does not affect a bit value. It is possible to clear several bits at a time. DSTR is cleared at reset.

DIMR DMA Internal Mask Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
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
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In DIMR, each bit corresponds to an interrupt request bit in the DSTR. When a bit is set, it enables the generation of an interrupt request to the LICs (Local Interrupt Controllers). All bits are cleared at reset and it is your responsibility to enable a channel interrupt request.

DEMR DMA External Mask Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
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
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In DEMR, each bit corresponds to an interrupt request bit in the DSTR. When a bit is set, it enables the generation of an interrupt request to the GIC. This interrupt is processed by the external host. DEMR is cleared at reset and it is your responsibility to enable a channel interrupt request.

Note: Interrupts should be enabled either by DIMR or DEMR to avoid undefined system behavior.

16.4.3 Bus Error Registers

If a system bus or a local bus error occurs on a DMA access, a non-maskable interrupt is routed to the SC140 core interrupt controller, and the DMA TEA Status Register (DTEAR) is updated. The interrupt service routine reads the appropriate DMA Transfer Error Address Register, PDMTEA for the system bus or LDMTEA for the local bus to determine the address on the bus where the error occurred. The channel that caused the bus error can be identified by reading the request number from the DMA Transfer Error Requestor Number Register, PDMTER for the system bus or LDMTER for the local bus. When a bus error occurs on a DMA transaction, all DMA activity related to the appropriate bus stops immediately. The DCHCR[ACTV] bit is deasserted by the hardware for all channels configured to service the other bus, guaranteeing memory coherency. The registers associated with bus errors are described in the sections that follow.

DTEAR DMA Transfer Error Address Status Register

Bit	0	1	2	3	4	5	6	7
	DBER_P	DBER_L	—					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

DTEAR reports bus error events recognized by the DMA controller on one of the buses. Upon recognition of a local bus or system bus error, the DMA controller sets its corresponding DBER bit. DTEAR is a memory-mapped register that can be read at any time. Bits are cleared by writing ones to them. Writing zeros has no effect.

Table 16-11. DTEAR Bit Descriptions

Name	Reset	Description	Settings
DBER_P 0	0	**DMA Channel System Bus Error** Indicates that the DMA channel on the system bus has terminated with an error during a read or write transaction.	The DMA transfer error address is read from PDMTEA. The channel number is read from PDMTER.
DBER_L 1	0	**DMA Channel Local Bus Error** Indicates that the DMA channel on the local bus has terminated with an error during a read or write transaction.	The DMA transfer error address can be read from LDMTEA, and the channel number from LDMTER.
— 2–7	0	Reserved. Write to zero for future compatibility.	

*x*DMTER DMA Transfer Error Requestor Number Registers

Bit	0	1	2	3	4	5	6	7
	RQNUM					—		
Type	R							
Reset	Undefined							

PDMTER and LDMTER contain the identification number of the current requestor that addressed the system bus or the local bus, respectively. The RQNUM of each transaction is held in these registers until the transaction completes. Both registers are undefined at reset.

Table 16-12. PDMTER and LDMTER Bit Descriptions

Name	Reset	Description	Settings
RQNUM 0–4	[1]	**Requestor Number** See the DMA Channel Configuration Register (DCHCRx) in **Section 16.4**, *DMA Programming Model,* on page 16-34.	Code number of the requestor accessing the bus when the bus error occurred.
— 5–7	[1]	Reserved. Write to zero for future compatibility.	
Note: Reserved. Write to zero for future compatibility.			

xDMTEA DMA Transfer Error Address Registers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ADDRESS															
Type	R															
Reset	Undefined															

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

PDMTEA holds the system address accessed during a DMA transfer error on the system bus. The LDMTEA holds the system address accessed during a DMA transfer error on the local bus. Both registers are undefined at reset.

Interrupt Processing 17

The MSC8113 interrupt system is optimized for a multi-processing environment and performs the following functions:

- Maximizes the localization of interrupt handling by each SC140 core using two interrupt controllers: PIC and LIC.
- Enables global distribution for important interrupt sources to all the SC140 cores.
- Provides a core-to-core signaling mechanism by virtual interrupt generation.
- Provides programmable routing of interrupt sources to interrupt lines.

Figure 17-1 illustrates the interrupt scheme. Each of the three extended cores contains a programmable interrupt controller (PIC) and a local interrupt controller (LIC). The SC140 core interfaces directly to the PIC, which handles interrupts from internal interrupt sources as well as some external interrupts. The PIC also receives nine interrupts from the LIC, which handles interrupts from the MSC8113 peripherals. A Global Interrupt Controller (GIC) handles interrupts from the SIU, internal signals, and external signals, and it also drives the $\overline{\text{INT\_OUT}}$ signal. The GIC generates virtual interrupts from any SC140 core or the external host to any SC140 core or to the $\overline{\text{INT\_OUT}}$ line. This configuration provides flexibility in the interrupt handling, by enabling any combination of SC140 cores and an external host to handle interrupts. The virtual system can generate interrupt pulses that can be captured locally in the LIC of the target SC140 core to maximize the localization of the interrupt handling.

The PIC supports 24 maskable interrupt sources as well as eight non-maskable ($\overline{\mathrm{NMI}}$) interrupt sources. It arbitrates between the interrupts according to their priority and sends an interrupt to the SC140 core along with the interrupt priority and the vector number of that interrupt in the interrupt table. The SC140 core uses this information to jump to the correct location in the interrupt vector table and also to support interrupt nesting using interrupt priority level indication. Both edge and level interrupt sources are supported. See **Section 17.1.3**, *Programmable Interrupt Controller (PIC)*, on page 17-17.

The LIC supports 64 maskable interrupt sources and consolidates them to eight lines directed to the PIC. The LIC enables four levels of interrupt priority for each group of 32 interrupt sources by programmable routing to one of four PIC interrupt inputs. It supports either edge or level interrupt sources, and generates level interrupt sources to the PIC. In edge mode, the LIC can generate a second-edge (or error) interrupt line to indicate second edge detection before the first edge is serviced. See **Section 17.1.2**, *Local Interrupt Controller (LIC)*, on page 17-9.

The GIC holds the virtual interrupt system that generates interrupts and $\overline{\mathrm{NMI}}$s to the SC140 cores or to the external world by writing to special memory addresses. The written value selects the interrupt destination. The GIC also concentrates internal and external interrupt sources, together with the virtual interrupts, and then generates an interrupt request either to the SC140 cores or to an external signal line. It receives interrupts from internal sources such as the periodic interrupt timer (PIT) or the Time Counter (TMCNT), and from external sources such as interrupt request lines ($\overline{\mathrm{IRQ}}$). For internal interrupt service, it samples the requests and routes them globally to the LICs and the PICs of all SC140s. Each SC140 core can separately enable these sources for its own service. For external interrupt service, the GIC concentrates all its enabled interrupt sources, in addition to DMA interrupt, to one $\overline{\mathrm{INT\_OUT}}$ request line. $\overline{\mathrm{INT\_OUT}}$ and $\overline{\mathrm{NMI\_OUT}}$ are open-drain outputs of the MSC8113 that enable several output signals to connect to the same input signal in the target device. $\overline{\mathrm{INT\_OUT}}$ can also be configured to have a full drive for fast deassertion time on a point-to-point connection, by setting the SIUMCR[INTODC] control bit.

The MSC8113 interrupt configuration allows each SC140 core to handle DSP-related interrupts while another external device, such as the MSC8101, the PowerQUICC II, or another MSC8113, handles other interrupts. To prevent conflict in the interrupt service, do not enable the same interrupt source to both the SC140 cores and an external host.

$\overline{\mathrm{NMI}}$s handled by the GIC can be routed to the SC140 cores through the PIC or to an external host. The $\overline{\mathrm{NMI}}$ handler is determined according to the NMIOUT bit in the Hard Reset Configuration Word (HRCW), described in **Section 5.6.1**. Routing $\overline{\mathrm{NMI}}$ signals to an external host makes it possible to build a system in which a single host handles all the $\overline{\mathrm{NMI}}$ signals. $\overline{\mathrm{NMI\_OUT}}$ is an open-drain output of the MSC8113 that enables a number of $\overline{\mathrm{NMI\_OUT}}$ signals to connect to the same $\overline{\mathrm{NMI}}$ input signal in the master device. For details on the GIC, see **Section 17.1.1**, *Global Interrupt Controller*, on page 17-4.

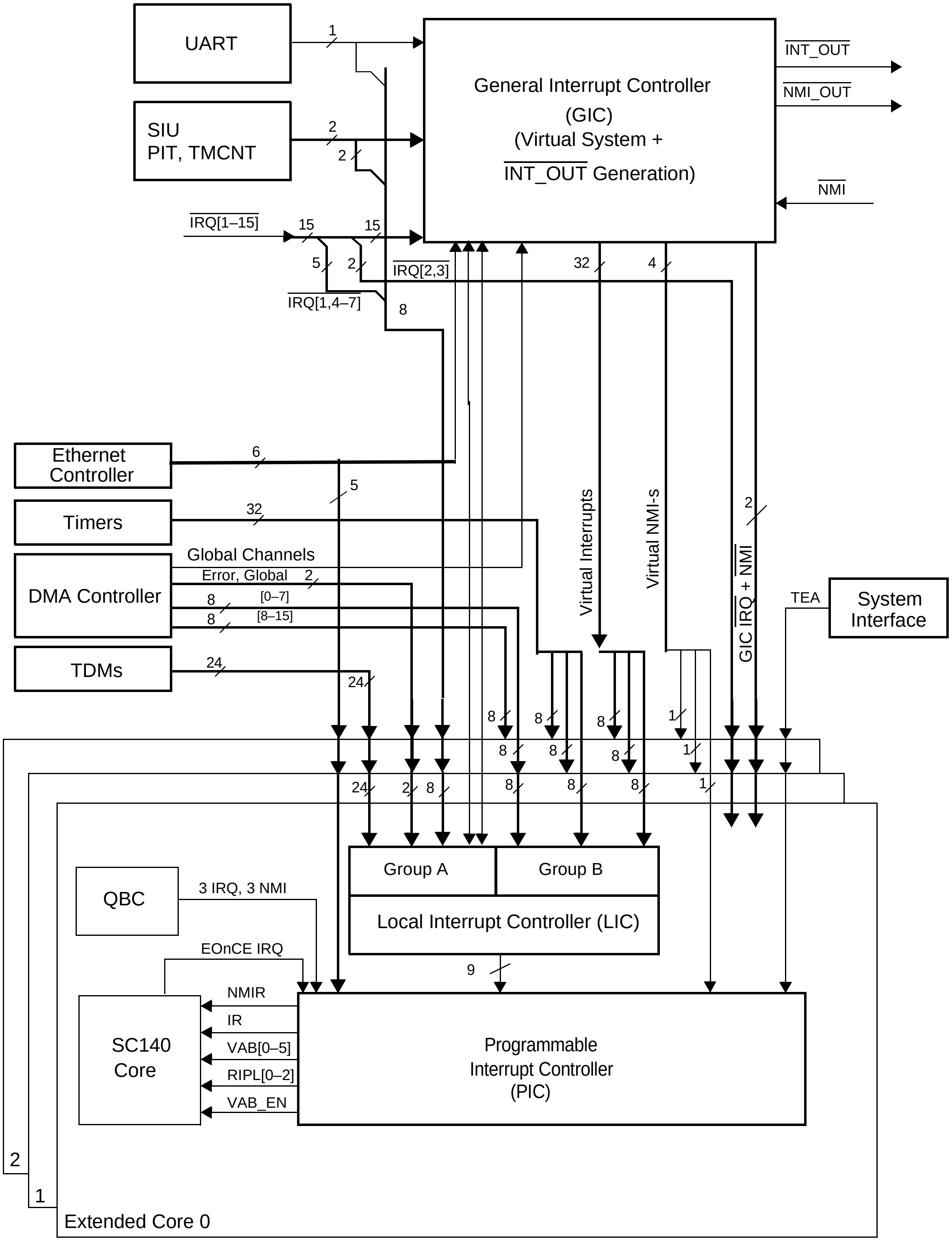


Figure 17-1. MSC8113 Interrupt Block Diagram

17.1 Architecture

This section focuses on the three interrupt controllers in the MSC8113 interrupt structure:

- Global interrupt controller (GIC)
- Local interrupt controller (LIC)
- Programmable interrupt controller (PIC)

17.1.1 Global Interrupt Controller

The GIC performs the following functions:

- Generates 24 virtual interrupts by write access to a special address (virtual address) with predefined data. The virtual interrupts are divided into three groups of eight interrupts, each group routed to the LIC of one SC140 core.
- Generates four virtual NMI pulses to the SC140 cores by a write access to a special NMI virtual address. One SC140 core can assert the $\overline{\text{NMI}}$ of another SC140 core.
- Collects interrupt sources from the UART, SIU interrupt sources, DMA system, 15 external sources ($\overline{\text{IRQ[1–15]}}$) and four of the virtual interrupt sources (line 0 of each group) and selectively enabling them for assertion of $\overline{\text{INT\_OUT}}$.

Collects external interrupt sources $\overline{\text{IRQ[8–15]}}$, configures them to edge/level, and selectively enables them for global distribution to all the SC140 PICs.

Figure 17-2 shows a functional block diagram of the GIC.

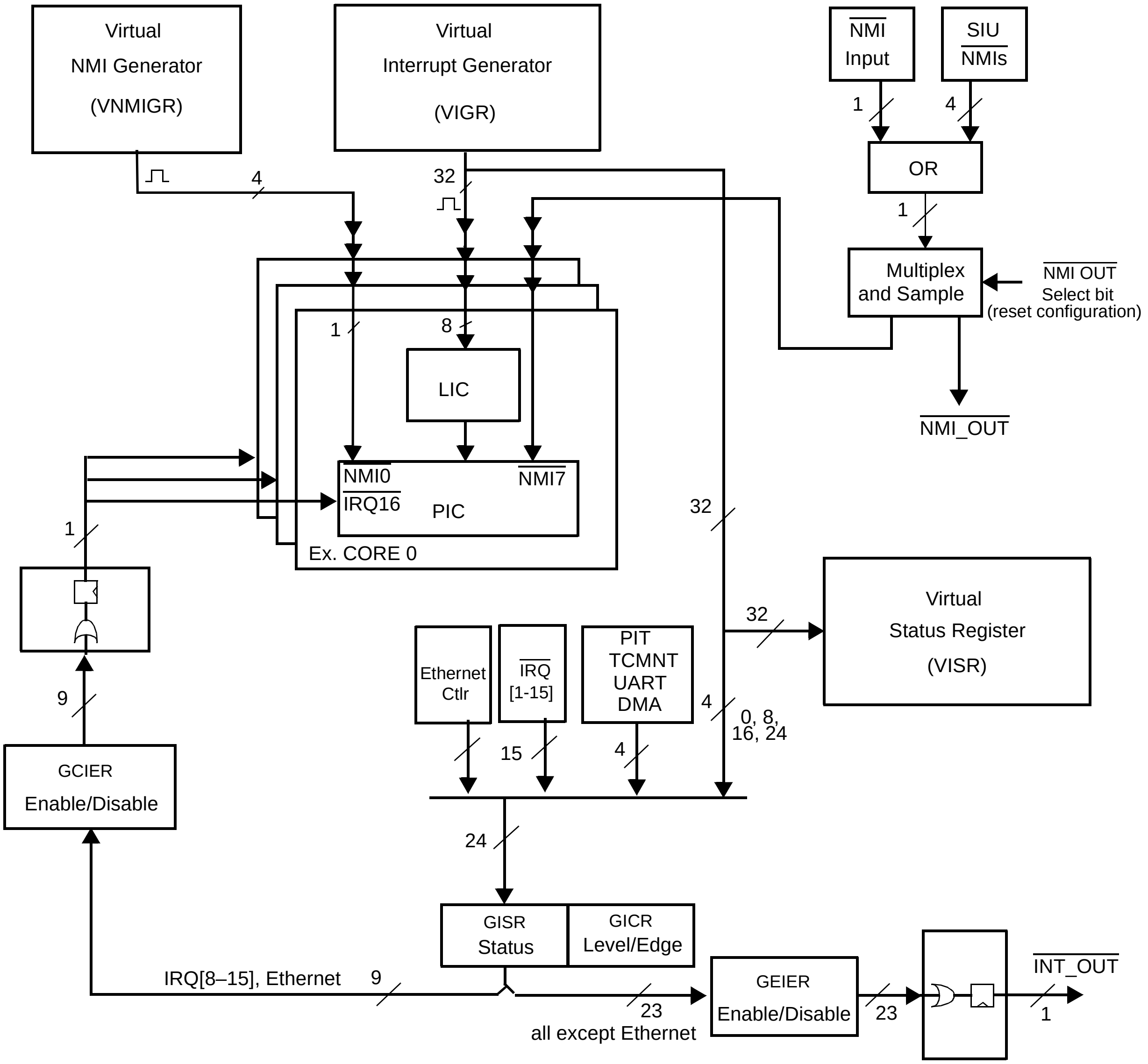


Figure 17-2. GIC Block Diagram

17.1.1.1 $\overline{\text{INT\_OUT}}$ Generation

The GIC provides an output interrupt line, which is a sum of the following interrupt sources. It holds separate enable and status bits and supports either edge or level mode for each of the sources:

- Four virtual Interrupt lines (0, 8, 16 and 24), see **Section 17.1.1.3**, *Virtual Interrupt Generation*.
- Two SIU interrupt sources: PIT and TMCNT (see **Section 4.1.3**, *Time Counter (TMCNT)* and **Section 4.1.4**, *Periodic Interrupt Timer (PIT)*).
- 15 external sources from $\overline{\text{IRQ}}$ lines
- One DMA interrupt source, which is a sum of 16 internal DMA sources. Each source can be internally enabled in the DMA controller (see **Chapter 16**, *Direct Memory Access (DMA) Controller*).
- One UART interrupt source, which is a sum of five internal sources (see **Chapter 21**, *UART*).

Table 17-1 lists the $\overline{\text{INT\_OUT}}$ sources. For details on GIC interrupt out source programming, see GICR on **page 17-26** and GEIER on **page 17-27**.

Table 17-1. GIC $\overline{\text{INT\_OUT}}$ Sources

No.	Source	Description
0	—	Reserved
1	—	Reserved
2	—	Reserved
3	—	Reserved
4	VS24	Virtual System interrupt 24
5	VS16	Virtual System interrupt 16
6	VS8	Virtual System interrupt 8
7	VS0	Virtual System interrupt 0
8	—	Reserved
9	—	Reserved
10	—	Reserved
11	—	Reserved
12	UART	UART Interrupt
13	TMCNT	Time Counter
14	PIT	Periodic Interrupt Timer
15	DMA	DMA global interrupt
16	$\overline{\text{IRQ15}}$	$\overline{\text{IRQ15}}$ Signal
17	$\overline{\text{IRQ14}}$	$\overline{\text{IRQ14}}$ Signal
18	$\overline{\text{IRQ13}}$	$\overline{\text{IRQ13}}$ Signal
19	$\overline{\text{IRQ12}}$	$\overline{\text{IRQ12}}$ Signal
20	$\overline{\text{IRQ11}}$	$\overline{\text{IRQ11}}$ Signal

Table 17-1. GIC INT_OUT Sources (Continued)

No.	Source	Description
21	IRQ10	IRQ10 Signal
22	IRQ9	IRQ9 Signal
23	IRQ8	IRQ8 Signal
24	IRQ7	IRQ7 Signal
25	IRQ6	IRQ6 Signal
26	IRQ5	IRQ5 Signal
27	IRQ4	IRQ4 Signal
28	IRQ3	IRQ3 Signal
29	IRQ2	IRQ2 Signal
30	IRQ1	IRQ1 Signal
31	—	Reserved

17.1.1.2 NMI or NMI_OUT Generation

The GIC receives the external and internal NMI sources and routes them either to the PICs of the SC140s or to the NMI_OUT lines. The NMI destination is selected at the reset configuration sequence by the NMIOUT bit in the HRCW. See **Section 5.6.1**, *Hard Reset Configuration Word*, on page 5-13. The following NMI sources generate either NMI or NMI_OUT (each SC140 core also has some internal NMI sources to indicate internal exceptions). **Table 17-2** references the section that describes each source configuration.

Table 17-2. NMI Generation Configuration

NMI Source	Description Reference
External NMI	Not configurable (always generates an NMI)
SIU Software Watchdog timer expiration	**Section 4.1.5**, *SIU and General Software Watchdog Timers*
SIU system bus monitor expiration on address only accesses	**Section 4.1.1**, *Bus Monitors*
SIU system bus data parity errors	Not configurable (always generates an NMI)
SIU memory bank atomic access lock period expiration on the system bus or local bus	**Section 12.1.7**, *Atomic Bus Operation*

17.1.1.3 Virtual Interrupt Generation

The GIC Virtual Interrupt System generates 24 edge interrupts, with eight interrupts per SC140 core. One interrupt from each group of eight interrupts also goes to the INT_OUT signal. An interrupt is generated by a write access of each SC140 core or by an external host CPU. The eight virtual interrupts go to the PIC through the LIC Group B.

The Virtual Interrupt System always operates as an edge source. The boot program initializes the appropriate LIC inputs to Edge mode. The Virtual Interrupt System also has a status register to indicate whether a virtual interrupt has been generated at least once, while not preventing the generation of another interrupt. The SC140 core that services the interrupt may clear this status bit by writing a value of one to it, or it may ignore this bit and work locally on its LIC. The LIC supports Dual-Edge mode, which may detect a second virtual interrupt while the first one is not yet serviced.

17.1.1.4 Virtual $\overline{\text{NMI}}$ Generation

The GIC Virtual $\overline{\text{NMI}}$ System generates three $\overline{\text{NMI}}$ signals, with one $\overline{\text{NMI}}$ to each SC140 core. $\overline{\text{NMI}}$ is generated by a write access of each SC140 core or by external host CPU. The Virtual $\overline{\text{NMI}}$ goes to the PIC ($\overline{\text{NMI0}}$). The Virtual $\overline{\text{NMI}}$ System does not have a status register; the status register is in the PIC (see the discussion of the IPRB register on **page 17-44**).

17.1.1.5 GIC Stop Mode

To put the GIC into Low-Power Stop mode, assert the SCR[GIC_STC] bit in the IPBus master. The GIC has two conditions for entering Low-Power Stop mode:

- IPBus Master block asserts the GIC stop request line.
- Both GCIER and GEIER disable all interrupts (hold zero).

When these conditions are both met, the GIC responds with a stop acknowledgment and shuts down most of its internal clocks, as described in the following paragraphs.

The GIC has the following functionality in Low-Power Stop mode:

- *Registers*. GCIER, GEIER and GISR are write protected. All the rest of the registers are accessible for both read and write. The GISR read value is meaningless.
- *Regular Interrupts*. No new regular interrupts are captured. Output interrupt lines to the SC140 cores and $\overline{\text{INT\_OUT}}$ are deasserted.
- *Virtual interrupts*. You can generate virtual interrupts to all the cores as in normal mode. VISR can be used to monitor and/or clear the pending status bits. Virtual interrupt sources 24, 16, 8, and 0 are not captured in GISR and do not assert $\overline{\text{INT\_OUT}}$.
- *Regular NMIs*. Internal and external $\overline{\text{NMI}}$ sources are handled the same way as in normal mode.
- *Virtual $\overline{\text{NMI}}$s*. You can generate virtual $\overline{\text{NMI}}$s to all the SC140 cores as in normal mode.

The following sequence is required for a smooth exit from GIC Low-Power Stop mode:

- Deassert GIC stop request by clearing SCR[GIC_STC] in the IPBus master.
- Clear all pending interrupts in the GISR by writing 0xFFFFFFFF to the register.
- Re-enable interrupt sources in GEIER or GCIER as required.

17.1.2 Local Interrupt Controller (LIC)

The LIC module complements the PIC. Its main function is interrupt concentration and localization in the SC140 core private peripheral address space to minimize the overhead of accessing the interrupt status registers at the origin and thus to maximize the performance of interrupt service routines. The LIC is optimally used in conjunction with peripherals that generate pulse interrupt requests (edge mode), but it also supports level operation mode, which is widely used in common peripherals. The LIC resides on the QBus together with the other SC140 core peripherals. It receives up to 64 interrupt sources and maps them to different PIC inputs. Interrupt priority between LIC sources is achieved by assigning a different priority level to each PIC interrupt originating in the LIC.

The LIC has the following functions:

- 64 interrupt input lines divided into two groups of 32 interrupts. In the MSC8113 device, these groups are separated into Group A interrupts and Group B interrupts.
- All interrupt sources are synchronized, and their polarity is hardwired to the actual source polarity.
- Each interrupt source has a primary interrupt status bit (for both level and edge modes) and a second edge error status (only in Dual Edge mode). All status bits can be polled without actually generating an interrupt request.
- All interrupt sources can be programmed to Level, Edge, or Dual Edge mode:
  - In Level mode, the primary status register continuously reflects the synchronized (sampled) status of the proper interrupt input line, allowing interrupt resolution at the source.
  - In Edge mode, the primary status register captures the active edge and ignores the interrupt source until another active edge appears after the SC140 core has deasserted the primary status bit.
  - In Dual Edge mode, in addition to the primary edge detection, if a second active edge appears before the primary status bit is cleared, a second edge error status bit is set. In all modes other than Dual Edge, this status bit is automatically cleared.
- Each interrupt group has four interrupt output lines for primary status bits, providing a total of eight lines, all connected to different PIC inputs. In each group, two bits, per each enabled interrupt source, map it to one of the four output lines. In addition, a sum of all enabled second edge error interrupt status bits generates a global second edge interrupt to the PIC.

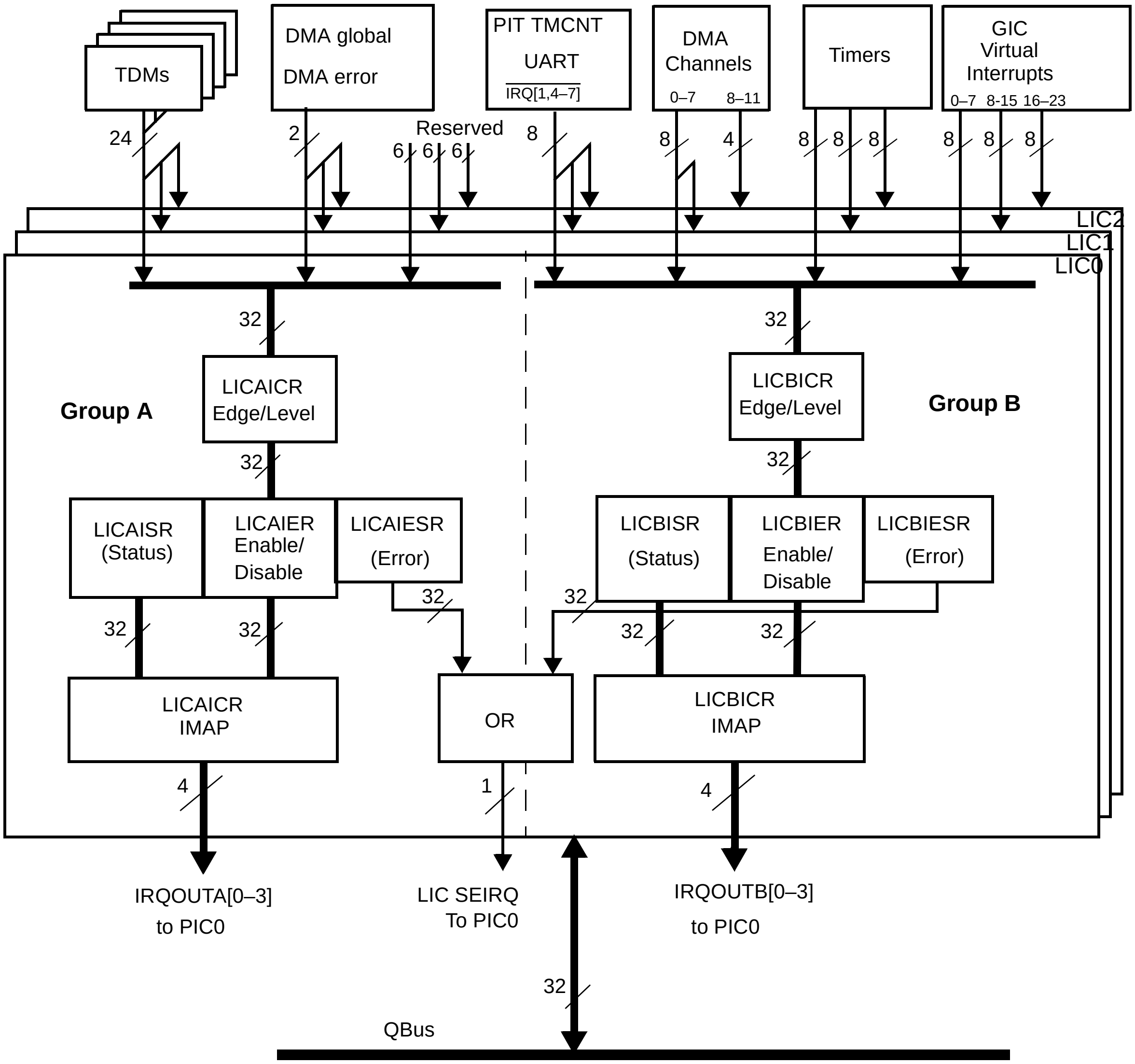


Figure 17-3. LIC Block Diagram

- All output lines towards the PIC are active low and should normally be programmed to level mode in the PIC. If a single LIC edge interrupt source is mapped to a single LIC interrupt output, the LIC may be programmed to level mode and the PIC programmed to edge mode, providing direct reflection of the interrupt edge to the PIC and edge detection at the PIC.
- Interrupt Group A has the following interrupt sources:
  - — Six interrupts from each TDM for a total of 24 sources.
  - — One DMA global channel interrupt and one DMA error interrupt (see **Section 16.4.2**, *DMA Status and Interrupt Registers*).
  - — Typically, the interrupt mapping is divided among the interrupt sources as DMA, TDM Tx, TDM Rx, and TDM Error, but any other combination is valid, depending on the application.

- Interrupt Group B has the following interrupt sources:
  - — Eight dedicated timers, with different timers for each SC140.
  - — Eight DMA channel interrupts, half of the channels connected to the LICs of SC140s 0 and 1, half to the LICs of SC140s 2 and 3.
  - — One global UART interrupt.
  - — Two global SIU interrupts from PIT and TMCNT.
  - — Five global $\overline{\text{IRQ}}$s.
  - — Eight virtual system interrupts from the GIC (see **Section 17.1.1.3**, *Virtual Interrupt Generation,* on page 17-7).
  - — Typically, the interrupt mapping is divided into one for DMA, one for SIU interrupts, one for timers, and one for virtual interrupts, but any other combination is valid, depending on the application.

Each interrupt source has the following programmable attributes:

- Two bits of IMAP control field map it to one of four interrupt output lines to the PIC.
- Two bits of EM (Edge Mode) select the interrupt source handling as level, edge, or dual edge mode.
- A primary status bit.
- In dual edge mode, a second edge error status bit. The sum of all second edge error status bits is sampled to generate a single second edge error detection interrupt output line to the PIC.

For details on LIC interrupts received at the PIC, see **Table 17-8**.

17.1.2.1 Resolving LIC Interrupts by the SC140 Cores

The SC140 cores support counting of leading bits using the CLB instruction. This feature can be used to achieve fast priority resolution between interrupts having the same priority level (that is, mapped to the same PIC input). The primary priority level is separated by mapping an interrupt source to different PIC inputs. Following is a simple SC140 core algorithm that uses both the PIC vector system and the CLB instruction for rapid detection of the LIC interrupt source to be serviced and for prioritizing the bit by location:

- Interrupts with different primary priority levels are mapped to different PIC inputs. Each group connects to four PIC inputs, and both groups share one PIC input for second edge error interrupts.
- While mapping and enabling a specific LIC interrupt source to a PIC input, the application should also set the appropriate bit in the 32-bit mask value associated with that input. Each bit set in this mask indicates that the specific LIC interrupt source is mapped to this PIC input. At the end of this set up, each PIC-mappable input from the LIC has an associated updated 32-bit mask value.

- When LIC interrupts are asserted and routed to one or more PIC inputs, interrupt arbitration in the PIC selects one of nine interrupt service routines (ISR), selected by the PIC vector, one for the second edge error and eight for the mapped interrupts. The ISR of mapped interrupts is associated with the 32-bit mask value updated at the initial setup, holding ‘1’ only for LIC source interrupts that are mapped to this PIC input. In addition, the ISR vector implicitly selects the proper LIC status to which it is related: the double edge error ISR is associated with LICAIESR or LICBIESR, four LIC group A interrupts are associated with LICAISR, and LIC group B interrupts are associated with LICBISR.
- The interrupt routine reads the appropriate 32-bit LIC status register and ANDs it with its unique 32-bit mask, filtering out only interrupts that are related to this ISR. This saves the long analysis required on the LIC interrupt mapping.
- The interrupt service routine may use a count leading bits instruction (CLB) to determine the left-most bit on that priority level and service it first. The result is the left-most bit from the interrupts mapped to the selected PIC input.
- After servicing, the ISR clears the serviced status bit in the LIC or in the source peripheral, as required. Then it can either redo the count leading bits for servicing the rest of the asserted interrupts mapped to this PIC input or simply exit the ISR.

Note: The interrupt service routine must wait until the deasserted interrupt line propagates all the way though the LIC and the PIC to the SC140 core interrupt line. The fastest way to work with LIC interrupt sources is programming the LIC to edge mode, which enables fast propagation to the core interrupt input.

17.1.2.2 Level Interrupt Mode

When an interrupt is programmed to be handled in level mode, the LIC continuously reflects the synchronized interrupt source at its dedicated status bit. If this interrupt is also enabled, it reflects the interrupt status at its mapped output. In this mode the LIC status bit is used only for reading the interrupt status locally instead of reading it at the peripheral. Second edge status bit operation is disabled and the second edge status bit is continuously cleared. To clear a level interrupt source, the SC140 core must access the interrupt origin directly at the peripheral status register. To prevent a false interrupt at the end of the interrupt service routine, the SC140 core must wait for the clearing operation at the peripheral to propagate the interrupt deassertion through the LIC synchronizer and output register and then though the PIC synchronizer and output register. In addition, if the write buffer takes over the write to the peripheral status register, the SC140 core must read back the status register to ensure proper flushing of the write buffer. Failing to do so may result in unpredictable delay of the actual write operation, which would cause a false interrupt detection from this source after the ISR is exited.

Level mode interrupts should be used in case an interrupt at the LIC input represents a sum of interrupt sources at the peripheral, since this configuration does not enable proper edge detection.

17.1.2.3 Edge Interrupt Mode

When an interrupt is programmed to be handled in edge mode, the LIC locally captures the peripheral-specific active edge of the interrupt line. In addition to the primary interrupt status bit, edge mode supports a secondary error status bit indicating second active edge detection. A sum of all second edge detections can generate one global second edge error interrupt. Once the active edge is detected while its primary status bit is not asserted, the primary status bit is set and remains set as a sticky bit until it is cleared by the SC140 core interrupt service routine. If a second active edge is detected while the primary status bit is set, then the second edge error status bit is set, and the error interrupt line is asserted.

17.1.2.4 DMA Interrupts

The DMA system generates 18 interrupt sources to the SC140 cores, 16 channel interrupts indicating a buffer empty condition, one global DMA error interrupt, and one global DMA interrupt that is the sum of all the 16 channel interrupts. Channel interrupt lines 0–7 are routed directly to LIC group B of SC140s 0 and 1, while channel interrupt lines 8–11 are routed directly to LIC group B of SC140 2. In addition, the sum of all EMA channel interrupts is routed globally to LIC group A of each SC140, enabling all channels be serviced by any SC140 core.

Typically, each SC140 core gets DMA channel interrupts that are related to its own activity. Following is an example of channel association to specific core interrupt lines:

- One interrupt related to the local M1 flyby address counter.
- One interrupt associated with an external DMA request.
- Interrupts associated with general-purpose DMA channels, or channels initiated by another SC140 core and used for core-to-core communication (DMA messaging system). For example, a flyby counter of another SC140 core may initiate a DMA transfer, and the associated interrupt is given to the receiving SC140 core upon data transfer completion.

17.1.2.5 LIC Interrupt Sources

The following tables list the LIC interrupt sources for each SC140 core. Group A gets global interrupts that are distributed in parallel to all SC140 cores and can be selectively enabled in each SC140 core. Group B contains some global interrupts and some private interrupt sources that are unique to each SC140 core.

Table 17-3. LIC Interrupt Group A Sources (Same for all SC140 Cores)

No.	Source	Description
0	TDM0RXER	TDM0 Receive Error (sum of TDM receive error detections).
1	TDM0RSTE	TDM0 Receive Second Threshold Event.
2	TDM0RFTE	TDM0 Receive First Threshold Event.
3	TDM0TXER	TDM0 Transmit Error (sum of TDM transmit error detections).
4	TDM0TSTE	TDM0 Transmit Second Threshold Event.

Table 17-3. LIC Interrupt Group A Sources (Same for all SC140 Cores) (Continued)

No.	Source	Description
5	TDM0TFTE	TDM0 Transmit First Threshold Event.
6	TDM1RXER	TDM1 Receive Error (sum of TDM receive error detections).
7	TDM1RSTE	TDM1 Receive Second Threshold Event
8	TDM1RFTE	TDM1 Receive First Threshold Event
9	TDM1TXER	TDM1 Transmit Error (sum of TDM transmit error detections).
10	TDM1TSTE	TDM1 Transmit Second Threshold Event
11	TDM1TFTE	TDM1 Transmit First Threshold Event
12	TDM2RXER	TDM2 Receive Error (sum of TDM receive error detections).
13	TDM2RSTE	TDM2 Receive Second Threshold Event
14	TDM2RFTE	TDM2 Receive First Threshold Event
15	TDM2TXER	TDM2 Transmit Error (sum of TDM transmit error detections).
16	TDM2TSTE	TDM2 Transmit Second Threshold Event
17	TDM2TFTE	TDM2 Transmit First Threshold Event
18	TDM3RXER	TDM3 Receive Error (sum of TDM receive error detections).
19	TDM3RSTE	TDM3 Receive Second Threshold Event
20	TDM3RFTE	TDM3 Receive First Threshold Event
21	TDM3TXER	TDM3 Transmit Error (sum of TDM transmit error detections).
22	TDM3TSTE	TDM3 Transmit Second Threshold Event
23	TDM3TFTE	TDM3 Transmit First Threshold Event
24	DMA	DMA global interrupt (sum of all channel interrupts)
25	DMA_ERROR	DMA error
26	—	Reserved
27	—	Reserved
28	—	Reserved
29	—	Reserved
30	—	Reserved
31	—	Reserved

Table 17-4. LIC Interrupt Group B Source for Core 0

No.	Source	Description
0	DMA0	DMA channel 0 interrupt
1	DMA1	DMA channel 1 interrupt
2	DMA2	DMA channel 2 interrupt
3	DMA3	DMA channel 3 interrupt
4	DMA4	DMA channel 4 interrupt
5	DMA5	DMA channel 5 interrupt
6	DMA6	DMA channel 6 interrupt
7	DMA7	DMA channel 7 interrupt
8	TIMER0A	Timer Block A Timer 0 Compare Flag
9	TIMER1A	Timer Block A Timer 1 Compare Flag
10	TIMER2A	Timer Block A Timer 2 Compare Flag

Table 17-4. LIC Interrupt Group B Source for Core 0 (Continued)

No.	Source	Description
11	TIMER3A	Timer Block A Timer 3 Compare Flag
12	TIMER8A	Timer Block A Timer 8 Compare Flag
13	TIMER9A	Timer Block A Timer 9 Compare Flag
14	TIMER10A	Timer Block A Timer 10 Compare Flag
15	TIMER11A	Timer Block A Timer 11 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	$\overline{\text{IRQ1}}$	$\overline{\text{IRQ1}}$ signal (global)
20	VIRQ0	Virtual Interrupt Number 0
21	VIRQ1	Virtual Interrupt Number 1
22	VIRQ2	Virtual Interrupt Number 2
23	VIRQ3	Virtual Interrupt Number 3
24	VIRQ4	Virtual Interrupt Number 4
25	VIRQ5	Virtual Interrupt Number 5
26	VIRQ6	Virtual Interrupt Number 6
27	VIRQ7	Virtual Interrupt Number 7
28	$\overline{\text{IRQ4}}$	$\overline{\text{IRQ4}}$ signal (global)
29	$\overline{\text{IRQ5}}$	$\overline{\text{IRQ5}}$ signal (global)
30	$\overline{\text{IRQ6}}$	$\overline{\text{IRQ6}}$ signal (global)
31	$\overline{\text{IRQ7}}$	$\overline{\text{IRQ7}}$ signal (global)

Table 17-5. LIC Interrupt Group B Source for Core 1

No.	Source	Description
0	DMA0	DMA channel 0 interrupt
1	DMA1	DMA channel 1 interrupt
2	DMA2	DMA channel 2 interrupt
3	DMA3	DMA channel 3 interrupt
4	DMA4	DMA channel 4 interrupt
5	DMA5	DMA channel 5 interrupt
6	DMA6	DMA channel 6 interrupt
7	DMA7	DMA channel 7 interrupt
8	TIMER4A	Timer Block A Timer 4 Compare Flag
9	TIMER5A	Timer Block A Timer 5 Compare Flag
10	TIMER6A	Timer Block A Timer 6 Compare Flag
11	TIMER7A	Timer Block A Timer 7 Compare Flag
12	TIMER12A	Timer Block A Timer 12 Compare Flag
13	TIMER13A	Timer Block A Timer 13 Compare Flag
14	TIMER14A	Timer Block A Timer 14 Compare Flag
15	TIMER15A	Timer Block A Timer 15 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)

Table 17-5. LIC Interrupt Group B Source for Core 1 (Continued)

No.	Source	Description
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	$\overline{\text{IRQ1}}$	$\overline{\text{IRQ1}}$ signal (global)
20	VIRQ8	Virtual Interrupt Number 8
21	VIRQ9	Virtual Interrupt Number 9
22	VIRQ10	Virtual Interrupt Number 10
23	VIRQ11	Virtual Interrupt Number 11
24	VIRQ12	Virtual Interrupt Number 12
25	VIRQ13	Virtual Interrupt Number 13
26	VIRQ14	Virtual Interrupt Number 14
27	VIRQ15	Virtual Interrupt Number 15
28	$\overline{\text{IRQ4}}$	$\overline{\text{IRQ4}}$ signal (global)
29	$\overline{\text{IRQ5}}$	$\overline{\text{IRQ5}}$ signal (global)
30	$\overline{\text{IRQ6}}$	$\overline{\text{IRQ6}}$ signal (global)
31	$\overline{\text{IRQ7}}$	$\overline{\text{IRQ7}}$ signal (global)

Table 17-6. LIC Interrupt Group B Source for Core 2

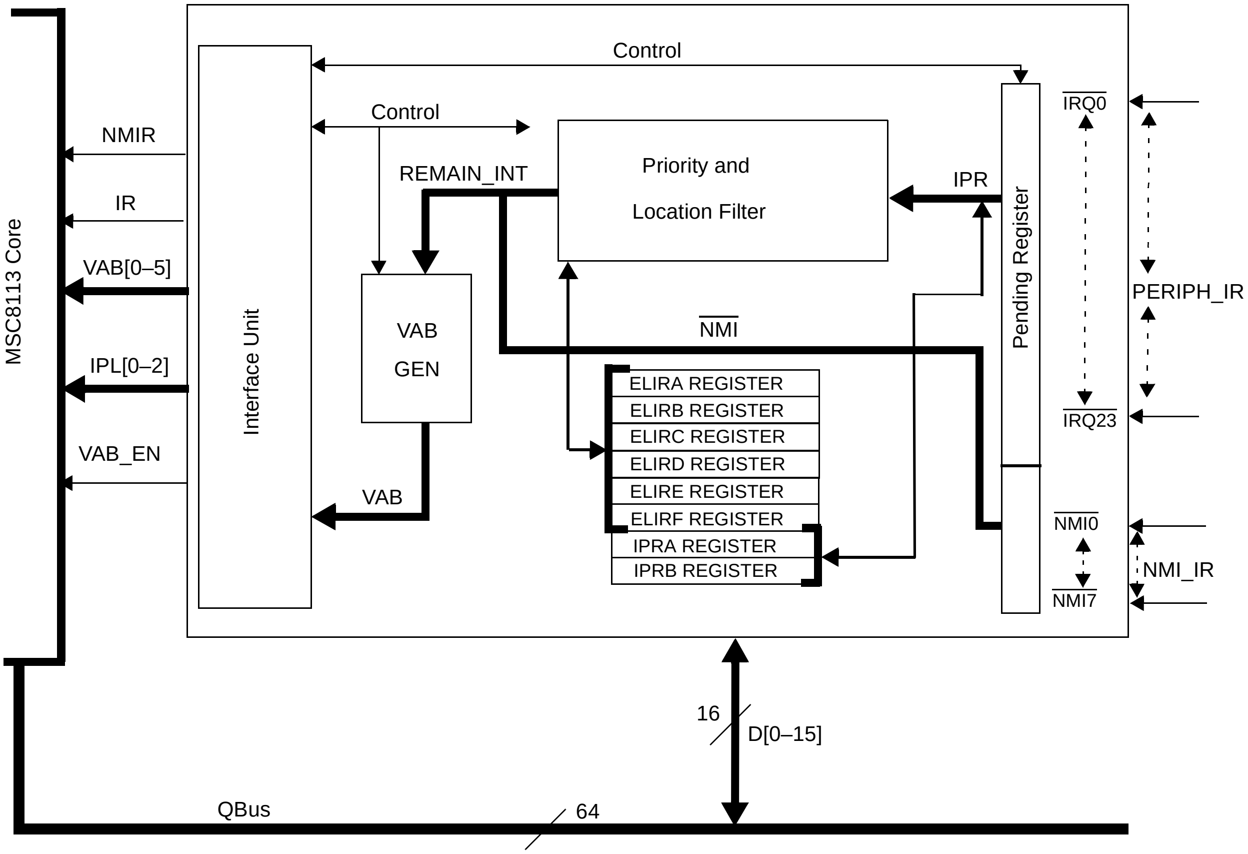
No.	Source	Description
0	DMA8	DMA channel 8 interrupt
1	DMA9	DMA channel 9 interrupt
2	DMA10	DMA channel 10 interrupt
3	DMA11	DMA channel 11 interrupt
4	DMA12	DMA channel 12 interrupt
5	DMA13	DMA channel 13 interrupt
6	DMA14	DMA channel 14 interrupt
7	DMA15	DMA channel 15 interrupt
8	TIMER0B	Timer Block B Timer 0 Compare Flag
9	TIMER1B	Timer Block B Timer 1 Compare Flag
10	TIMER2B	Timer Block B Timer 2 Compare Flag
11	TIMER3B	Timer Block B Timer 3 Compare Flag
12	TIMER8B	Timer Block B Timer 8 Compare Flag
13	TIMER9B	Timer Block B Timer 9 Compare Flag
14	TIMER10B	Timer Block B Timer 10 Compare Flag
15	TIMER11B	Timer Block B Timer 11 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	$\overline{\text{IRQ1}}$	$\overline{\text{IRQ1}}$ signal (global)
20	VIRQ16	Virtual Interrupt Number 16
21	VIRQ17	Virtual Interrupt Number 17
22	VIRQ18	Virtual Interrupt Number 18
23	VIRQ19	Virtual Interrupt Number 19
24	VIRQ20	Virtual Interrupt Number 20

Table 17-6. LIC Interrupt Group B Source for Core 2 (Continued)

No.	Source	Description
25	VIRQ21	Virtual Interrupt Number 21
26	VIRQ22	Virtual Interrupt Number 22
27	VIRQ23	Virtual Interrupt Number 23
28	IRQ4	IRQ4 signal (global)
29	IRQ5	IRQ5 signal (global)
30	IRQ6	IRQ6 signal (global)
31	IRQ7	IRQ7 signal (global)

17.1.3 Programmable Interrupt Controller (PIC)

The MSC8113 PIC is a peripheral module that serves the IRQ and NMI signals received from MSC8113 peripherals and I/O lines. The PIC is memory-mapped to the SC140 and is accessed via the SC140 QBus. The PIC includes 32 inputs for IRQ signals and NMI signals: eight asynchronous edge-triggered NMI inputs and the 24 asynchronous edge-triggered or level-triggered IRQ inputs. The PIC has an auto-vector interrupt generation that supports eight priority levels.



Note: Bold lines are bus lines; the thinner lines are control and data lines.

Figure 17-4. PIC Block Diagram

17.1.4 Peripheral Bus (QBus) Interface

The QBus interface provides the control and status registers, buffering of the internal bus from the QBus, and address decoding generation. The various interrupt request control and status registers of the QBus interface are described in **Section 17.3.3**, *PIC Registers*. **Table 17-7** summarizes the interface registers that are accessible to the SC140 through the QBus interface.

Table 17-7. PIC Interface Registers

Register Name	Description	Page
ELIRA	Edge/Level-Triggered Interrupt Priority Register A	**page 17-41**
ELIRB	Edge/Level-Triggered Interrupt Priority Register B	**page 17-41**
ELIRC	Edge/Level-Triggered Interrupt Priority Register C	**page 17-41**
ELIRD	Edge/Level-Triggered Interrupt Priority Register D	**page 17-42**
ELIRE	Edge/Level-Triggered Interrupt Priority Register E	**page 17-42**
ELIRF	Edge/Level-Triggered Interrupt Priority Register F	**page 17-42**
IPRA	Interrupt Pending Register A	**page 17-43**
IPRB	Interrupt Pending Register B	**page 17-44**

17.1.4.1 Interrupt Request Generation

When the PIC detects an $\overline{\mathrm{IRQ}}$ on one or more of its inputs, it arbitrates each $\overline{\mathrm{IRQ}}$ according to its priority level and location and generates the following:

- An $\overline{\mathrm{IRQ}}$ signal to the SC140 core, indicating that an $\overline{\mathrm{IRQ}}$ input has requested interrupt service by the SC140 core.
- An IPL[2–0] signal indicating the priority of the $\overline{\mathrm{IRQ}}$.
- An entry in the predefined VAB, determined by the location of the $\overline{\mathrm{IRQ}}$.

17.1.5 Interrupt Routing

The MSC8113 PIC serves a total of 24 $\overline{\mathrm{IRQ}}$ and eight $\overline{\mathrm{NMI}}$. Each $\overline{\mathrm{IRQ}}$ can be configured as edge-triggered or level-triggered and can be assigned a priority in the range 0 through 7, where priority 0 masks the interrupt. On reset, all $\overline{\mathrm{IRQ}}$ signals are masked (set to priority 0) and configured as level-triggered. On bootstrap, $\overline{\mathrm{IRQ20}}$ (see **Table 17-8**) is configured as edge-triggered. The $\overline{\mathrm{NMI}}$ relative priority is fixed, with $\overline{\mathrm{NMI0}}$ assigned the lowest priority and $\overline{\mathrm{NMI7}}$ the highest. $\overline{\mathrm{NMI}}$ signals are always edge-triggered.

The PIC programming model enables you to ensure that specific sets execute in sequence by masking all interrupts, using the **di** (disable interrupts) instruction. The restriction imposed by **di** is removed by issuing the EI (enable interrupts) instruction.

To mask interrupts up to a specified priority level, set the selected priority level in the SC140 status register via the interrupt mask bits I[2–0]. The core handles only $\overline{\text{NMI}}$ signals, or interrupts with an IPL higher than the current interrupt mask value. At reset these bits are set, and all interrupts are disabled. The interrupt mask bits, I2, I1 and I0, reflect the current IPL of the SC140 core. Refer to the *SC140 DSP Core Reference Manual* for details on the SC140 core status registers. The interrupt programming model consists of:

- Setting the interrupt table base address in the VBA Register
- Programming the PIC ELIRx Registers
- Masking interrupts in the core status register
- Programming the interrupt service routines in the appropriate addresses starting from the base address in VBA.

The memory allocation for each interrupt routine is 64 bytes, which constitutes four program fetches. SC140 instructions are encoded as two to four bytes, with a minimum instruction size of one word. An average of 20 instructions can be held in the allocated memory area. To further extend the code size, the use of service routines is recommended, as shown in the example in **Section 17.2.3**, *Clearing Pending Requests*. The address calculation is based on the VBA Register and the VAB vector, as shown in **Figure 17-5**.

Address bits

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
VBA[31–12]																				VAB[5–0]						0	0	0	0	0	0

Figure 17-5. Interrupt Service Routine Address Construction

Table 17-8 summarizes the routing of MSC8113 interrupts. Unless stated otherwise, all $\overline{\text{IRQ}}$ signals are level-triggered. For details on $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ signals, refer to the relevant chapters The PIC handles interrupts $\overline{\text{IRQ[0–23]}}$ and $\overline{\text{NMI[0–7]}}$.

Table 17-8. MSC8113 Interrupt Routing

VAB[0–5]	Signal	Description	Service Routine Address (Offset from VBA)
0x0	TRAP	Internal exception (generated by trap instruction)	0x0
0x1	—	Reserved	0x40
0x2	ILLEGAL	Illegal instruction or set[1]	0x80
0x3	DEBUG	Debug exception (EOnCE)	0xC0
0x4	OVERFLOW	Overflow exception (DALU)	0x100
0x5	—	Reserved	0x140
0x6	DEFAULT $\overline{\text{NMI}}$	In VAB disabled mode only	0x180
0x7	DEFAULT $\overline{\text{IRQ}}$	In VAB disabled mode only	0x1C0
0x8–0x1F	—	Reserved	0x200–0x7FF
0x20	$\overline{\text{IRQ0}}$	Ethernet Ring 0 receive frame event	0x800
0x21	$\overline{\text{IRQ1}}$	Ethernet Ring 1 receive frame event	0x840

Table 17-8. MSC8113 Interrupt Routing (Continued)

VAB[0–5]	Signal	Description	Service Routine Address (Offset from VBA)
0x22	$\overline{\text{IRQ2}}$	Ethernet Ring 2 receive frame event	0x880
0x23	$\overline{\text{IRQ3}}$	Ethernet Ring 3 receive frame event	0x8C0
0x24	$\overline{\text{IRQ4}}$	Ethernet Transmit frame event	0x900
0x25	$\overline{\text{IRQ5}}$	Ethernet controller Receive Inter Frame Gap Status Interrupt (RIFGSI) **Note:** RIFGSI is only used in SMII mode.	0x940
0x26	$\overline{\text{IRQ6}}$	LIC IRQOUTA0 Group A	0x980
0x27	$\overline{\text{IRQ7}}$	LIC IRQOUTA1 - Group A	0x9C0
0x28	$\overline{\text{IRQ8}}$	LIC IRQOUTA2 - Group A	0xA00
0x29	$\overline{\text{IRQ9}}$	LIC IRQOUTA3 - Group A	0xA40
0x2A	$\overline{\text{IRQ10}}$	Reserved	0xA80
0x2B	$\overline{\text{IRQ11}}$	QBus controller (local bus contention)	0xAC0
0x2C	$\overline{\text{IRQ12}}$	QBus controller (p-x contention)	0xB00
0x2D	$\overline{\text{IRQ13}}$	QBus controller (misaligned data error)[2]	0xB40
0x2E	$\overline{\text{IRQ14}}$	LIC IRQOUTB0 Group B	0xB80
0x2F	$\overline{\text{IRQ15}}$	External $\overline{\text{IRQ2}}$ (edge/level configurable)	0xBC0
0x30	$\overline{\text{IRQ16}}$	GIC - global interrupt	0xC00
0x31	$\overline{\text{IRQ17}}$	External $\overline{\text{IRQ3}}$ (edge/level configurable)	0xC40
0x32	$\overline{\text{IRQ18}}$	LIC IRQOUTB1 Group B	0xC80
0x33	$\overline{\text{IRQ19}}$	Reserved	0xCC0
0x34	$\overline{\text{IRQ20}}$	EOnCE interrupt (edge-triggered)	0xD00
0x35	$\overline{\text{IRQ21}}$	LIC IRQOUTB2 Group B	0xD40
0x36	$\overline{\text{IRQ22}}$	LIC IRQOUTB3 Group B	0xD80
0x37	$\overline{\text{IRQ23}}$	LICSEIRQ - LIC Second Edge IRQ (Groups A and B)	0xDC0
0x38	$\overline{\text{NMI0}}$	GIC Virtual $\overline{\text{NMI}}$ of this core	0xE00
0x39	$\overline{\text{NMI1}}$	Reserved	0xE40
0x3A	$\overline{\text{NMI2}}$	QBus controller (memory write error)	0xE80
0x3B	$\overline{\text{NMI3}}$	QBus controller (misaligned program error)	0xEC0
0x3C	$\overline{\text{NMI4}}$	QBus Controller (bus error–unmapped memory space)[3]	0xF00
0x3D	$\overline{\text{NMI5}}$	System interface block TEA on System bus	0xF40
0x3E	$\overline{\text{NMI6}}$	Reserved	0xF80
0x3F	$\overline{\text{NMI7}}$	SIU $\overline{\text{NMI}}$ (from GIC), for example, Software watchdog, external $\overline{\text{NMI}}$, parity error, bus monitor	0xFC0

Notes: **1.** A typical cause for this exception is a branch to an address where no code is loaded or a hardware problem has caused a corruption of code read from an external device. In rare instances, however, because there is some latency in the interrupt processing, the illegal code can alter system registers (such as PLL, memory banks, and so forth) before the interrupt is handled. In these rare cases, the interrupt may not be serviced and the system may lock up, requiring a system recovery (stop the process or reset the system, for example).

2. For **rte/d**, **rts/d**, and **rtstk** instructions, an indirect change of flow may not invoke this interrupt for non-aligned addresses.

3. Access to unmapped space includes implicit access, such as prefetching subsequent addresses when executing an instruction even if the fetched words are not used.

17.2 Interrupt Programming Examples

This section describes how to use the LIC and the PIC programming model for $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ signals. The programming examples include the following functionality:

- Setting the interrupt base address in the VBA Register
- Initializing the stack pointer
- Masking interrupts in the MSC8113 status register
- Masking, unmasking and programming PIC IR properties in the ELIRx registers
- Configuring the LIC Configuration register EMx and IMAPx in the LICICR
- Masking, unmasking interrupts in the LICIER registers
- Clearing a pending $\overline{\text{IRQ}}$ in the IPRx register
- Using interrupt service routines longer than 64 bytes

17.2.1 Initialization

The VBA is a 32-bit read/write register that holds the 20 MSB of the interrupt table base address. Consequently, the 12 LSB of this register must be cleared. At bootstrap, the VBA is initialized to the ROM base address (0x01077000), and the stack pointers of the cores are initialized to 0x01076f80 (core 0), 0x01076fa0 (core 1), and 0x01076fc0 (core 2). You can change these values before issuing a call to any subroutine, since this address may not be available for the stack, depending on the application. At reset, the SC140 cores disable all maskable interrupts.

When an IR occurs, the status register is pushed onto the stack, and the interrupt priority level (IPL) of the current IR is written to SR[23–21]. All IRs with a priority level less than or equal to the IPL of the current IR are masked. The following example programs the interrupt base address in VBA, initializes the stack pointer and enables interrupts with priority levels 5 or 6 only. All interrupts with priority level 4 or less are masked.

```
...
;Programming the VBA register to address 0x5000
move.l #$5000,vba
;Initializing the stack pointer to address 0x32000
(200KB)
move.l #$32000,r0
nop
tfra r0,sp
...
...
; Masking interrupts of priority 0,1,2,3,4
bmclr #$0040,sr.h
...
```

17.2.2 LIC and PIC Programming

The ELIRA–ELIRF are 16-bit read/write PIC control registers by which you configure the priority level and select the trigger mode for each interrupt. See **Section 17.3.3**, *PIC Registers*, on page 17-40. On reset, all $\overline{\text{IRQ}}$ signals are masked (set to priority 0) and configured as level-triggered. The following example shows how to assign priority 5 to the GIC global interrupt and priority 4 to the LIC IRQOUTB1 interrupt. In the LIC, VIRQ0 is enabled in second-edge mode, and mapped to IRQOUTB1.

Example 17-1. Assigning Interrupt Priorities

```
...
; BASE0 is 0x00f00000
ELIRE equ $00f09c20    ; PIC Edge/Level-Triggered
Interrupt Priority Register E
LICBICR1 equ $00f0ac48 ; LIC Group B Interrupt
Configuration Register 1
LICBIER equ $00f0ac60  ; LIC Group B Interrupt
Enable Register
IRQ16 equ $30c00       ; GIC global interrupt
routine
IRQ18 equ $30c80       ; LICBICR1 interrupt routine
; VBA is set to 0x30000 (offset: 192KB)
move.l #$30000,vba

; assign priority 5 to GIC global interrupt, level
mode (irq 16)
; assign priority 4 to LIC IRQOUTB1, level mode (irq
18)
move.w #$0506,ELIRE

; configure the LIC-B, EM20='10' set to edge trigger
- second-edge mode
; configure the LIC-B IMAP20='01' route the
interrupt line to IRQOUTB1
move.l #$00090000,LICBICR1

; Enable LIC-B, interrupt number 20 (VRIQ0)
move.l #$00100000,LICBIER
...
org p:IRQ16
; interrupt service routine for GIC
rte
org p:IRQ18
; interrupt service routine for LIC
rte
```

17.2.3 Clearing Pending Requests

If the size of the interrupt routine is larger than 64 bytes, you can use service routines to accommodate unlimited code size. The following example illustrates a typical interrupt routine that uses a service routine. This example also demonstrates the use of the **DI** and **EI** instructions to disable and enable IRs, respectively. For details, refer to **Section 17.3.3.3**, *Interrupt Pending Registers,* on page 17-43.

Example 17-2. Typical Interrupt Routine

```
IPRB equ $00f09c38
...
org p:IRQ16
; interrupt routine for GIC
di ; disable any IR
jsr GIC_IRQ
nop
ei ; enable IR
rte
...
org p:GIC_IRQ
; Code to locate GIC interrupt source and clear
status bit at the source.
; To override write-buffer delay - read-back the
status register in which the.
; bit had been cleared. The PIC pending bit IPRB[0]
is cleared by the
; interrupt propagation to IPRB[0].
...
; service routine to handle GIC
move.w #$1,IPRB
; interrupt service routine to handle GIC
...
rts
```

17.3 Interrupts Programming Model

The interrupt registers are divided into three groups: GIC, LIC, and PIC.

17.3.1 GIC Programming Model

The GIC registers reside on the 256 KB address space of the IPBus and have a constant offset. They can be accessed through the SQBus, the system bus, and the DSI. The addresses of the GIC registers for accesses through the SQBus are presented in **Section 8.5**, *IPBus Address Space*, on page 8-12. The addresses of the GIC registers for accesses through the system bus are presented in **Section 8.7**, *System Bus Address Space*, on page 8-55. The addresses of the GIC registers for accesses through the DSI is presented in **Section 8.8**, *DSI Address Map*, on page 8-61. The GIC has one interrupt configuration register, two interrupt status registers, and four interrupt control registers:

- Virtual Interrupt Generation Register (VIGR), **page 17-24**.
- Virtual Interrupt status register (VISR), **page 17-25**.
- Virtual NMI Generation Register (VNMIGR), **page 17-25**.
- GIC Interrupt Configuration Register (GICR), **page 17-26**.
- GIC External Interrupt Enable Register (GEIER), **page 17-27**.
- GIC Core Interrupt Enable Register (GCIER), **page 17-28**.
- GIC Interrupt Status Register (GISR), **page 17-29**.

VIGR Virtual Interrupt Generation Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	W															

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	—	—	—	—	—	—	CORENUM		—	—	—	—	—	VIRQNUM		
Type	W															

VIGR generates virtual interrupt pulses (edge interrupt source), according to the written data. A read from VIGR returns all zeros. There are four groups of eight interrupts, each associated with one SC140. A write access to VIGR with appropriate values to both the CORENUM and VIRQNUM fields generates one of eight interrupt pulses to the LIC of the selected SC140 (both fields must be written in the same access). The generated interrupt pulse also sets an interrupt status bit in the VISR, selected by the concatenation of {CORENUM,VIRQNUM}. For example, if virtual interrupt number 5 (“101”) is generated to core number 2 (“10”), VS number 21 (“10101”) is set in the VISR.

A set status bit in the VISR does not block generation of interrupt pulses by additional write accesses. In addition, interrupt pulse number 0 of each SC140 core also sets one of four status bits in the GISR, each can be enabled by a proper bit in GEIER to assert the external $\overline{\text{INT\_OUT}}$ line.

Note: For proper virtual interrupt operation, the LIC must be programmed to edge mode for the virtual interrupt pulse sources.

Table 17-9. VIGR Bit Descriptions

Name	Reset	Description	Settings
— 0–21	—	Reserved. Write to zero for future compatibility.	

Table 17-9. VIGR Bit Descriptions (Continued)

Name	Reset	Description	Settings
CORENUM 22-23	—	**Core Number Selection** Set an interrupt to the selected SC140.	00 Set an interrupt to core number 0. 01 Set an interrupt to core number 1. 10 Set an interrupt to core number 2. 11 Reserved.
— 24–28	—	Reserved. Write to zero for future compatibility.	
VIRQNUM 29-31	—	**VS Interrupt Number Selection**	000Select VS Interrupt number 0. ... 111 Select VS Interrupt number 7.

VISR Virtual Interrupt Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	VS31	VS30	VS29	VS28	VS27	VS26	VS25	VS24	VS23	VS22	VS21	VS20	VS19	VS18	VS17	VS16
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**
	VS15	VS14	VS13	VS12	VS11	VS10	VS9	VS8	VS7	VS6	VS5	VS4	VS3	VS2	VS1	VS0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the VISR corresponds to one virtual interrupt source, selected by proper write access to the VIGR. Each group of eight virtual interrupt pulses is routed to a different SC140 core LIC (Core 2 gets VS[23–16], Core 1 gets VS[15–8], and Core 0 gets the eight least significant bits—VS[7–0]). When the interrupt pulse is generated by this write access, the GIC sets the corresponding status bit. It is the responsibility of the interrupt service routine of the destination SC140 core to clear only the correct status bits by writing ones to them. Writing zeros to status bits has no effect on their status. A set status bit does not block the generation of another virtual interrupt pulse by additional writes to VIGR with appropriate values.

VNMIGR Virtual NMI Generation Register

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

VNMIGR generates a virtual NMI pulse to the selected SC140. A read from VNMIGR returns all zeros. Write access to VNMIGR with appropriate value to the CORENUM field generates an NMI pulse to the selected core. The generated NMI pulse sets the NMI0 status bit in the selected SC140 PIC IPRB register. See **Table 17-8**.

Table 17-10. VNMIGR Bit Descriptions

Name	Reset	Description	Settings
— 0–21	—	Reserved. Write to zero for future compatibility.	
CORENUM 22-–23	—	**Core Number Selection** Set NMI0 of the selected SC140 core.	00 Set $\overline{\text{NMI0}}$ of core 0. 01 Set $\overline{\text{NMI0}}$ of core 1. 10 Set $\overline{\text{NMI0}}$ of core 2. 11 Reserved
— 24–31	—	Reserved. Write to zero for future compatibility.	

GICR GIC Interrupt Configuration Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	VS24	VS16	VS8	VS0	ETHAE	—	—	—	UART	TMCNT	PIT	DMA
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	1	1	1	1	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
	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GICR configures each interrupt source to edge- or level-trigger mode. **Table 17-11** shows the settings for the interrupt trigger mode bit. For proper operation of virtual interrupts, the VSx bits must be programmed to edge mode.

Table 17-11. GICR Bit Descriptions

Name	Reset	Description	Settings
— 0–3	0	Reserved. Write to zero for future compatibility.	
VS[24,16,8,0] 4–7	0	**Virtual Source 24, 16, 8, or 0 Interrupt**	0 Level Mode. 1 Edge mode.
ETHAE 8	0	**Ethernet Another Event Interrupt**	0 Level Mode. 1 Edge mode.
— 9–11	0	Reserved. Write to zero for future compatibility.	
UART 12	0	**UART Interrupt**	0 Level Mode. 1 Edge mode.
TMCNT 13	0	**TMCNT Interrupt**	0 Level Mode. 1 Edge mode.
PIT 14	0	**PIT Interrupt**	0 Level Mode. 1 Edge mode.

Table 17-11. GICR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA 15	0	**DMA Interrupt**	0 Level Mode. 1 Edge mode.
IRQ[15–1] 16–30	0	**Interrupt Request 15–1**	0 Level Mode. 1 Edge mode.
— 31	0	Reserved. Write to zero for future compatibility.	

GEIER GIC External Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	VS24	VS16	VS8	VS0	—	—	—	—	UART	TMCNT	PIT	DMA
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
	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in GEIER enables one interrupt source to assert the $\overline{\text{INT\_OUT}}$ line.

Table 17-12. GEIER Bit Descriptions

Name	Reset	Description	Settings
— 0–3	0	Reserved. Write to zero for future compatibility.	
VS[24,16,8,0] 4–7	0	**Virtual Source 24, 16, 8, or 0 Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
— 8–11	0	Reserved. Write to zero for future compatibility.	
UART 12	0	**UART Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
TMCNT 13	0	**TMCNT Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
PIT 14	0	**PIT Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
DMA 15	0	**DMA Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
IRQ[15–1] 16–30	0	**Interrupt Request 15–1**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to output $\overline{\text{INT\_OUT}}$.
— 31	0	Reserved. Write to zero for future compatibility.	

GCIER GIC Core Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	—	—	—	—	ETHAE	—	—	—	—	—	—	—
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
	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	—	—	—	—	—	—	—	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GCIER enables assertion of the *GIC Global interrupt* input to PIC $\overline{\text{IRQ16}}$, which is global to all the SC140 cores. The corresponding status bit resides in GISR, in the same bit locations. The same status bit can be enabled to assert either $\overline{\text{INT\_OUT}}$ (by GEIER) or the SC140 core GIC interrupt (by GCIER). However, you should enable only one destination per status bit to avoid conflict between internal and external interrupt handlers. Reserved bits in GCIER should be cleared for future compatibility.

Table 17-13. GCIER Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
ETHAE 8	0	**Ethernet Another Event Interrupt**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to the PIC $\overline{\text{IRQ16}}$.
— 9–15	0	Reserved. Write to zero for future compatibility.	
IRQ[15–8] 16–23	0	**Interrupt Request 15–8**	0 Disable interrupt. 1 Enable interrupt and route the interrupt line to the PIC $\overline{\text{IRQ16}}$.
— 24–31	0	Reserved. Write to zero for future compatibility.	

GISR GIC Interrupt Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	VS24	VS16	VS8	VS0	ETHAE	—	—	—	UART	TMCNT	PIT	DMA
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
	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the GISR corresponds to one interrupt source. In level mode, the status bit is read-only and continuously reflects the synchronized interrupt source status. The interrupt service routine must clear the interrupt source at its origin. In edge mode, the status bit captures the active edge of the interrupt source, and the interrupt service routine can clear it by writing a one to the bit. Writing zeros has no effect. All GISR status bits can be enabled for interrupt generation on the $\overline{\text{INT\_OUT}}$ line by GEIER, while lines $\overline{\text{IRQ[15–8]}}$ can be enabled to generate a global GIC interrupt by the GCIER to all the SC140 cores.

17.3.2 LIC Programming Model

Each LIC group has four interrupt configuration registers, one enable register, and two status registers:

- LIC Group A Interrupt Configuration Register (LICAICR[0–3]), **page 17-30**.
- LIC Group A Interrupt Enable Register (LICAIER), **page 17-37**.
- LIC Group A Interrupt Status Register (LICAISR), **page 17-38**.
- LIC Group A Interrupt Error Status Register (LICAIESR), **page 17-39**.
- LIC Group B Interrupt Configuration Register (LICBICR[0–3]), **page 17-33**.
- LIC Group B Interrupt Enable Register (LICBIER), **page 17-37**.
- LIC Group B Interrupt Status Register (LICBISR), **page 17-38**.
- LIC Group B Interrupt Error Status Register (LICBIESR), **page 17-39**.

Each LIC resides only in its associated core memory space, and all LIC blocks are seen by their associated SC140 on the same addresses.

17.3.2.1 LIC Interrupt Configuration Registers

LICAICR[0–3] configure the 32 interrupts of group A for edge/level modes and map each interrupt to one of four PIC inputs.

LICAICR0 LIC Group A Interrupt Configuration Register 0

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM31		IMAP31		EM30		IMAP30		EM29		IMAP29		EM28		IMAP28	
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
	EM27		IMAP27		EM26		IMAP26		EM25		IMAP25		EM24		IMAP24	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-14. LICAICR0 Bit Descriptions

Name	Reset	Description	Settings
EM[31–24] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 31–24.	00 Level Mode. The corresponding interrupt status bit in LICAISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICAIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICAISR is set, the interrupt is captured by the corresponding bit in the LICAIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[31–24] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 31–24.	00 Route an enabled interrupt line through IRQOUTA0 into the PIC. 01 Route an enabled interrupt line through IRQOUTA1 into the PIC. 10 Route an enabled interrupt line through IRQOUTA2 into the PIC. 11 Route an enabled interrupt line through IRQOUTA3 into the PIC.

LICAICR1 LIC Group A Interrupt Configuration Register 1

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM23		IMAP23		EM22		IMAP22		EM21		IMAP21		EM20		IMAP20	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EM19		IMAP19		EM18		IMAP18		EM17		IMAP17		EM16		IMAP16	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Table 17-15. LICAICR1 Bit Descriptions

Name	Reset	Description	Settings
EM[23–16] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 23–16.	00 Level Mode. The corresponding interrupt status bit in LICAISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICAIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICAISR is set, the interrupt is captured by the corresponding bit in the LICAIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[23–16] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 23–16.	00 Route an enabled interrupt line through IRQOUTA0 into the PIC. 01 Route an enabled interrupt line through IRQOUTA1 into the PIC. 10 Route an enabled interrupt line through IRQOUTA2 into the PIC. 11 Route an enabled interrupt line through IRQOUTA3 into the PIC.

LICAICR2 LIC Group A Interrupt Configuration Register 2

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM15		IMAP15		EM14		IMAP14		EM13		IMAP13		EM12		IMAP12	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EM11		IMAP11		EM10		IMAP10		EM9		IMAP9		EM8		IMAP8	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 17-16. LICAICR2 Bit Descriptions

Name	Reset	Description	Settings
EM[15–8] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 15–8.	00 Level Mode. The corresponding interrupt status bit in LICAISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICAIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICAISR is set, the interrupt is captured by the corresponding bit in the LICAIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[15–8] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 15–8.	00 Route an enabled interrupt line through IRQOUTA0 into the PIC. 01 Route an enabled interrupt line through IRQOUTA1 into the PIC. 10 Route an enabled interrupt line through IRQOUTA2 into the PIC. 11 Route an enabled interrupt line through IRQOUTA3 into the PIC.

LICAICR3 LIC Group A Interrupt Configuration Register 3

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM7		IMAP7		EM6		IMAP6		EM5		IMAP5		EM4		IMAP4	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EM3		IMAP3		EM2		IMAP2		EM1		IMAP1		EM0		IMAP0	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0

Table 17-17. LICAICR3 Bit Descriptions

Name	Reset	Description	Settings
EM[7–0] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 7–0.	00 Level Mode. The corresponding interrupt status bit in LICAISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICAIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICAISR is set, the interrupt is captured by the corresponding bit in the LICAIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[7–0] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 7–0.	00 Route an enabled interrupt line through IRQOUTA0 into the PIC. 01 Route an enabled interrupt line through IRQOUTA1 into the PIC. 10 Route an enabled interrupt line through IRQOUTA2 into the PIC. 11 Route an enabled interrupt line through IRQOUTA3 into the PIC.

LICBICR0 LIC Group B Interrupt Configuration Register 0

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM31		IMAP31		EM30		IMAP30		EM29		IMAP29		EM28		IMAP28	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	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
	EM27		IMAP27		EM26		IMAP26		EM25		IMAP25		EM24		IMAP24	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

LICBICR[0–3] registers configure the 32 interrupts of group B for edge/level modes and map each interrupt to one of four PIC inputs.

Table 17-18. LICBICR0 Bit Descriptions

Name	Reset	Description	Settings
EM[31–24] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 31–24.	00 Level Mode. The corresponding interrupt status bit in LICBISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICBIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICBISR is set, the interrupt is captured by the corresponding bit in the LICBIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.

Table 17-18. LICBICR0 Bit Descriptions (Continued)

Name	Reset	Description	Settings
IMAP[31–24] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 31–24.	00 Route an enabled interrupt line through IRQOUTB0 into the PIC. 01 Route an enabled interrupt line through IRQOUTB1 into the PIC. 10 Route an enabled interrupt line through IRQOUTB2 into the PIC. 11 Route an enabled interrupt line through IRQOUTB3 into the PIC.

LICBICR1 LIC Group B Interrupt Configuration Register 1

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	EM23		IMAP23		EM22		IMAP22		EM21		IMAP21		EM20		IMAP20	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	EM19		IMAP19		EM18		IMAP18		EM17		IMAP17		EM16		IMAP16	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-19. LICBICR1 Bit Descriptions

Name	Reset	Description	Settings
EM[23–16] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 23–16.	00 Level Mode. The corresponding interrupt status bit in LICBISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second edge status bit in LICBIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in the LICBISR is set, the interrupt is captured by the corresponding bit in the LICBIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[23–16] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 23–16.	00 Route an enabled interrupt line through IRQOUTB0 into the PIC. 01 Route an enabled interrupt line through IRQOUTB1 into the PIC. 10 Route an enabled interrupt line through IRQOUTB2 into the PIC. 11 Route an enabled interrupt line through IRQOUTB3 into the PIC.

LICBICR2 LIC Group B Interrupt Configuration Register 2

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM15		IMAP15		EM14		IMAP14		EM13		IMAP13		EM12		IMAP12	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	EM11		IMAP11		EM10		IMAP10		EM9		IMAP9		EM8		IMAP8	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Table 17-20. LICBICR2 Bit Descriptions

Name	Reset	Description	Settings
EM[15–8] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 15–8.	00 Level Mode. The corresponding interrupt status bit in LICBISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICBIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICBISR is set, the interrupt is captured by the corresponding bit in the LICBIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[15–8] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 15–8.	00 Route an enabled interrupt line through IRQOUTB0 into the PIC. 01 Route an enabled interrupt line through IRQOUTB1 into the PIC. 10 Route an enabled interrupt line through IRQOUTB2 into the PIC. 11 Route an enabled interrupt line through IRQOUTB3 into the PIC.

LICBICR3 LIC Group B Interrupt Configuration Register 3

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EM7		IMAP7		EM6		IMAP6		EM5		IMAP5		EM4		IMAP4	
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
	EM3		IMAP3		EM2		IMAP2		EM1		IMAP1		EM0		IMAP0	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-21. LICBICR3 Bit Descriptions

Name	Reset	Description	Settings
EM[7–0] 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	0	Edge mode selection for interrupt source 7–0.	00 Level Mode. The corresponding interrupt status bit in LICBISR continuously reflects the interrupt source (read only). 01 Single Edge Mode. The interrupt second edge is ignored, and the corresponding second-edge status bit in LICBIESR is constantly cleared. 10 Second Edge Detection Mode. When the interrupt generates a second edge while the corresponding bit in LICBISR is set, the interrupt is captured by the corresponding bit in the LICBIESR and the LICSEIRQ global interrupt line is asserted towards the PIC. 11 Reserved.
IMAP[7–0] 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	0	Map selection of interrupt source 7–0.	00 Route an enabled interrupt line through IRQOUTB0 into the PIC. 01 Route an enabled interrupt line through IRQOUTB1 into the PIC. 10 Route an enabled interrupt line through IRQOUTB2 into the PIC. 11 Route an enabled interrupt line through IRQOUTB3 into the PIC.

17.3.2.2 LIC Interrupt Enable Registers

LICAIER LIC Group A Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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
	E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LICBIER LIC Group B Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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
	E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LICAIER and LICBIER enable/disable assertion of group A and B LIC interrupts at the appropriate PIC inputs.

Table 17-22. LICAIER, LICBIER Bit Descriptions

Name	Reset	Description	Settings
E[31–0] 31–0	0	**Enable/Disable Interrupt Source 31–0** When the individual bit is cleared, the corresponding status bit in LICAISR or LICBISR can be polled in level mode (read only) or edge mode. The second edge status bit is constantly cleared, to eliminate a second edge error interrupt.	0 Disable interrupt. Interrupt line does not pass an interrupt to the selected PIC input. 1 Enable interrupt. The interrupt source is routed by the IMAP field to the appropriate PIC interrupt input.

17.3.2.3 LIC Interrupt Status Registers

LICAISR LIC Group A Interrupt Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
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**
	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LICBISR LIC Group B Interrupt Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16
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**
	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the interrupt status registers corresponds to one interrupt source. In level mode, the status bit continuously reflects the synchronized value of the interrupt source, and it is read-only. In edge mode the status bit captures the active edge of the interrupt source, and can be cleared by writing one to it. Writing zero has no effect. When an interrupt source is configured to second edge detection mode, and the second edge appears while the status bit in LICAISR or LICBISR is set, the appropriate second edge error status bit is set in LICAIESR or LICBIESR, and a second edge interrupt is asserted at the PIC input.

Note: If a status bit in LICAIESR or LICBIESR is set, it blocks the reassertion of the corresponding status bit in LICAISR or LICBISR in second-edge mode.

LICAIESR LIC Group A Interrupt Error Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ES31	ES30	ES29	ES28	ES27	ES26	ES25	ES24	ES23	ES22	ES21	ES20	ES19	ES18	ES17	ES16
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
	ES15	ES14	ES13	ES12	ES11	ES10	ES9	ES8	ES7	ES6	ES5	ES4	ES3	ES2	ES1	ES0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LICBIESR LIC Group B Interrupt Error Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ES31	ES30	ES29	ES28	ES27	ES26	ES25	ES24	ES23	ES22	ES21	ES20	ES19	ES18	ES17	ES16
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
	ES15	ES14	ES13	ES12	ES11	ES10	ES9	ES8	ES7	ES6	ES5	ES4	ES3	ES2	ES1	ES0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the interrupt error status registers corresponds to one interrupt source. When the interrupt is disabled or not configured to second edge mode, the second edge status bit is constantly reset. When the interrupt source is enabled and configured to second edge mode, if the primary status bit is set while the second active edge is detected, the second edge error status bit is set. It can be cleared by writing one to it, while writing zero has no effect. When at least one status bit is set in LICAIESR or LICBIESR, a global LIC second edge error $\overline{\text{IRQ}}$ (LICSEIRQ) is asserted at PIC $\overline{\text{IRQ23}}$ input.

Note: If a status bit in LICAIESR or LICBIESR is set, it blocks the reassertion of the corresponding status bit in LICAISR or LICBISR in second-edge mode.

17.3.3 PIC Registers

The PIC registers have the following functional types:

- Six edge-triggered/level-triggered interrupt priority registers (ELIRA through ELIRF).
- Two interrupt pending registers (IPRA and IPRB).

Each PIC resides only in its associated core memory space, and all PIC blocks are accessible to their associated SC140 cores on the same addresses.

17.3.3.1 Edge/Level-Triggered Interrupt Priority Registers

The six ELIRs are 16-bit read/write registers by which the SC140 core determines the interrupt priority level (IPL) and trigger mode of the interrupt requests received at each of the 24 maskable PIC inputs. These registers are software programmable. Each of the six edge-triggered/level-triggered interrupt priority registers, ELIRA through ELIRF, defines a bank of four maskable IR inputs, as shown in **Table 17-23**.

Table 17-23. PIC Edge/Level-Triggered Interrupt Priority Registers

Register Name	Description	Bank	IR Inputs
ELIRA	PIC Edge/Level-Triggered Interrupt Priority Register A	A	0–3
ELIRB	PIC Edge/Level-Triggered Interrupt Priority Register B	B	4–7
ELIRC	PIC Edge/Level-Triggered Interrupt Priority Register C	C	8–11
ELIRD	PIC Edge/Level-Triggered Interrupt Priority Register D	D	12–15
ELIRE	PIC Edge/Level-Triggered Interrupt Priority Register E	E	16–19
ELIRF	PIC Edge/Level-Triggered Interrupt Priority Register F	F	20–23

Each register defines the interrupt trigger mode and IPL for four inputs. For each input, three bits define the priority level, and one bit specifies the trigger mode for the interrupt.

17.3.3.2 Interrupt Priority Structure and Mode

Eight of the 32 PIC inputs are $\overline{\text{NMI}}$s that cannot be programmed. The $\overline{\text{NMI}}$s are always assigned the highest priority, regardless of their source. Each of the remaining 24 inputs can be programmed to one of seven maskable priority levels, IPL 0 through IPL 6, with a corresponding numeric value of 1 through 7. The highest maskable priority is IPL 6. **Table 17-24** lists the possible settings for the three interrupt priority level bits, with their corresponding value and IPL. A value of zero in these three bits indicates that interrupts are disabled on this input.

Table 17-24. Interrupt Priority Level Bits

PILxx0	PILxx1	PILxx2	Enabled	Value	IPL
0	0	0	No	0	—
0	0	1	Yes	1	0
0	1	0	Yes	2	1
0	1	1	Yes	3	2
1	0	0	Yes	4	3

Table 17-24. Interrupt Priority Level Bits (Continued)

PILxx0	PILxx1	PILxx2	Enabled	Value	IPL
1	0	1	Yes	5	4
1	1	0	Yes	6	5
1	1	1	Yes	7	6

The PIC supports both edge-triggered and level-triggered interrupt requests. $\overline{\text{NMI}}$s are always edge-triggered. The mode for the 24 programmable inputs can be defined as either edge-triggered or level-triggered.

Note: Unless specified otherwise in **Table 17-8**, *MSC8113 Interrupt Routing*, on page 17-19, all maskable PIC interrupt sources should be programmed as level-triggered.

Table 17-25 shows the settings for the interrupt trigger mode bit.

Table 17-25. Interrupt Trigger Mode Bit

PEDxx	Trigger Mode
0	Negative Level-Triggered
1	Negative Edge-Triggered

ELIRA Edge/Level-Triggered Interrupt Priority Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED3	PIL30	PIL31	PIL32	PED2	PIL20	PIL21	PIL22	PED1	PIL10	PIL11	PIL12	PED0	PIL00	PIL01	PIL02
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ELIRA defines the trigger mode and IPL for IR inputs 0 through 3.

ELIRB Edge/Level-Triggered Interrupt Priority Register B

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED7	PIL70	PIL71	PIL72	PED6	PIL60	PIL61	PIL62	PED5	PIL50	PIL51	PIL52	PED4	PIL40	PIL41	PIL42
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ELIRB defines the trigger mode and IPL for IR inputs 4 through 7.

ELIRC Edge/Level-Triggered Interrupt Priority Register C

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED11	PIL110	PIL111	PIL112	PED10	PIL100	PIL101	PIL102	PED9	PIL 90	PIL91	PIL 92	PED8	PIL80	PIL81	PIL82
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ELIRC defines the trigger mode and IPL for IR inputs 8 through 11.

ELIRD Edge/Level-Triggered Interrupt Priority Register D

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED15	PIL150	PIL151	PIL152	PED14	PIL140	PIL141	PIL142	PED13	PIL130	PIL131	PIL132	PED12	PIL120	PIL121	PIL122
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ELIRD defines the trigger mode and IPL for IR inputs 12 through 15.

ELIRE Edge/Level-Triggered Interrupt Priority Register E

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED19	PIL190	PIL191	PIL192	PED18	PIL180	PIL181	PIL182	PED17	PIL170	PIL171	PIL172	PED16	PIL160	PIL161	PIL162
Type	R/W															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ELIRE defines the trigger mode and IPL for IR inputs 16 through 19.

ELIRF Edge/Level-Triggered Interrupt Priority Register F

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PED23	PIL230	PIL231	PIL232	PED22	PIL220	PIL221	PIL222	PED21	PIL210	PIL211	PIL212	PED20	PIL200	PIL201	PIL202
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

ELIRF defines the trigger mode and IPL for IR inputs 20 through 23.

Table 17-26. ELIRA–ELIRF Bit Descriptions

Name	Reset	Description	Settings
PED xx 0, 4, 8, 12	0	**Trigger Mode for IR Input xx** Defines whether the interrupt is in edge-triggered or level-triggered mode.	0 Level-triggered mode. 1 Edge-triggered mode.
PILxx0–PILxx2 1–3, 5–7, 9–11, 13–15	0	**Priority Level for IR Input xx** Defines the interrupt priority level (IPL). IPL 6 is the highest priority. If the value is zero, interrupts are disabled on this input.	000 Interrupts disabled. 001 IPL 0 (lowest priority). 010 IPL 1. 011 IPL 2. 100 IPL 3. 101 IPL 4. 110 IPL 5. 111 IPL 6 (highest priority).

The MSC8113 boot procedure configures ELIRF to the appropriate trigger mode and sets the corresponding $\overline{\text{IRQ}}$ signals. The value of ELIRF after boot is 0x0008, which sets $\overline{\text{IRQ20}}$ (EOnCE interrupt) to edge-triggered mode. For a summary of the routing of MSC8113 interrupts, refer to **Table 17-8**.

17.3.3.3 Interrupt Pending Registers

The PIC interrupt pending registers, IPRA and IPRB, are 16-bit read/write registers that the SC140 core uses to:

- Monitor pending interrupts.
- Reset edge-triggered interrupts.

Reading the two interrupt pending registers, you can view the status of all current $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ signals. Each bit in the registers represents one of the 32 inputs. If an $\overline{\text{IRQ}}$ is configured as level-triggered, its corresponding interrupt pending (IP) bit reflects the status of the $\overline{\text{IRQ}}$ signal. The IP bit is set if at least one $\overline{\text{IRQ}}$ is pending and reset if there are no $\overline{\text{IRQ}}$ signals pending. When the corresponding $\overline{\text{IRQ}}$ is configured as edge-triggered, its IP bit is set for every new negative edge detected on the $\overline{\text{IRQ}}$. A value of 1 written to the IP bit indicates that the corresponding $\overline{\text{IRQ}}$ is acknowledged. This feature is used for both $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$ signals to indicate to the PIC that the SC140 core has acknowledged the corresponding edge-triggered interrupt source and that the PIC should ignore any request from the corresponding interrupt source until its next negative edge. Each bit in the interrupt pending registers corresponds to an interrupt source, as shown in **Table 17-27**. PIC Interrupt Pending Register A (IPRA) defines the status of the first 16 programmable $\overline{\text{IRQ}}$ signals, while PIC Interrupt Pending Register B (IPRB) defines the remaining eight programmable $\overline{\text{IRQ}}$ and the eight $\overline{\text{NMI}}$ signals.

Table 17-27. PIC Interrupt Pending Registers

Register Name	Description	Bank	Inputs	IR/NMI
IPRA	PIC Interrupt Pending Register A	A	0–15	IRs
IPRB	PIC Interrupt Pending Register B	B	16–23 24–31	IRs NMIs

IPRA PIC Interrupt Pending Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	IP15	IP14	IP13	IP12	IP11	IP10	IP9	IP8	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPRA reflects the status for IR inputs 0 through 15.

Table 17-28. IPRA Bit Descriptions

Name	Reset	Description	Settings
IP15 0	0	**Status of IR Input 15** The trigger mode of the interrupt, level-triggered or edge-triggered, determines the meaning of the status when its value is set.	Level-triggered mode: 0 No IR pending. 1 IR pending. Edge-triggered mode: 0 No IR pending. 1 IR acknowledged by the SC140 core. The PIC ignores any request from the interrupt source for this input until its next negative edge.
IP14–0 1–15	0	**Status of IR input 14–0** The description and settings are the same as IP15 for IR inputs 14–0.	

IPRB PIC Interrupt Pending Register B (IPRB)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	IP31	IP30	IP29	IP28	IP27	IP26	IP25	IP24	IP23	IP22	IP21	IP20	IP19	IP18	IP17	IP16
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPRB reflects the status for $\overline{\text{IRQ}}$ inputs 16 through 23 and $\overline{\text{NMI}}$ inputs 24 through 31.

Table 17-29. IPRB Bit Descriptions

Name	Reset	Description	Settings
IP31 0	0	**Status of $\overline{\text{NMI}}$ Input 31** (edge-triggered only). When this bit is set, The PIC ignores any request from the interrupt source for this input until its next negative edge.	0 No $\overline{\text{NMI}}$ pending. 1 $\overline{\text{NMI}}$ acknowledged by the SC140 core.
IP30–24 1–7	0	**Status of $\overline{\text{NMI}}$ Input 30–24** The settings are the same as IP31 for inputs 30–24.	
IP23 8	0	**Status of IR Input 23** The trigger mode of the interrupt, level-triggered or edge-triggered, determines the meaning of the status when its value is set.	Level-triggered mode: 0 No IR pending. 1 IR pending. Edge-triggered mode: 0 No IR pending. 1 IR acknowledged by the SC140 core. The PIC ignores any request from the interrupt source for this input until its next negative edge.
IP22–16 9–15	0	**Status of IR input 22–16** The description and settings are the same as IP23 for inputs 22–16.	

VBA Vector Base Address Register

VBA, which resides in the SC140 core, is an important part of the interrupt programming model. The VBA register allows you to determine the base address for the interrupt vector table by writing it to the VBA Register. At reset the value of the 20-bit wide VBA Register is set to zero. The offset for each exception vector is predefined. There are 64 possible exception vector locations. The spacing between two exception vectors is 32 words (four full execution sets).

Debugging 18

The dedicated user-accessible test access port (TAP) is designed to be compatible with the **IEEE** Std. 1149.1 test access port and boundary scan architecture. Problems associated with testing high-density circuit boards led to development of this standard under the sponsorship of the test technology committee of **IEEE** and the joint test action group (JTAG). The MSC8113 supports circuit-board test strategies based on this standard. This chapter covers aspects of JTAG that are specific to the MSC8113. It includes the items that the standard requires to be defined, with additional information specific to the MSC8113 device. For details on the standard, refer to the **IEEE** Std. 1149.1 documentation.

The JTAG port also provides access to the Enhanced On-Chip Emulator (EOnCE) module, a dedicated block for debugging applications. Therefore, this chapter includes information on registers and functionality of the EOnCE module that are specific to the MSC8113. For details on the EOnCE module functionality, see the *SC140 DSP Core Reference Manual*.

The SC140 core EOnCE module interfaces with the SC140 core and its peripherals non-intrusively so that you can examine registers, memory, or on-device peripherals, thus facilitating hardware and software development on the SC140 core-based devices. Special circuits and dedicated signals on the SC140 core avoid sacrificing user-accessible internal resource. As the DSP applications grow in both size and complexity, the EOnCE module provides many features of the breakpoints, conditional breakpoints, breakpoints on data-bus values, and event detection that offer the user non-destructive access to peripherals, variety in profiling, a program tracing buffer, and real-time access to memory.

18.1 Overview

The MSC8113 TAP consists of five dedicated signal lines, a 16-state TAP controller, and three test data registers. A Boundary Scan Register (BSR) links most of the device signal connections into a single shift register. The test logic, which uses static logic design, is independent of the device system logic. The MSC8113 JTAG can do the following:

- Perform boundary scan operations to check circuit-board electrical continuity.
- Bypass the MSC8113 for a given circuit-board test by effectively reducing the Boundary Scan Register (BSR) to a single cell.
- Sample the MSC8113 system connections during operation and transparently shift out the result in the BSR. Preload values to outputs prior to circuit board testing.
- Disable the drive to outputs during circuit board testing.
- Access the EOnCE controller and circuits to control a target system.
- Give entry to Debug mode.
- Query identification information (manufacturer, part number and version) from an MSC8113-based device.
- Force test data onto the outputs of an MSC8113-based device while replacing its BSR in the serial data path with a single-bit register.

Note: Precautions must be taken to ensure that the **IEEE** Std. 1149.1-like test logic does not interfere with non-test operation.

To access the JTAG registers, shift the appropriate command into the JTAG instruction register and then shift the required value into the register. See **Section 18.3** for a discussion of the JTAG instructions. **Figure 18-1** shows the MSC8113 JTAG 5-bit instruction register and the following test registers:

- Boundary Scan Register (BSR). Regarding the length of the BSR, The boundary scan bit definitions vary according to the specific chip implementation of the MSC8113 and are described by the BSDL file on the product website
- 1-bit Bypass Register
- 32-bit Identification Register (ID)
- 32-bit General Purpose Register (GPR)
- 32-bit Parallel Input Register (PIREG)

Table 18-1 lists the test access port (TAP) signals.

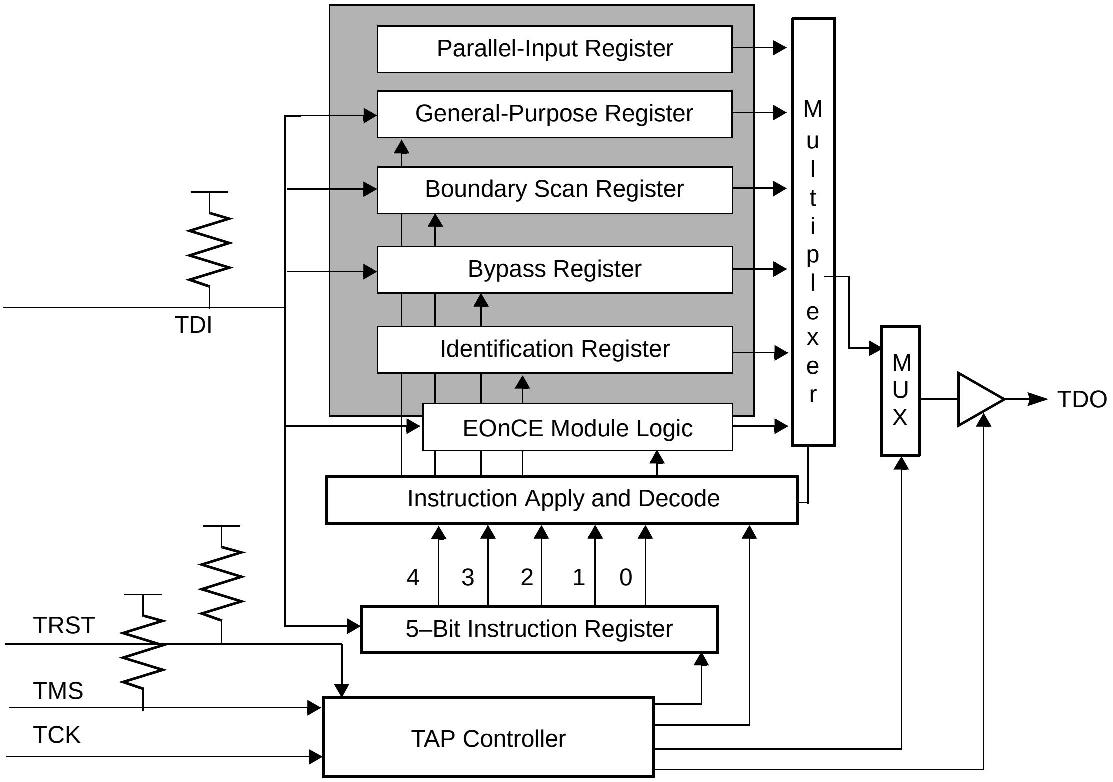


Figure 18-1. Test Logic Block Diagram

Table 18-1. TAP Signals

Signal	Description
TCK	A test clock input to synchronize the test logic.
TMS	A test mode select input (with an internal pull-up resistor) that is sampled on the rising edge of TCK to sequence the TAP controller state machine.
TDI	A test data input (with an internal pull-up resistor) that is sampled on the rising edge of TCK.
TDO	A data output that can be tri-stated and actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO changes on the falling edge of TCK.
TRST	An asynchronous reset (with an internal pull-up resistor) that provides initialization of the TAP controller and other logic required by the standard.

18.2 TAP Controller

The TAP controller interprets the sequence of logical values on the TMS signal. This synchronous state machine controls the operation of the JTAG logic. The value adjacent to each arc in **Figure 18-2** represents the value of the TMS signal sampled on the rising edge of the TCK signal. For a description of the TAP controller states, refer to the **IEEE** Std. 1149.1 documentation.

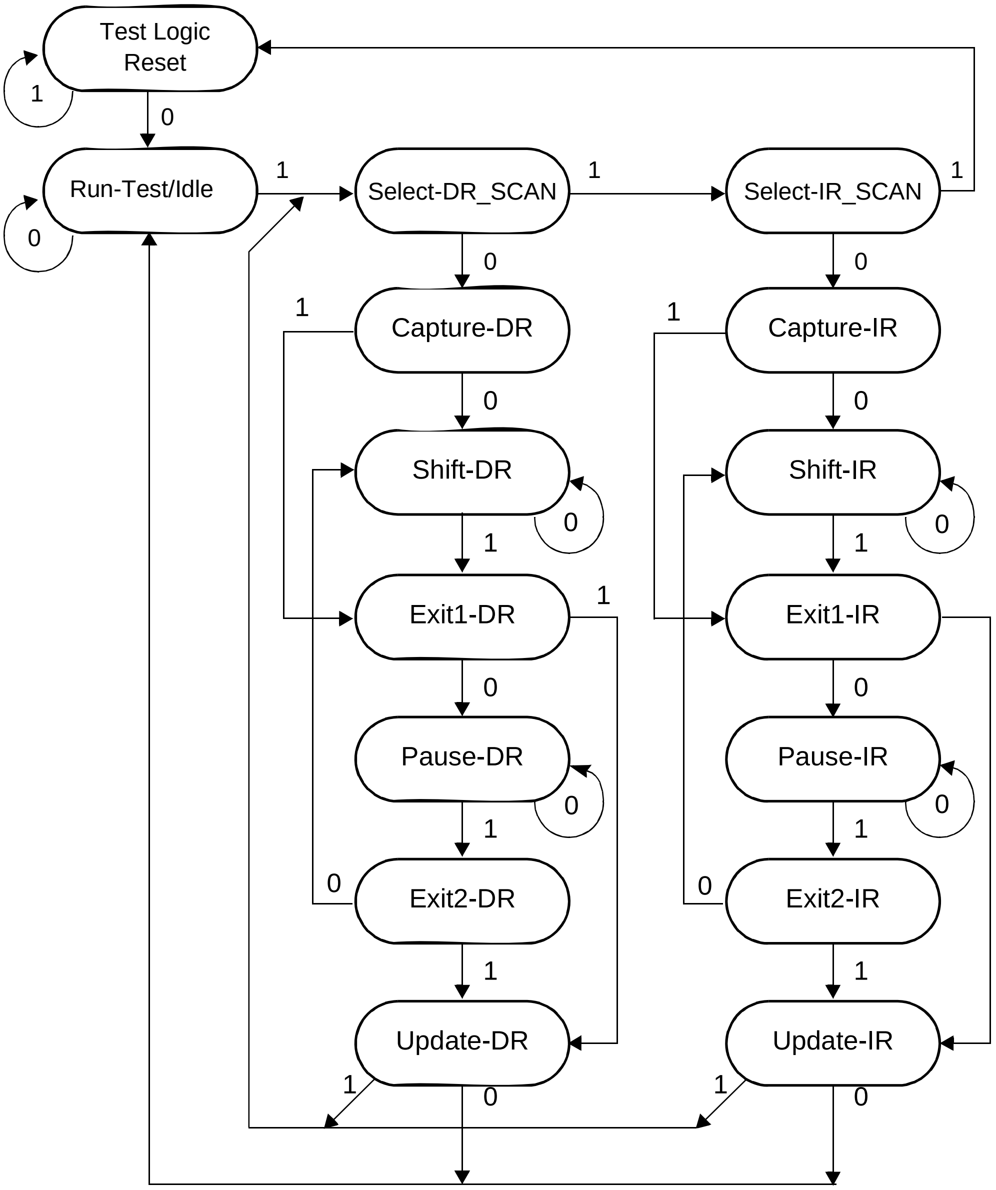


Figure 18-2. TAP Controller State Machine

18.3 Instruction Decoding

The MSC8113 includes the three mandatory public instructions EXTEST, SAMPLE/PRELOAD, and BYPASS and also supports the optional CLAMP and HIGHZ instructions defined by **IEEE Std.** 1149.1. The following public instructions perform key functions:

- ENABLE_EONCE enables the JTAG port to communicate with the EOnCE circuitry.
- DEBUG_REQUEST enables the JTAG port to force the MSC8113 into Debug mode.
- CHOOSE_EONCE allows the operation of multiple EOnCE devices. This instruction should always execute before the first ENABLE_EONCE instruction and should shift a 1 to the SC140 EOnCE module choose cells for each module that you want to enable. Since there are three internal EOnCE modules, you must shift 4 bits to the choose cells. For details, see **Section 18.4**.

The MSC8113 includes a 5-bit instruction register without parity, consisting of a shift register with five parallel outputs. Data is transferred from the shift register to the parallel outputs during the UPDATE-IR controller state. The five bits decode the ten unique instructions listed in **Table 18-3*Instruction Decoding,*** on page 18>-6. All other encoding, with the exception of the manufacturer’s private instructions, is reserved for future enhancements and is decoded as BYPASS.

The parallel output of the Instruction Register is reset to 0b00010 in the test-logic-reset controller state, which is equivalent to the IDCODE instruction. During the CAPTURE-IR controller state, the parallel inputs to the instruction shift register are loaded with the code 01 in the least significant bits, as required by the standard. The most significant bits are loaded with the values upd_ack, cores1, cores0, as shown in **Table 18-2** and **Figure 18-3**. Two bits of the GPR are configured to select an SC140 core, whose status is output from the multiplexer. Therefore, the status of all SC140 cores can be viewed serially by updating the GPR between each SC140 core status reading. Alternatively, all three SC140 cores can be viewed simultaneously from the PIREG. For details on core states, refer to the *SC140 DSP Core Reference Manual*.

Table 18-2. Instruction Register Capture and SC140 Core Status Values

Name/bits	Description	Settings
upd_ack 4	**Update Acknowledge** Indicates whether the selected SC140 EOnCE module has executed the last instruction dispatched to it	0 EOnCE module has executed the last instruction dispatched to it. 1 EOnCE module has not executed the last instruction dispatched to it.
cores[1–0] 3–2	**Core Status** Reflects the status of the selected SC140 core	00 Core is executing instructions. 01 Core is in WAIT or STOP mode. 10 Core is waiting for bus. 11 Core is in debug mode.
— 1–0	Contains value required by the JTAG standard	Read-only

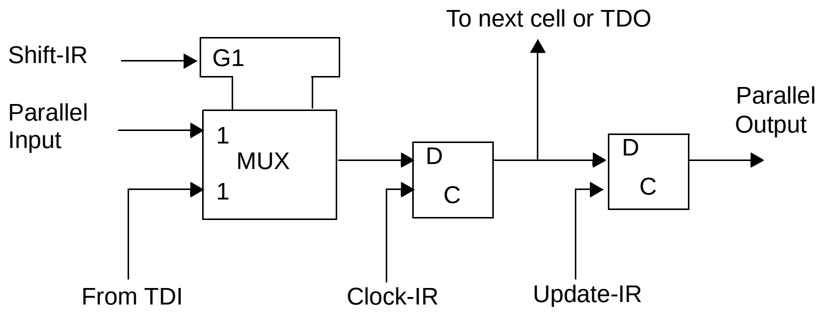


Figure 18-3. Instruction Register (IR) Configuration

Table 18-3 describes the 5-bit instructions coded in the Instruction Register.

Table 18-3. Instruction Decoding

Bits 4-0	Instruction	Description
00000	EXTEST	Selects the Boundary Scan Register (BSR). EXTEST also asserts internal reset for the MSC8113 system logic to force a predictable internal state while external boundary scan operations are performed. By using the TAP, the register can: • Scan user-defined values into the output buffers • Capture values presented to inputs • Control the direction of bidirectional signals • Control the output drive of tri-statable outputs For details on the function and use of EXTEST, refer to the **IEEE** Std. 1149.1 documentation.
00001	SAMPLE/PRELOAD	Initializes the BSR output cells prior to the selection of EXTEST. This initialization ensures that known data appears on the outputs when an EXTEST instruction is entered. SAMPLE/PRELOAD also provides a means to obtain a snapshot of system data and control signals. **Note:** Since there is no internal synchronization between the TCK and CLKOUT, to achieve meaningful results, you must provide some form of external synchronization between the JTAG operation at TCK frequency and the system operation CLKOUT frequency.
00010	IDCODE	Selects the ID Register. This instruction is a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP. The ID Register configuration is as follows: • Bits 31–28: Version Information • Bits 27–12: Customer Part Number • Bits 11–1: Manufacturer Identity One application of the ID Register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge that conform to the **IEEE** Std. 1149.1, it is desirable to allow for a system diagnostic controller unit to blindly interrogate a board design and determine the type of each component in each location. This information is also available for factory process monitoring and for failure mode analysis of assembled boards. The manufacturer identity number is 0b00000001110. The customer part number consists of two parts: design center number (bits 27–22) and a sequence number (bits 21–12). The design center number is 0b000110. Once the IDCODE instruction is decoded, it selects the ID Register, which is a 32-bit data register. The Bypass Register loads a logic at the start of a scan cycle, whereas the ID Register loads a logic 1 into its least significant bit. Consequently, examination of the first bit of data shifted out of a component during a test data scan sequence, immediately following exit from test-logic-reset controller state, shows whether such a register is included in the design. As required by the **IEEE** Std. 1149.1, the operation of the test logic has no effect on the operation of the internal system logic when the IDCODE instruction is selected.

Table 18-3. Instruction Decoding (Continued)

Bits 4-0	Instruction	Description
00011	CLAMP	Optional in the **IEEE** Std. 1149.1. This public instruction selects the one-bit Bypass Register as the serial path between TDI and TDO, while allowing signals driven from the component to be determined from the Boundary Scan Register. During testing of ICs on PCBs, it may be necessary to place static guarding values on signals that control logic operations not involved in the test. The EXTEST instruction could be used for this purpose, but since it selects the BSR, the required guarding signals would be loaded as part of the complete serial data stream shifted in, both at the start of the test and each time a new test pattern is entered. Since the CLAMP instruction allows guarding values to be applied using the BSR of the appropriate ICs while selecting their Bypass Registers, it allows much faster testing than EXTEST. Data in the boundary scan cell remains unchanged until a new instruction is shifted in. **Note:** The CLAMP instruction also asserts internal reset for the MSC8113 system logic to force a predictable internal state while external boundary scan operations are performed.
00100	HIGHZ	Optional in the **IEEE** Std. 1149.1. It is a manufacturer's public instruction to prevent back-drive of the outputs during circuit-board testing. When HIGHZ is invoked, all output drivers, including the two-state drivers, are turned off (that is, high impedance). The HIGHZ instruction selects the Bypass Register. It also asserts internal reset for the MSC8113 system logic to force a predictable internal state while external boundary scan operations are performed.
00101	—	Reserved
00110	ENABLE_EONCE	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to perform system debug functions. When the ENABLE_EONCE instruction is decoded, TDI and TDO connect directly to the EOnCE registers. The EOnCE controller selects the specific EOnCE register connected between TDI and TDO, depending on the EOnCE instruction being executed. All communication with the EOnCE controller is through the SELECT-DR-SCAN path of the JTAG TAP Controller. Before the ENABLE_EONCE instruction is selected, the CHOOSE_EONCE instruction should be executed to define which EOnCE is to be activated. **Note:** This instruction is valid only if the core processor is running.
00111	DEBUG_REQUEST	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to generate a debug request signal to the MSC8113. When the DEBUG_REQUEST instruction is decoded, TDI and TDO connect to the EOnCE registers. In addition, ENABLE_EONCE is active and forced to request Debug mode from the MSC8113, in order to perform system debug functions. Before the DEBUG_REQUEST instruction is selected, the CHOOSE_EONCE instruction should be executed to define which EOnCE is to be selected for DEBUG_REQUEST. **Note:** Issuing this instruction does not ensure that the SC140 core enters the debug state. Monitor the core status to make sure that it has stopped.
01000	PRIVATE	Manufacturer's private instruction. **Note:** Selecting this instruction many cause *unpredictable* operation of the device.
01001	CHOOSE_EONCE	Not included in the **IEEE** Std. 1149.1. This instruction enables selected SC140 EOnCE modules. All instructions executed after this one target only the selected EOnCE set. Therefore, this instruction always executes, regardless of the selected EOnCE set.
01010	—	Reserved
01011	—	Reserved
01100	PRIVATE	Manufacturer's private instruction. **Note:** Selecting this instruction many cause *unpredictable* operation of the device.
01101	LOAD_GPR	Not included in the **IEEE** Std. 1149.1: LOAD GPR When programming the GPR, use only the bits permitted in **Table 18-6**.
01110	PRIVATE	Manufacturer's private instruction. **Note:** Selecting this instruction many cause *unpredictable* operation of the device.
01111	—	Reserved

Table 18-3. Instruction Decoding (Continued)

Bits 4-0	Instruction	Description
...	...	Reserved
11100	---	Reserved
11101	READ_PIREG	Not included in the **IEEE** Std. 1149.1: read Parallel Input Register (PIREG). **Note:** Use only the bits specified in **Table 18-7**. Other bits should be disregarded.
11110	PRIVATE	Manufacturer's private instruction. **Note:** Selecting this instruction many cause *unpredictable* operation of the device.
11111	BYPASS	Selects the single-bit Bypass Register. This creates a shift-register path from TDI to the Bypass Register and, finally, to TDO, circumventing the 573-bit BSR register. This instruction enhances test efficiency when a component other than the MSC8113-based device is the device under test. When the current instruction selects the Bypass Register, the shift-register stage is set to a logic zero on the rising edge of TCK in the CAPTURE-DR controller state. Therefore, the first bit to be shifted out after the Bypass Register is selected is always a logic zero.

18.4 Multi-Core JTAG and EOnCE Module Concept

The MSC8113 uses JTAG TAP for standard defined testing compatibilities and for multi-core EOnCE module control and EOnCE module interconnection control. The MSC8113 has *three* internal EOnCE modules, one module per SC140 core. The EOnCE modules interconnect in a chain and are configured and directed by the JTAG TAP controller. **Figure 18-4** shows the chained connection.

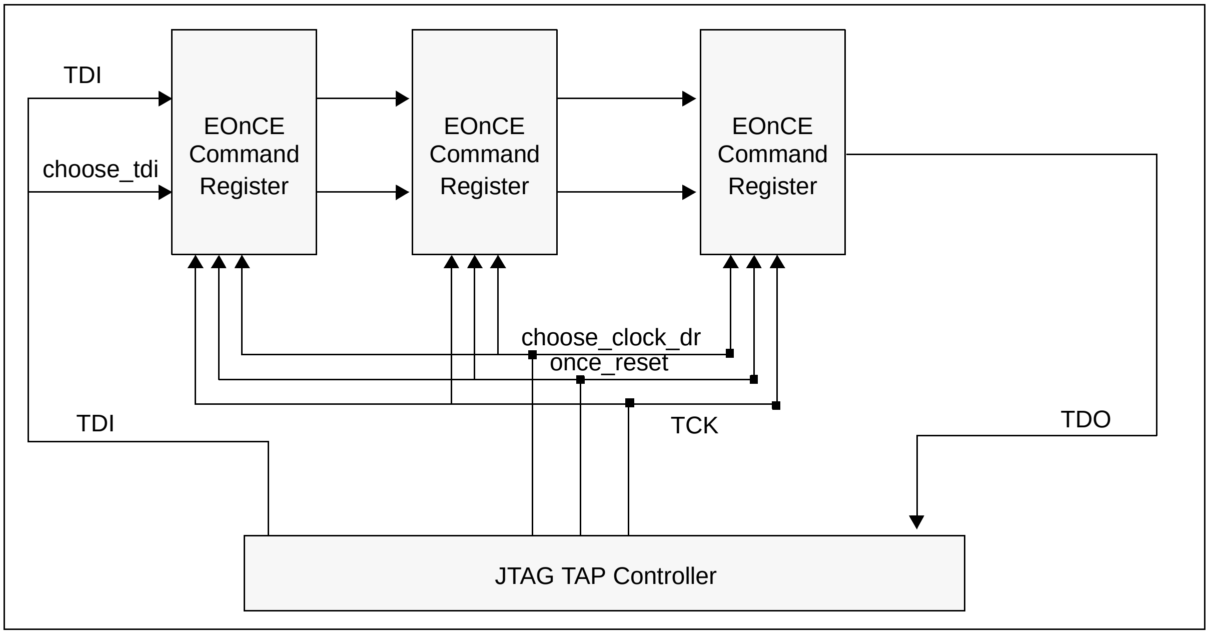


Figure 18-4. JTAG TAP Controller and EOnCE Module Multi-Core Interconnection

Each of the *three* MSC8113 EOnCE modules has an interface to a JTAG port. The interface is active even when a reset signal to the SC140 core is asserted. However, system reset must be deasserted to allow a proper interface with the cores. This interface is synchronized with the internal clocks derived from the JTAG TCK clock. Each EOnCE module includes an EOnCE controller, an event counter (used by the MSC8113 ICache to count hits/misses in cache; see **Section 9.4.1**), an event detector unit, a synchronizer, an event selector, and a trace unit.

Note: For details on the EOnCE module features, consult the *SC140 DSP Core Reference Manual*.

The JTAG port performs the following tasks via the JTAG-EOnCE module interface:

- Chooses one or more EOnCE module blocks (CHOOSE_EOnCE instruction)
- Issues a debug request to the EOnCE module (DEBUG_REQUEST instruction)
- Writes an EOnCE command to the EOnCE Command Register (DEBUG_REQUEST or ENABLE_EOnCE instruction)
- Reads and writes to internal EOnCE registers (DEBUG_REQUEST or ENABLE_EOnCE instruction)

18.4.1 Enabling the EOnCE Module

The CHOOSE_EOnCE mechanism integrates multiple cores and thus multiple EOnCE modules on the same device. Using the CHOOSE_EOnCE instruction, you can selectively activate one or more of the EOnCE modules on the MSC8113. The EOnCE modules selected by the CHOOSE_EOnCE instruction are cascaded as shown in **Figure 18-5**. Only selected EOnCE modules respond to ENABLE_EOnCE and DEBUG_REQUEST instructions from the JTAG. All EOnCE modules are deselected after reset.

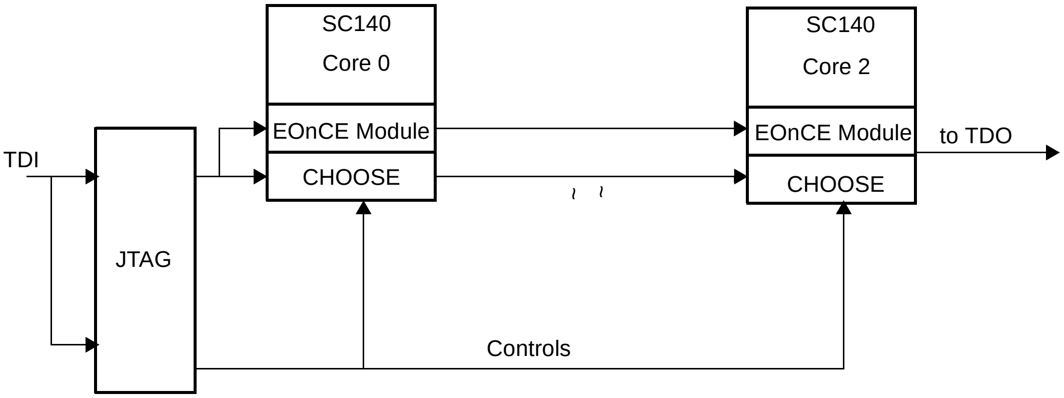


Figure 18-5. Cascading Multiple EOnCE Modules

Since all the EOnCE modules are cascaded, the selection procedure is performed serially. The sequence is:

1. Select the CHOOSE_EONCE instruction.
2. Enter at Shift_DR state the serial stream that specifies the modules to be selected (1 = selected, 0 = not selected).

The number of bits in this stream, that is, the number of clocks in this state, is equal to the number of selected SC140 cores in the cascade, which is *three*. This state is indicated by the CHOOSE_CLOCK_DR signal. For example, for the three SC140 cores on the MSC8113, to activate the third core in the cascade, which is the closest to TDO and the farthest from TDI, the data is 1,0,0,0 (first a one, then three zeros). If the data is 1,0,1,0, then both the second and the fourth cells are selected.

Only the EOnCE command register (ECR) should be accessed in cascaded mode. To do this, first enter the CHOOSE_EONCE instruction and set ENABLE_ONCE to 1 for all cores. Then shift in the cascaded value for all ECRs in series. When the shift is ended and the controller issues a SHIFT_UPDATE, all registers are updated in parallel. However, it is not guaranteed that this occurs in the same SC140 clock cycle for all cores.

Note: Accessing any other EOnCE register in cascaded mode may fail. Always select a single core at a time to access any other EOnCE register.

18.4.2 DEBUG_REQUEST and ENABLE_EOnCE Commands

After completing the CHOOSE_EOnCE instruction, you can execute DEBUG_REQUEST and ENABLE_EOnCE instructions. More than one such instruction can execute, and other instructions can be placed between them, as well as between them and the CHOOSE_EOnCE instruction. The EOnCE modules selected in the CHOOSE_EOnCE instruction remain selected until the next CHOOSE_EOnCE instruction. The DEBUG_REQUEST or ENABLE_EOnCE instruction is shifted in during the SHIFT-IR state, as are all JTAG instructions.

18.4.3 Reading/Writing EOnCE Registers Through JTAG

An external host can read or write almost every EOnCE register through the JTAG interface by performing the following steps:

1. Execute the CHOOSE_EOnCE command in the JTAG.
2. Send the data showing which EOnCE module is chosen. This command enables the JTAG to manage multiple EOnCE modules in a device.
3. Execute the ENABLE_EOnCE command in the JTAG.
4. Write the EOnCE command into the EOnCE Command register (ECR); that is, enter the JTAG TAP state machine into the SHIFT-DR state and then give the required command on the TDI input signal.

After the command is shifted in, the JTAG TAP state machine must enter the UPDATE-DR state. The data shifted via the TDI is sampled into the ECR. If, for example, the command written into the ECR is *Write EDCA0_CTRL*, then the host must again enter the JTAG into SHIFT-DR and shift the required data, which is to be written into the EDCA0_CTRL, via TDI. If the command is *read some register*, then the DR chain must be passed again and the contents of the register are shifted out through the TDO output signal. When JTAG shifts data to the EOnCE module, the lsb of the data is shifted first. See **Figure 18-6**.

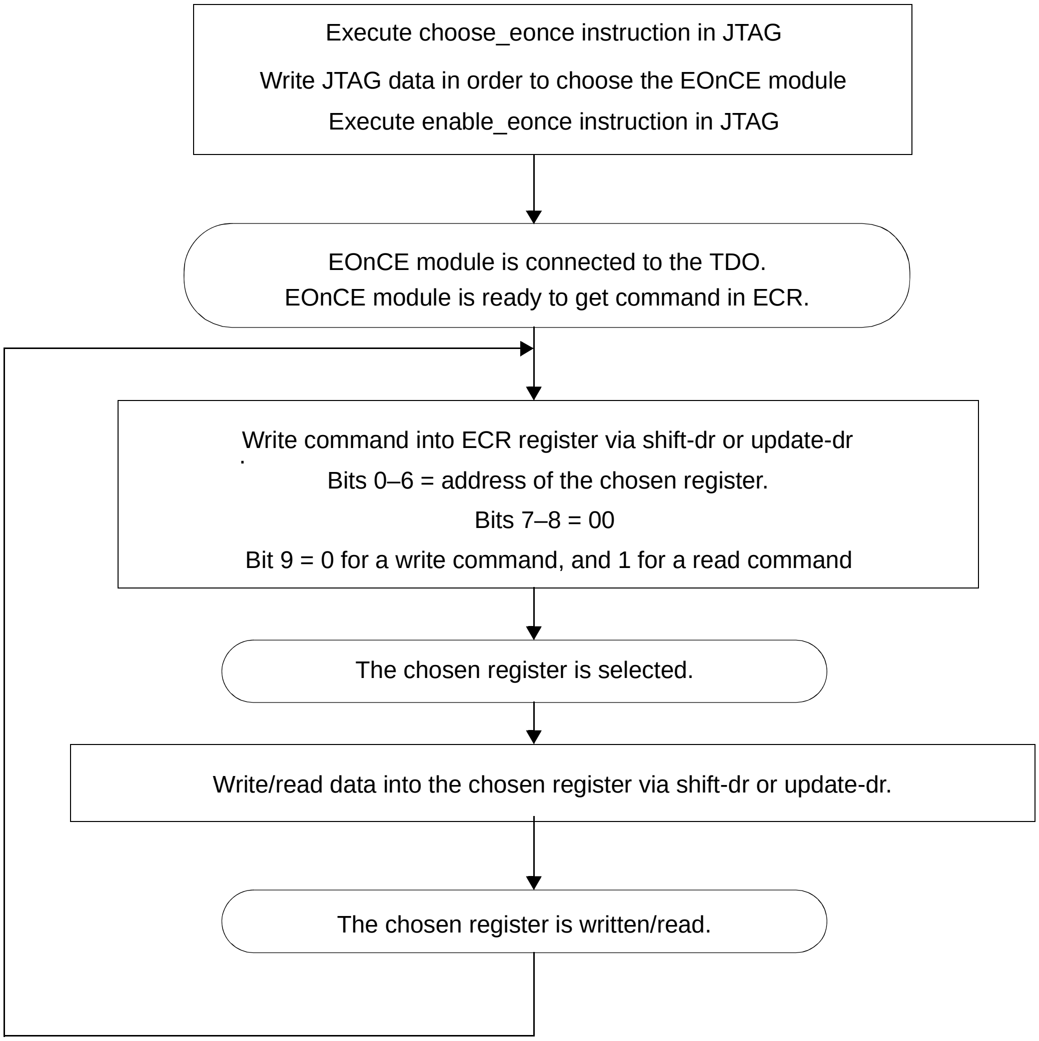

Execute choose_eonce instruction in JTAG
Write JTAG data in order to choose the EOnCE module
Execute enable_eonce instruction in JTAG
EOnCE module is connected to the TDO.
EOnCE module is ready to get command in ECR.
Write command into ECR register via shift-dr or update-dr
Bits 0–6 = address of the chosen register.
Bits 7–8 = 00
Bit 9 = 0 for a write command, and 1 for a read command
The chosen register is selected.
Write/read data into the chosen register via shift-dr or update-dr.
The chosen register is written/read.

Figure 18-6. Reading and Writing EOnCE Registers Via the JTAG TAP

18.5 Signalling a Debug Request

The EE[0–1] signals connect to each of the MSC8113 EOnCE modules. EE0 is an input that signals a debug request; EE1 is an output signal that acknowledges the request or acts as an output of event detector 1.

Note: Asserting EE0 does not guarantee that the cores enter debug mode.The signal can be masked internally. Monitor the core status by shifting out the contents of PIREG or by issuing an instruction and observing the TDO value shifted out.

18.5.1 EE_CTRL Modifications for the MSC8113

The relevant paragraph from the EOnCE module chapter of *SC140 DSP Core Reference Manual* is reproduced here with the appropriate amendments.

EE_CTRL EE Control Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—												EE1DEF		EE0DEF	
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Boot	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1

The modes for EE[0–1] are restricted as follows:

Table 18-4. EE0 Definition (EE0DEF), Bits 1–0

EE0DEF		EE0 Definition
0	0	Reserved
0	1	Reserved
1	0	Input
1	1	Input: Debug Request

The EE0DEF bits program EE0, either to enable Address Event Detection Channel 0 by providing an input to the EOnCE module in the event detection unit and the event selector to or to generate an EOnCE event. EE0 can be programmed to enter the SC140 core into Debug mode (the default) right after the SC140 core is reset. Holding EE0 at logic value 1 during and after the reset puts the SC140 core into Debug mode before the first dispatch occurs. In this mode, asserting EE0 also causes an exit from the STOP or WAIT processing states of the SC140 core. If you want some SC140 cores running and others in Debug mode, you must disable either the inputs of the running SC140 cores or the outputs of the stopped SC140 cores. To block EE0, set it as an input and mask the EE0 event in the event selector mask register (see the next section on programming the ESEL_DM Register).

Table 18-5. EE1 Definition (EE1DEF), Bits 3–2

EE1DEF		EE1 Definition
0	0	Output: Detection by EDCA1
0	1	Output: Debug Acknowledge
1	0	Reserved
1	1	Reserved

The EE1DEF bits program EE1. The signal can be programmed as an output of the EOnCE module to indicate detection by Address Event Detection Channel 1 (working as a toggle) or to indicate that the DSP has entered Debug mode (Debug Acknowledge). To disable EE1, EDCA1 must be disabled and the mode set to 00.

Note: If the boot code is not executed, you must initialize the EE1DEF bits to 01 (output debug acknowledge) to activate EE1 as debug acknowledge. The default value (00) does not activate EE1 as debug acknowledge. If the EE1DEF bits are not initialized correctly, EE1 as a core output will always be 0, meaning that no debug acknowledge is sent to the other cores (as a trigger to enter Debug mode) or to the EE1 output. Therefore, if the boot code is bypassed, it is the user’s responsibility to initialized EE1DEF correctly to use the debug acknowledge.

18.5.2 Event Selector Register Programming

There are four event selector registers. The Event Selector Mask Debug Mode (ESEL_DM) register in the EOnCE programs the event selectors for the debug events. The MSC8113 only supports the EE0 signals. Also, there is a requirement to block triggering from EE0 if only some SC140 cores must enter Debug mode.

ESEL_DM Event Selector Mask Debug Mode Register

	Bit 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	EDCA0	EDCA1	EDCA2	EDCA3	EDCA4	EDCA5	—		EDCD	COUNT	EE0	—				DEBUGEV
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESEL_DM has one bit for every source of the event selector. Setting the appropriate bit configures the related source to cause “Enter Debug mode.” If all the bits are set to zero, the event selector does not enter the MSC8113 into Debug mode. In the MSC8113, bits 11–14 should always be written to 0. If EE0 is to be blocked, bit 10 is cleared. If bit 10 is cleared and the JTAG TAP controller does not issue a debug request, the SC140 core can enter Debug mode only via events that relate to execution of instructions in the SC140 core (for example, EDCA events, instructions, or counter values). This may create problems in synchronized entry to Debug mode in the multicore environment.

The Event Selector Debug Interrupt (ESEL_DI) register, Event Selector Enable Trace Buffer (ESEL_ETB), and Event Selector Disable Trace Buffer (ESEL_DTB) registers generate a debug exception, enable the trace buffer, and disable the trace buffer, respectively. These registers contain the same fields as the ESEL_DM register for input selectors. Because the MSC8113 only supports EE0, bits 11–14 of these registers should always be written to 0.

18.5.3 EDCA1_CTRL Register Programming

EDCA1_CTRL controls the activation of one of the address event detection channels (EDCA1). Only bits 10–13 are described here. The other bits are described in the EOnCE module chapter in the *SC140 Core Reference Manual*. If EE1 must be disabled, clear the EDCAEN field. This disables EDCA1. Otherwise, EDCA1 is used as any other EDCA.

18.5.4 Real-Time Debug Request

All the SC140 cores can enter Debug mode in several ways. The EE0 debug input request of all three SC140 cores is wired to the output of an “OR” gate that sums the state of all EE1 outputs of the other SC140 cores and the external EE0 signal (see **Figure 18-7**). Therefore, if any one SC140 core sets its EE1 output (that is, enters Debug mode) or EE0 is asserted, the debug request input on all SC140 cores is asserted. EE1 is activated when at least one of the SC140 cores enters Debug mode.

Note: The EE0 input initiates Debug mode, and the EE1 output is the debug acknowledge indication.

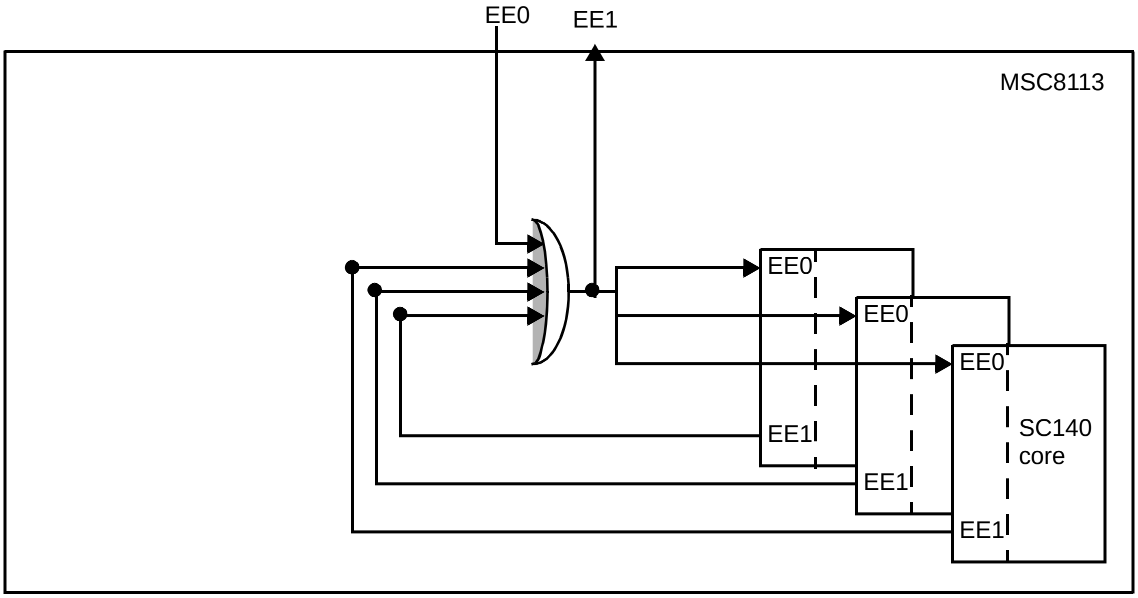


Figure 18-7. Selected SC140 Core Issues a Debug Request to All Other SC140 Cores

18.5.5 Exiting Debug Mode

When an SC140 core enters Debug mode (this is checked by EOnCE module status bits through JTAG as shown in **Table 18-2**), the EE0 internal signal of that SC140 core EOnCE module is masked, preventing any more debug requests. When all the SC140 cores exit Debug mode, the EE0 internal signals of all SC140 cores are unmasked, enabling further debug requests. To restart the SC140 cores, a **go** instruction is scanned into all three SC140 cores. When the scan completes, the update launches all three SC140 cores.

Note: When multiple cores are in Debug mode, issuing simultaneous **go** instructions to such cores does not guarantee that the cores exit Debug mode on the same clock cycle.

No retriggering occurs through EE0. For stepping, the same arrangement is used with the **step** instruction. All SC140 cores are enabled via the CHOOSE_EONCE command, and then a **step** instruction is scanned into all three SC140 cores. When the scan is done, the update launches all three SC140 cores simultaneously. No retriggering occurs through EE0.

18.5.6 Accessing EOnCE Registers Through JTAG in Real Time

When performing an access to EOnCE registers through the JTAG while the SC140 core is executing instructions (that is, not in Debug mode), the access may not be serviced if the core is frozen. After performing such an access, the user or the driver software should wait for the update acknowledge indication. If the update acknowledge (upd_ack) bit is not set (see **page 18-22**), the debug software should retry the access. Update acknowledge can be checked by using one of the two following methods:

- Read the two core status bits and the upd_ack bit that are shifted out when the JTAG command is shifted in. The core select bits in the General Purpose Register (GPR) indicate the core to which the status bits belong. For details, see **page 18-22**. This the method currently used by the debugger tool.
- Shift out and read the contents of the Parallel Input Register (PIREG) to check the status of all three cores. For details, see **page 18-22**.

18.5.7 External Debug Exception Request

You can request an interrupt through software by issuing a JTAG DEBUG_REQUEST command or via hardware by asserting the EE0 signal. Either request may cause the SC140 core to perform a debug exception instead of entering Debug mode, if the EMCR[IME] bit is set. In addition, if the interrupt request is made while the core is frozen, the request may be lost. To ensure interrupt acceptance, you should implement a double-acknowledge software protocol that communicates between the software agent that asserts the interrupt request and the debug exception handler that runs on the SC140 core.

18.5.8 Generating a Debug Exception From an EDCA PC Detection Event

When the EOnCE module is configured to generate a debug exception as a result of setting an EDCA to generate a PC detection event, do not place the PC detection with the following execution sets:

- A change-of-flow instruction (such as JMP, BT, or SKIPLS).
- Immediately following a conditional change-of-flow instruction (such as JF or IFT JMP).

18.6 Tracing in the MSC8113

The trace buffer in each EOnCE module in the MSC8113 is 8 KB. Use of a trace buffer in the MSC8113 requires specific procedures, depending on how you access the trace buffer.

If you read the trace buffer through software, you must perform the following steps before reading the buffer:

1. Ensure that the program reading the trace buffer is in internal memory.
2. Ensure that the local bus to M1 memory interface is not active.
3. Flush the write buffer before reading the trace buffer.

If you read the trace buffer through the JTAG interface, you must perform the following steps before reading the buffer:

1. Ensure that the EOnCE module is in Debug mode.
2. Ensure that the local bus to M1 memory interface is not active.
3. Flush the write buffer before reading the trace buffer.

After you read the trace buffer, you must perform the following post-processing steps:

1. Search in the trace data for the following trace sequence:
   ```
   A1 -> A2
   A1 -> I1
   ```
   where A1 and A2 are the change of flow instructions, and I1 is the start of an interrupt. This sequence represents a case where a change of flow instruction (A2) was aborted due to a pending interrupt.
2. Replace the flow with the following trace:
   ```
   A1 -> A2
   A2 -> I1
   ```

If the EOnCE module is configured to enable or disable tracing as a result of setting an EDCA to generate a PC detection event, do not place the PC detection with the following execution sets:

- A change-of-flow instruction (such as JMP, BT, or SKIPLS)
- Immediately following a conditional change-of-flow instruction (such as JF or IFT JMP)

18.7 General JTAG Mode Restrictions

The control afforded by the output enable signals using the **bsr** and the **extest** instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. You must avoid situations in which the MSC8113 output drivers are enabled into actively driven networks. There are two constraints on the JTAG interface.

- The TCK input is not blocked in low-power Stop mode. To minimize power consumption, connect TCK externally to V_{CC} or GND.
- There are two methods to ensure that the JTAG test logic does not conflict with the system logic by forcing TAP into the test-logic-reset controller state. During power-up, $\overline{\text{TRST}}$ must be externally asserted to force the TAP controller into this state. After power-up, TMS must be sampled as a logic one for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{CC}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

To save power when JTAG is not in use, the MSC8113 should be in the following state:

- To enter or to remain in the Low-Power Stop mode, the TAP controller must be in the test-logic-reset state. Leaving the TAP controller test-logic-reset state negates the ability to achieve low power but does not otherwise affect device functionality.
- The TCK input is not blocked in Low-Power Stop mode. To consume minimal power, the TCK input should externally connect to V_{CC} or ground.
- TMS and TDI include internal pull-up resistors. In Low-Power Stop mode, these two signals should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

18.8 JTAG and EOnCE Module Programming Model

18.8.1 Identification Register

The JTAG ID register is a 32-bit read-only factory-programmed register that distinguishes the component on a board according to the **IEEE** Std. 1149.1. The fields are defined as follows:

JTAGID JTAG Identification (ID) Register **JTAG port access only**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1
	Version				Design Center						Sequence Number										Manufacturer identity											1

- Version information corresponds to the revision number. The MSC8113 first mask set is 0b0000.
- Design Center number is 0b000110.
- Sequence Number for the MSC8113 is 0b0010000101
- Manufacture identity is 0b00000001110.
- The final 1 is required by the **IEEE** Std. 1149.1.

Note: The hexadecimal value stored in this register is 0x0188501D.

Later mask sets will have a different number. Refer to the website listed on the back cover of this manual for the information about the contents of this register for current device revisions.

18.8.2 Boundary Scan Register (BSR)

The MSC8113 BSR contains bits for most device signals and control signals. All MSC8113 bidirectional signals have two registers for boundary scan data and are controlled by an associated control bit in the BSR. The boundary scan bit definitions vary according to the specific chip implementation of the MSC8113 and are described by the BSDL file on the product website. **Figure 18-8** through **Figure 18-11** show various BSR cell types.

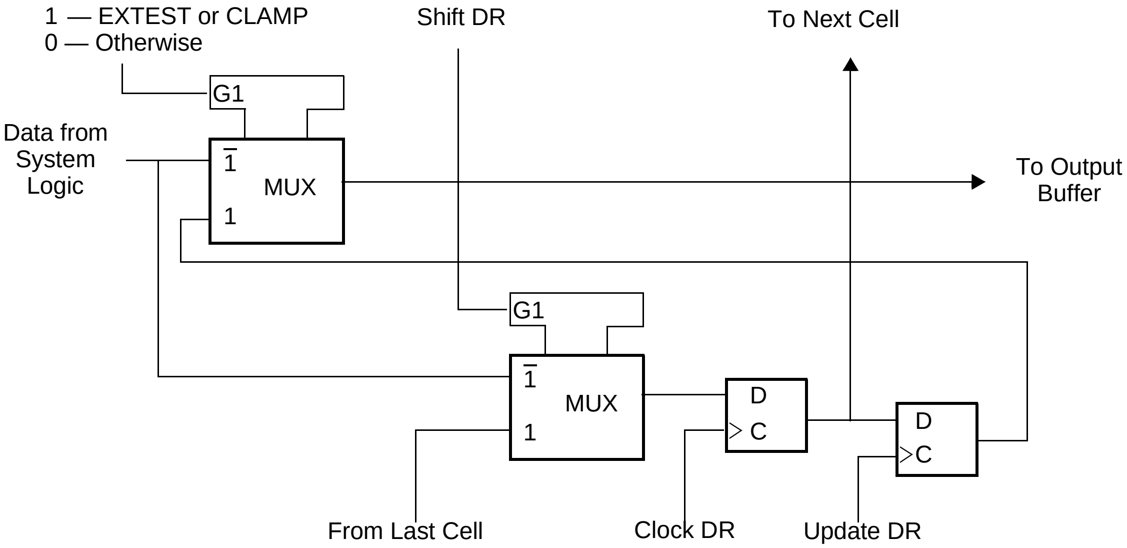


Figure 18-8. Output Signal Cell (O.PIN)

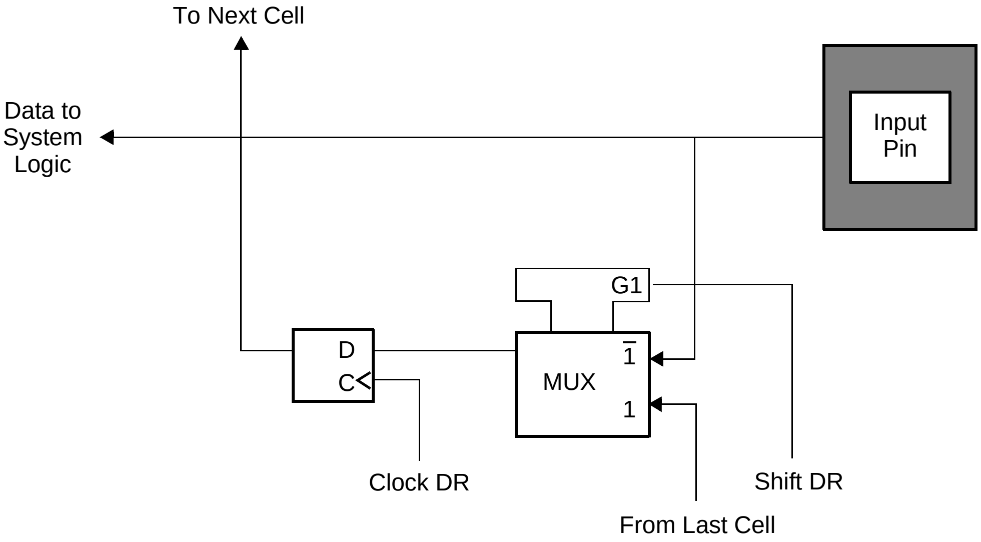


Figure 18-9. Observe-Only Input Signal Cell (I.OBS)

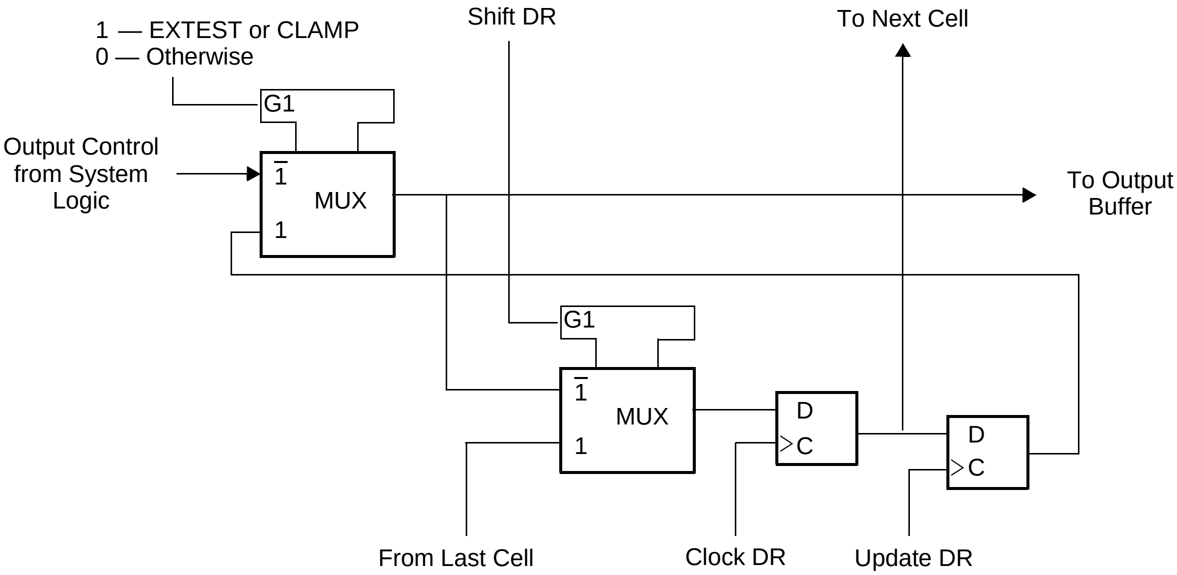


Figure 18-10. Output Control Cell (IO.CTL)

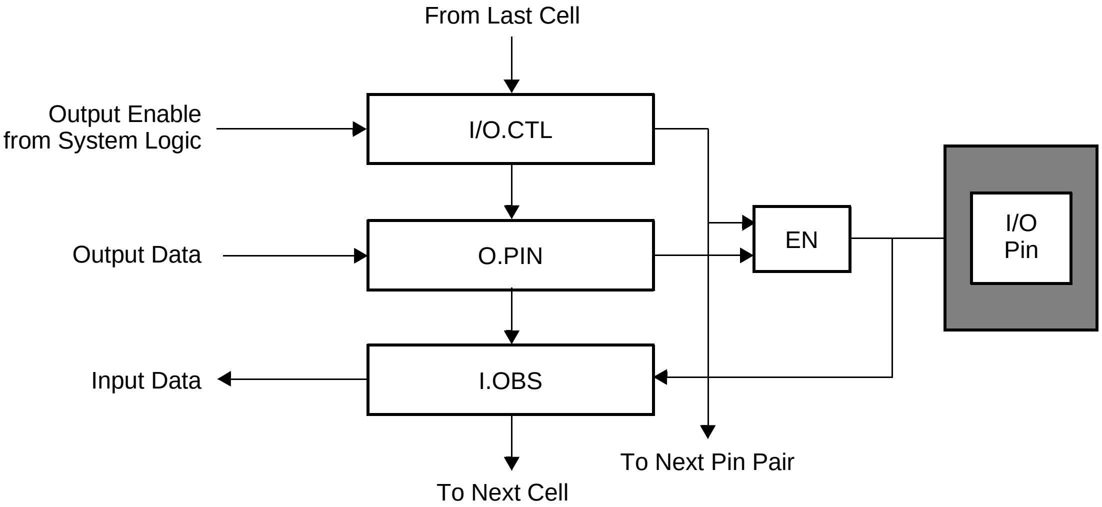


Figure 18-11. General Arrangement of Bidirectional Signal Cells

The control bit value controls the output function of the bidirectional signal. One or more bidirectional data cells can be serially connected to a control cell. Bidirectional signals include two scan cells for data (IO.Cell) as shown in **Figure 18-11**, and these bits are controlled by the cell shown in **Figure 18-10**. It is important to know the boundary scan bit order and signals that are associated with them. The BSDL file on the product website describes the boundary scan serial string. The three MSC8113 cell types described in this file are depicted in **Figure 18-8** through **Figure 18-10**, which describe the cell structure for each type.

18.8.3 Shift Registers

The shift registers include the Bypass Register, General-Purpose Register (GPR), BSR, Identification Register, and Parallel Input register.

18.8.3.1 Bypass Register

The Bypass Register is a single-bit shift register (see **Figure 18-12**). When selected, it creates a shift-register path of one bit from TDI to TDO. When the Bypass Register is selected, the shift-register stage is set to a logic zero on the rising edge of TCK in the CAPTURE-DR controller state.

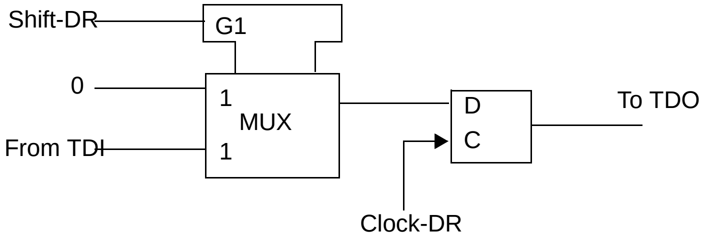


Figure 18-12. Bypass Register Configuration

18.8.3.2 Identification Register

When the Identification Register is selected, the shift-register stage is set to a logic value equal to IDCODE on the rising edge of TCK in the CAPTURE-DR controller state. It can then be shifted out in the SHIFT-DR controller state. See **Figure 18-13**.

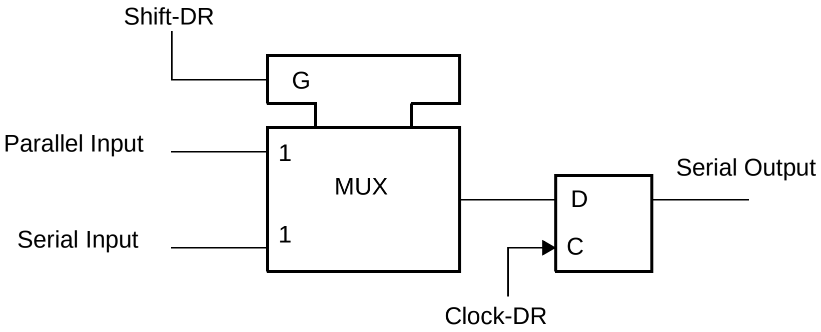


Figure 18-13. Identification Register Configuration (ID)

18.8.3.3 General-Purpose Register

GPR JTAG General-Purpose Register **JTAG port access only**

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

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

During a shift in of any JTAG instruction, the bits shifted out reflect the status of the SC140 core (see **Table 18-2**). To select the SC140 core to which these bits belong, two bits are used in the JTAG GPR. The GPR shifts in from TDI and out to TDO. The GPR[ISRSEL] bits select which EOnCE module and SC140 core status bits are reflected in the capture of any JTAG instruction. All other encodings are reserved and should be written with a value of 0. Writing a 1 to any of these bits may result in improper operation.

Table 18-6. GPR Bit Descriptions

Name	Reset	Description	Settings
ISRSEL 0–1	0	**Instruction Status Core Select** Defines the SC140 core to which the bits output during instruction register shifts belong.	00 Core 0. 01 Core 1. 10 Core 2. 11 Reserved.
— 10–31	0	Reserved. Write to zero for future compatibility.	

18.8.3.4 Parallel Input Register

PIREG JTAG Parallel Input Register **JTAG port access onl**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—							core2 cores[1–0]		core2 upd_ack	core1 cores[1–0]		core1 upd_ack	core0 cores[1–0]		core0 upd_ack
Type	R															
Reset	0															

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

You can observe the status of all three SC140 cores by programming the GPR and shifting in a JTAG instruction three times. However, it is easier to observe the status of all three SC140 cores at once by shifting out the contents of the parallel input register. The Parallel Input Register (PIREG) is selected using the READ PIREG command and then shifting out 32 bits from the PIREG. The bits shifted out reflect the status of the SC140 cores. See **Table 18-7** for the bit-field definitions.

Table 18-7. PIREG Bit Descriptions

Number	Reset	Description	Settings
— 31–25	0	Reserved.	
core 2 cores 24–23	0	**Core 2 Core Status** Reflects the status of core 2	00 Core is executing instructions. 01 Core is in Wait or Stop mode. 10 Core is waiting for bus. 11 Core is in debug mode.
core 2 upd_ack 22	0	**Core 2 Update Acknowledge** Indicates whether the core 2 SC140 EOnCE module has executed the last instruction dispatched to it	0 EOnCE module has executed the last instruction dispatched to it. 1 EOnCE module has not executed the last instruction dispatched to it.
core 1 cores 21–20	0	**Core 1 Core Status** Reflects the status of core 1	00 Core is executing instructions. 01 Core is in Wait or Stop mode. 10 Core is waiting for bus. 11 Core is in debug mode.
core 1 upd_ack 19	0	**Core 1 Update Acknowledge** Indicates whether the SC140 core 1 EOnCE module has executed the last instruction dispatched to it.	0 EOnCE module has executed the last instruction dispatched to it. 1 EOnCE module has not executed the last instruction dispatched to it.
core 0 cores 18–17	0	**Core 0 Core Status** Reflects the status of core 0.	00 Core is executing instructions. 01 Core is in Wait or Stop mode. 10 Core is waiting for bus. 11 Core is in debug mode.
core 0 upd_ack 16	0	**Core 0 Update Acknowledge** Indicates whether the SC140 core 0 EOnCE module has executed the last instruction dispatched to it.	0 EOnCE module has executed the last instruction dispatched to it. 1 EOnCE module has not executed the last instruction dispatched to it.
— 15–0	0	Reserved.	

Internal Peripheral Bus (IPBus) 19

The internal peripheral bus (IPBus) is connected to the following modules:

- TDM interface
- UART
- Timers
- GPIOs
- Hardware semaphore registers
- General interrupt controller (GIC)
- Direct slave interface (DSI)
- Ethernet controller

This chapter briefly describes each of these peripherals, indicates the chapters in which they are discussed in detail, and describes the IPBus functionality and programming model.

19.1 TDM Interface

The TDM interface is composed of four identical and independent TDM modules, each supporting 256 channels running at up to 66 Mbps with 2-,4-, 8-, and 16-bit data sizes. The TDM bus connects gluelessly to most T1/E1frames as well as to common buses such as the ST-BUS. Each TDM module operates in independent or shared mode when receiving or transmitting data:

- In independent mode, there are different sync, clock, and data links for receive and transmit modules.
- In shared sync and clock mode, the clock and sync are shared between the transmit and receive modules with different data links for each module.
- In shared data link mode, the receive and transmit modules share sync, clock, and full duplex data links between the transmit and receive modules. The clock and the sync signals can also be shared between the TDM modules.

Note: For details, see **Chapter 20**, *TDM Interface*.

19.2 UART

The UART, also known as the serial communication interface (SCI), provides a full-duplex port for serial communications with other devices. This interface uses two dedicated signals: transmit data (TXD) and receive data (RXD). Both signal lines are available for general-purpose I/O (GPIO) when they are not configured for operation.

Note: For details, see **Chapter 21**, *UART*.

19.3 Timers

The MSC8113 device contains 32 timers of 16 bits each that serve as frequency dividers, watchdog timers, clock generators, and event counters. Each timer receives input from one of 15 sources: six external inputs, eight timer outputs, or the local bus clock (BUSES_CLOCK).

Note: For details, see **Chapter 22**, *Timers*.

19.4 GPIOs

The MSC8113 has 32 general-purpose I/O (GPIO) signals. Each signal line in the I/O port is configured either as a GPIO signal or as a dedicated peripheral interface signal. In addition, fifteen of the signal lines can generate interrupts to the global interrupt controller. Each signal is configured as an input or output (with a register for data output that is read or written at any time). All output signals can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, the signal drives a zero voltage but goes to three-states when driving a high voltage.

Note: **Chapter 23**, *GPIO*.

19.5 Hardware Semaphore Registers

The MSC8113 hardware semaphores (HS) block holds eight coded hardware semaphores. A coded hardware semaphore provides a simple way to achieve a *lock* operation via a single write access, eliminating the need for sophisticated read-modify-write software mechanisms. Using the hardware semaphores, external hosts such as DSI external masters can protect internal and external shared resources and ensure coherency in any sequence of operations on these resources. The coded hardware semaphore is an eight-bit register with a selective write mechanism.

Note: **Chapter 15**, *Hardware Semaphores*.

19.6 Global Interrupt Controller (GIC)

The GIC performs the following functions:

- Generates 32 virtual interrupts by write access to a special address (virtual address) with predefined data. The virtual interrupts are divided into four groups of eight interrupts, each group routed to the LIC of one SC140 core.
- Generates four virtual $\overline{\text{NMI}}$ interrupt pulses, each of which goes to one SC140 core, by write access to a special virtual $\overline{\text{NMI}}$ address. One SC140 core can assert the $\overline{\text{NMI}}$ of another SC140 core.
- Collects interrupt sources from the UART, SIU interrupt sources, DMA system, Ethernet controller, 15 external sources ($\overline{\text{IRQ[1–15]}}$) and four of the virtual interrupt sources (line 0 of each group) and selectively enables them for assertion of INT_OUT.
- Collects external interrupt sources $\overline{\text{IRQ[8–15]}}$, configures them to edge/level, and selectively enables them for global distribution to all the SC140 PICs.

Note: For details, see **Chapter 17**, *Interrupt Processing*.

19.7 Direct Slave Interface (DSI)

The Direct Slave Interface (DSI) gives an external host direct access to the MSC8113 device. It provides the following slave interfaces to an external host:

- Asynchronous interface giving the host single accesses (with no external clock).
- Synchronous interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8113 internal bus clock.

Note: For details, see **Chapter 14**, *Direct Slave Interface (DSI)*.

19.8 Ethernet Controller

The Ethernet controller is designed to comply with the **IEEE** Std. 802.3™ and supports 10 Mbps and 100 Mbps operation with the media-independent interface (MII) and the reduced media-independent interface (RMII). The Ethernet controller works with minimal SC140 core intervention and operates in two modes:

- Full Duplex mode, for connecting the Ethernet to an on-board ethernet switch.
- Half-Duplex mode, for connecting the Ethernet to an on-board physical layer (PHY).

The Ethernet controller is designed to comply with the **IEEE** Std. 802.3 and supports 10 Mbps and 100 Mbps operation with the media-independent interface (MII) and the reduced media-independent interface. The Ethernet controller supports full and half duplex MII, RMII, and SMII interfaces and can receive frames into and transmit frames from M1, M2 or external memory.

Note: For details, see **Chapter 25**, *Ethernet Controller*.

19.9 IPBus Functionality

The purpose of the IPBus is to control and configure the peripheral modules (TDM, timers, UART, DSI, HS, GIC, GPIO, and the Ethernet controller) and handle UART data transfers. Each SC140 core accesses the IPBus through the SQBus for the internal configuration. The IPBus is a 32-bit wide single-master, multi-slave bus. The IPBus master arbitrates between the SQBus and the local bus, and its slaves are TDMs, timers, UART, DSI, HS, GIC, Ethernet controller, and GPIO. The IPBus master forwards the accesses from the SQBus and from the local bus to the IPBus and controls its operation. The local bus operates on the BUSES_CLOCK clock, and the SQBus operates on CORES_CLOCK clock. When a simultaneous request occurs from both the local bus and the SQBus, the SQBus receives higher priority and wins the arbitration.

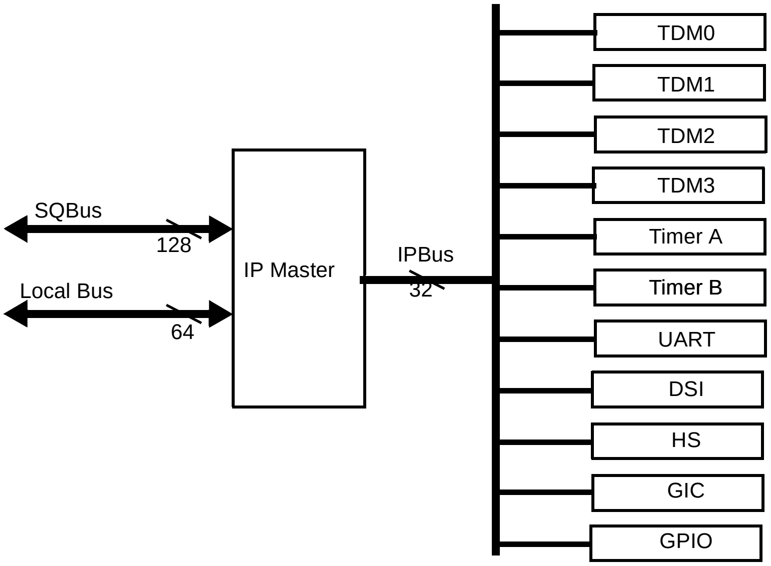


Figure 19-1. IPBus Block Diagram

19.10 Stop Options

A Stop option is provided for power management. It disables clocks for different modules, and it functions differently for each slave module. TDMs and timers enter Stop mode when there is no access and the module is disabled by the user. UART, DSI, Ethernet Controller, and GIC enter Stop mode according to the Stop Control Register bits.These modules must not be accessed while they are in Stop mode. There is no Stop for the HS and GPIO modules. The Stop option applies to the different slave modules as follows:

- *Stop for TDMs*. When you disable the specific TDM (both the receiver and transmitter of the TDM are disabled), and there is no access to the TDM via the IP master, the TDM automatically shuts down its clock. The clock is turned on when the TDM is enabled or accessed (see **Chapter 20**, *TDM Interface*).
- *Stop for Timers*. When a specific timer is not enabled (disable the timers by clearing the TE bit in all 16 TCRs) and there is no access to it via the IP master, the timer automatically shuts down its clock. The clocks are turned on when the timer is enabled or accessed (see **Chapter 22**, *Timers*).
- *Stop for UART*. The UART uses the stop bit for the clock stop. Its stop acknowledge is unconditioned, but entering the Stop mode during a transmission or reception results in invalid data (see **Chapter 21**, *UART*).
- *Stop for DSI*. The DSI uses the stop bit to complete the clock stop. Its stop acknowledge is conditioned by the setting of the DDR[DSISTP] bit and an empty write buffer (no pending accesses). Upon receiving a request from a stop request bit, it continues to perform all the accesses waiting in the write buffer and does not perform prefetches into the read buffer (see **Chapter 14**, *Direct Slave Interface (DSI)*).
- *Stop for Ethernet Controller*. The Ethernet Controller uses the stop bit for the clock stop. Its stop acknowledge is conditioned by setting the SCR1[ETH_STC] bit and disabling the Interrupt Events (see **Chapter 25**, *Ethernet Controller*).
- *Stop for GIC*. The GIC uses the stop bit to stop its clocks in a sequence. It keeps pending interrupts and enables handling them, but it does not detect new interrupts. It acknowledges a stop request only when there are no more enabled pending interrupts (see **Chapter 17**, *Interrupt Processing*).

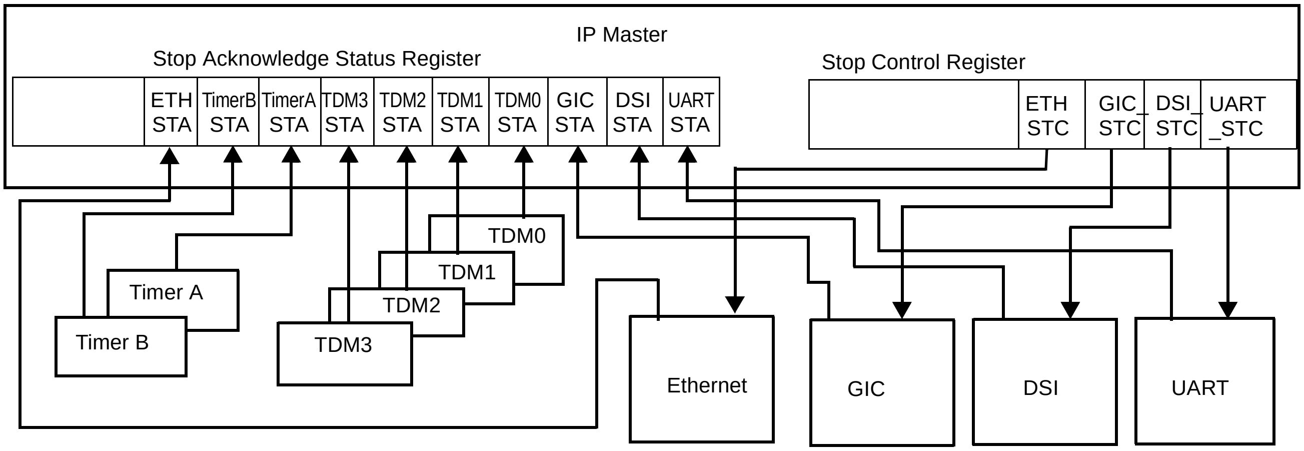


Figure 19-2. Stop Mode for Different IPBus Modules

19.11 IPBus Programming Model

Refer to **Section 8.5**, *IPBus Address Space*, on page 8-12 for the IPBus base address.

SCR Stop Control Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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
	—												ETH_STC	GIC_STC	DSI_STC	UART_STC
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SCR defines which slave blocks are requested to enter Stop mode. The register includes control bits for the UART, DSI, Ethernet, and GIC blocks. To set a request for the specific module to enter Stop mode, you should set the appropriate bit in the SCR. Clear this bit when there is no stop request.

Table 19-1. SCR Bit Descriptions

Name	Reset	Description	Settings
— 0–28	0	Reserved. Write to zero for future compatibility.	
ETH_STC 29	0	**Ethernet Controller Stop** Determines whether the Ethernet module is requested to enter Stop mode.	0 Ethernet Controller is not requested to enter Stop mode. 1 Ethernet Controller is requested to enter Stop mode.
GIC_STC 29	0	**GIC Stop** Determines whether the GIC module is requested to enter Stop mode.	0 GIC is not requested to enter Stop mode. 1 GIC is requested to enter Stop mode.
DSI_STC 30	0	**DSI Stop** Determines whether the DSI module is requested to enter Stop mode.	0 DSI is not requested to enter Stop mode. 1 DSI is requested to enter Stop mode.
UART_STC 31	0	**UART Stop** Determines whether the UART module is requested to enter Stop mode.	0 UART is not requested to enter Stop mode. 1 UART is requested to enter Stop mode.

SASR Stop Acknowledge Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
—	—															
Type	R															
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
—	—	—	—	—	—	—	ETH_ STA	TimerB _STA	TimerA _STA	TDM3_ STA	TDM2_ STA	TDM1_ STA	TDM0_ STA	GIC_ STA	DSI_ STA	UART_ STA
Type	R															
Reset	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0

SASR accumulates stop acknowledge data from all the IPBus peripherals. When any peripheral enters Stop mode, the appropriate bit in the SASR is set. When the peripheral exits Stop mode, the bit is cleared. SASR data can be read through the SQBus or the local bus.

Table 19-2. SASR Bit Descriptions

Name	Reset	Description	Settings
— 0–21	0	Reserved. Read returns zero.	
ETH_STA 22	0	**Ethernet Stop Ack** Determines whether the Ethernet module is in Stop mode.	0 Ethernet Controller is not in Stop mode. 1 Ethernet Controller is in Stop mode.
TimerB_STA 23	1	**Timer B Stop Ack** Determines whether the Timer B module is in Stop mode.	0 Timer B is not in Stop mode. 1 Timer B is in Stop mode.
TimerA_STA 24	1	**Timer A Stop Ack** Determines whether the Timer A module is in Stop mode.	0 Timer A is not in Stop mode. 1 Timer A is in Stop mode.
TDM3_STA 25	1	**TDM3 Stop Ack** Determines whether the TDM3 module is in Stop mode.	0 TDM3 is not in Stop mode. 1 TDM3 is in Stop mode.
TDM2_STA 26	1	**TDM2 Stop Ack** Determines whether the TDM2 module is in Stop mode.	0 TDM2 is not in Stop mode. 1 TDM2 is in Stop mode.
TDM1_STA 27	1	**TDM1 Stop Ack** Determines whether the TDM1 module is in Stop mode.	0 TDM1 is not in Stop mode. 1 TDM1 is in Stop mode.
TDM0_STA 28	1	**TDM0 Stop Ack** Determines whether the TDM0 module is in Stop mode.	0 TDM0 is not in Stop mode. 1 TDM0 is in Stop mode.
GIC_STA 29	0	**GIC Stop Ack** Determines whether the GIC module is in Stop mode.	0 GIC is not in Stop mode. 1 GIC is in Stop mode.
DSI_STA 30	0	**DSI Stop Ack** Determines whether the DSI module is in Stop mode.	0 DSI is not in Stop mode. 1 DSI is in Stop mode.
UART_STA 31	0	**UART Stop Ack** Determines whether the UART module is in Stop mode.	0 UART is not in Stop mode. 1 UART is in Stop mode.

TDM Interface 20

The MSC8113 Time-Division Multiplexing (TDM) interface enables communication among many devices over a single bus. Traffic is managed according to a time-division multiplexing method in which only one device drives the bus (transmit) for each channel. Each device drives its active transmit channels and samples its active receive channels when its channel is active. It is the system designer's responsibility to guarantee that there is no conflict in transmit channel allocation.

The TDM interface is composed of four identical and independent TDM modules, each supporting 256 channels running at up to 66 Mbps with 2-, 4-, 8-, and 16-bit word size. The TDM bus connects gluelessly to most T1/E1 frames as well as to common buses such as the ST-BUS. Each TDM module operates in independent or shared mode when receiving or transmitting data:

- In independent mode, there are different sync, clock, and data links for receive and transmit.
- In shared sync and clock mode, the clock and the sync are shared between the transmit and receive with different data links for the receive and transmit.
- In shared data link mode, the receive and transmit share sync, clock, and full duplex data links between the transmit and receive. The clock and the sync signals can also be shared between the TDM modules.

Note: When the TDM interface is used, for correct operation, the TDM clock must run continuously, even during idle data time slots.

At any time, each channel is individually set to active or inactive. An on-the-fly hardware A-law/µ–law conversion is supported for 8-bit channels. A channel is transparent or A-law/µ–law. Its data is collected in its own buffer location independently from other channel buffers. Buffer size is 16 MB for transparent channels and 32 MB for A-law/µ–law channels. The direction of the bits in the channel (MSB first/LSB first) is configured globally for each TDM module. The direction of TSYN is set to input or output. The polarity of the clock (sample/drive at clock rise or fall) is independently configured for the receiver and transmitter. The polarity of TSYN/RSYN/FSYN is configured to positive or negative.

The four TDM modules have an I/O matrix that routes the clock and sync signals between the TDM modules and the MSC8113 signal lines. The TDM is configured by all three SC140 cores through the interface to the IPBus (see **Figure 20-1**). Data is received and transmitted from the TDM modules to the channel buffers through the local bus. **Figure 20-2** shows the TDM block diagram and the receive and transmit data flows. The dashed line depicts the transmit data flow from the local bus to the I/O matrix; the solid line depicts the receive data flow from the I/O matrix to the receive buffers on the local bus. Serial data received from the I/O matrix is packed and stored in the TDM local memory buffer. From the local memory buffer, the data is converted according to A/µ transformation (if needed) and re-packed for transaction to the local bus. Data transmission occurs in a similar way but in reverse order. The channel data is transferred from the transmit data buffers being converted by the A/µ logic and stored in the TDM local memory buffer. Then the data is transmitted to the transmit serial block and to the I/O matrix.

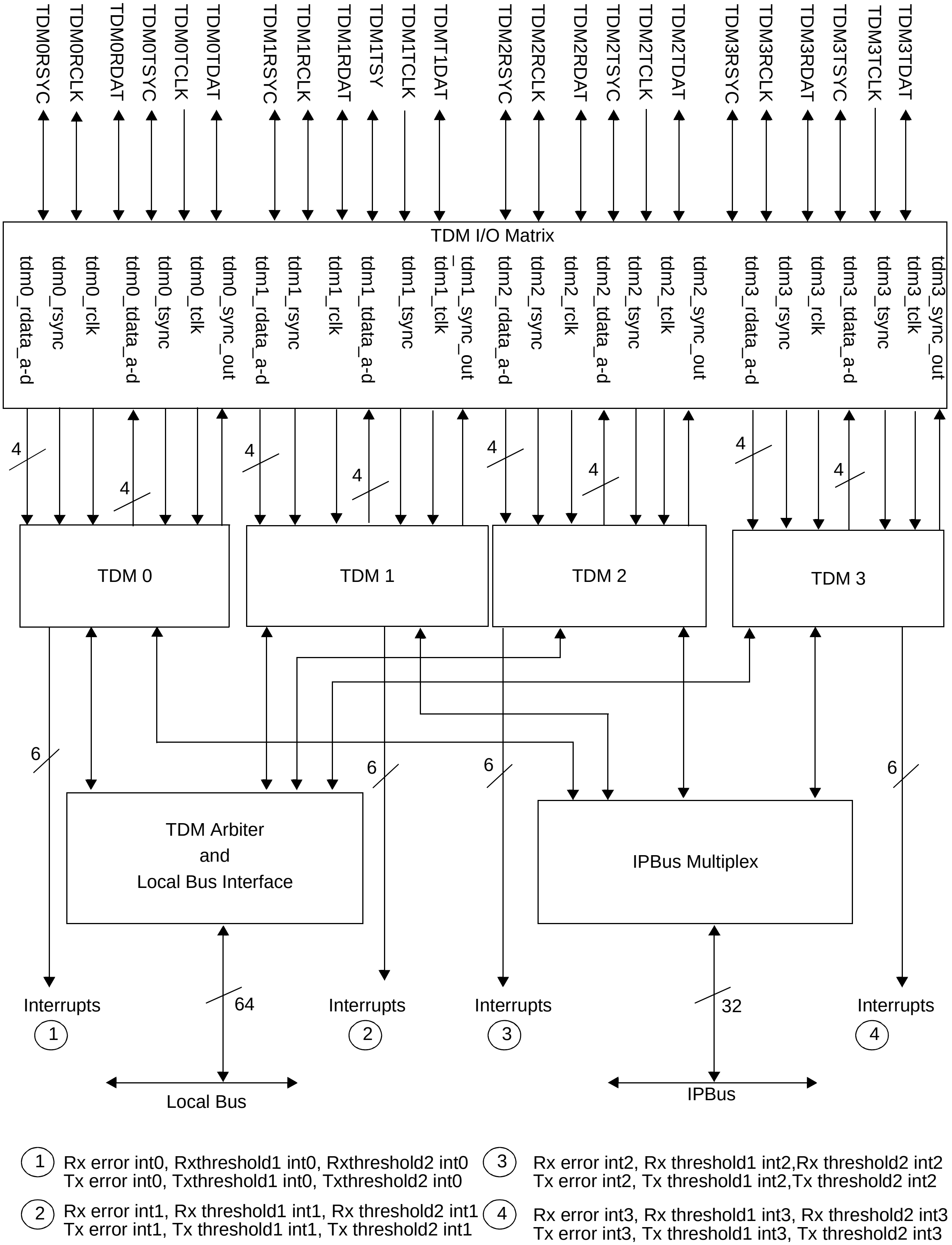


Figure 20-1. General TDM Module Interface

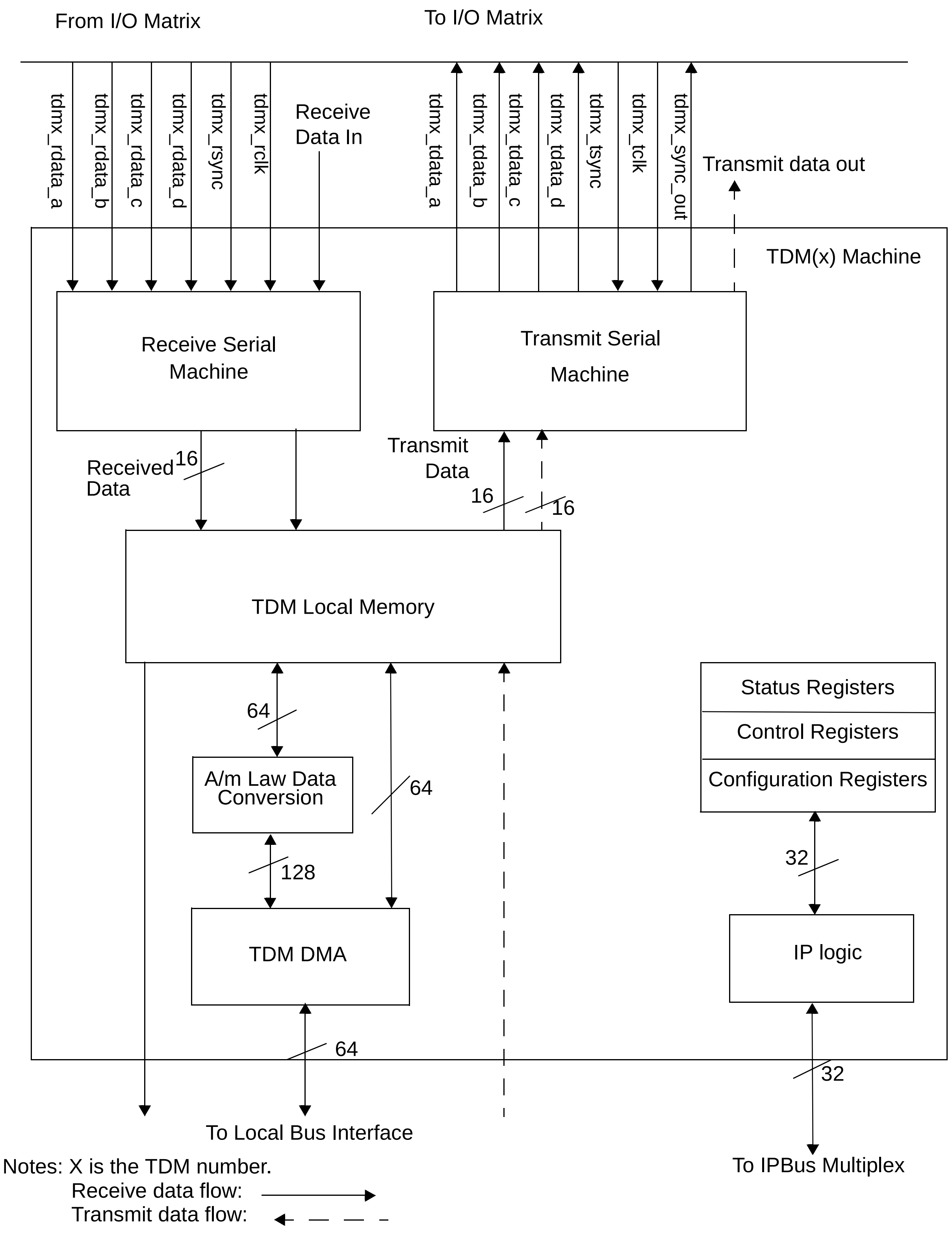


Figure 20-2. TDM Block Diagram

20.1 Typical Configurations

The TDM connects in various configurations. **Figure 20-3** shows two MSC8113 devices that connect point-to-point. Data transmits from the device on the left to the device on the right or *vice versa*.

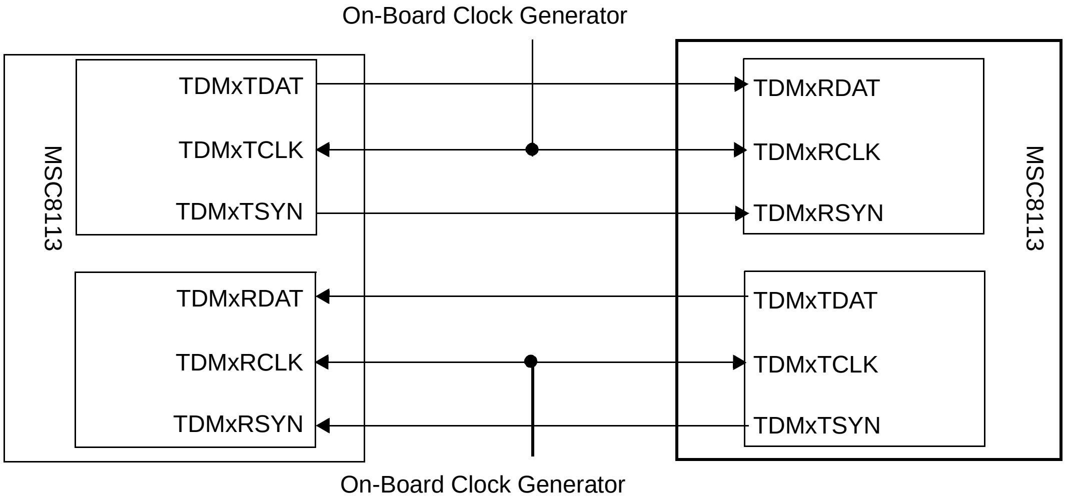


Figure 20-3. TDM Point-to-Point Configuration

Figure 20-4 depicts a TDM point to multi-point configuration. Multiple MSC8113 devices connect on the same TDM bus, which connects to the network through a framer.

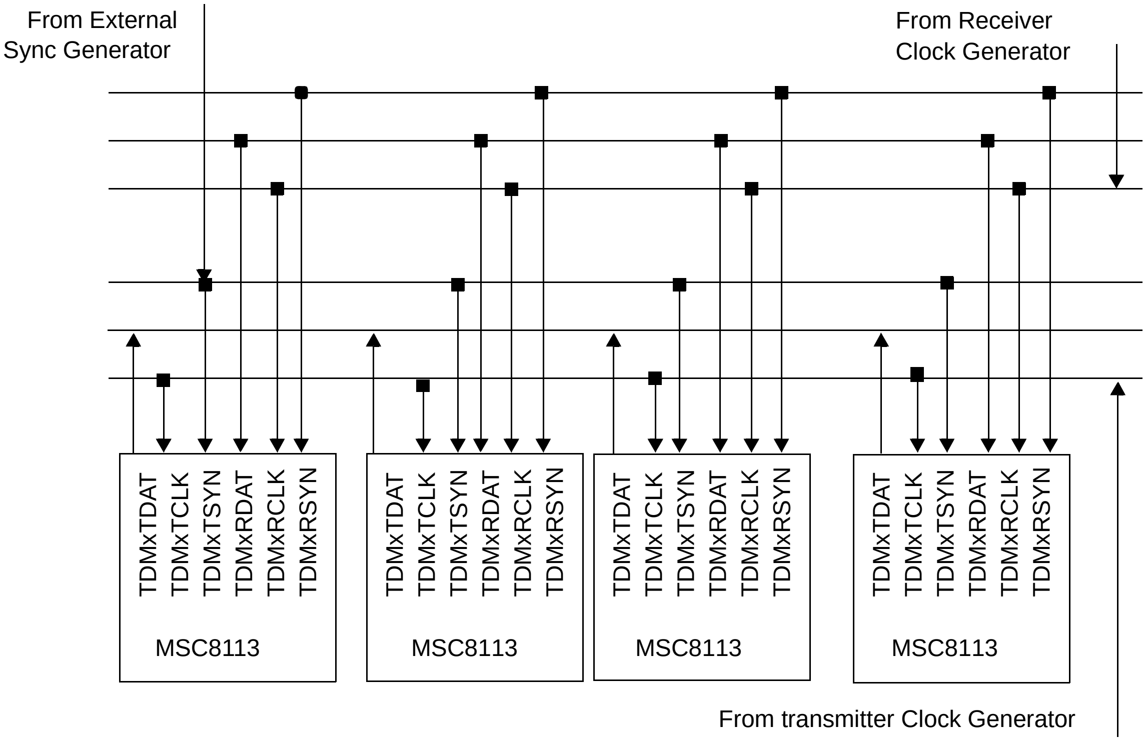


Figure 20-4. TDM Point-to-Multi-Point Configuration

Figure 20-5 depicts an application in which all the TDM modules share the sync and the clock (see **Figure 20-11**). Therefore, each TDM module supports one or two active links. In this example, 16 receive link and 16 transmit links connect to two MSC8113 devices.

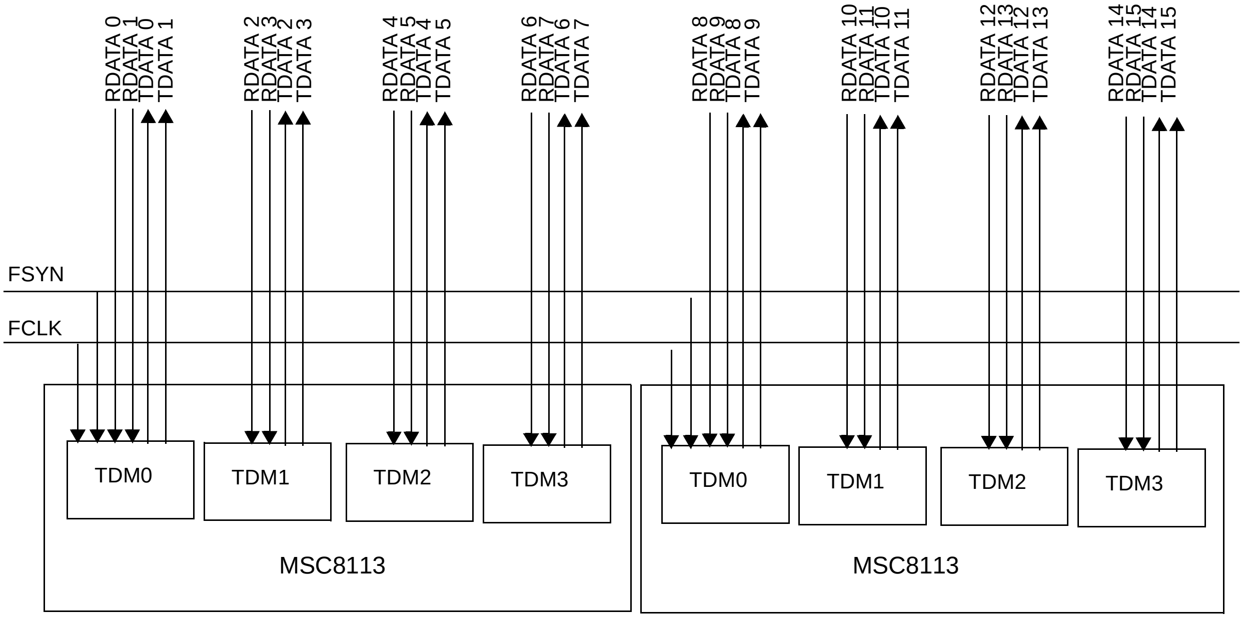


Figure 20-5. Common Frame Sync and Clock

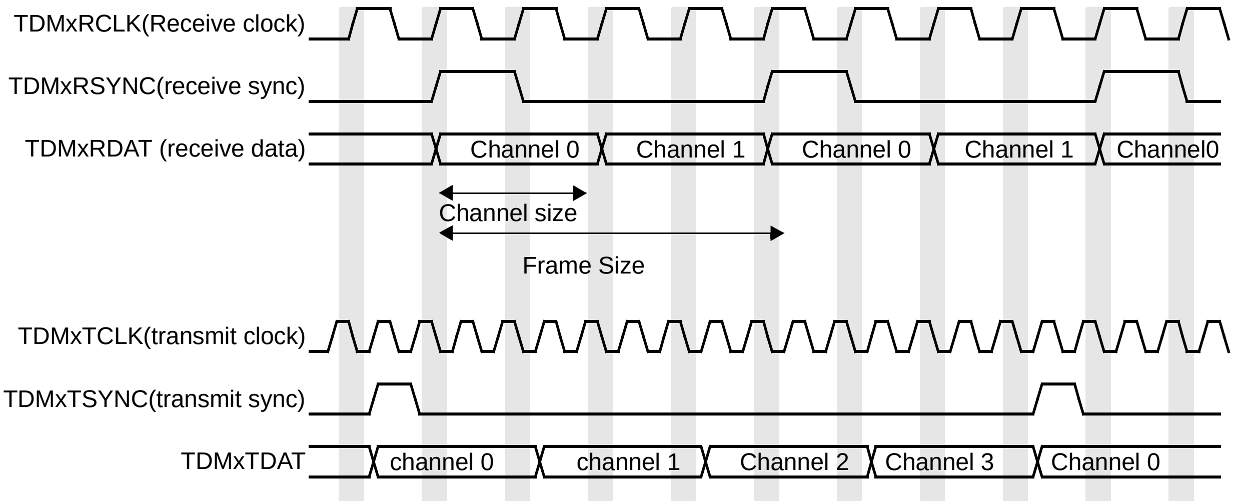
20.2 TDM Basics

Multiple TDM channels are transferred sequentially in a structure called a frame. The frame start is identified by a frame sync signal that is briefly asserted at the beginning of every frame. Each of the four TDM modules can receive or transmit up to 256 channels at a granularity of two. The number of receive channels is determined by the RNCF field in the TDMx Receive Frame Parameters Register (TDMxRFP) (see page 20-47). The number of transmit channels is determined by the TNCF field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 20-49).

The size of all the channels (for each TDM module) is unified and it can be 2-), 4-, 8-, and 16 bits. The receive channel size is determined by the RCS field in the TDMxRFP; the transmit channel size is determined by the TCS field in the TDMxTFP (refer to page 20-49).

When the TDM connects to a T1 framer, the RT1 field in the TDMx Receive Frame Parameters Register (TDMxRFP) (see page 20-47) and the TT1 field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 20-49) should be set. The T1 frame contains 193 bits (24 channels of 8 bits each) when the first bit of the frame is a Frame Alignment bit that is not used by the TDM. At the T1 received frame, the Frame Alignment bit is removed and does not transfer to the main memory. At the transmit T1 frame, the bit is not driven out.

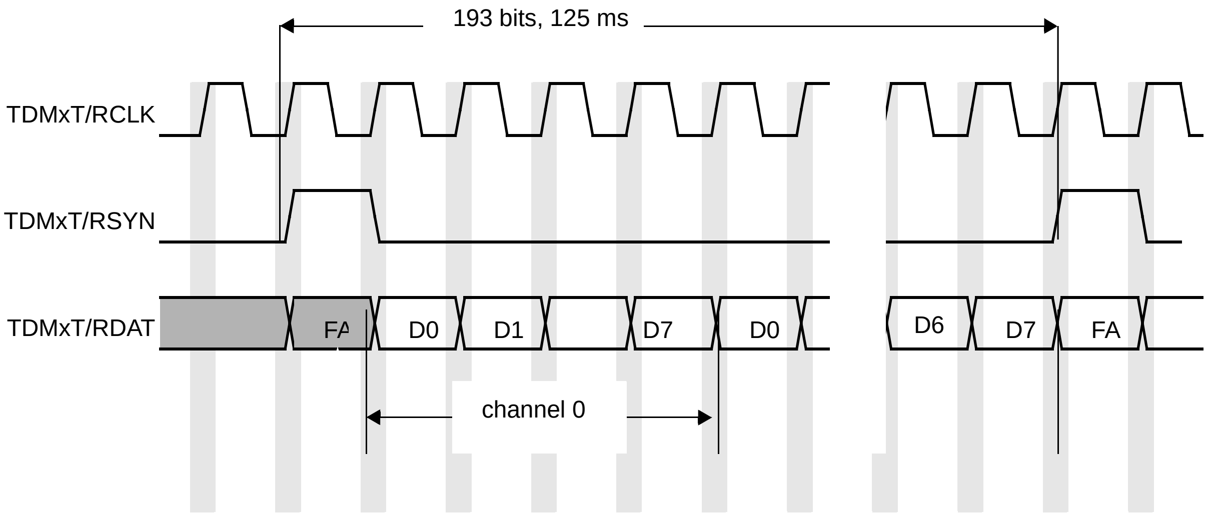
Figure 20-6 shows an example of a TDMx interface. The receive frame contains two 2-bit channels. The transmit frame contains four 4-bit channels. **Figure 20-7** shows an example of T1 frame. Note that in T1 mode, the first bit of the frame is not used by the TDM.



Receive Frame parameters: RNCF[7–0] = 8 × 1 (2 channels), RCS = 4 × 1 (2 bits) and RT1 = 0 (nonT1).

Transmit Frame parameters: TNCF[7–0]= 8 × 3 (4 channels), TCS = 4 × 3 (4 bits). and TT1 = 0 (non T1)

Figure 20-6. TDM Frames



Receive Frame parameters: RNCF[7–0] = 8 × 23 (24 channels), RCS = 4 × 7 (8 bits) and RT1 = 1 (T1 mode).

Transmit Frame parameters: TNCF[7–0] = 8 × 23 (24 channels), TCS = 4 × 7 (8 bits) and TT1 = 1 (T1 mode)

Figure 20-7. T1 Frame

20.2.1 Common Signals for the TDM Modules

The sync and clock signals can be shared among the TDM modules or separate for each TDM module. When the CTS bit of the TDMx General Interface Register (see page 20-36) is equal to 1, the TDM modules share sync and clock signals. In this mode, the common signals connect to the following signal lines:

- The transmit sync/frame sync connects to TDM0TSYN (receive and transmit share the same sync signal).
- The transmit clock/frame clock connects to TDM0TCLK (receive and transmit share the same clock signal).
- The receive sync connects to TDM1TSYNC.
- The receive clock connects to TDM1TCLK.
- When the TSO bit is set to a value of 1 (see page 20-45), the sync out signal drives out through TDM0TSYN.

The configuration registers (see page 20-36) should be identical for the TDM modules that share signals. There are only three possibilities for sharing TDMs: TDM0 and TDM1; TDM0, TDM1, and TDM2; or TDM0, TDM1, TDM2, and TDM3. **Figure 20-8** illustrates a common receive sync, receive clock, transmit sync, and transmit clock for TDM 0 and TDM 1. When the CTS bit of the TDMx General Interface Register (see page 20-36) is cleared, the TDM modules do not share signals. In **Figure 20-8**, TDM2 and TDM3 do not share signals with other TDM modules.

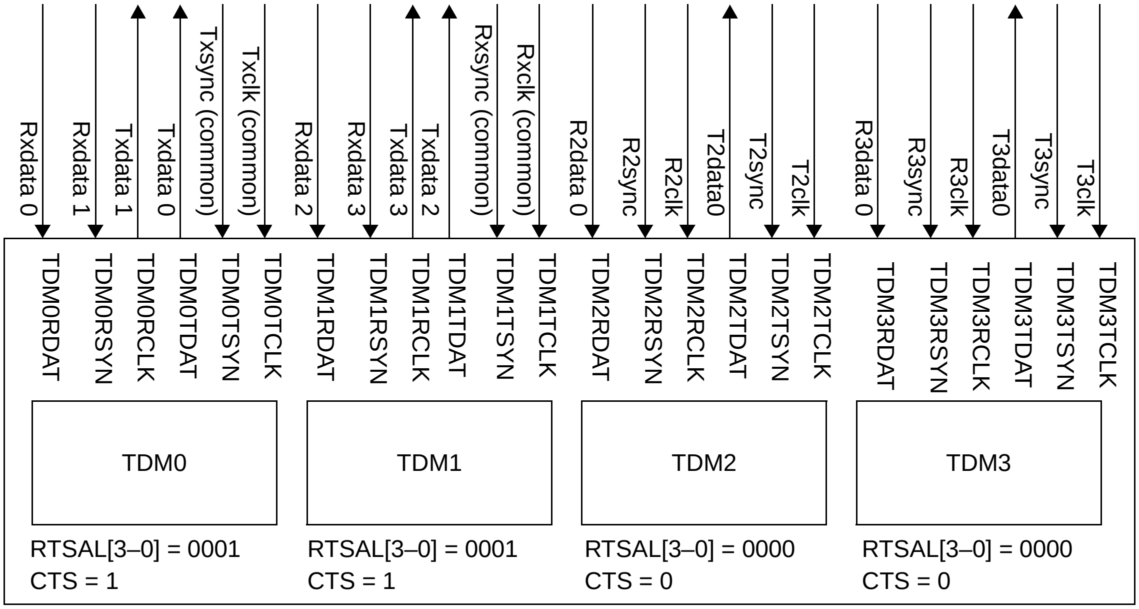


Figure 20-8. TDM Modules Model

In **Figure 20-9**, all four TDM modules share the same receive clock and sync and the same transmit clock and sync. The receive clock connects to the TDM1TCLK port. The transmit clock connects to TDM0TCLK, the receive sync connects to TDM1TSYN, and the transmit sync connects to TDM0TSYNC. Each module has two active data links. Notice that TDM2TSYN, TDM2TCLK, TDM3TSYN, and TDM3TCLK are not used.

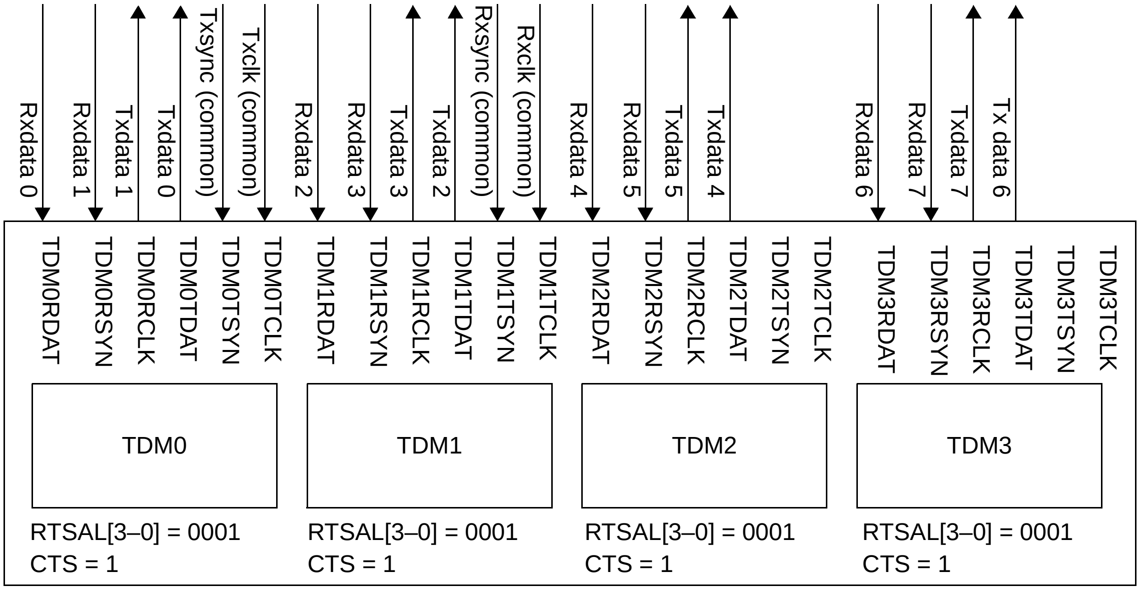


Figure 20-9. Shared Receive Sync and Clock and Transmit Sync and Clock

In **Figure 20-10**, all four TDM modules share the same frame sync, clock, and data links. Notice that TDM1TCLK, TDM1TSYN, TDM2TCLK, TDM2TSYN, TDM3TCLK, and TDM3TSYN are not used.

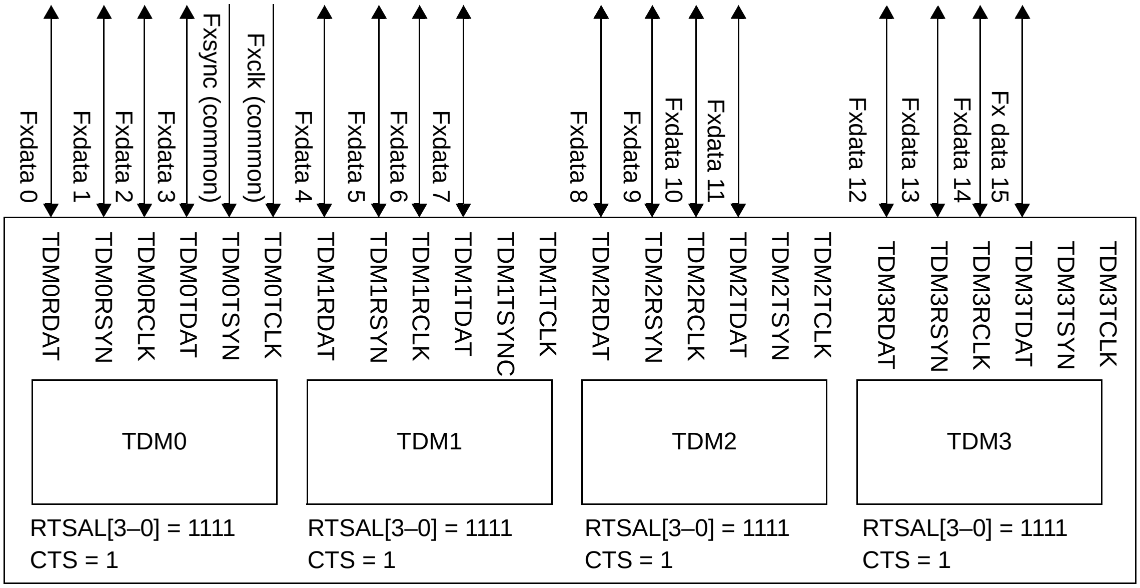


Figure 20-10. Shared Frame Sync, Clock, and Data Links

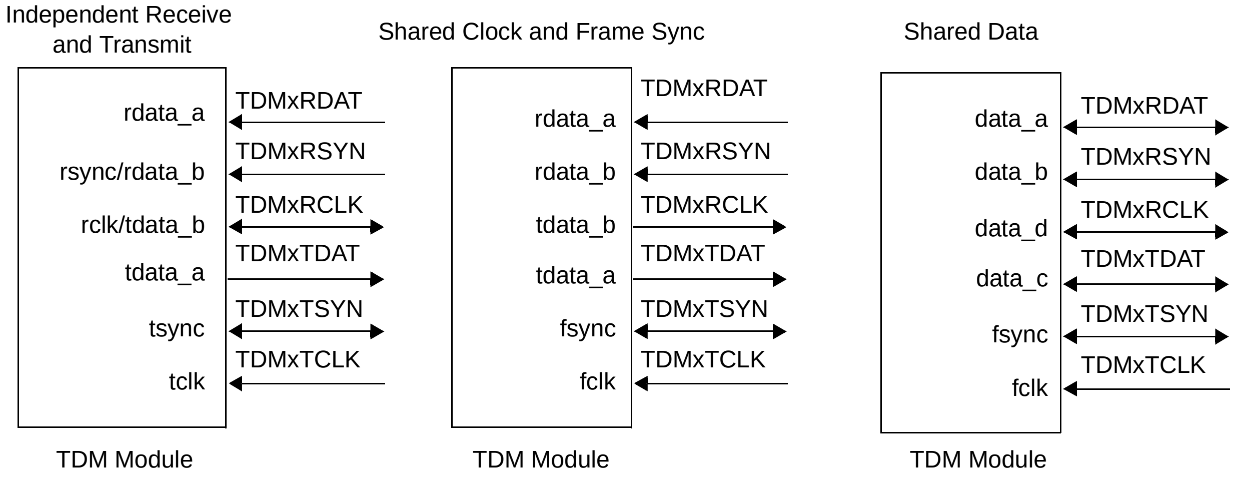
20.2.2 Receiver and Transmitter Independent or Shared Operation

The TDM operates with the transmit and receive operations running either independently or shared, as illustrated in **Figure 20-11**. When the two most significant bits of the RTSAL field in the TDMx General Interface Register (see page 20-36) equal 0b00, the receive and the transmit are independent as illustrated on the left side of **Figure 20-11**. In this mode, there is one input receive data link and one output transmit data link. If the TDM shares signals with other TDM modules (TCS = 1), it can receive two data links and it can output two data links.

When bits 3 and 2 of the RTSAL field in the TDMx General Interface Register (see page 20-36) equal 0b01, the receive and transmit are shared as illustrated in the middle of **Figure 20-11**. The transmit and the receive share the Frame Sync (FSYN) and the Frame Clock (FCLK) signals. The number of receive and the transmit active links can be one or two. The direction of the receive links is input, and the direction of the transmit links is output.

When bits 3 and 2 of the RTSAL field in the TDMx General Interface Register (see page 20-36) equal 0b11, the receive and the transmit are shared as illustrated on the right side of **Figure 20-11**. The transmit and the receive share the Frame Sync (FSYN), the Frame Clock (FCLK), and the data signals. In this mode, the data links are full duplex and are used for both transmit and receive, so the number of active links can be 1, 2, or 4.

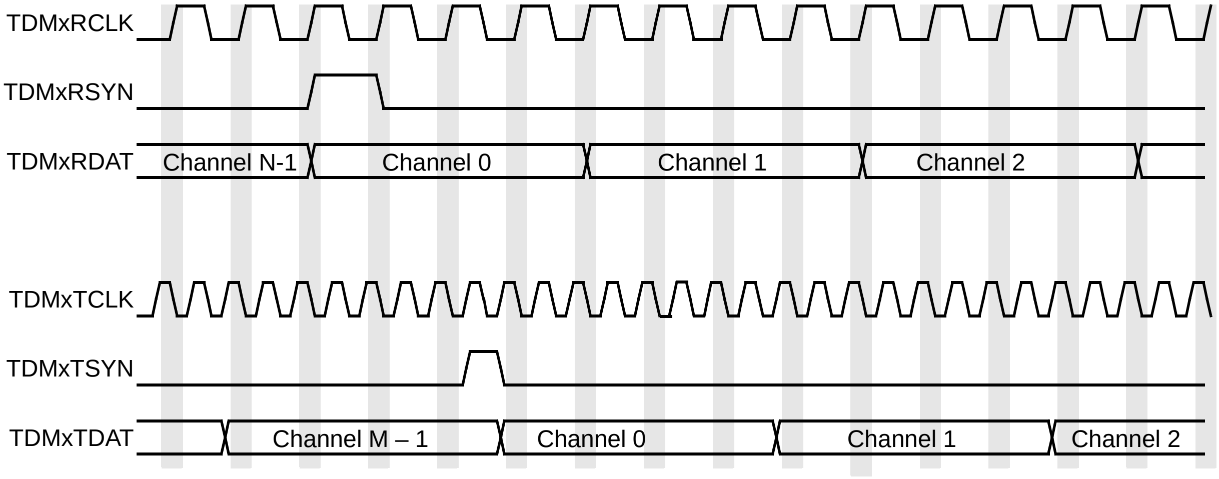
When RTSAL [1–0] equals 0b11, there are four active links: DATA_A, DATA_B, DATA_C, and DATA_D. When RTSAL[1–0] equals 0b01, there are two active links: DATA_A and DATA_B. When RTSAL[1–0] equals 0b00, there is one active link, DATA_A.



x Defines the TDM number.
FSYNC (frame sync) specifies that the receiver and transmitter share the same sync.
FCLK (frame clock) specifies that the receiver and transmitter share the same clock.

Figure 20-11. TDM Module Sharing Modes

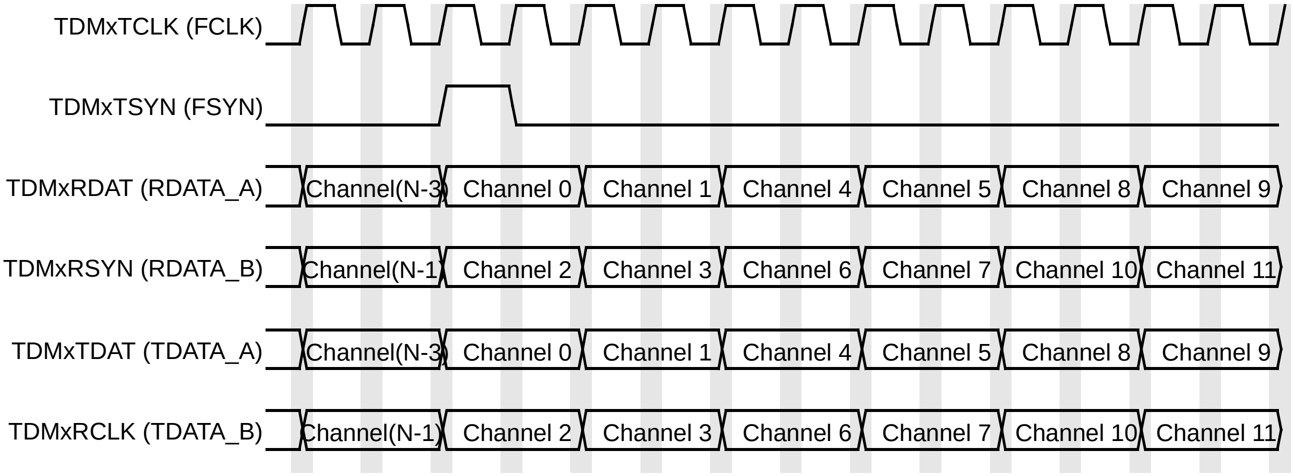
Figure 20-12 describes the TDM interface when the receive and transmit are totally independent (TDMxGIR[RTSAL] = 0b0000). The TDMxRCLK is not synchronized to the TDMxTCLK. They differ according to the sync location relative to the beginning of the frame and the number of bits.



X The TDM number.
N The number of channel in the receive TDM frame.
M The number of channels in the transmit TDM frame.

Figure 20-12. Receive and Transmit Totally Independent

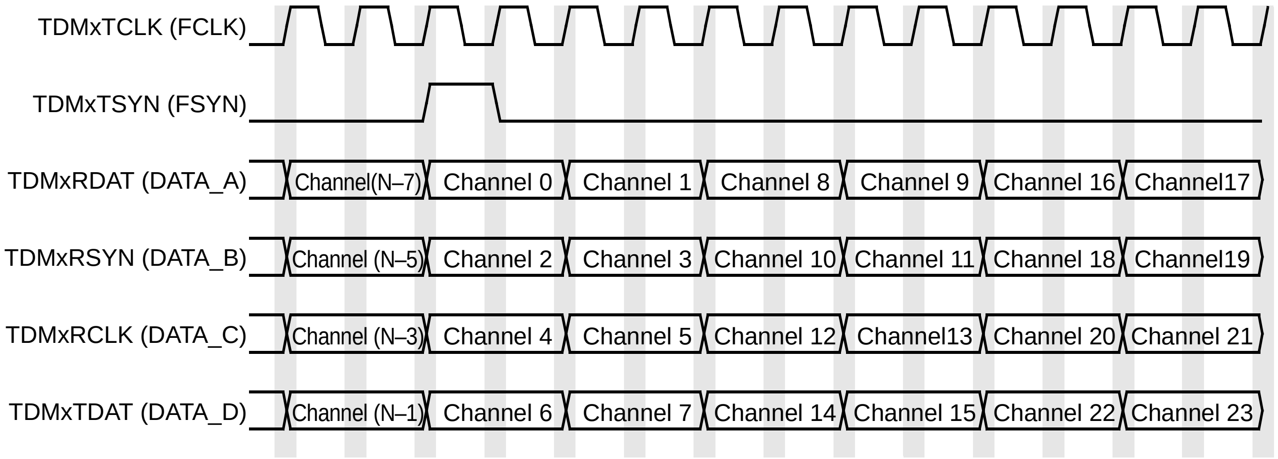
Figure 20-13 describes the TDM timing interface when TDMxGIR[RTSAL] = 0b0101. The frame sync and the clock is shared between the receive and transmit, and links A and links B are active. RDAT and RSYN are used as received data links, and TDAT and RCLK are used as transmit data links. Channels are organized in pairs: channel i and channel i+1 are always received/transmitted on the same link, one after another.



x The TDM number.
N The number of channels in a TDM frame.

Figure 20-13. Shared Sync and Clock (Two Active Data Links)

Figure 20-14 shows the TDM timing interface when the RTSAL field is set to 0b1111.The frame sync, the clock, and the data links are common. All four data links are active and are used for both transmit and receive.



x The TDM number.

N Number of channels in a TDM frame.

The data links are bidirectional.

The clock and the sync are common.

Figure 20-14. Receive and Transmit Share Sync, Clock, and Data (Four Active Links)

Note: The number of channels in a frame must be a multiple of 2 × the number of data links.

Table 20-1 shows the number of channels each active link supports.

Table 20-1. Maximum Number of Channels Per Active Link

Number of Active Links	1 Active Link	2 Active Links	4 Active Links
Maximum channel number per active link in one TDM module	256	128	64

The TDM bit rate depends on:

- *System bus clock*. The TDM processes the data with the system bus clock, so the maximum data bit rate is limited to half the rate of the system bus clock.
- *Number of active links*. The total bit rate is shared by all active links, so one active link supports the highest bit rate.
- *Channel width*. When there are more bits per channel, there are fewer channels per second, so higher bit rates can be processed per second

Table 20-2 describes the maximum bit rate as a function of these parameters. Factors other than the width of the channel can affect the bit rate, for example, capacity on the data links.

Table 20-2. Factors Affecting Maximum Bit Rate

Channel Width (Bits)	1 Active Link	2 Active Links	4 Active Links
2	BUSES_CLOCK/8	BUSES_CLOCK/12	BUSES_CLOCK/20
4	BUSES_CLOCK/4	BUSES_CLOCK/6	BUSES_CLOCK/10
8	BUSES_CLOCK/2	BUSES_CLOCK/3	BUSES_CLOCK/5
16	BUSES_CLOCK/2	BUSES_CLOCK/2	BUSES_CLOCK/2.5

Table 20-3 describes the maximum TDM bit rate when the system bus clock works at its typical value of 133 MHz.

Table 20-3. Maximum Bit Rate When the System Bus Clock Runs at 133 MHz

Channel Width (Bits)	1 Active Link	2 Active Links	4 Active Links
2	16.63 Mbps	11.08 Mbps	6.65 Mbps
4	33.25 Mbps	22.17 Mbps	13.3 Mbps
8	66.5 Mbps	44.3 Mbps	26.6 Mbps
16	66.5 Mbps	66.5 Mbps	53.2 Mbps

20.2.3 TDM Data Structures

TDM data structures are stored in transmit and receive local memory, as follows:

- *TDM receive local memory*. Received data is stored in 256 8-byte entries located in addresses between 0x0000–0x07FF, which is offset from the TDMx receive local memory (see **Chapter 8**, *Memory Map*). This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive Number of Buffers Register (TDMxRNB) (discussed on page 20-64). Channel C in buffer B is the 8 bytes starting at (256/(RNB + 1) × B + C) × 8.
- *TDM transmit local memory*. Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800– 0x1FFF, which is offset from the TDMx receive local memory (see **Chapter 8**, *Memory Map*). This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at (256/(TNB + 1) × B + C) × 8.

Figure 20-15 shows an example of TDM local memory that contains four transmit buffers and one receive buffer. Up to 32 transmit bytes of channel 2 are located in four buffers (TNB = 3). Only 8 receive bytes of channel 2 are located in one buffer (RNB = 0). Each buffer contains 8 bytes per channel.

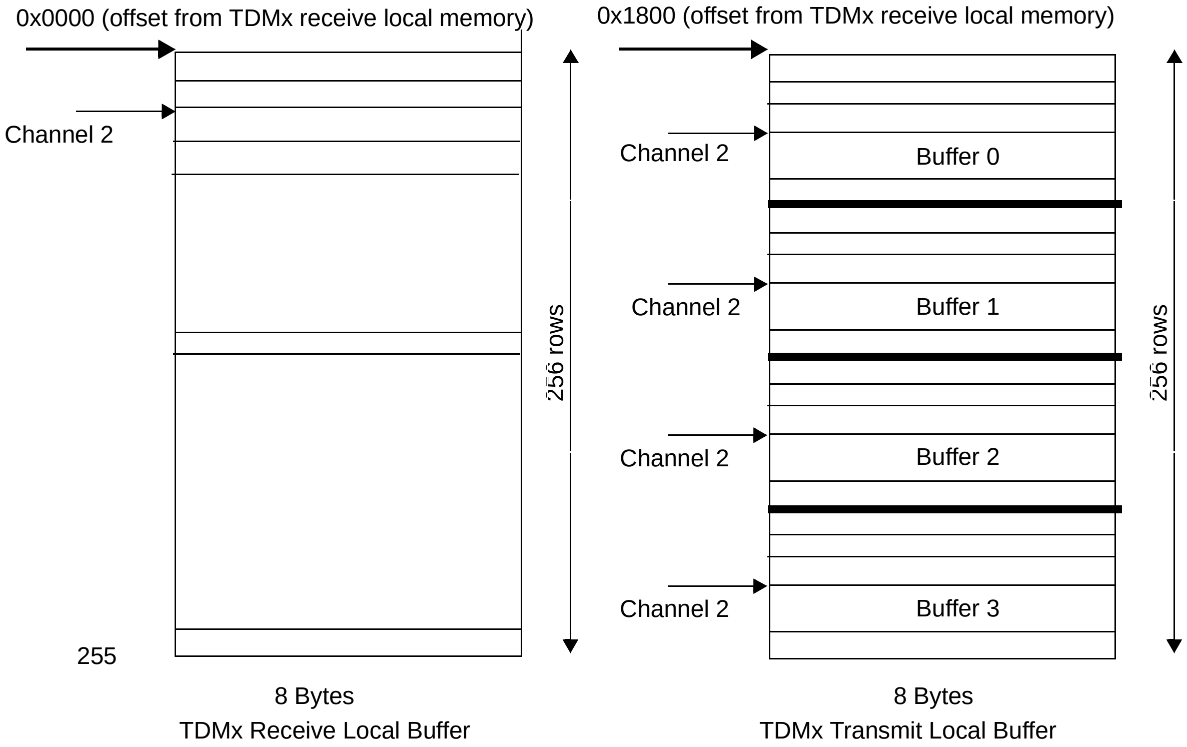

0x0000 (offset from TDMx receive local memory)
Channel 2
255
256 rows
8 Bytes
TDMx Receive Local Buffer
0x1800 (offset from TDMx receive local memory)
Channel 2
Buffer 0
Channel 2
Buffer 1
Channel 2
Buffer 2
Channel 2
Buffer 3
256 rows
8 Bytes
TDMx Transmit Local Buffer

Figure 20-15. TDM Local Buffer (Receive and Transmit)

When the TDM transmit local memory is accessed through the IPBus interface to addresses with 8-byte alignment, data is written to the 4 LSB of the memory row. **Figure 20-16** describes the TDMx local memory after write access of 0x01234567 to address 0x1800 (offset from TDMx Receive Local Memory) and 0x89ABCDEF to address 0x1804 (offset from TDMx Receive Local Memory) through the IPBus interface. In this case, the 0x89ABCDEF data is transmitted before the 0x01234567 data.

	8 Bytes															
0x1800	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7

Figure 20-16. TDMx Local Memory Write Example

When the TDM receive local memory is accessed through the IPBus interface from addresses with 8-byte alignment, data is read from the 4 LSB of the memory row. **Figure 20-17** describes a row in the TDMx local memory, in which the 0x00112233 data is received before the 0x44556677 data. In this example, the data to be read through the IPBus interface from address 0x1000 (offset from TDMx Receive Local Memory) is 0x44556677, and the data to be read from address 0x1004 (offset from TDMx Receive Local Memory) is 0x00112233.

	8 Bytes															
0x1000	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7

Figure 20-17. TDMx Local Memory Read Example

20.2.4 Serial Interface

This section covers issues related to the serial interface, such as how to configure the frame sync and how to control the data order of the bits in the channel.

20.2.4.1 Sync Out Configuration

TDMxTSYN is programmed as either an input or output by writing 1 to the TSO bit in the Transmit Interface Register (TDMxTIR) (see page 20-45). When the TSO bit value is equal to 1, the sync_out signal connects to the sync out signal in the TDM I/O matrix and is output via TDMxTSYN. When the TDM modules share a sync and clock signals (the CTS bit is set), then the TDMx[TSO] bits should be equal for all the TDM modules and they determine whether the sync arrives from the board or is generated by the TDM0 transmitter. Configuring the sync out signal involves the parameters listed in **Table 20-4**.

Table 20-4. Parameters in Configuring the Frame Sync (TDMxTIR[TSO] = 1)

Task	Register
Control the length of the sync_out signal. If the SOL bit is clear then the sync_out width is one transmit bit, else the sync_out length is one transmit channel.	TDMxTIR[SOL], **page 20-45**
Control the transmit clock edge on which the sync_out is driven out. If the SOE bit is clear, the sync_out is driven out on the rising edge of the transmit clock.	TDMxTIR[SOE], **page 20-45**
Control the sync_out level. The sync out level must be identical to the transmit sync. It is determined by the TSL configuration field.	TDMxTIR[TSL], **ppage 20-45**
Control the sync_out distance. The distance between two consecutive sync out events is constant and equal to one transmit frame. The transmit frame length is determined by the transmitter configuration fields TCS,TNCF, TT1 and RTSAL[1–0]. The distance is = (TCS + 1) × (TNCF + 1) / (RTSAL[1–0] + 1) + TT1.	TDMxTFP[TNCF], **page 20-49** TDMxTFP[TCS], **page 20-49** TDMxTFP[TT1], **page 20-49** TDMxGIR[RTSAL], **page 20-36**

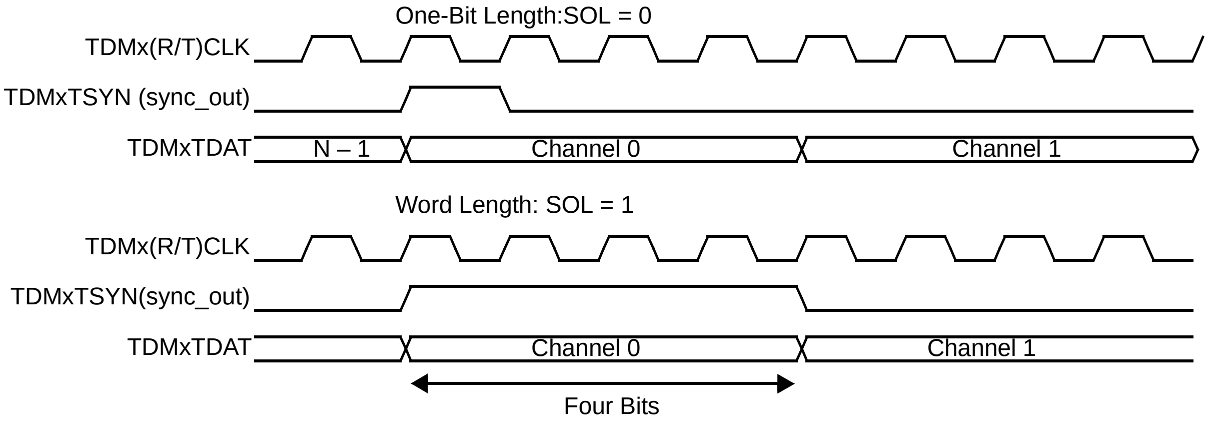


Figure 20-18. Sync Length Selection

20.2.4.2 Sync In Configuration

TDMxRSYN is an input that identifies the beginning of the received frame. TDMxTSYN can be an input or output from the TDM, but the transmitter refers to the transmit sync as an input because the connection between the sync_out signal and the transmit sync (tsync) occurs only in the TDM I/O matrix. **Figure 20-19** illustrates the relation between the data, the sync, and the clock for various configurations. The receive data and frame sync are sampled with the rising or falling edge of the receive clock. The transmit frame sync is sampled with the rising or falling edge of the transmit clock. The transmit data drives out at the rising or falling edge of the transmit clock. The delay between the first data bit of the frame and the sync is referred to as the rising edge of the sync. **Table 20-5** lists the frame sync controls.

Table 20-5. Transmit and Receive Frame Configuration

Control	Register
Which receive clock edge samples the receive frame sync. If RFSE is clear, the receive frame sync is sampled on the rising edge of the receive clock.	TDMxRIR[RFSE] bit, page 20-43
Which transmit clock edge samples the transmit frame sync. If TFSE is clear, the transmit frame sync is sampled on the rising edge of the transmit clock.	TDMxTIR[TFSE] bit, page 20-45
Which receive clock samples the receive data. If RDE is clear, the receive data sync is sampled on the rising edge of the receive clock.	TDMxRIR[RDE] bit, page 20-43
Which transmit clock edge drives out the data. If TDE is clear, then the transmit data is driven out on the rising edge of the transmit clock.	TDMxTIR[TDE] bit, page 20-45
Determines the receive sync level. If RSL is clear the receive sync level is high.	TDMxRIR[RSL] bit, page 20-43
Determines the transmit sync level. If TSL is clear the transmit sync level is high.	TDMxTIR[TSL] bit, page 20-45
Determines the timing of the receive frame sync signal relative to the first data bit of the receive frame.	TDMxRIR[RFSD] field, page 20-43
Determines the timing of the transmit frame sync signal relative to the first data bit of the transmit frame.	TDMxTIR[TFSD] field, page 20-45

The receive delay when the receive sync and the receive data are not sampled at the same clock edge is RFSD + 0.5. The transmit data can be driven out before the transmit sync sample. Therefore, the transmit delay when the transmit sync and transmit data are sampled/driven out at the same clock edge is (TFSD – 1). And when the sync and the data sampled/driven out at different clock edge is (TFSD – 1 + 0.5).

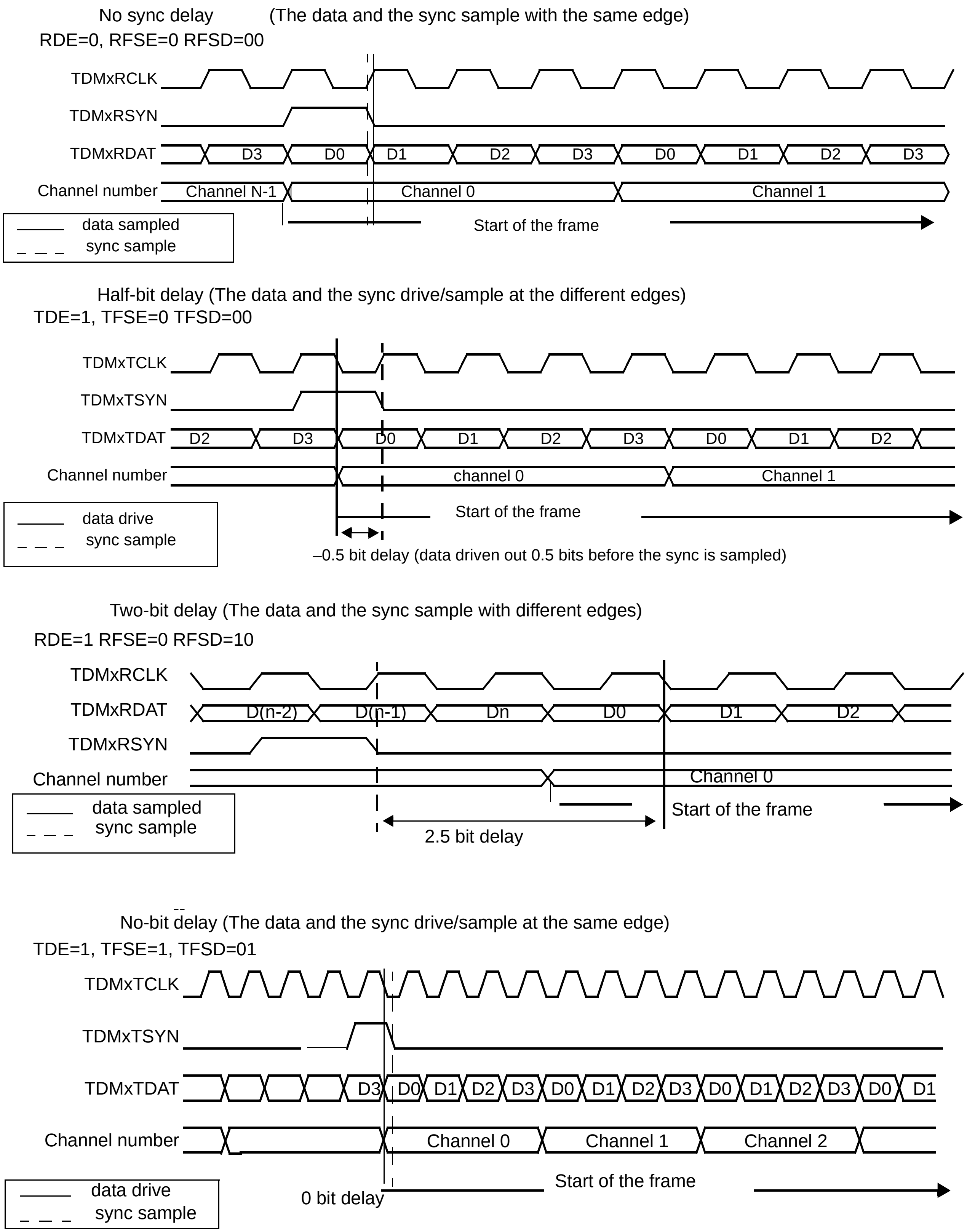


Figure 20-19. Frame Sync Configurations

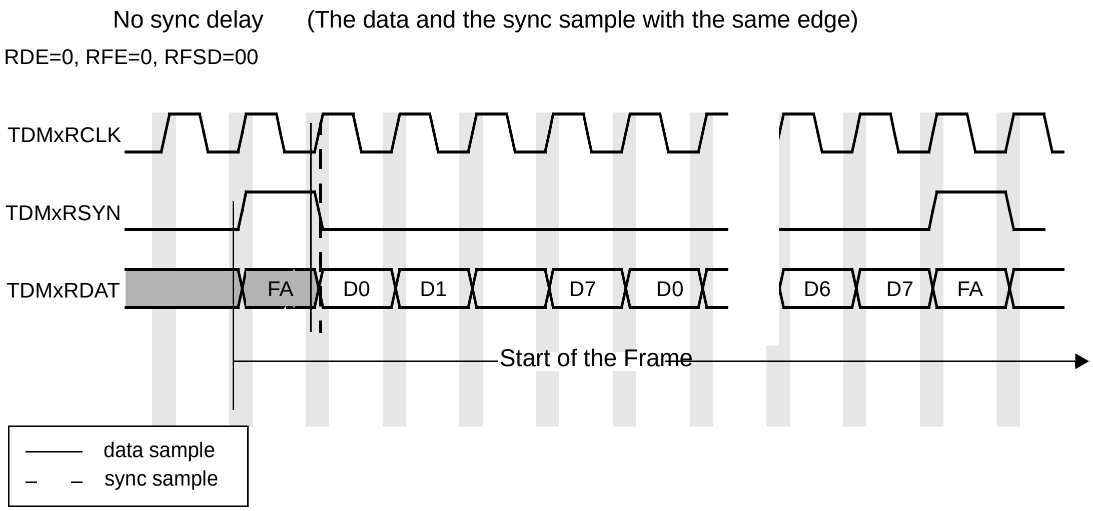


Figure 20-20. Frame Sync Configuration At T1 Mode

Figure 20-21 illustrates how the polarity of the receive sync and the transmit sync signals is controlled by the TDMxRIR[RSL] and the TDMxTIR[TSL] bits.

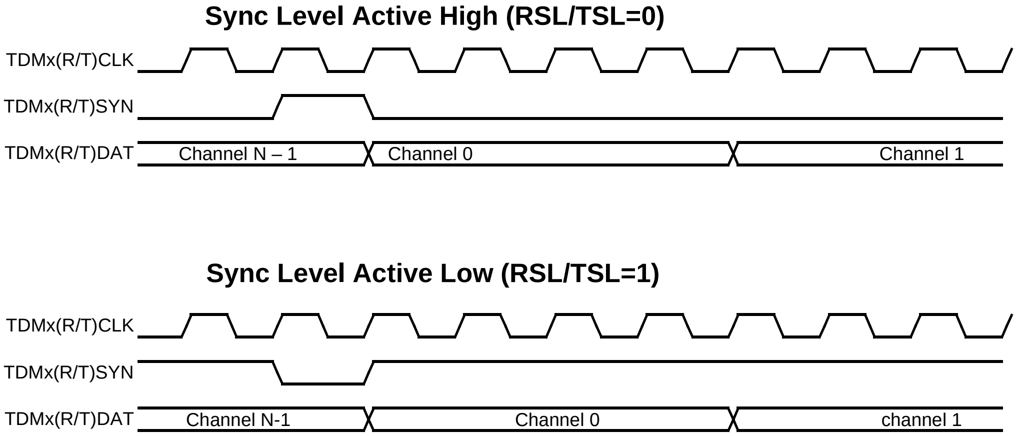


Figure 20-21. Frame Sync Polarity

20.2.4.3 Serial Interface Synchronization

The TDM module enables communication among many devices over a single bus. The receive and transmit of each TDM frame is identified by a frame sync signal that is asserted at the beginning of every frame. The frame sync synchronization is necessary when more than one device drives the bus. **Figure 20-22** shows the state diagram of the frame sync synchronization.

The details of the state diagram are as follows:

- *HUNT* (0b00). A sync event is constantly sought. As soon the sync event is detected, the state machine changes to a WAIT state. During the Hunt state, data is neither received nor transmitted.
- *WAIT* (0b01). At least one sync has been detected. The next sync event is accepted after one TDM frame. If the sync appears in the correct position, the state changes to the PRESYNC state (0b11). If the sync does not appear, the state returns to the hunt state. During the WAIT state, data is neither received nor transmitted.
- *PRESYNC* (0b11). Two sync events have been detected and the distance between the syncs is one TDM frame. If the sync event is recognized early, the state returns to the WAIT state. Otherwise, the machine transfers to the SYNC state at the last bit of the TDM frame. During PRESYNC state, data is neither received nor transmitted.
- *SYNC* (0b10). At least one sync event has appeared exactly where it was expected. This state is maintained as long as the sync event continues to appear where expected. If a sync is missed or a sync event is recognized early, the state changes to the HUNT state (0b00). During the SYNC state, data is both received and transferred.

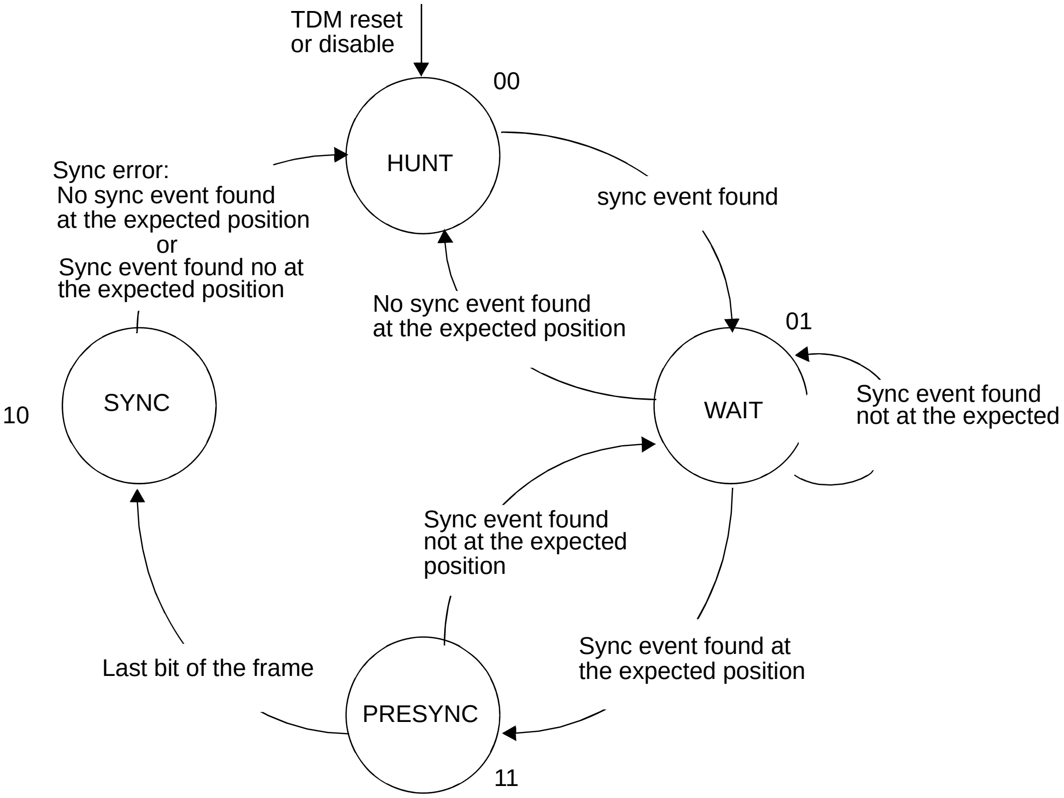


Figure 20-22. Frame Sync Synchronization State Diagram

The TDM receiver synchronizes on the receive frame sync (rsync). The state of the receive frame sync synchronization is indicated by the TDMxRSR[RSSS] field (see page 20-67). During the HUNT, WAIT, and PRESYNC states, the received data is not transferred to the buffers in the main memory for processing. When the receive sync synchronization is lost, the state transfers from SYNC to HUNT (the TDMxRER[RSE] bit is asserted) (see page 20-65). If the TDMxRIER[RSEEE] bit (see page 20-60) is also set, a receive error interrupt is generated. The transmit frame sync synchronization state is indicated by the TDMxTSR[TSSS] field (see page 20-68). During the HUNT, WAIT, and PRESYNC states, new data is not driven out. If the Transmit Always Out (TDMxTIR[TAO]) field (see page 20-45) is set, then the last data is driven out until the frame sync synchronization state returns to SYNC state. If the TDMxTIR[TAO] bit is clear, data is not driven out and TDMxTDAT is tri-stated. When the transmit sync synchronization is lost, the TDMxTER[TSE] bit (see page 20-66) is asserted. If the TDMxTIER[TSEIE] bit (see page 20-61) is also set, a transmit error interrupt is generated. The frame sync synchronization state can identify different problems. In the initial design stages, the frame sync summarization state indicates whether the TDM programming matches the actual TDM stream. During operation, the synchronization state and the error interrupts may indicate errors in the TDM module signal processing.

Note: Error interrupts from the TDM are driven directly to the LIC by the TDMx Receive Error Interrupt and TDMx Transmit Error Interrupt bits (TDMxRER[RSE] and TDMxTER[TSE]) when the interrupts are enabled. Therefore, the interrupt handler should clear these bits by writing a 1 to them before clearing the LIC-related status register and before returning to normal operation (exiting the interrupt handler).

20.2.4.4 Reverse Data Order

Figure 20-23 illustrates how the bit order of the stored data relates to the bit order of the receive or the transmit data. The TDMxRIR[RRDO] bit defines how the receive channel data is stored in memory. If TDMxRIR[RRDO] is clear, the first bit of the received channel data is stored as the most significant bit. The TDMxTIR[TRDO] bit selects the transmit data bits order. If

TDMxTIR[TRDO] is clear, the most significant bit of the memory is transmitted as the first transmit data.

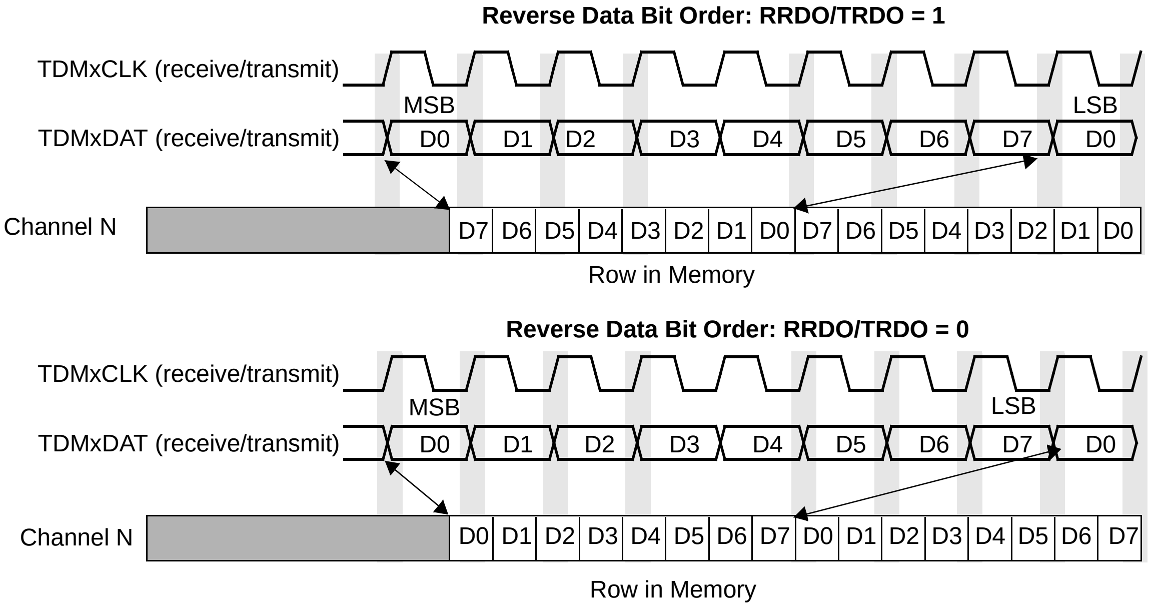

Reverse Data Bit Order: RRDO/TRDO = 1
TDMxCLK (receive/transmit)
TDMxDAT (receive/transmit)
MSB
LSB
D0 D1 D2 D3 D4 D5 D6 D7 D0
Channel N
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
Row in Memory
Reverse Data Bit Order: RRDO/TRDO = 0
TDMxCLK (receive/transmit)
TDMxDAT (receive/transmit)
MSB
LSB
D0 D1 D2 D3 D4 D5 D6 D7 D0
Channel N
D0 D1 D2 D3 D4 D5 D6 D7 D0 D1 D2 D3 D4 D5 D6 D7
Row in Memory

Figure 20-23. Reserve Bit Order

20.2.5 TDM Local Memory

Received data is temporarily stored in the TDM receive local memory until it is transferred to the receive buffers mapped on the local bus. A single data transfer from TDM local memory to a memory-mapped region on the local bus transfers at least 64 bits of data. Each channel can store more than 64 bits before the data is written to the buffers mapped on the local bus. The TDMxRFP[RCDBL] field (see page 20-47) provides an upper boundary on the number of receive bits that can be stored in the TDM local memory. The receive data latency is defined as the time between receiving data and the time when it is available for processing by an SC140 core. Reducing the TDMxRFP[RCDBL] value reduces the receive data latency. However, reading the TDM local memory imposes more strict latency requirements on the local bus. The maximum receive data latency is calculated as maximum receive data latency = RCDBL / (RCS) × (receive frame time).

When the amount of received data exceeds the size of the TDM receive local memory, the TDMxRER[OLBE] bit is set (see page 20-65). If the TDMxRIER[OLBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the local bus and therefore cannot write the data into the destination memory (the data buffer).

Data transmitted from memories mapped on the local bus is temporarily stored in the TDM transmit local memory until it is transferred externally. A single data transfer from the local bus to TDM local memory transfers at least 64 bits of data. Each channel can store more than 64 bits before it is transmitted externally. The TDMxTFP[TCDBL] field provides an upper boundary on the number of transmit bits that can be stored in TDM local memory. The transmit data latency is defined as the time between when the data is read from the buffers mapped to the local bus and when it is transmitted externally. Reducing the TDMxTFP[TCDBL] value reduces the transmit data latency. However, writing the TDM local memory imposes more strict latency requirements on the local bus. The maximum transmit data latency is calculated as maximum transmit data latency = TCDBL / (TCS) × (transmit frame time).

When the TDM cannot transfer data from data buffers to TDM local memory, an underrun occurs. When the TDM transmit local memory is empty, the TDMxTER[ULBE] bit (see page 20-66) is set and the TDMxTIER[ULBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the local bus and therefore cannot read the data from the source memory into TDM transmit local memory. The minimum latency is achieved when the RCDBL/TCDBL field is clear (only 64 bits are stored in the TDM local memory). For example, the minimum latency for a T1 application with 8 bits per channel and a frame length of 125 µs is equal to 1 µs. T1 minimum latency= 64/8 × 125 µs.

20.2.6 Buffers Mapped on the Local Bus

Each receive or transmit data channel is stored in a different buffer mapped on the internal local bus. This buffer can be located in the M1 memory of one of the SC140 cores or in the M2 memory, which is shared by all the SC140 cores.

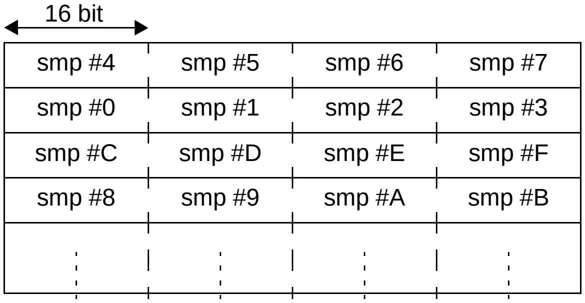
20.2.6.1 Data Buffer Size and A/µ-law Channels

Data buffer size is identical for all receive channels belonging to a TDM module and is indicated in the TDMxRDBS[RDBS] field. Data buffer size is also identical for all the transmit data buffers and is indicated in the TDMxTDBS[TDBS] field (see page 20-51). An exception is the A/µ-law channels (buffer size × 2). When the TDMxRCPRn[RCONV] field (see page 20-58) indicates that a channel is an A-law channel, the received 8 bits are converted into a 13-bit PCM sample padded with three zeros on the right. This channel therefore occupies 16 bits per 8 received bits, essentially occupying double the size. When the TDMxRCPRn[RCONV] field indicates that a channel is a µ-law channel, the received 8 bits are converted into a 14-bit PCM sample padded with two zeros on the right. This channel also occupies 16 bits per 8 received bits, essentially occupying double the size.

When the TDMxTCPRn[TCONV] field (see page 20-59) indicates that a channel is an A-law channel, the transmitted 13 bits are converted into an 8-bit PCM sample. This channel therefore occupies 16 bits (13 bits padded with three zeros at the right) per 8 transmit bits, essentially occupying double the size. When the TDMxTCPRn[TCONV] field indicates that a channel is a µ-law channel, the received 14 bits are converted into an 8-bit PCM sample. This channel also occupies 16 bits (14 bits padded with two zeros at the right) per 8 transmit bits, essentially occupying double the size. The A/μ-law conversion is performed according to the ITU-T recommendation G.711.

Note: The minimum buffer size for both transmit and receive is 16 bytes (that is, the RDBS/TDBS value is 0x00000F). The maximum buffer size for both transmit and receive is 16 MB (that is, the RDBS/TDBS value is 0xFFFFFF), but it can be further limited by the main memory size according to the number of channels in the frame.

Figure 20-24 shows how the samples are stored in the receive main data buffer (if the receive channel is A/μ law) or the transmit main data buffer (if the transmit channel is A/μ law).



smp #4	smp #5	smp #6	smp #7
smp #0	smp #1	smp #2	smp #3
smp #C	smp #D	smp #E	smp #F
smp #8	smp #9	smp #A	smp #B
⋮	⋮	⋮	⋮

Figure 20-24. Receive/Transmit Main Data Buffer For A-Law/μ-Law Channel

20.2.6.2 Data Buffer Address

The address of a receive buffer is a function of the following:

- *Receive Global Base Address*. TDMxRGBA[RGBA], page 20-52.
- *Receive Channel Data Base Address*. TDMxRCPRn[RCDBA] field, page 20-58. RGBA << 16 + RCDBA points to the first byte of receive data buffer *n*. The four lsbs of RCDBA must be 0000.
- *Receive Data Buffer Displacement*. TDMxRDBDR[RDBD] field, page 20-62. Adding this field to the first byte of receive data buffer *n* indicates the location to which the TDM will write next: RGBA << 16 + RCDBA + RDBD is the current write pointer to the receive data buffer *n*. In most cases, the RDBD can be used to indicate that data is written to the buffer and can be processed. However, in some cases in which the local bus is extremely busy and the TDM priority is low, the pointer may be updated before the data is actually to the internal memory. This typically affects the last channel transmitted. In all cases, when configured to reflect the last buffer configuration, assertion of the receive buffer threshold interrupt indicates that the data was updated in memory.

The address of a transmit buffer is a function of the following:

- *Transmit Global Base Address*. TDMxTGBA[TGBA] field, page 20-52
- *Transmit Channel Data Base Address*. TDMxTCPRn[TCDBA] field, page 20-59. TGBA << 16 + TCDBA points to the first byte of transmit data buffer *n*. The four lsbs of TCDBA must be 0000.

- *Transmit Data Buffer Displacement*. TDMxTDBDR[TDBD] field, page 20-63. Adding this field to the first byte of transmit data buffer *n* indicates the location to which the TDM will read next: TGBA << 16 + TCDBA + TDBD is the current read pointer of transmit data buffer *n*. The TDBD can be used to show which data is already read from the buffer so that the buffer can be filled with new data.

Note: For A/µ-law channels the RDBD and the TDBD fields should be doubled before use.

Figure 20-25 illustrates the pointers associated with receive and transmit buffers that are mapped on the local bus.

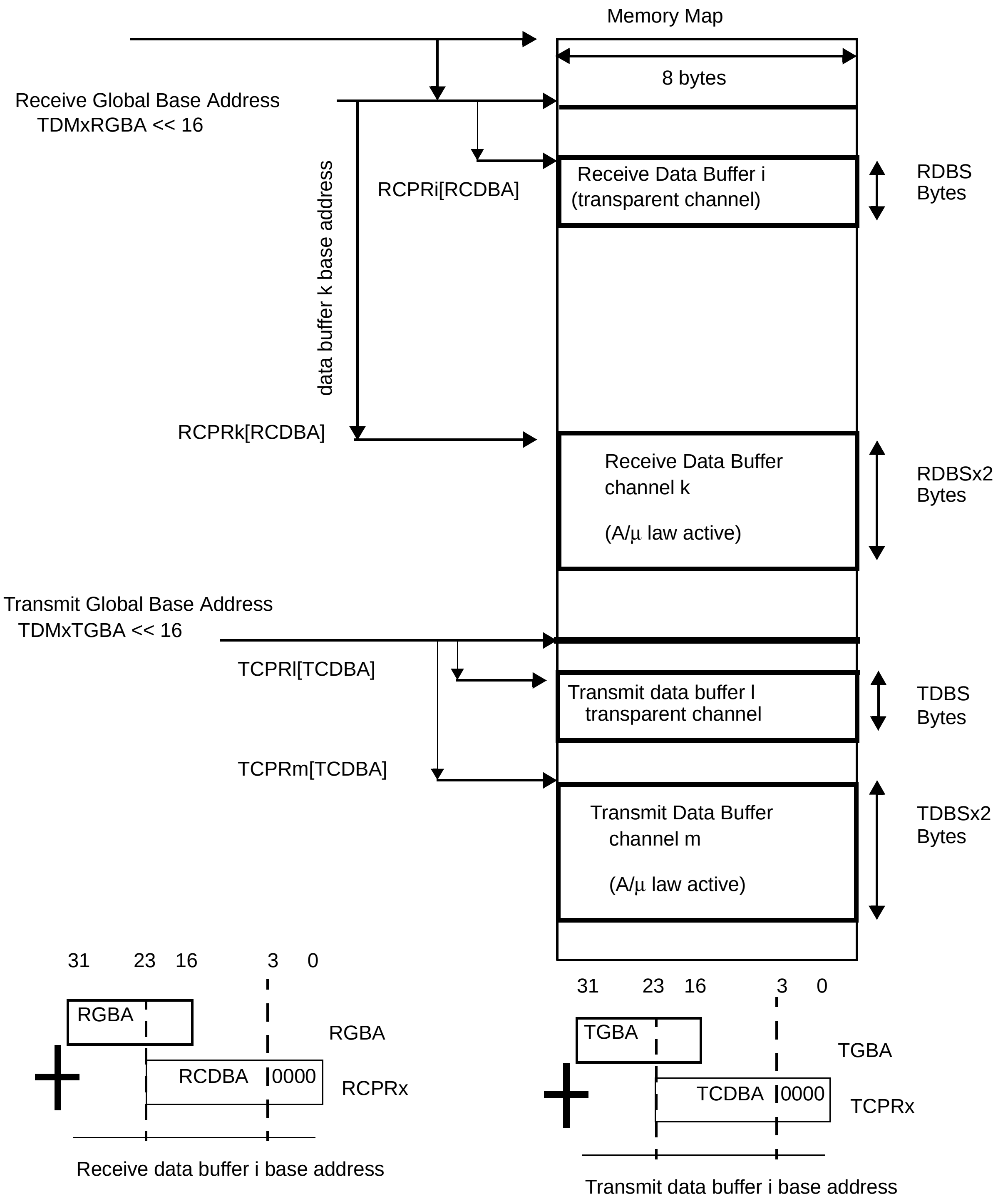


Figure 20-25. Data Buffer Location in Main Memory

20.2.6.3 Threshold Pointers and Interrupts

The receive data buffers share two threshold levels. The TDM notifies the SC140 core each time it fills the receive buffer up to a threshold level. An example use of thresholds is the implementation of double buffering with the first threshold in the middle of a buffer and the second at the last eight bytes of the buffer.

When the TDM receiver fills the receive buffer in the local bus to an offset defined by the first threshold, which is the TDMxRDBFT[RDBFT] field (see page 20-55), the TDMxRER[RFTE] bit is set. If the TDMxRIER[FTREE] bit is also set, a first threshold interrupt is generated. The interrupt can generate as pulse or level, as determined by the TDMxRIR[RFTL] bit. The SC140 core can now read all the receive buffers from their beginning up to the byte to which the first threshold (RDBFT) points. Meanwhile, the TDM keeps writing new data to the second part of the buffer.

When the TDM receiver fills the receive buffer in the local bus up to an offset defined by the second threshold, which is the TDMxRDBST[RDBST] field (see page 20-57), the TDMxRER[RSTE] bit is set. If the TDMxRIER[RSEEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxRIR[RSTL] bit. The SC140 core can now read all the receive buffers up to the byte to which the second threshold (TDMxRDBST[RDBST]) points. Meanwhile, the TDM keeps writing new data to the first part of the buffer.

Note: The TDMxRER[RFTE] and TDMxRER[RSTE] bits are set and the associated interrupts are generated when the TDM performs the first access after the buffers reach the associated threshold level and not immediately when the threshold is reached.

The transmit data buffers also share two threshold levels. The TDM notifies the SC140 core each time it reads from the transmit buffer to a threshold level. When the TDM transmitter reads the transmit buffer in the local bus to an offset defined by the first threshold, which is the TDMxTDBFT[TDBFT] field, the TDMxTER[TFTE] bit is set. If the TDMxTIER[TFTEE] bit is also set, a first threshold interrupt is generated. The interrupt can generate as pulse or level, as determined by the TDMxTIR[TFTL] bit. The SC140 core can now fill all the transmit buffers from their beginning up to the byte to which the first threshold (TDBFT) points. Meanwhile, the TDM continues reading new data from the second part of the buffer.

When the TDM transmitter reads the transmit buffer in the local bus up to an offset defined by the second threshold, which is the TDMxTDBST[TDBST] field, the TDMxTER[TSTE] bit is set. If the TDMxTEIR[TSTEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxTIR[TSTL] bit. The SC140 core can now fill all the transmit buffers from their beginnings to the byte to which the second threshold (TDBST) points. Meanwhile, the TDM keeps reading new data from the buffer.

The TDM threshold interrupt can be programmed as pulse or level (TDMxRIR[RFTL] and TDMxTIR[TFTL]). When level interrupt is used, you must clear the relevant bit (TDMxRER[RFTE/RSTE] and TDMxTER[TFTE/TSTE]) during interrupt handling. Otherwise, the TDM does not generate a new interrupt when it reaches the threshold the next time. Pulse interrupt mode does not require that these bits be cleared, because in this mode, the TDM continues to generate an interrupt toward the LIC every time it reaches the threshold. The LIC must be programmed to comply with the selected interrupt mode.

Figure 20-26 shows the threshold pointers for transparent and A/µ law channels.

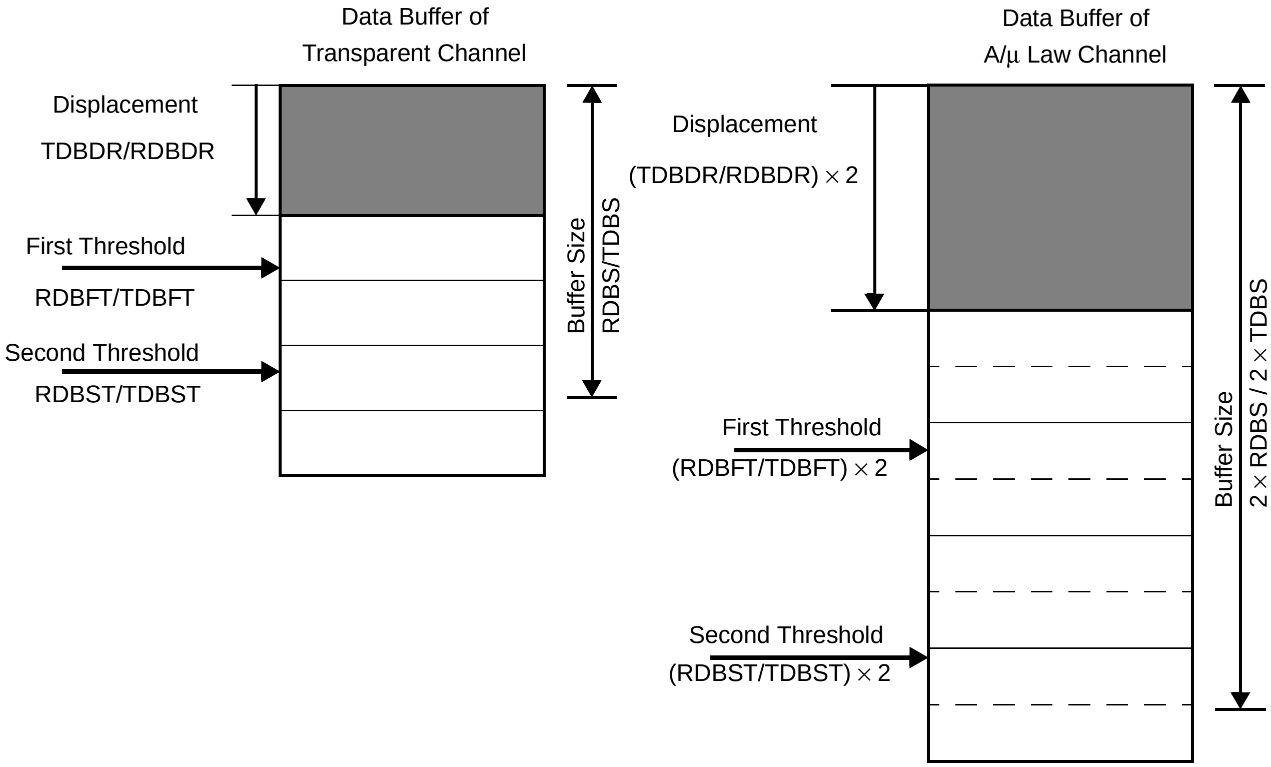


Figure 20-26. Main Memory Buffers Threshold Pointers

The TDMxRDBFT[RDBFT], TDMxRDBST[RDBST], TDMxTDBFT[TDBFT], and TDMxTDBST[TDBST] fields are control fields and can therefore be updated while the TDM is active. For example, to invoke an interrupt for each 64 bits written to the local bus memory, the interrupt routine that handles the receive first threshold interrupt should include:

```
If (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0xF))
     then TDMxRDBFT[RDBFT] = 0x0
else if (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0x7))
     then TDMxRDBFT[RDBFT] = 0x8
else TDMxRDBFT[RDBFT] = TDMxRDBFT[RDBFT] + 0x10
```

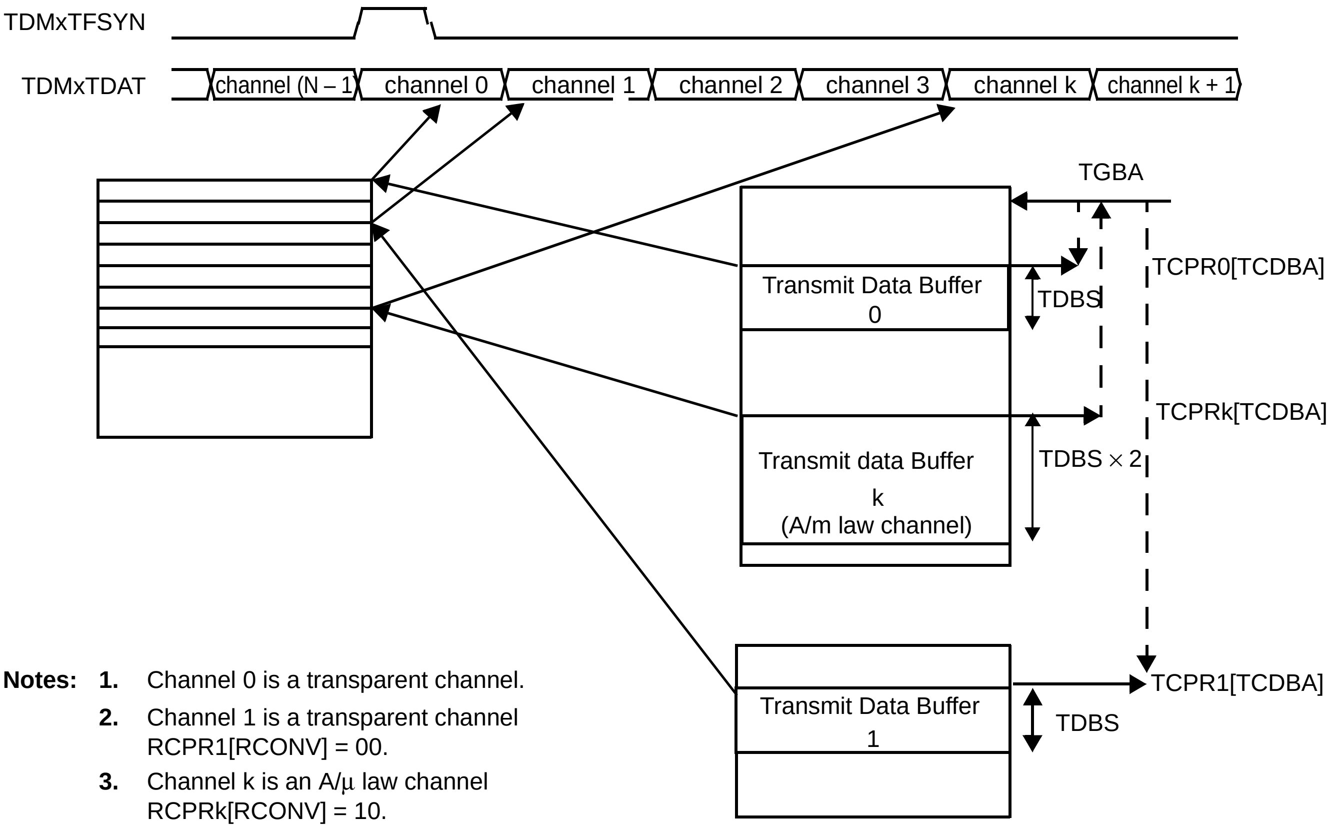
The interrupt routine that handles the receive second threshold interrupt should include:

```
If (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0xF))
    then TDMxRDBST[RDBST] = 0x0
else if (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0x7))
    then TDMxRDBST[RDBST] = 0x8
else TDMxRDBST[RDBST] = TDMxRDBST[RDBST] + 0x10
```

20.2.6.4 Unified Buffer Mode

When the TDMxRFP[RUBM] bit is set (see page 20-47), the two receive channels are directed to one buffer in the local bus. The buffer parameters are stored in the TDMxRCPR0. The number of channels must be two (RNCF = 0x01), the number of active links must be one (RTSAL = 0b0000 or 0b0100 or 0b1100), and the number of bits per channel must be four, eight, or sixteen (RCS = 0b0011 or 0b0111 or 0b1111). The channel parameters of channels 0 and 1 are located in the TDMxRCPR0 register. Unified Buffer mode essentially creates a one-channel link that is typically used in point-to-point connections. When TDMxTFP[TUBM] =1, data is transmitted from one buffer into two transmit channels, each four, eight, or sixteen bits wide.

Figure 20-27 describes the transmit data flow in independent data buffers mode (TDMxTFP[TUBM] = 0). Each data channel transfers data from an independent data buffer to the TDM local memory, and transmit data is read out serially from the local memory.



Notes:
1. Channel 0 is a transparent channel.
2. Channel 1 is a transparent channel RCPR1[RCONV] = 00.
3. Channel k is an A/µ law channel RCPRk[RCONV] = 10.

Figure 20-27. Transmit Data Buffer in Independent Data Buffer Mode (TUBM=0)

Figure 20-28 illustrates the receive data flow in receive unified buffer mode. All the received channels are stored in the TDM local memory and then written into their unified buffer in the local bus.

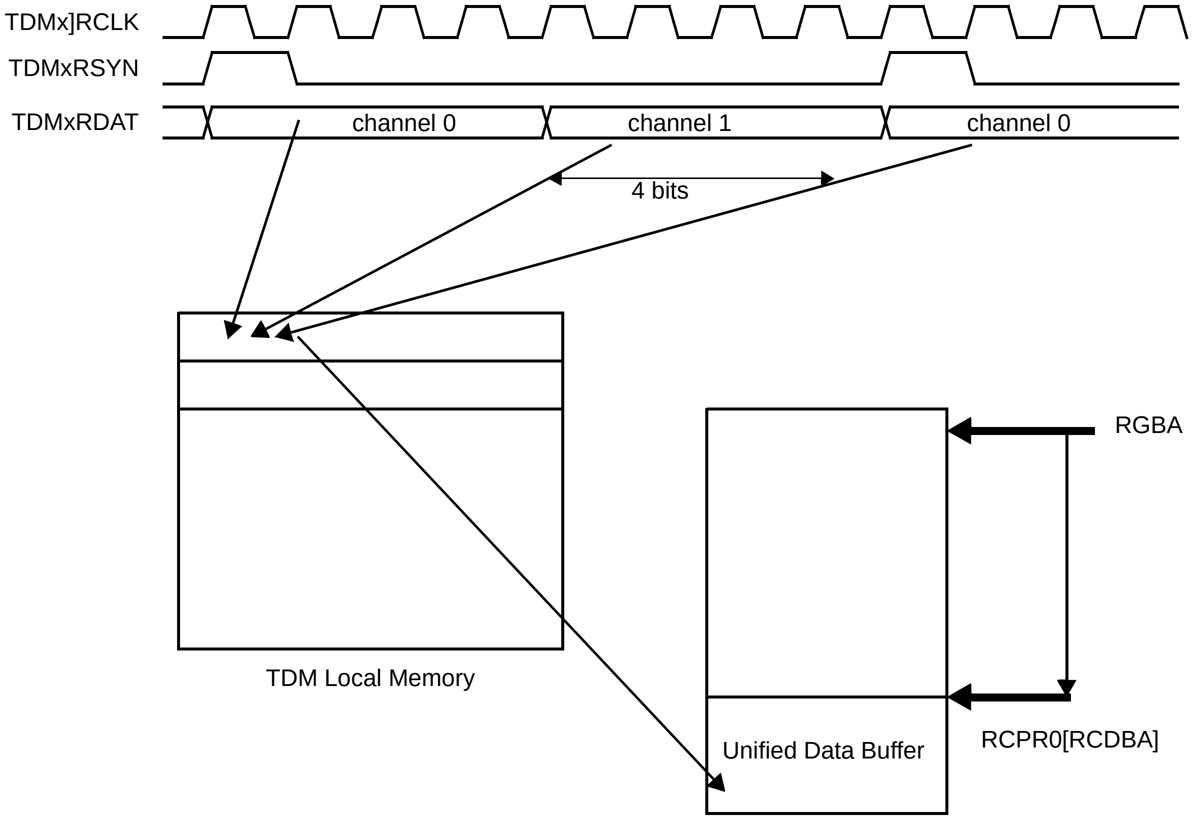


Figure 20-28. Receive Unified Buffer Mode (RUBM = 1)

Note: When the receiver is configured as Unified Buffer Mode, the RRDO bit in the TDMxRIR should be cleared. When the transmitter is configured as Unified Buffer Mode, the TRDO bit in the TDMxTIR should be cleared.

20.2.7 Adaptation Machine

Each TDM module has an adaptation machine that counts the number of bits between frame SYNCs. This module can be used to determine the frame size in bits. MSC8113 boot code uses this module during boot from TDM to determine whether the TDM boot master is a T1 (193 bits) or an E1 (256 bits) interface.

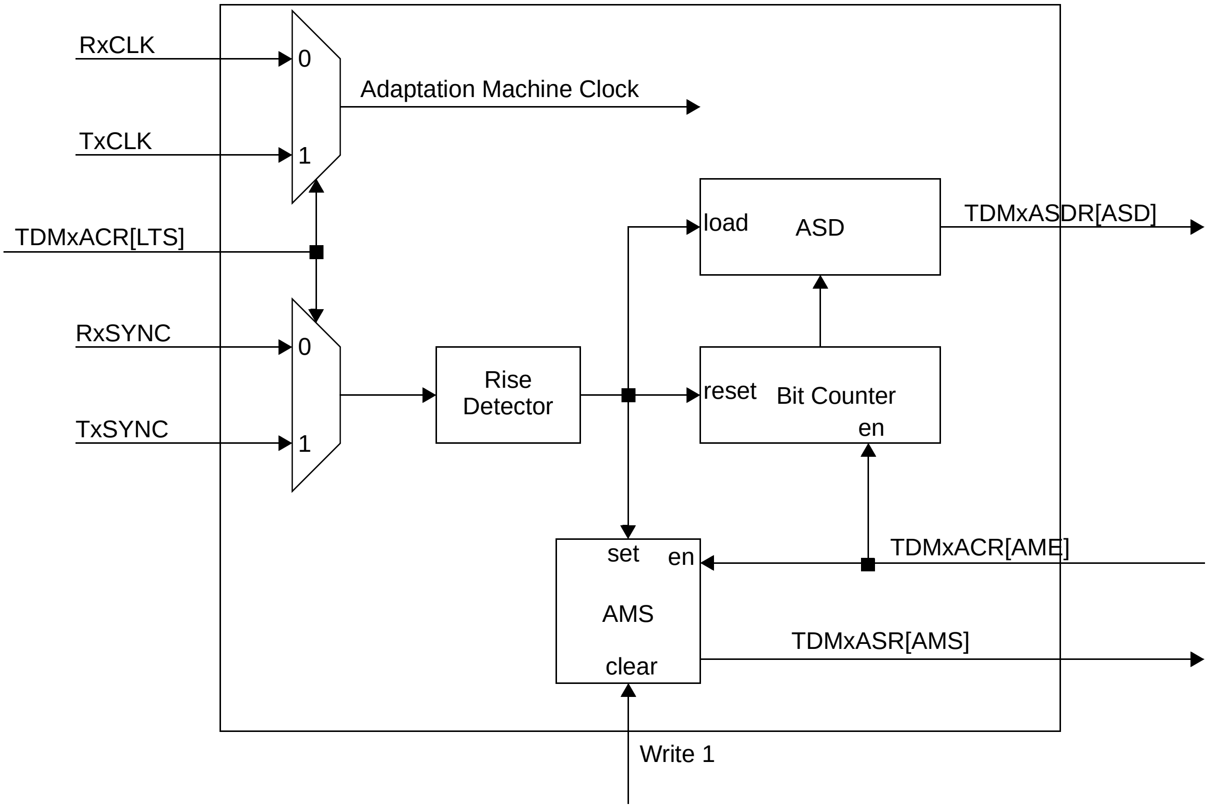


Figure 20-29. Adaptation Machine Block Diagram

Figure 20-29 shows the adaptation machine block diagram. The adaptation machine can work with either the transmit or receive frame. The LTS bit in the TDMx Adaptation Control Register (TDMxACR) defines whether the adaptation machine is fed with the transmit or with the receive frame sync and clock. The adaptation machine samples the sync only at the rising edge of the associated clock. When enabled, the adaptation machine detects a frame sync, stores the Bit Counter in the ASD field in the TDMx Adaptation Sync Distance Register (TDMxASDR), resets the Bit Counter, and sets the AMS bit in the TDMx Adaptation Status Register (TDMxASR).

The following steps define how to use the adaptation machine:

1. Configure the LTS bit to define whether the adaptation machine is fed with the Transmit or with the receive frame sync and clock. (See the TDMxACR TDMx Adaptation Control Register on page 20-53).
2. Set the AME bit in the TDMxACR to enable the adaptation machine.
3. Wait for AMS bit in the TDMxASR to be set to 1. (See the TDMxASR TDMx Adaptation Status Register on page 20-67).
4. Read the value of the ASD field in the TDMxASDR. (See the TDMxASDR TDMx Adaptation Sync Distance Register on page 20-62).
5. Clear the AMS bit by writing a 1 to the AMS bit in the TDMxASR.

6. Repeat steps 3–5 until you read the same value from the ASD field for 20 consecutive times. At this time the ASD value is valid and can be used to configure the TDM receiver or transmitter.
7. Clear the AME bit in the TDMxACR to disable the adaptation machine.
8. Configure the receiver or transmitter according to the following parameters:

— ASD value.
— Number of active links.
— Channel size.
— SYN

20.3 TDM Power Saving

The MSC8113 TDMs use the stop mode of different clocks to save power. Each TDM has three clock domains: transmit serial, receive serial, and the system clock. The transmit serial clock is not supplied to the TDM module when the transmitter is disabled, that is, the TDMxTCR[TEN] bit and the TDMxTSR[TENS] are both clear. The receive serial clock is not supplied to the TDM module when the receiver is disabled, TDMxRCR[REN] bit and TDMxRSR[RENS] bit are both clear. The system clock automatically stops when the TDM is disabled, that is, both transmitter and receiver are disabled. In addition, the TDM registers get the system clock only at reset or during an IPBus access.

Each TDM has a status bit in the Stop Ack Status Register (SASR) (see page 19-7), which indicates the TDM system clock activity status.

20.4 Channel Activation

The TACT and RACT bits in the Transmit/Receive Channel Parameter Registers (see page 20-58 and page 20-59) are enabled during the receiver/transmitter operation to control the channel activation. If the TACT/RACT bit is clear, the channel is not active. Otherwise, it is active. The procedure for activating an inactive receive channel (C) is as follows:

1. Verify that the active (RACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the receive TDM local memory.

   The receive local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at (256 / (RNB + 1) × B + C) × 8. (See ***Section 20.2.3, TDM Data Structures***, on page 20-13). For example, if the number of buffers is four, the SC140 core should write the initialization value to all four receive buffers. Initializing the receive TDM local memory prevents invalid data from being received by the channel buffer in the main memory.
3. Set the TDMxRCPRC[RACT] bit (C indicates the channel number).

The procedure for activating an inactive transmit channel (C) is as follows:

1. Verify that the active (TACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the transmit TDM local memory.

   The transmit local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at (256 / (TNB + 1) × B + C) × 8. (See ***Section 20.2.3, TDM Data Structures***, on page 20-13). Initializing the transmit TDM local memory prevents invalid data from being transmitted out.
3. Set the TDMxTCPRC[TACT] bit (C indicates the channel number).

   For example, if the SC140 core needs to activate receive channel 2 and the number of receive buffers is 4 (RNB[3–0] = 0011), it should write the initialization value to the following addresses (which are offsets from the TDMx receive local memory, see **Chapter 8**, *Memory Map*):

   — 0x0010–0x0017 (the channel location in buffer 0).
   — 0x0210–0x0217 (the channel location in buffer 1).
   — 0x0410–0x0417 (the channel location in buffer 2).
   — 0x0610–0x0617 (the channel location in buffer 3).

20.5 Loopback Support

In Loopback Test mode, the receiver receives the same data that is transmitted. The frame clock should supply to the TDM, and the frame sync can be generated internally or supplied externally. The receiver and transmitter share the frame sync, frame clock, and data links (RTSAL[3–2] = 0b11). The number of data links can be 1, 2, or 4 and is determined by the RTSAL[1–0] bits. All the receive and transmit frame channels are active. The procedure for loopback is as follows:

1. Configure the RTSAL field in the GIR register (see page 20-36) to shared data links mode- RTSAL[3–2] = 0b11. The number of data links can be 1, 2, or 4.
2. Configure the receive and transmit frame parameters to be the same. The configuration of the RFP register should be identical to that of the TFP register (see page 20-47 and page 20-49).
3. Configure the TDMx Transmit Interface Register (TDMxTIR) and the TDMx Receive Interface Register (TDMxRIR) according to the following instructions:

   — Set the Transmit Sync Out (TSO) bit to according to the Transmit Sync signal direction (input or output) used in your system.
   — Set the Receive Frame Sync Delay field to 0x00 and the Transmit Frame Sync Delay field to 0x01.

- Set both the Receive Frame Sync Edge (RFSE) bit and the Transmit Frame Sync Edge (TFSE) bits to 1. The sync samples at the negative edge.
- The value of the Receive Sync level bit should be identical to that of the Transmit Sync Level field (RSL = TSL).
- Clear the Receive Data Edge bit (RDE = 0x0), so that the receive data is sampled on the positive edge.
- Set the Transmit Data Edge bit (TDE = 0x1) to transmit data driven at the negative edge.

4. Set the receive active RACT bit of all the channels to 1.
5. Set the transmit active TACT bit of all the channels to 1.
6. Set the TDMxTCR[TEN] bit.
7. Set the TDMxRCR[REN] bit.

20.6 TDM Initialization

After reset, all TDM registers are reset. **Table 20-6** describes the TDM signal direction after reset.

Table 20-6. TDM Signal Direction at Reset

TDM Signal	Signal Direction
TDMXRCLK	input
TDMxRDAT	input
TDMxRSYN	input
TDMxTCLK	input
TDMxTDAT	input
TDMxTSYN	input

The TDMxRCR[REN] bit (see page 20-54) enables the receiver part of the TDM module. When TDMxRCR[REN] is clear, the receive TDM is disabled, but all the registers retain their values except for the TDMx Receive Data Buffers Displacement Register (TDMxRDBDR). The TDMxTCR[TEN] bit (see page 20-54) enables the transmit part of the TDM module. When TDMxTCR[TEN] is clear, the transmit TDM is disabled, but all the registers retain their values except for the TDMx Transmit Data Buffers Displacement Register (TDMxTDBDR).

The correct flow for initializing the TDM is as follows:

1. Perform a hardware reset to disable the receiver and the transmitter.
2. Program all the configuration registers (their default value is zero). Program all the control registers, except the TDMx Receive Control Register (TDMxRCR) and the TDMx Transmit Control Register (TDMxTCR).
3. Fill the sync data in all the TDM receive local memory.

The received data is stored in 256 entries of 8 bytes each located in the addresses between 0x0000–0x07FF. This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers, starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive number of Buffers Register (TDMxRNB) (see page 20-64). Channel C in buffer B is the 8 bytes starting at (256 / (RNB + 1) × B + C) × 8. (Refer to ***Section 20.2.3, TDM Data Structures***, on page 20-13 for details)

4. Fill the sync data in all the TDM transmit local memory.

   Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800–0x1FFF. This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at (256/(TNB+1) × B+C) × 8.

5. Clear the TDMxRER and TDMxTER event registers by writing a value of 0xF to each of them.

6. Set the TDMxRCR[REN] bit and/or the TDMxTCR[TEN] bit.

20.7 TDM Programming Model

The handshake between the TDM module and the SC140 core occurs via a set of registers, data structures in the memory, and interrupts. All TDM registers are mapped into the IPBus address space. See **Chapter 8**, *Memory Map* for details on IPBus addressing. There are four modules (TDM 0–3), each with its own region in the IPBus address space. Within the module address space, the area is divided into spaces for configuration registers, control registers, and status registers as follows:

- *Configuration registers*. Set the operation modes and provide indications for all channels. They are set before the TDM is enabled and should not be changed while the TDM is active.
- *Control registers*. Set the channel specific parameters individually for each channel and the threshold pointers. These registers can be changed during operation.
- *Status registers*. Read-only registers that can be accessed any time.

This section describes the TDM module registers, which are listed as follows:

- TDMx General Interface Register (TDMxGIR), page 20-36.
- TDMx Receive Interface Register (TDMxRIR), page 20-43.
- TDMx Transmit Interface Register (TDMxTIR), page 20-45.
- TDMx Receive Frame Parameters (TDMxRFP), page 20-47.
- TDMx Transmit Frame Parameters (TDMxTFP), page 20-49.

- TDMx Receive Data Buffer Size (TDMxRDBS), page 20-51.
- TDMx Transmit Data Buffer Size (TDMxTDBS), page 20-51.
- TDMx Receive Global Base Address (TDMxRGBA), page 20-52.
- TDMx Transmit Global Base Address (TDMxTGBA), page 20-52.
- TDMx Adaptation Control Register (TDMxACR), page 20-53.
- TDMx Receive Control Register (TDMxRCR), page 20-54.
- TDMx Transmit Control Register (TDMxTCR), page 20-54.
- TDMx Receive Data Buffers First Threshold (TDMxRDBFT), page 20-55.
- TDMx Transmit Data Buffers First Threshold (TDMxTDBFT), page 20-56.
- TDMX Receive Data Buffers Second Threshold (TDMxRDBST), page 20-57.
- TDMx Transmit Data Buffers Second Threshold (TDMxTDBST), page 20-58.
- TDMx Receive Channel Parameter Register 0–255 (TDMxRCPR[0–255]), page 20-58.
- TDMx Transmit Channel Parameter Register 0–255 (TDMxTCPR[0–255]), page 20-59.
- TDMx Receive Interrupt Enable Register (TDMxRIER), page 20-60.
- TDMx Transmit Interrupt Enable Register (TDMxTIER), page 20-61.
- TDMx Adaptation Sync Distance Register (TDMxASDR), page 20-62.
- TDMx Receive Data Buffers Displacement Register (TDMxRDBDR), page 20-62.
- TDMx Transmit Data Buffers Displacement Register (TDMxTDBDR), page 20-63.
- TDMx Receive Number of Buffers (TDMxRNB), page 20-64.
- TDMx Transmitter Number of Buffers (TDMxTNB), page 20-64.
- TDMx Receive Event Register (TDMxRER), page 20-65.
- TDMx Transmit Event Register (TDMxTER), page 20-66.
- TDMx Adaptation Status Register (TDMxASR), page 20-67.
- TDMx Receive Status Register (TDMxRSR), page 20-67.
- TDMx Transmit Status Register (TDMxTSR), page 20-68.

Two additional registers in the system bus address space include information on the TDM transaction on the local bus in case an error occurs on the bus:

- Local Bus GTD Transfer Error Address (LGTDTEA), page 20-69.
- GTD Transfer Error Requestor Number Register (LGTDTER), page 20-70.

20.7.1 Configuration Registers

TDMxGIR TDM*x* General Interface Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—											CTS	RTSAL			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*GIR defines the TDM*x* interface operation mode.

Table 20-7. TDM*x*GIR Bit Descriptions

Name	Reset	Description	Settings
— 0–26	0	Reserved. Write to zero for future compatibility.	
CTS 27	0	**Common TDM Signals** Defines whether the TDM shares sync and clock signals with other TDM modules. The four TDMs can share signals as follows: • TDM0,TDM1,TDM2, and TDM 3 do not share signals. (TDM0CTS=0, TDM1CTS=0, TDM2CTS=0, TDM3CTS=0) • TDM0 and TDM1 share signals, but TDM2 and TDM 3 do not share signals with the other TDM modules. (TDM0CTS=1, TDM1CTS=1, TDM2CTS=0, TDM3CTS=0) • TDM0,TDM1, and TDM2 share signals, but TDM3 does not share signals with the other TDM modules. (TDM0CTS=1, TDM1CTS=1, TDM2CTS=1, TDM3CTS=0) • TDM0,TDM1,TDM2,TDM3 share signals. (TDM0CTS=1, TDM1CTS=1, TDM2CTS=1, TDM3CTS=1). **Table 20-8** on page 20-37 describes the functionality of the TDM signals as a function of the **RTSAL** field. **Note:** If the TDM modules share sync and clock signals, then the RFP, TFP,RIR, and TIR registers should be configured the same way for all the TDM modules.	0 The TDM does not share signals with other TDM modules 1 The TDM shares sync and clock signals with other TDM modules. Refer to **Table 20-8**.

Table 20-7. TDM*x*GIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RTSAL 28–31	0	**Receive and Transmit Sharing and Active Links** Defines the TDM serial interface operating mode. It determines whether the TDM transmit and receive paths are independent or share the same clock and sync. It also determines whether the TDM receive and transmit share the data links. Bits 2 and 3 determine the receive and transmit sharing mode, and bits 1 and 0 determine the number of active data links. **Note:** If RTSAL [3–2]= 01 or 11, some parameters of the receive and transmit path should be the same. The value of the TDMxRFP[RNCF], RCS and RT1 fields should be equal to that of the TDMxTFP[TNCF], TCS, and TT1 fields. The value of the TDMxRIR[RFSE] and TDMxRIR[RSL] fields should be equal to the that of the TDMxTIR[TFSE] and TDMxTIR[TSL] fields, respectively. These fields are described on page 20-43 through page 20-45. For details, see ***Section 20.2.1, Common Signals for the TDM Modules***, on page 20-8. **Note:** Unused signals should not be configured as dedicated signals in the PAR.	0000 The receive and transmit are independent.The TDM receives one data link and transmits one data link. 0001The receive and transmit are independent. The TDM receives two data links and transmits two data link***s (valid only if CTS=1). 0010Reserved. 0011Reserved. 0100The receive and transmit share the frame clock and frame sync.The TDM receives one data link and transmits one data link. 0101The receive and transmit share the frame sync and frame clock. The TDM receives two data links and transmits two data links. 0110Reserved. 0111Reserved. 1000Reserved. 1001Reserved. 1010Reserved. 1011Reserved. 1100 The receive and transmit share the frame sync, frame clock, and one full duplex data link. 1101The receive and transmit share the frame sync, frame clock, and two full duplex data links. 1110Reserved. 1111The receive and transmit share the frame sync, frame clock, and four full duplex data links. Refer to **Table 20-9**.

Table 20-8. TDM Signal Configuration When TDM Modules Share Signals

RTSAL[0–3] Field Value	Description
0000	Receive clock: TDM1TCLK Transmit clock: TDM0TCLK Receive sync: TDM1TSYN Transmit sync: TDM0TSYN Receive data links: TDMxRDAT Transmit data links: TDMxTDAT Unused signals: TDMxRCLK, TDMxRSYN. **Note:** The x specifies the TDM number of TDMs that share signals. For example if TDM0, TDM1, and TDM2 share signals, then x is equal to 0,1, and 2 and the receive data links are TDM0RDAT, TDM1RDAT, and TDM2RDAT.

Table 20-8. TDM Signal Configuration When TDM Modules Share Signals (Continued)

RTSAL[0–3] Field Value	Description
0001	Receive clock: TDM1TCLK Transmit clock: TDM0TCLK Receive sync: TDM1TSYN Transmit sync: TDM0TSYN Receive data links: TDMxRDAT, TDMxRSYN Transmit data links: TDMxTDAT, TDMxRCLK **Note:** The x specifies the number of the TDM and any one of the shared TDM modules.
0100	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Receive data links: TDMxRDAT Transmit data links: TDMxTDAT Unused signals: TDMxRCLK, TDMxRSYN, TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCLK, TDM1RCLK, TDM0RSYN, TDM1RSYN, TDM1TCLK, and TDM1TSYN.
0101	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Receive data links: TDMxRDAT, TDMxRSYN Transmit data links: TDMxTDAT, TDMxRCLK Unused signals: TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCLK and TDM1TSYN.
1100	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout and is used for receive and transmit) TDMxRDAT Unused signals: TDMyTCLK, TDMyTSYN, TDMxRCLK, TDMxRSYN, and TDMxTDAT TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCLK, TDM1RCLK, TDM0RSYN, TDM1RSYN, TDM0TDAT, TDM1TDAT, TDM1TCLK, and TDM1TSYN.
1101	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout it and it is used for receive and transmit): TDMxRDAT, TDMxRSYN Unused signals: TDMyTCLK, TDMyTSYN, TDMxRCLK, TDMxTDAT TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except of TDM0. for example if TDM0 and TDM1 shared signals then the unused signals are TDM0RCLK, TDM1RCLK, TDM0TDAT, TDM1TDAT, TDM1TCLK, and TDM1TSYN.
1111	Frame clock (receive and transmit share the same clock): TDM0TCLK Frame sync (receive and transmit share the same sync): TDM0TSYN Full duplex data links (the data link is inout and is used for receive and transmit): TDMxRDAT,TDMxRSYN,TDMxTDAT,TDMxRCLK Unused signals: TDMyTCLK, TDMyTSYN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCLK and TDM1TSYN.

Table 20-9. Configuring TDM Signals With the RTSAL and CTS Fields

No.	CTS	RTSAL [0–3]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
0	0	0000	receive data (RDATA_A)	receive sync	receive clock	transmit data (TDATA_A)	transmit sync	transmit clock	The TDM does not share signals with others TDM modules. Independent mode. One active data link.
	direction		Input	Input	Input	Output	Inout	Input	
1	0	0001	Reserved						
2	0	0010	Reserved						
3	0	0011	Reserved						
4	0	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with others TDM modules. Receive and transmit share sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	
5	0	0101	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
6	0	0110	Reserved						
7	0	0111	Reserved						

Table 20-9. Configuring TDM Signals With the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [0–3]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
8	0	1100	data link (DATA_A)	not used	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
9	0	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
10	0	1110	Reserved						
11	0	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	
12	1	0000	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. One active data link.
	direction		Input			Output	Inout	Input	

Table 20-9. Configuring TDM Signals With the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [0–3]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
13	1	0001	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
14	1	0010	Reserved						
15	1	0011	Reserved						
16	1	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit shared sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	
17	1	0101	receive data RDATA_A	receive data RDATA_B	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
18	10	0110	Reserved						
19	1	0111	Reserved						

Table 20-9. Configuring TDM Signals With the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [0–3]	TDMxRDAT	TDMxRSYN	TDMxRCLK	TDMxTDAT	TDMxTSYN	TDMxTCLK	Comments
20	1	1100	data link (DATA_A)	not used	not used	not used	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
21	1	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync/ not used	frame clock/ not used	The TDM share the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
22	1	1110	Reserved						
23	1	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	

Note: Frame sync specifies that the receiver and transmitter use the same sync. Frame clock specifies that the receiver and transmitter use the same clock. If data link specifies that the direction is inout, the signal is used for receive and transmit.

TDM*x*RIR TDM*x* Receive Interface Register

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	RFTL	RSTL	—								RFSD		RSL	RDE	RFSE	RRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIR defines the TDM*x* receiver interface operation.

Table 20-10. TDM*x*RIR Bit Descriptions

Name	Reset	Description	Settings
— 0–14	0	Reserved. Write to zero for future compatibility.	
— 15	0	Reserved. Write to one for future compatibility. The boot program writes a 1 to this bit. If you do not use the boot program, write a 1 to this bit after reset for proper operation.	
RFTL 16	0	**Receive First Threshold Level** Determines whether the receive first threshold interrupt is pulse or level. For details, see **Section 20.2.6.3**.	0 Receive first threshold interrupt is pulse. 1 Receive first threshold interrupt is level.
RSTL 17	0	**Receive Second Threshold Level** Determines whether the receive second threshold interrupt is pulse or level. For details, see **Section 20.2.6.3**.	0 Receive second threshold interrupt is pulse. 1 Receive second threshold interrupt is level.
— 18–25	0	Reserved. Write to zero for future compatibility.	
RFSD 26–27	0	**Receive Frame Sync Delay** With the RDE and the RFSE bits, determines the number of clocks between the receive sync signal and the first data bit of the receive frame. For examples, see **Section 20.2.4.2.**	Refer to **Table 20-11**. **Note:** If the receive channel size is 2 (RCS = 0x1), then the RFSD field value can be only 0 or 1.
RSL 28	0	**Receive Sync Level** Determines the polarity of the receive sync signal. For details, see **Figure 20-21**.	0 Receive sync is active on logic 1. 1 Receive sync is active on logic 0.
RDE 29	0	**Receive Data Edge** Determines whether the receive data signal is sampled on the rising or falling edge of the receive clock. For details see **Section 20.2.4.2**.	0 The receive data signal is sampled on the rising edge of the receive clock. 1 The receive data signal is sampled on the falling edge of the receive clock.

Table 20-10. TDM*x*RIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RFSE 30	0	**Receive Frame Sync Edge** Determines whether the receive frame sync signal is sampled on the rising or falling edge of the receive clock. For details, see **Section 20.2.4.2**.	0 The receive frame sync signal is sampled with the rising edge of the receive clock. 1 The receive frame sync signal is sampled on the falling edge of the receive clock.
RRDO 31	0	**Receive Reversed Data Order** For examples, see **Section 20.2.4.4**.	0 The first bit of a received channel is stored as the most significant bit in the internal memory. 1 The first bit of a received channel is stored as the least significant bit in the internal memory

Table 20-11. Received Data Delay for Receive Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Receive Clocks[1]
00	0	0	0.0
00	0	1	0.5
00	1	0	0.5
00	1	1	0.0
01	0	0	1.0
01	0	1	1.5
01	1	0	1.5
01	1	1	1.0
10	0	0	2.0
10	0	1	2.5
10	1	0	2.5
10	1	1	2.0
11	0	0	3.0
11	0	1	3.5
11	1	0	3.5
11	1	1	3.0

Note: Receive clocks is the number of receive clocks between the sample of the receive frame sync and the sample of first data bit of the received frame.

TDM*x*TIR TDM*x* Transmit Interface Register

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	TFTL	TSTL	TSO	TAO	SOL	SOE	-—				TFSD		TSL	TDE	TFSE	TRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxTIR defines the TDM *x* transmitter interface operation.

Table 20-12. TDM*x*TIR Bit Descriptions

Name	Reset	Description	Settings
— 0–14	0	Reserved. Write to zero for future compatibility.	
— 15	0	Reserved. Write to one for future compatibility. The boot program writes a 1 to this bit. If you do not use the boot program, write a 1 to this bit after reset for proper operation.	
TFTL 16	0	**Transmit First Threshold Level** Determines whether the Transmit first threshold interrupt is pulse or level. For details, see **Section 20.2.6.3.**	0 Transmit first threshold interrupt is pulse. 1 Transmit first threshold interrupt is level.
TSTL 17	0	**Transmit Second Threshold Level** Determines whether the Transmit second threshold interrupt is pulse or level. For details, see **Section 20.2.6.3.**	0 Transmit second threshold interrupt is pulse. 1 Transmit second threshold interrupt is level.
TSO 18	0	**Transmit Sync Output** Determines whether the transmit sync is driven out by the TDM transmitter or it input to the TDM transmitter. For details, see **Section 20.2.4.1.**	0 Transmit sync is input. 1 Transmit sync is output.
TAO 19	0	**Transmit Always Output** Determines whether the TDM transmitter drives TDMxDAT for the inactive channels.	0 The TDM transmitter does not drive the TDMxDAT for inactive channels. 1 The TDM transmitter drives the TDMxDAT regardless of whether the channel is active.
SOL 20	0	**Sync Out Length** Indicates whether the TDMXTSYN is asserted for one cycle of TDMxTCLK or is asserted for the duration of the first channel in the frame. For details, see **Section 20.2.4.1**.	0 The sync_out width is one bit. 1 The sync_out width is equal to the channel width.
SOE 21	0	**Sync Out Edge** Determines whether the sync out signal is driven out with the rising or falling edge of the transmit clock. For details, see **Section 20.2.4.1**.	0 The transmit frame sync signal is driven out on the rising edge of the transmit clock. 1 The transmit frame sync signal is driven out on the falling edge of the transmit clock.
22–25	0	Reserved. Write to zero for future compatibility.	

Table 20-12. TDM*x*TIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TFSD 26–27	0	**Transmit Frame Sync Delay** With the TDE and the TFSE bits, determines the number of clocks between the transmit sync signal and the first data bit of the transmit frame. For examples, see **Section 20.2.4.2**.	Refer to **Table 20-13** on page 20-46. **Note:** If the transmit channel size is 2 (TCS = 0x1) then the TFSD field value can be only 0 or 1.
TSL 28	0	**Transmit Sync Level** Determines the polarity of the transmit sync signal. For details, see **Section 20.2.4.2**.	0 Transmit sync is active on logic 1. 1 Transmit sync is active on logic 0.
TDE 29	0	**Transmit Data Edge** Determines whether the transmit data is driven out on the rising or falling edge of the transmit clock. For details, see **Section 20.2.4.2**.	0 The transmit data is driven out on the rising edge of the transmit clock. 1 The transmit data is driven out on the falling edge of the transmit clock.
TFSE 30	0	**Transmit Frame Sync Edge** Determines whether the transmit frame sync signal is sampled with the rising or falling edge of the receive clock. For details, see **Section 20.2.4.2**.	0 The transmit frame sync signal is sampled with the rising edge of the transmit clock. 1 The transmit frame sync signal is sampled with the falling edge of the transmit clock.
TRDO 31	0	**Transmit Reversed Data Order** For examples, see **Section 20.2.4.4**.	0 The most significant bit of the memory is sent out at the first transmit data bit. 1 The least significant bit of the memory is sent out at the first transmit data bit.

Table 20-13. Transmit Data Delay for Transmit Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Transmit Clocks[1]
00	0	0	–1 [2]
00	0	1	–0.5 [2]
00	1	0	–0.5 [2]
00	1	1	–1 [2]
01	0	0	0
01	0	1	0.5
01	1	0	0.5
01	1	1	0
10	0	0	1
10	0	1	1.5
10	1	0	1.5
10	1	1	1
11	0	0	2
11	0	1	2.5
11	1	0	2.5
11	1	1	2

Notes: **1.** Transmit clocks is the number of transmit clocks between the first transmit frame sync sample and the first data bit of the frame that is driven out.

2. The field value is negative because the data is driven out before the transmit frame sync sample.

TDM*x*RFP TDM*x* Receive Frame Parameters

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								RNCF							
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
	—					RCDBL			—		RCS				RT1	RUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RFP defines the TDM*x* receive frame parameters.

Table 20-14. TDM*x*RFP Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
RNCF 8–15	0	**Receive Number of Channels in a TDM Frame** Specifies the total number of channels that are received in the TDM modules. One TDM frame can contain 2–256 channels at a granularity of two. **Notes: 1.** RNCF[8-15] = (number of channels that received on one active link) × (number of active data links) – 1, the number of active data links is specified in the RTSAL field. **2.** If RCDBL field is clear, then the minimum number of channels is limit. The minimum receive number of channels is 128 / (receive channel size) + 2. For example, if the channel size is 4 bits, then the receive TDM frame should contain at least 34 channels. **Table 20-15** describes the RNCF valid value as a function of the RTSAL field (Receive and Transmit Sharing and Active Links). For details, see **Section 20.2.4**.	0x00 Reserved. 0x01 2 received channels. 0x02 Reserved. 0x03 4 received channels. 0x04 Reserved. . . . 0xFD 254 received channels. 0xFF 256 received channels. 0xFD 254 received channels. 0xFE Reserved. 0xFF 256 received channels. **Note:** The even values are reserved.

Table 20-14. TDMxRFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 16–20	0	Reserved. Write to zero for future compatibility.	
RCDBL 21-23	0	**Receive Channel Data Bits Latency** Defines the maximum amount of receive channel bits stored in the TDM local memory before they are transferred for processing. RCDBL determines the maximum data latency in the following way: Maximum data latency= (RCDBL)/RCS × (receive frame duration). For details, see **Section 20.2.5**. **Notes:** **1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically the latency it much smaller. **2.** RCS is field at RFP register defines the channel size. **3.** The minimum number of receive channel is limited if the RCDBL field is clear.The minimum receive number of channels is 128/(receive channel size) + 2.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 24-25	0	Reserved. Write to zero for future compatibility.	
RCS 26–29	0	**Receive Channel Size** Determines the receiver channel size – 1. For details, see **Section 20.2.5**. **Note:** In Receiver Unified Buffer mode (RUBM = 1), the channel size must be 4 bits (RCS = x3).	0000 Reserved. 0001 The receiver channel size is 2 bits. 0010 Reserved. 0011 The receiver channel size is 4 bits. 0100–0110 Reserved. 0111 Receiver channel size is 8 bits. 1000–1110 Reserved. 1111 Receiver channel size is 16 bits.
RT1 30	0	**Receive T1 frame** Determines whether the receive frame is T1 frame or non T1. **Note:** In T1 mode the channel size must be 8 bits (RCS = 0x7) and the number of channels must be 24 × (number of links). For example, if the number of link is 2 (RTSAL[1–0] = 01), the number of channels should be 48 (RNCF = 0x2F). For details, see **Section 20.2**.	0 The receive frame is non T1 frame. 1 The receive frame is T1 frame.
RUBM 31	0	**Receive Unified Buffer Mode** Indicates that all the received data is directed to one buffer. When RUBM is set, the number of channels must be 2 (RNCF = 0x01), the number of active links must be 1 (RTSAL = 0b0000 or RTSAL = 0100), and the number of bits per channel should be 4 (RCS = 0b0011). The channel parameters of channels 0 and 1 are located in the TDMxRCPR0 (See page 20-58). For details, see **Section 20.2.6.4**. **Note:** When this bit is set, the TDMxRIR[RRDO] bit should be cleared.	0 Each channel is written to a different data buffer in the local bus. 1 All the channels are written to the same data buffer in the local bus.

Table 20-15 describes the RNCF valid value as a function of the RTSAL[0–1] field.

Table 20-15. RNCF[7–0] Valid Values

RTSAL[1–0]	Number of Active Links	RNCF[7–0] Suffix	Valid Value of RNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved.		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

TDM*x*TFP TDM*x* Transmit Frame Parameters

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								TNCF							
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
	—					TCDBL			—		TCS				TT1	TUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TFP defines the TDM*x* transmit frame parameters.

Table 20-16. TDM*x*TFP Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
TNCF 8–15	0	**Transmit Number of Channels in a TDM Frame** Specifies the total number of channels that are transmitted in the TDM modules. One TDM frame contains 2–256 channels. **Notes: 1.** TNCF[8–15] = (number of channels that transmit on one active link) × (number of active data links) – 1. the number of active data links is specified in the RTSAL field. **2.** If TCDBL field is cleared, the minimum number of channels is limit. The minimum transmit number of channels is 128 / (transmit channel size) + 2. for example if the transmit channel size is 16 bits then the transmit TDM frame should contain at least 10 channels. The number of active data links is specified in the RTSAL field. **Table 20-17** describes the TNCF valid value as a function of the TDMxGIR[RTSAL] field (Receive and Transmit Sharing and Active Links). For details, see **Section 20.2.4**.	0x00 Reserved. 0x01 2 Transmit channels. 0x02 Reserved. 0x03 4 Transmit channels. 0x04 Reserved. . . . 0xFE Reserved. 0xFF 256 Transmit channels. The even values are reserved.
— 16–20	0	Reserved. Write to zero for future compatibility.	

Table 20-16. TDM*x*TFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCDBL 21–23	0	**Transmit Channel Data Bits Latency** Defines the maximum transmit channel bits that can be stored in the TDM local memory before it transfers output. TCDBL determines the maximum data latency in the following way: Maximum data latency = (TCDBL) / TCS × (transmit frame duration). See **Section 20.2.5**. **Notes: 1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically, actual latency is much smaller. **2.** TDMxTFP[TCS] defines the transmit channel size. **3.** The minimum number of transmit channel is limit if the RCDBL field is clear. The minimum transmit number of channels is 128 / (transmit channel size) + 2.For example see **TNCF** field.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 24–25	0	Reserved. Write to zero for future compatibility.	
TCS 26–29	0	**Transmit Channel Size** Determines the transmitter channel size – 1. For details, see **Section 20.2.4**. **Note:** In Transmit Unified Buffer mode, TCS should be 4 bits (TCS = 0x3).	0000 Reserved 0001 The transmitter channel size is 2 bits. 0010 Reserved 0011 The transmitter channel size is 4 bits. 0100 Reserved. 0101 Reserved. 0110 Reserved. 0111 The transmitter channel size is 8 bits. 1000– 1110 Reserved 1111 The transmitter channel size is 16 bits.
TT1 30	0	**Transmit T1 Frame** Determines whether the TDM transmitter drives a T1 frame or non T1 frame. **Note:** In T1 mode, the channel size must be 8 (TCS = 0x7) and the number of channels 24 × (number of links). For example, if the number of links is 1 (RTSAL[1–0] = 00, the number of channels should be 24 (TNCF = 0x17). For details, see **Section 20.2**.	0 Transmit frame is non T1 frame. 1 Transmit frame is T1 frame.
TUBM 31	0	**Transmit Unified Buffer Mode** Indicates that all the transmit data is transferred from one buffer. When TUBM is set, the number of channels must be 2 (TNCF = 0x01), the number of active links must be 1 (RTSAL = 0b0000 or 0b0100), and the number of bits per channel should be 4 (TCS = 0b0011). The parameters of channels 0 and 1 are located in Transmit Channel Parameter 0 (refer to page 20-59). For details, see **Section 20.2.6.4**. **Note:** When this bit is set, the TDMxTIR[TRDO] bit should be cleared.	0 Each channel is read from a different data buffer in the local bus. 1 All the channels are read from the same data buffer in the local bus.

Table 20-17 describes the TNCF valid value as function of the RTSAL field.

Table 20-17. TNCF[7–0] Valid Values

RTSAL[2–0]	Number of Active Links	TNCF[7–0] Suffix	Valid Value of TNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

TDM*x*RDBS TDM*x* Receive Data Buffer Size

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

TDM*x*RDBS defines the receive data buffers size in bytes.The buffers for A/μ-law channels are double size.

Table 20-18. TDM*x*RDBS Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
RDBS 8–31	0	**Receive Data Buffers Size** Receive data buffers size equals the receive data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 29–31 must be set to “111”. For details, see ***Section 20.2.6.1, Data Buffer Size and A/m-law Channels***, on page 20-22. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

TDM*x*TDBS TDM*x* Transmit Data Buffer Size

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

TDM*x*TDBS defines the transmit data buffer size in bytes. The buffers for A/µ channels are double size.

Table 20-19. TDM*x*TDBS Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
TDBS 8–31	0	**Transmit Data Buffers Size** Transmit data buffers size equals the transmit data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 29–31 must be set to “111.” For details, see **Section 20.2.6.1**. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

TDM*x*RGBA TDM*x* Receive Global Base Address

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	RGBA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RGBA determines the 16 most significant bits global base address of the receiver data buffers. The actual address of each receive data buffer is RCDBA + (RGBA << 16). See **Section 20.2.6.2**.

Table 20-20. TDMxRGBA Bit Descriptions

Name	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RGBA 16–31	0	**Receive Global Base Address** Determines the global base address of the receiver data buffers. It is added to the channel data buffer address and to the current receive displacement to generate the actual data address.

TDM*x*TGBA TDM*x* Transmit Global Base Address

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	TGBA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TGBA determines the 16 most significant bits global base address of the transmitter data buffers. The actual address of each transmit data buffer is TCDBA + (TGBA << 16). See **Section 20.2.6.2**.

Table 20-21. TDM*x*TGBA Bit Descriptions

Name	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
TGBA 16–31	0	**Transmit Global Base Address** Determines the global base address of the transmit data buffers. It is added to channel data buffer address and to the current transmit displacement to generate the actual address.

20.7.2 Control Registers

TDMxACR TDMx Adaptation Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—														LTS	AME
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxACR controls the activation/deactivation of the TDMx adaptation machine. The propagation of the enable/disable to the adaptation machine is delayed because is clocked by the serial clock.

Table 20-22. TDMxACR Bit Descriptions

Name	Reset	Description	Settings
— 0–29	0	Reserved. Write to zero for future compatibility.	
LTS 30	0	**Learn Transmit Sync** Determines whether the adaptation machine learns the transmit sync or the receive sync.	0 Adaptation machine learns the receive sync. 1 Adaptation machine learns the transmit sync.
AME 31	0	**Adaptation Machine Enable** Determines whether the adaptation machine is enabled or disabled.	0 Adaptation machine is disabled. 1 Adaptation machine is enabled

TDM*x*RCR TDM*x* Receive Control Register

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

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

TDM*x*RCR controls the activation/deactivation of the TDM*x* Receiver. Receiver activation is valid only when the RENS bit is clear.

Table 20-23. TDM*x*RCR Bit Descriptions

Name	Reset	Description	Settings
— 0–30	0	Reserved. Write to zero for future compatibility.	
REN 31	0	**Receive Enable** Determines whether the receive TDM is enabled or disabled. **Note:** Setting this bit is the last step in initializing the receiver.	0 Receiver is disabled. 1 Receiver is enabled.

TDM*x*TCR TDM*x* Transmit Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	Reserved															TEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TCR controls the activation/deactivation of the TDM*x* Transmitter. Transmitter activation is valid only when the TENS bit is clear.

Table 20-24. TDM*x*TCR Bit Descriptions

Name	Reset	Description	Settings
— 0–30	0	Reserved. Write to zero for future compatibility.	

Table 20-24. TDM*x*TCR Bit Descriptions

Name	Reset	Description	Settings
TEN 31	0	**Transmit Enable** Determines whether the transmit TDM is enabled or disabled. Setting this bit is the last step in initializing the transmitter.	0 Transmitter is disabled. 1 Transmitter is enabled.

TDM*x*RDBFT TDM*x* Receive Data Buffers First Threshold

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—								RDBFT							
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**
	RDBFT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RDBFT determines the first threshold of the receive data buffers. When the receive buffers are filled up to the first threshold defined by this field—the TDM*x* Receive Data Buffers Displacement Register (TDM*x*RDBDR) = TDM*x* Receive Data Buffers First Threshold (TDM*x*RDBFT) + 8—the RFTE bit in the TDMx Receive Event Register (TDM*x* RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* receiver is enabled. For details, see **Section 20.2.6.3**.

Table 20-25. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
RDBFT 8–31	0	**Receive Data Buffer First Threshold** Determines the location of the first threshold in the receive data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

TDMxTDBFT TDM*x* Transmit Data Buffers First Threshold

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

TDM*x*TDBFT determines the first threshold of the transmit data buffers. When the transmit buffers are emptied up to the first threshold defined by this field—the TDM*x* Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the TDM*x* Transmit Data Buffers First Threshold (TDM*x*TDBFT) + 8—the TFTE bit in the TDM*x* Transmit Event Register (TDM*x*TER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDMx transmitter is enabled. For details, see **Section 20.2.6.3.**

Table 20-26. TDM*x*TDBFT Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
TDBFT 8–31	0	**Transmit Data Buffer First Threshold** Determines the location of the first threshold in the transmit data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

TDM*x*RDBST TDM*x* Receive Data Buffers Second Threshold

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Reserved								RDBST							
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
	RDBST															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RDBST determines the second threshold of the receive data buffers. When the receive buffers are filled up to the second threshold defined by this field—the Receive Data Buffers Displacement Register (TDM*x*RDBDR) = the Receive Data Buffers Second Threshold (TDM*x*RDBST) + 8—the RSTE bit in the TDMx Receive Event Register (TDM*x*RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDM*x* receiver is enabled. For details, see **Section 20.2.6.3**.

Table 20-27. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
RDBST 8–31	0	**Receive Data Buffer Second Threshold** Determines the location of the second threshold in the receive data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

TDM*x*TDBST TDM*x* Transmit Data Buffers Second Threshold

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—								TDBST							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

TDM*x*TDBST determines the second threshold of the transmit data buffers. When the transmit buffers are emptied up to the second threshold defined by this field—the Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the Transmit Data Buffers Second Threshold (TDM*x*TDBST) + 8—then the TSTE bit in the TDM*x* Transmit Register (TDM*x*TER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* transmitter is enabled. For details, see **Section 20.2.6.3**.

Table 20-28. TDM*x*TDBST Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
TDBST 8–31	0	**Transmit Data Buffer Second Threshold** Determines the location of the second threshold in the transmit data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

TDM*x*RCPRn TDM*x* Receive Channel Parameter Register n

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	RACT	RCONV		Reserved					RCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	RCDBA															
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

TDM*x*RCPRn determines the parameters for channel 0 to channel 255. The TDM*x*RCPRn[RACT] bit can be changed at any time during the receiver operation. All other fields can only be changed when TDM*x*RCPRn[RACT] is cleared. The read/write access to TDM*x*RCPRn registers can done only to 32 bits, write or read of byte or word is not valid. The register reset value is unknown.

Note: All TDM*x*RCPRn with an index number (n) less than or equal to the TDM*x*RFP[RNCF] bit (see page 20-47) should be valid when setting the corresponding TDM*x*RCR[REN] bit (see page 20-54).

Table 20-29. TDM*x*RCPRn Bit Descriptions

Name	Reset	Description	Settings
RACT 0	—	**Receive Channel Active** Set when the receive channel *n* is active.	0 The channel is non-active. 1 The channel is active.
RCONV 1–2	—	**Receive Channel Convert** Determines the type of the incoming channel n: Transparent, A-law, or μ-Law.	00 Receive channel n is a transparent channel. 01 Receive channel n is a μ-Law channel. 10 Receive channel n is an A-Law channel. 11 Reserved.
— 3–7	—	Reserved. Write to zero for future compatibility.	
RCDBA 8–31	—	**Receive Channel Data Buffer Base Address** Determines the offset of the data buffer *n* base address from the Receive Global Base Address (RGBA).The RCDBA value should be 16 byte aligned; that is, the four LSB should be 0. For details, see **Section 20.2.6.2**.	0x000000–0xFFFFF0.

TDM*x*TCPRn TDM*x* Transmit Channel Parameter Register n

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	TACT	TCONV		—					TCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	TCDBA															
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The TDM*x*TCPRn registers determine the parameters for channel 0 to channel 255. The registers can change any time during the transmitter operation. The TDM*x*TCPRn[TACT] bit can be changed at any time during the transmitter operation. All other fields can only be changed when TDM*x*TCPRn[TACT] is cleared. The read/write access to TDM*x*TCPRn registers can done only as a 32-bit access. A write or read of a byte or a word is not valid.The register reset value is indeterminate.

Note: All TDM*x*TCPRn with an index number (n) less than or equal to the TDM*x*TFP[TNCF] bit (see page 20-49) should be valid when setting the corresponding TDM*x*TCR[TEN] bit (see page 20-54).

Table 20-30. TDM*x*TCPRn Bit Descriptions

Name	Reset	Description	Settings
TACT 0	—	**Transmit Channel Active** Set when the transmit channel n is active.	0 The channel is non-active. 1 The channel is active.
TCONV 1–2	—	**Transmit Channel Convert** Determines the type of the transmit channel n: Transparent, A-law, or µ-Law.	00 Transmit channel *n* is a transparent channel. 01 Transmit channel *n* is a µ-Law channel. 10 Transmit channel *n* is an A-Law channel. 11 Reserved.
— 3–7	—	Reserved. Write to zero for future compatibility.	
TCDBA 8–31	—	**Transmit Channel Data Buffer Base Address** Determines the offset of the transmit data buffer *n* base address from the Transmit Global Base Address (TGBA). The TCDBA value should be 16 byte aligned; that is, the four LSB should be clear. For details, see **Section 20.2.6.2**.	0x000000–0xFFFFF0.

TDM*x*RIER TDM*x* Receive Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—												RSEEE	OLBEE	RFTEE	RSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIER has the same bit format as the TDM*x*RER registers. If an RIER bit is clear, the corresponding event in the TDM*x*RER registers is masked (see page 20-65).

Table 20-31. TDM*x*RIER Bit Descriptions

Name	Reset	Description	Settings
— 0–27	0	Reserved. Write to zero for future compatibility.	
RSEEE 28	0	**Receive Sync Error Event Enable** Enable assertion of the receive error interrupt when the Receive Sync Error (RSE) bit is set (see page 20-65).	0 Receive sync error is masked. 1 Receive sync error is enabled.
OLBEE 29	0	**Overrun Local Buffer Event Enable** Enable assertion of an interrupt when the Overrun Local Buffer Event (OLBE) bit is set (see page 20-65).	0 Overrun Local buffer event is masked. 1 Overrun Local buffer event is enabled.
RFTEE 30	0	**Receive First Threshold Event Enable** Enable assertion of the receive first threshold interrupt when the Receive First threshold Event (RFTE) bit is set (see page 20-65).	0 Receive first threshold interrupt is disabled. 1 Receive first threshold interrupt is enabled.

Table 20-31. TDM*x*RIER Bit Descriptions (Continued)

Name	Reset	Description	Settings
RSTEE 31	0	**Receive Second Threshold Event Enabled** Enable assertion of the receive second threshold interrupt when the Receive Second Threshold Event (RSTE) bit is set (see page 20-65).	0 Receive second threshold interrupt is disabled. 1 Receive second threshold interrupt is enabled

TDM*x*TIER TDM*x* Transmit Interrupt Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—												TSEIE	ULBEE	TFTEE	TSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TIER has the same bit format as the TDM*x*TER registers. If a TDM*x*TIER bit is clear, the corresponding event in the TDM*x*TER is masked (see page 20-66).

Table 20-32. TDM*x*TIER Bit Descriptions

Name	Reset	Description	Settings
— 0-27	0	Reserved. Write to zero for future compatibility.	
TSEIE 28	0	**Transmit Sync Error Event Enabled** Enable assertion of the transmit error interrupt when the Transmit Sync Error (TSE) bit is set. See page 20-66.	0 Transmit sync error interrupt is disabled. 1 Transmit sync error interrupt is enabled.
ULBEE 29	0	**Underrun Local Buffer Event Enabled** Enable assertion of an interrupt when the Underrun Local Buffer Event (ULBE) bit is set. See page 20-66.	0 Underrun Local buffer event is masked. 1 Underrun Local buffer event is enabled.
TFTEE 31	0	**Transmit First Threshold Event Enabled** Enable assertion of the transmit first threshold interrupt when the Transmit First Threshold Event (TSTE) bit is set. See page 20-66.	0 Transmit first threshold interrupt is disabled. 1 Transmit first threshold interrupt is enabled.
TSTEE 31	0	**Transmit Second Threshold Event Enabled** Enable assertion of the transmit second threshold interrupt when the Transmit second Threshold Event (TSTE) bit is set.	0 Transmit second threshold interrupt is disabled. 1 Transmit second threshold interrupt is enabled.

20.7.3 Status Registers

TDMxASDR TDMx Adaptation Sync Distance Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								—							
Type	R															
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
	—					ASD										
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxASDR indicates the number of receive/transmit bits between the last two consecutive receive/transmit sync events. The register value updates each time the TDMxASR[AMS] bit is set.

Table 20-33. TDMxASDR Bit Descriptions

Name	Reset	Description	Settings
— 0–20	0	Reserved. Write to zero for future compatibility.	
ASD 21–31	0	Adaptation Sync Distance Indicate the number of bits between the last two consecutive sync events. If the TDMxACR[LTS] bit is set, the ASD field indicates the number of transmit bits between the last two transmit sync events. If the LTS bit is clear, the value indicates the number of receive bits between the last two receive sync events.	0x000 The number of bits between the last two sync events is 1. 0x001 The number of bits between the last two sync events is 2. . . . 0x7FF The number of bits between the last two sync events is 2048.

TDM*x*RDBDR TDM*x* Receive Data Buffers Displacement Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—								RDBD							
Type	R															
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
	RDBD															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RDBDR points to the current displacement in the receive data buffers where the received data should be written by the TDM DMA. For details, see **Section 20.2.6.2**.

Table 20-34. TDM*x*RDBDR Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
RDBD 8–31	0	**Receive Data Buffer Displacement** Points to the current displacement of the received data in the data buffers. The value is unified to all the transparent channels and is doubled for A/µ law channels.	0 to (RDBS – 7) = Receive Data Buffer Size.

TDM*x*TDBDR TDM*x* Transmit Data Buffers Displacement Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—								TDBD							
Type	R															
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**
	TDBD															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TDBDR points to the current displacement in the transmit data buffers of the data that should be read by the TDM DMA. For details, see **Section 20.2.6.2**.

Table 20-35. TDMxTDBDR Bit Descriptions

Name	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
TDBD 8–31	0	**Transmit Data Buffer Displacement** Points to the current displacement of the transmit data in the transmit data buffers. The value is unified to all the transparent channels and is doubled for A/µ law channels.	The register value can range from 0x000000 to the Transmit Data Buffer (TDBS – 7).

TDM*x*RNB TDM*x* Receive Number of Buffers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—											RNB				
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RNB holds the number of buffers in the TDM receive local buffer. Using this register, you can calculate the location of all the bytes belonging to any channel in the TDM local memory.

Table 20-36. TDMxRNB Bit Descriptions

Name	Reset	Description	Settings
— 0–26	0	Reserved. Write to zero for future compatibility.	
RNB 27–31	0	**Receive Number of Buffers** Holds the number of buffers in the TDM receive local buffer. For details, see **Section 20.2.5**. **Note:** The number of receive buffers equals RNB + 1.	0x001 buffer. 0x012 buffers. 0x034 buffers. 0x078 buffers. 0x0F 16 buffers. 0x1F32 buffers. The other values are reserved.

TDM*x*TNB TDM*x* Transmitter Number of Buffers

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—											TNB				
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxTNB holds the number of transmit buffers in the TDM transmit local buffer.

Table 20-37. TDMxTNB Bit Descriptions

Name	Reset	Description	Settings
— 0–26	0	Reserved. Write to zero for future compatibility.	

Table 20-37. TDMxTNB Bit Descriptions (Continued)

Name	Reset	Description	Settings
TNB 27–31	0	**Transmit Number of Buffers** Holds the number of buffers in the TDM transmit local buffer. **Note:** The number of transmit buffers equals TNB + 1.	0x001 buffer. 0x012 buffers. 0x034 buffers. 0x078 buffers. 0x0F 16 buffers. 0x1F32 buffers. The other values are reserved.

TDM*x*RER TDM*x* Receive Event Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—												RSE	OLBE	RFTE	RSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RER contains the status of the receive data buffers and general receive events. The register can be read at any time. Bits are cleared by writing ones to them; writing zero has no effect.

Table 20-38. TDM*x*RER Bit Descriptions

Name	Reset	Description	Settings
— 0–27	0	Reserved. Write to zero for future compatibility.	
RSE 28	0	**Receive Sync Error** Indicates whether a sync error has occurred. RSE is set when the receive frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the receive signals of the TDM module. For details, see **Section 20.2.4.3**	0 Normal operation. No receive error has occurred. 1 Receive sync error has occurred.
OLBE 29	0	**Overrun Local Buffer Event** Indicates whether an overrun event has occurred in TDM local memory. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the local bus and therefore cannot write the data into the destination memory (data buffer). For details, see **Section 20.2.5**.	0 No overrun event has occurred in the TDM local memory. 1 An overrun event has occurred in the TDM local memory.
RFTE 30	0	**Receive First Threshold Event** This field is set when the first thresholds of all the received data buffers are filled with received data. The first threshold pointer is determined by the Receive Data Buffer First Threshold field (RDBFT). For details, see **Section 20.2.6.3**.	0 No receive first threshold event has occurred. 1 A receive first threshold event has occurred.

Table 20-38. TDM*x*RER Bit Descriptions (Continued)

Name	Reset	Description	Settings
RSTE 31	0	**Receive Second Threshold Event** This field is set when the second thresholds of all the receive data buffers are filled with received data. The second threshold pointer is determined by the Receive Data Buffer Second Threshold. (RDBST) field. For details, see **Section 20.2.6**	0 No receive second threshold event has occurred. 1 A receive second threshold event has occurred.

TDM*x*TER TDM*x* Transmit Event Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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**
	—												TSE	ULBE	TFTE	TSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TER contains the status of the transmit data buffers and general transmit events.The register can be read at any time. Bits are cleared by writing ones to them; writing zero has no effect.

Table 20-39. TDM*x*TER Bit Descriptions

Name	Reset	Description	Settings
— 0–27	0	Reserved. Write to zero for future compatibility.	
TSE 28	0	**Transmit Sync Error** Indicates whether a sync error has occurred. TSE is set when the transmit frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a transmit frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the transmit signals of the TDM module. For details, see **Section 20.2.4.3.**	0 Normal operation. No transmit sync error has occurred. 1 A transmit sync error has occurred.
ULBE 29	0	**Underrun Local Buffer Event** Indicates whether an underrun event has occurred in the TDM local buffer. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the local bus and therefore cannot read the data from the data buffers to the TDM local memory. For details, see **Section 20.2.5**.	0 No underrun event has occurred in the TDM local memory. 1 An underrun event has occurred in the TDM local memory.
TFTE 30	0	**Transmit First Threshold Event** Indicates whether a first threshold event has occurred. TFTE is set when the first threshold of all the transmit data buffers is empty. The first threshold pointer is determined by the Transmit First Threshold Register (TDMxTFTR). For details, see **Section 20.2.6.3**.	0 No transmit first threshold event has occurred. 1 A transmit first threshold event has occurred.

Table 20-39. TDM*x*TER Bit Descriptions (Continued)

Name	Reset	Description	Settings
TSTE 31	0	**Transmit Second Threshold Event** Indicates whether a transmit second threshold event has occurred. TSTE is set when the second threshold of all the transmit data buffers is empty. The second threshold pointer is determined by the transmit Second Threshold Register (TDMxTSTR). For details, see **Section 20.2.6.3**.	0 No transmit second threshold event has occurred. 1 A transmit second threshold event has occurred.

TDMxASR TDMx Adaptation Status Register

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

TDMxASR contains the status of the adaptation machine. The register can be read at any time. Bits are cleared by writing ones to them; writing zero has no effect.

Table 20-40. TDMxASR Bit Descriptions

Name	Reset	Description	Settings
— 0–30	0	Reserved. Write to zero for future compatibility.	
AMS 31	0	**Adaptation Machine Status** Indicates the status of the adaptation machine. If the bit is set, new sync arrive and the Adaptation Sync Distance Register (TDMxASDR) loads the new value.	0 No sync arrives and TDMxASDR does not contain a new value. 1 New sync arrives and the TDMxASDR register contains a new value that the SC140 core should read.

TDMxRSR TDM*x* Receive Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—													RSSS		RENS
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RSR contains the receiver status. It indicates whether the receiver is synchronized on the receive sync and the receiver is enabled or disabled.

Table 20-41. TDM*x*RSR Bit Descriptions

Name	Reset	Description	Settings
— 0–28	0	Reserved. Write to zero for future compatibility.	
RSSS 29-30	0	**Receive Sync Synchronization Status** Indicates the status of the receive sync synchronization. When the synchronization state is SYNC, the serial part synchronized on the received sync and the received data transfer to the buffer in main memory for processing. **Note:** For details, see **Section 20.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
RENS 31	0	**Receive Enable Status** Indicates whether all the receiver parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clocks domains. **Note:** If the serial clock is not toggling, this bit may not reflect updated values.	0 The receiver machine is disabled. 1 The receiver machine is enabled.

TDM*x*TSR TDM*x* Transmit Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—													TSSS		TENS
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TSR contains the status of the transmitter. It indicates whether the transmitter is synchronized on the transmit sync and whether it is enabled or disabled.

Table 20-42. TDM*x*TSR Bit Descriptions

Name	Reset	Description	Settings
— 0–28	0	Reserved. Write to zero for future compatibility.	
TSSS 29-30	0	**Transmit Sync Synchronization Status** Indicates the transmit sync synchronization status. When the synchronization state is SYNC, the serial part is synchronized on the transmit sync and new transit data is driven out. For details, see **Section 20.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
TENS 31	0	**Transmit Enable Status** Indicates whether all the transmitter parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clock domains. **Note:** If the serial clock is not toggling, this bit may not reflect updated values.	0 The transmit machine is disabled. 1 The transmit machine is enabled.

20.7.4 System Bus Registers

LGTDTEA Local Bus GTD Transfer Error Address

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	ADDRESS															
Type	R															
Reset	undefined															

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

LGTDTEA holds the system address accessed during a TDM transfer error on the Local bus. The Local Bus Transfer Error Status and Control Register (L_TESCR1) indicates whether an error has occurred on the local bus and L_TESCR1[TC] indicates whether the TDM initiated the error access (see page 12-112). The register is undefined at reset.

LGTDTER GTD Transfer Error Requestor Number Register

Bit	0	1	2	3	4	5	6	7
	RQNUM					—		
Type	R							
Reset	Undefined							

LGTDTER contains the identification number of the current requestor that addressed the local bus. The RQNUM of each transaction is held in this register until the transaction completes. The register is undefined at reset.

Table 20-43. LGTDTER Bit Descriptions

Name	Reset	Description	Settings
RQNUM 0–4	—	**Requestor Number** Indicates the number of the requestor accessing the bus when the bus error occurred.	00000 Receive TDM0. 00001 Transmit TDM0. 00010 Receive TDM1. 00011 Transmit TDM1. 00100 Receive TDM2. 00101 Transmit TDM2. 00110 Receive TDM3. 00111 Transmit TDM3. The other values are reserved.
— 5–7	—	Reserved.	

UART 21

The UART, also known as the serial communication interface (SCI), is a full-duplex port for serial communications with other devices. This interface uses two dedicated signals: transmit data (UTXD) and receive data (URXD) (see **Figure 21-1**). The external connections shared by these signals with GPIO[27–28] are available as general-purpose I/O (GPIO) signals when they are not configured for UART operation.

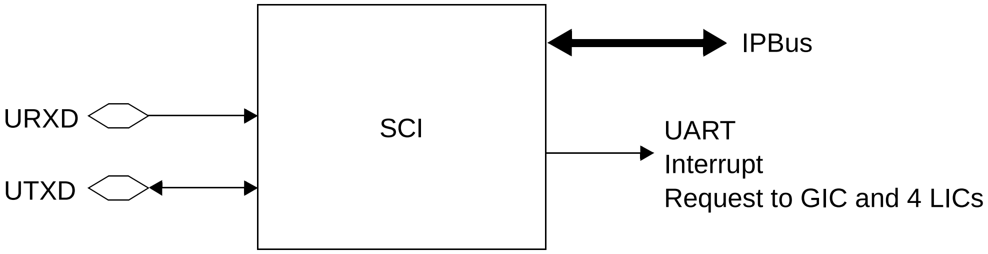


Figure 21-1. UART Interface

The UART is accessible, via the IPBus, to an external host or to each of the SC140 cores. An external host accesses the IPBus via the local bus, and each SC140 core accesses the IPBus via the SQBus.

The UART generates one interrupt signal that connects to each of the SC140 core LICs so that each SC140 core can service UART interrupts. The UART interrupt signal also connects to the GIC, which drives $\overline{\text{INT\_OUT}}$ so that an external host using interrupts on $\overline{\text{INT\_OUT}}$ signal can service UART interrupts. For details on UART interrupt signal connectivity to LICs and GIC, refer to **Chapter 17**, *Interrupt Processing*. When accepting an interrupt request, an SC140 core or external host should read the UART status register (SCISR) to identify the interrupt source and service it accordingly. During reception, the UART generates an interrupt request when a new character is available to the UART data register, SCI Data Register (SCIDR). An SC140 core or external host then reads the character from the data register. During transmission, the UART generates an interrupt request when its data register can be written with new character. An SC140 core or external host then writes the character to the data register.

As **Figure 21-2** shows, the UART allows full duplex, asynchronous, non-return-to-zero (NRZ) serial communication between the MSC8113 and remote devices, including other MSC8113 devices. The UART transmitter and receiver operate independently, although they use the same baud-rate generator and same character length. An SC140 core monitors the status of the UART, writes the data to be transmitted, and processes received data.

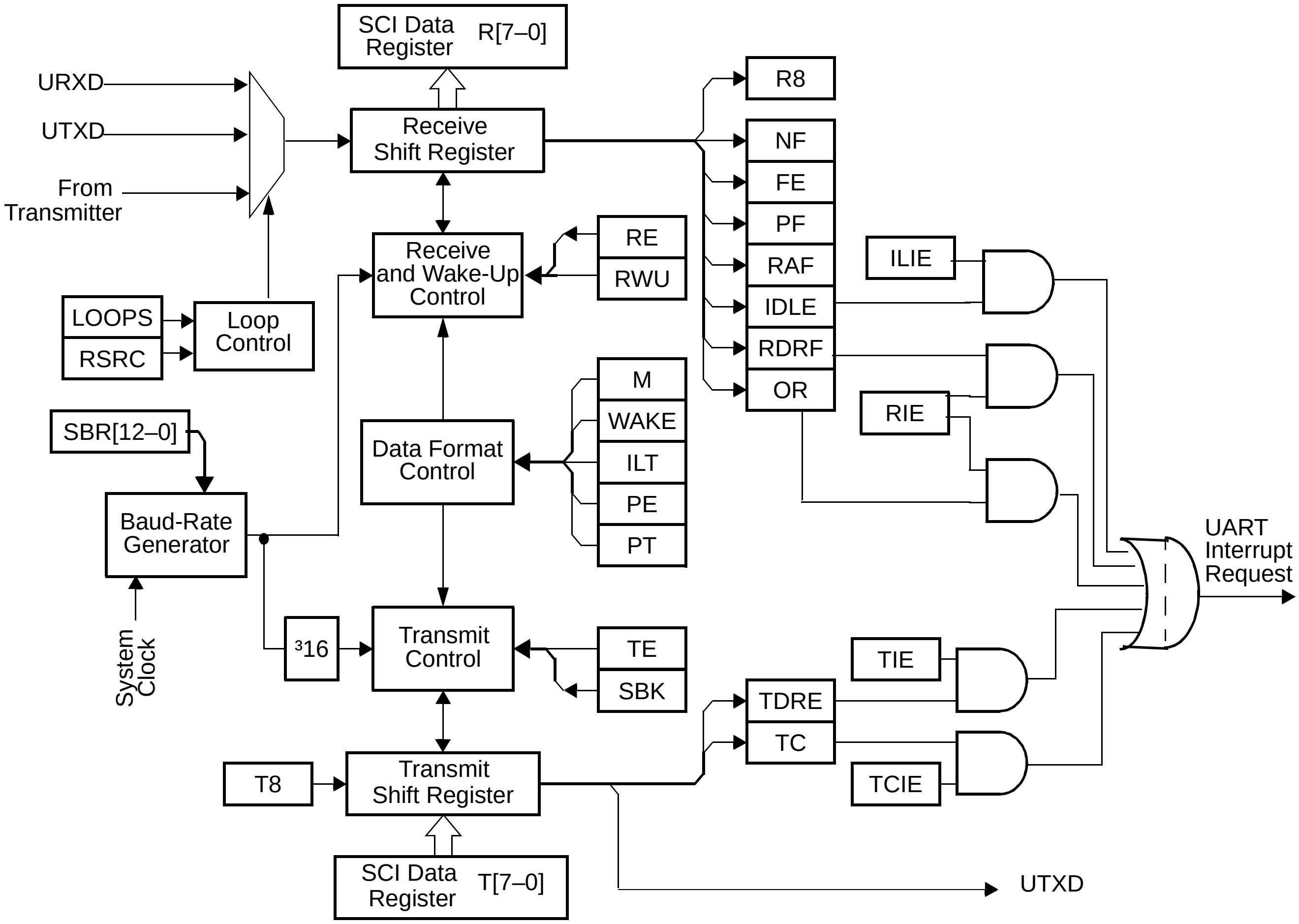
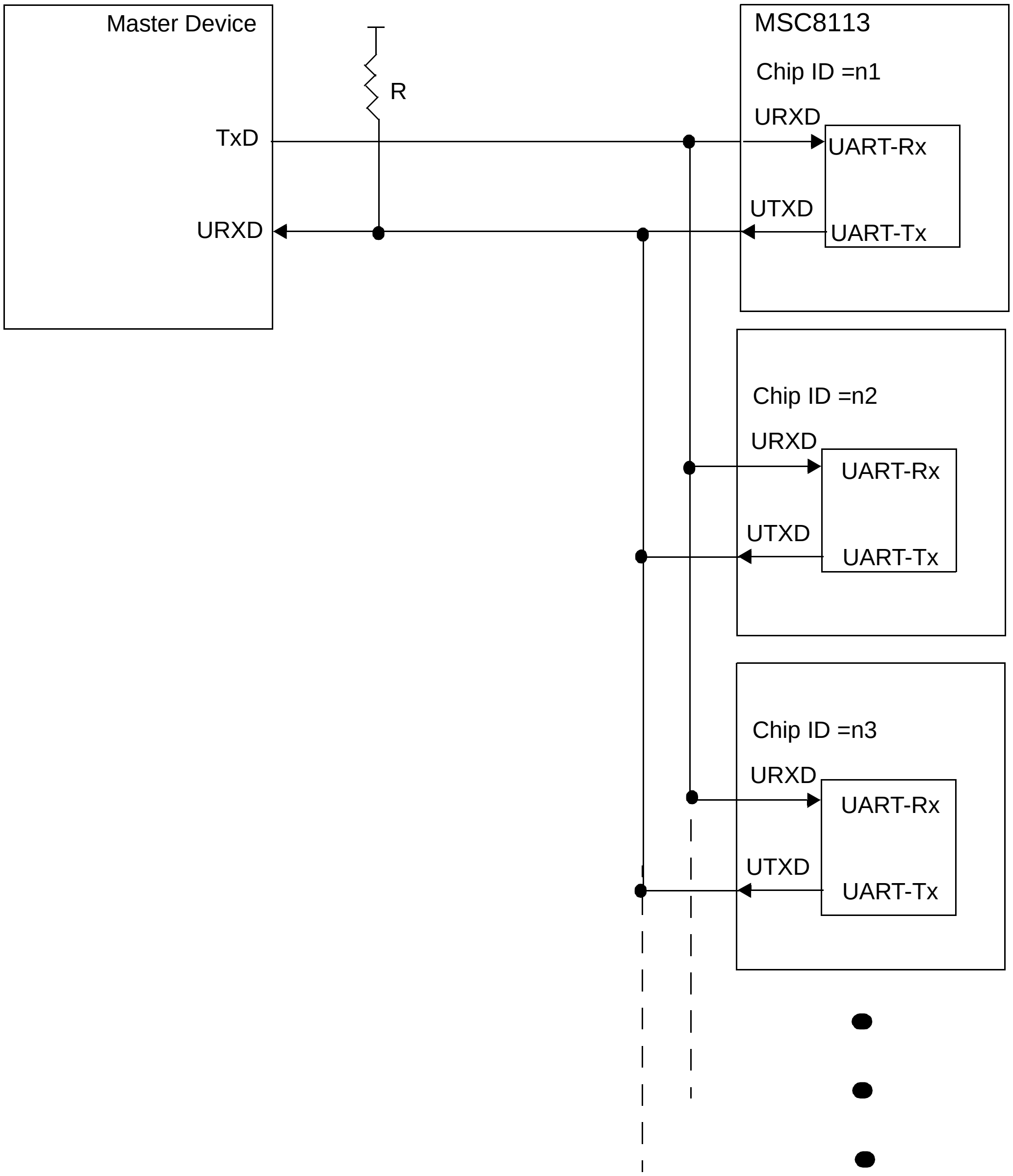


Figure 21-2. UART Block Diagram

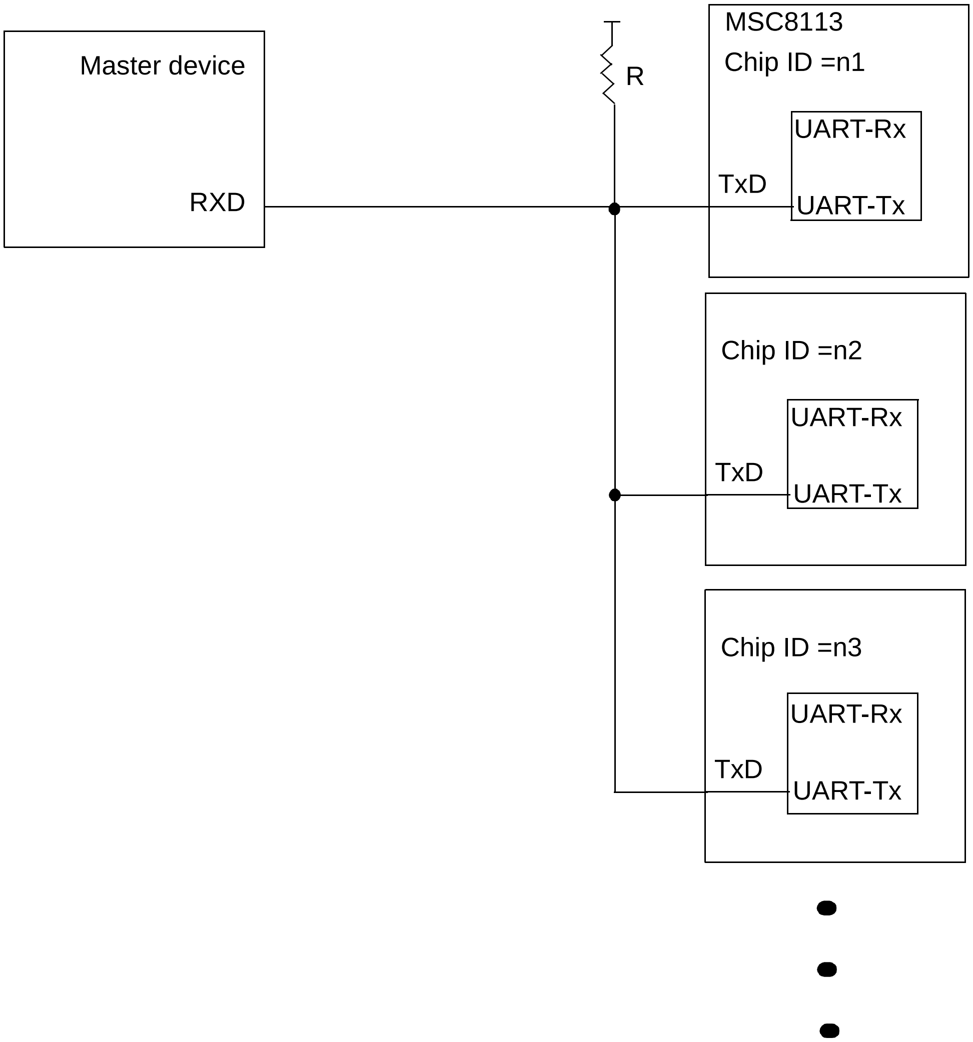
Figure 21-3 shows the full duplex UART system in which the MSC8113 UART transmits and receives simultaneously. A higher-level protocol should handle the full duplex communication to guarantee that no more than one slave UART transmits to the URXD signal of the master at a given time. Receiver wake-up can obtain such a protocol (see **Section 21.2.7**, *Receiver Wake-Up*). The UART UTXD signal can be configured with full CMOS drive or with open-drain drive (see **Chapter 23**, *GPIO*). In both cases, the external pull-up resistor is needed to avoid floating input at the URXD of the master.



Note: The RC value on the MultiPoint TxD may limit system baud rate.

Figure 21-3. Full Duplex Multiple UART System

Figure 21-4 shows the UART on a single-wire connection of a half duplex system. The UTXD signal must be configured with open-drain drive (see **Chapter 23**, *GPIO*) and an external pull-up resistor. For details on single-wire, see **Section 21.4.2**, *Single-Wire Operation*.



Note: The RC value on the MultiPoint UTXD might limit system baud rate.

Figure 21-4. Single-Wire Connection

The UART uses the standard NRZ mark/space data format illustrated in **Figure 21-5**.

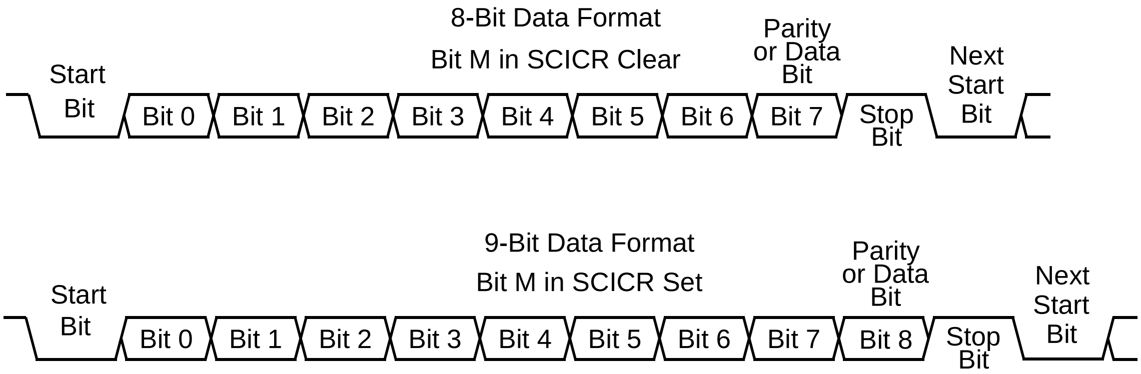


Figure 21-5. UART Data Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI Control Register 1 (SCICR) configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits, including a start bit and a stop bit.

Table 21-1. Examples of 8-Bit Data Format

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Note: The address bit identifies the frame as an address character. The address bit is bit 7 (M = 0) or bit 8 (M = 1) See **Section 21.2.7**, *Receiver Wake-Up*.				

Setting the M bit configures the UART for nine-bit data characters. When the UART is configured for 9-bit data characters, the ninth data bit is the T8 or R8 bit in the SCIDR. T8 remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits, including a start bit and a stop bit.

Table 21-2. Example of 9-Bit Data Format

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1
Note: The address bit identifies the frame as an address character. The address bit is bit 7 (M = 0) or bit 8 (M = 1). See **Section 21.2.7**, *Receiver Wake-Up*.				

A 13-bit modulus counter in the baud-rate generator derives the baud rate for both the receiver and the transmitter. A value ranging from 1 to 8191 is written to the SBR[12–0] bits to determine the system clock divisor. Writing a 0 to SBR[12–0] disables the baud-rate generator. The SBR bits are in the SCI Baud-Rate Register (SCIBR). The baud-rate clock is synchronized with the bus clock and drives the receiver. The baud-rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time. Baud-rate generation is subject to two sources of error:

- Integer division of the system clock may not give the exact target frequency.
- Synchronization with the bus clock can cause phase shift.

Refer to **Section 21.2.2**, *Data Sampling*, for details on adjusting to the received baud rate at the receiver.

Table 21-3 lists some examples of achieving target baud rates with a system clock frequency of 100 MHz, using the following formula:

$$UART\ baud\ rate = System\ clock/(16 \times SCIBR[12\text{–}0])$$

Table 21-3. Baud Rates (System Clock = 100 MHz)

Bits SBR	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (Percentage)
14	7,142,857	446,429	460,000	3.04
27	3,703,704	231,481	230,000	0.64
54	1,851,852	115,741	115,000	0.64
98	1,020,408	63,776	64,000	0.35
163	613,497	38,343.6	38,400	0.15
326	306,748	19,171.8	19,200	0.15
651	153,610	9600.6	9600	0.006
1302	76,804.9	4800.3	4800	0.006
2604	38,402.5	2400.15	2400	0.006
5208	19,201.2	1200.08	1200	0.006
Note: The error relates only to the first source error.				

21.1 Transmitter

The UART transmitter accommodates either 8-bit or 9-bit data characters. The state of the M bit in the SCI Control Register (SCICR) determines the length of the data characters. When 9-bit data is transmitted, bit T8 in the SCIDR is the ninth bit (bit 8).

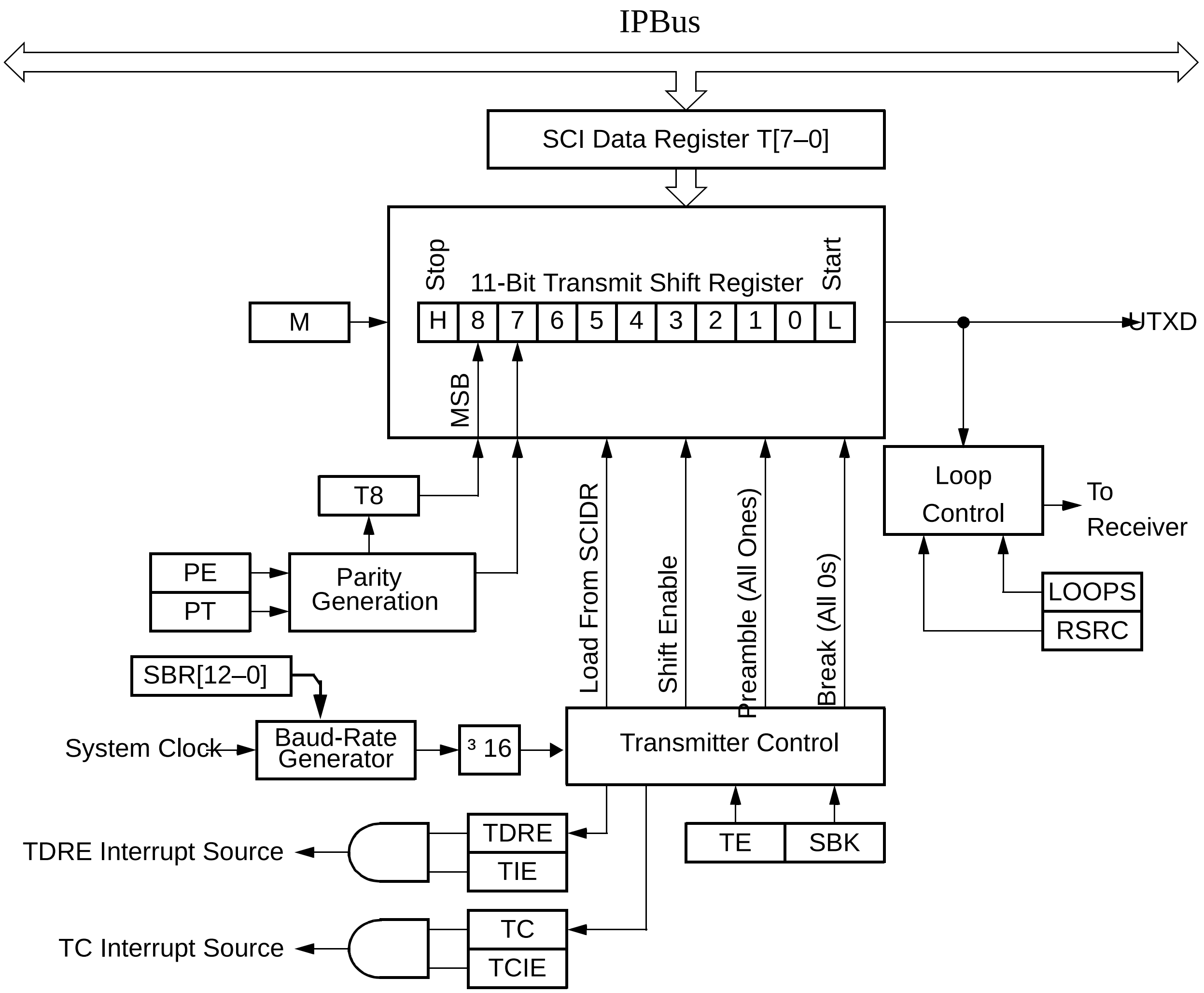


Figure 21-6. Transmitter Block Diagram

21.1.1 Character Transmission

To transmit data, one of the SC140 cores or an external host writes the data character to the SCI Data Register (SCIDR), which is then transferred to the transmitter shift register. The transmitter shift register then shifts out the data bits on the UTXD signal, after it prefaces them with a start bit and appends them with a stop bit. The SCI data register is the write-only buffer between the IPBus and the transmit shift register.

The UART also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDR) to the transmitter shift register. If the Transmit Interrupt Enable (TIE) bit in the SCICR is set, the TDRE flag asserts a UART interrupt request. The transmit interrupt service routine responds to this flag by writing another character to the transmitter buffer (SCIDR), while the shift register is still shifting out the first character. If the TDRE flag is set and no new data or break character transferred to the shift register, the UART sets a flag, transmit complete (TC) and UTXD becomes idle.

Begin a UART transmission as follows:

1. Configure the UART:

   a. Select a baud rate. Write the appropriate value to the SCIBR to start the baud-rate generator. Note that the baud-rate generator is disabled when the baud rate is zero. Writing to the 5 MSB (SBR[12–8]) bits of the SCIBR has no effect without also writing to the 8 LSB of SCIBR (SBR[7–0]).

   b. Configure GPIO28 for UART UTXD (see **Chapter 23**, *GPIO*):

      - Set PAR[DD28] and PSOR[SO28] (PAR[DD28] = PSOR[SO28] = 1) to connect the UART UTXD signal to the output connection.
      - Set PDIR[DR28] (PDIR[DR28] = 1).

   c. Write to the SCICR to configure data length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT) and enable the transmit and receive interrupts as required (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK). A preamble character is now shifted out of the transmitter shift register.

2. Perform the transmit procedure for each character:

   a. Poll the TDRE flag by reading the SCISR or responding to the UART interrupt. Keep in mind that the TDRE reset value is one.

   b. If the TDRE flag is set, write the data to be transmitted to SCIDR, where the ninth bit is written to the T8 bit in SCIDR if the UART is in 9-bit data format. Reading TDRE bit in the SCISR and then writing new data to T[7–0] in the SCIDR clears the TDRE flag. Otherwise, the last data transmitted and then UTXD goes to idle condition, that is, a logic 1 (high).

3. Repeat step 2 for each subsequent transmission.

Note: The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDR, which occurs 9/16ths of a bit time *after* the start of the stop bit of the previous frame.

Note: When the shift register is empty (the TC and TDRE flags are set), transmission starts until one bit time after the data register is written. If only the TC interrupt source is enabled (SCICR[TCIE] = 1, SCICR[TIE] = 0), then you must ensure at least one bit time interval between successive writes to the SCIDR to enable the transmitter software to write twice to the SCIDR per interrupt.

Setting the Transmitter Enable (SCICR[TE]) bit to 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCIDR into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (msb) of the data character is the parity bit.

When the transmit shift register is not transmitting a character, UTXD goes to the idle condition, logic 1. When software clears the SCICR[TE] bit, the transmitter relinquishes control of UTXD.

Note: If SCIDDR[22] is set, UTXD is driven by a logic 0 (pulled down). Otherwise, if SCIDDR[22] is cleared, the UTXD signal is not driven. PAR[DD28] can be cleared to use GPIO28 as general-purpose I/O signal, but set PDIR[DR28] and PDAT[D28] if the UTXD signal must be idle.

If software clears SCICR[TE] while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. Then the transmitter relinquishes control of UTXD even if there is data pending in the SCI data register. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing SCICR[TE].

To separate messages with preambles with minimum idle line time, use the following sequence between messages (see also **Figure 21-7**, *Queuing an Idle Character*):

1. Write the last character of the first message to the SCIDR.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Insert a preamble by clearing and then setting the SCICR[TE] bit.
4. Write the first character of the second message to the SCIDR.

Another way to separate messages with idle line is to wait until the TC flag is set after writing the last character of the first message to SCIDR, indicating this character has already been transmitted. When TC is set, UTXD goes idle. Then, after some idle line time, write the first character of the second message to SCIDR.

21.1.2 Break Characters

Setting the send break bit (SCICR[SBK]) to a value of 1 loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character is ten logic 0s (if M = 0) or eleven logic 0s (if M = 1). As long as SCICR[SBK] is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

21.1.3 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. The length of idle characters depends on the M bit in SCICR. The preamble is a synchronizing idle character that begins the first transmission initiated after the SCICR[TE] bit is written from 0 to 1. Clearing and then setting the SCICR[TE] bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

Note: When queuing an idle character, return the SCICR[TE] bit to logic 1 before the stop bit of the current frame shifts out to UTXD. Setting SCICR[TE] after the stop bit appears on UTXD discards data previously written to the SCI data register. Toggle the SCICR[TE] bit for a queued idle character while the TDRE flag is set and immediately before writing the next character to the SCI data register. See **Figure 21-7**, *Queuing an Idle Character*.

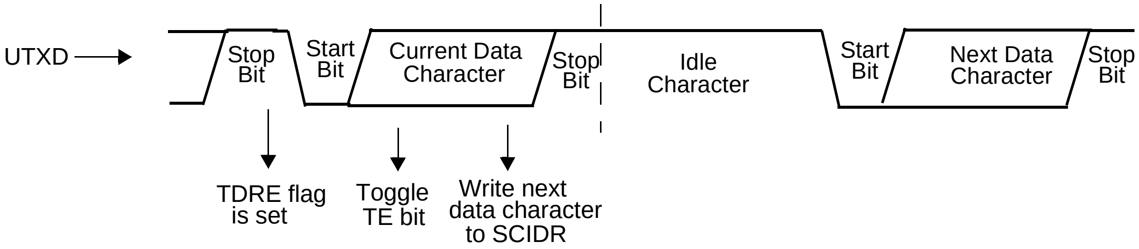


Figure 21-7. Queuing an Idle Character

21.1.4 Parity Bit Generation

The UART can be configured to enable parity bit generation by the parity enable bit (SCICR[PE]). The parity type bit (SCICR[PT]) determines whether to place even or odd parity at T8 (if M = 1) or at T7 (if M = 0) bits of SCIDR.

21.2 Receiver

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the SCICR[M] bit determines the length of data characters. When receiving 9-bit data, bit R8 in the SCIDR is the ninth bit (bit 8).

21.2.1 Character Reception

During a UART reception, the receive shift register shifts a frame in through URXD. The SCI data register is the read-only buffer between the IPBus and the receive shift register. After a complete frame shifts into the receive shift register, the data portion of the frame (the character) is transferred to the SCI data register. The receive data register full flag, RDRF, in SCISR is set, indicating that the received character can be read.

The overrun flag, OR, is set when software fails to read the SCIDR before the receive shift register receives the next character. If the receive interrupt enable bit (SCICR[RIE]) is also set, the Receive Data Register Full (RDRF) or the OR flags generate an interrupt request.

Begin an SCI reception as follows:

1. Configure the SCI:
   a. Select the target baud rate and write the appropriate value to the SCI Baud Rate Register (SCIBR). Note that the baud-rate generator is disabled when the baud rate is zero. Writing to 5 MSB bits of SCIBR (SBR[12–8]) has no effect without also writing to 7 LSB of SCIBR (SBR[7–0]).

   b. Configure GPIO27 for UART URXD (see **Chapter 23**, *GPIO*):

   — Set PAR[DD27] and PSOR[SO27] (PAR[DD27] = PSOR[SO27] =1) to connect the UART URXD signal to the external connection.

   — Clear PDIR[DR27] (PDIR[DR27] = 0).

   c. Write to the SCICR to configure data length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT) and enable the transmitter, interrupts, receive, and wake up as required (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK).If the SBK bit is set, the receiver wakes up if there are particular conditions on the URXD signal according to the WAKE control bit. Refer to **Section 21.2.7**, *Receiver Wake-Up*.
2. Perform the reception procedure for each character:
   a. Poll the RDRF flag by reading the SCISR or responding to the UART interrupt.

   b. If the RDRF flag is set, read the data to be received from SCIDR, where the ninth bit is read from R8 bit in SCIDR if the SCI is in 9-bit data format. Reading RDRF bit at SCISR and then reading new data from SCIDR clears RDRF flag.
3. Repeat step 2 for each subsequent reception.

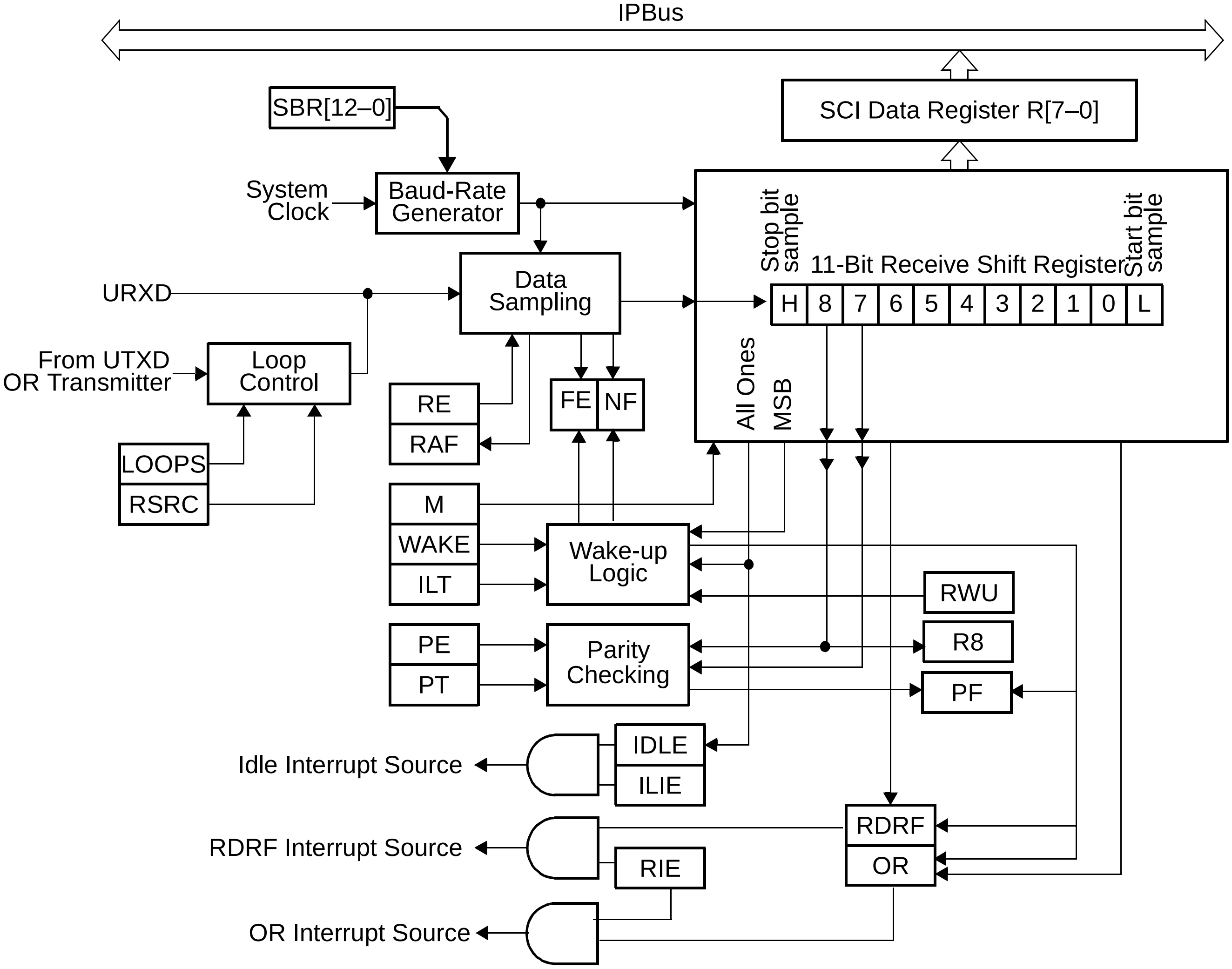


Figure 21-8. UART Receiver Block Diagram

21.2.2 Data Sampling

The receiver samples URXD at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch between the baud rate generated by RT clock and the target baud rate, the RT clock (see **Figure 21-9**) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data sampling logic searches for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock logic begins to count to 16.

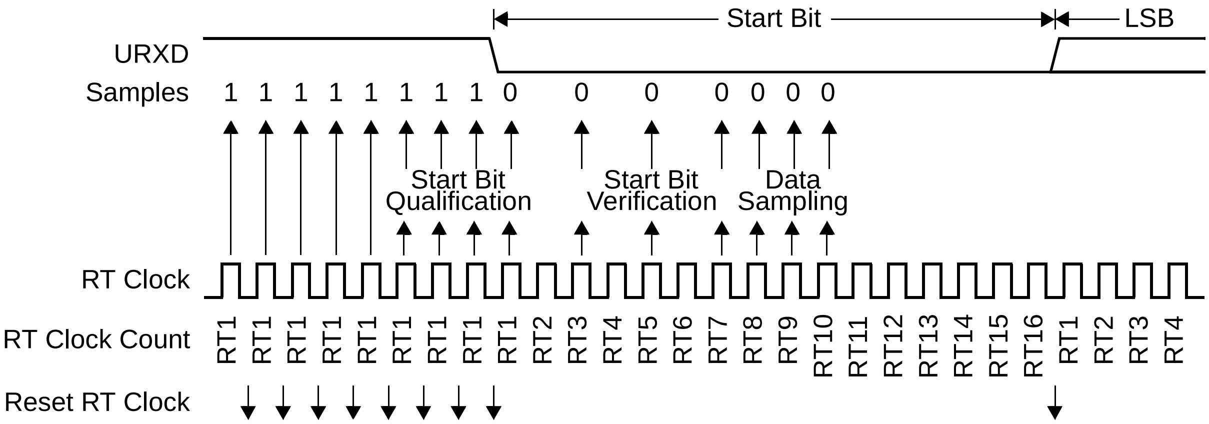


Figure 21-9. Receiver Data Sampling

To verify the start bit and to detect noise, data sampling logic takes samples at RT3 and RT5. If both samples are logic 1 the RT counter is reset and a new search for a start bit begins, else also RT7 sample is taken. If at least two samples (from RT3, RT5, and RT7) are logic 0 then the start bit is perceived. The noise flag, NF, is set if two samples are logic 0 and one is logic 1. **Table 21-4** summarizes the results of the start bit verification samples.

Table 21-4. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
11 (RT7 sample is not taken)	No	0

If start bit verification is not successful, the RT counter is reset and a new search for a start bit begins. To determine the value of a data bit and to detect noise, sample logic takes samples at RT8, RT9, and RT10. The data bit value is determined by the majority of the samples. The noise flag, NF, is set if not all samples have the same logical value. **Table 21-5** summarizes the results of the data bit samples.

Table 21-5. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Note: The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, sample logic takes samples at RT8, RT9, and RT10. The noise flag, NF, is set if not all samples have the same logical value (the same as for data bit sampling). If the majority of the samples are logic 0 framing error flag, FE, is set. **Table 21-6** summarizes the results of the stop bit samples.

Table 21-6. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In **Figure 21-10** the start bit verification samples RT3 and RT5 determine that the first logic 0 detected is noise and not the beginning of a start bit. The RT counter is reset and the start bit search resumes. The noise flag is not set because the noise occurred before the start bit was found.

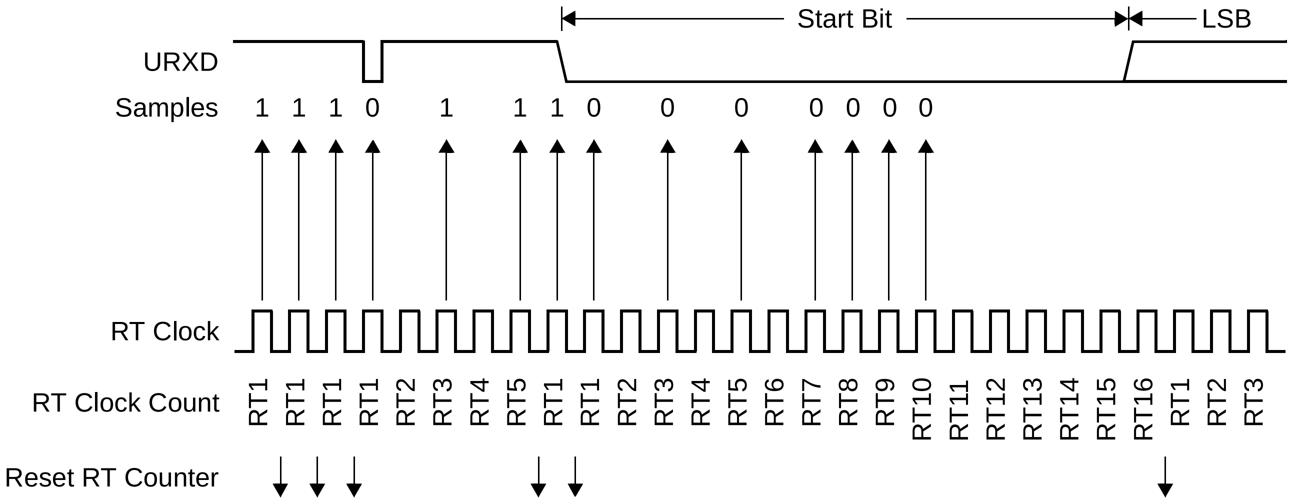

Start Bit
LSB
URXD
Samples
1 1 1 0 1 1 1 0 0 0 0 0 0 0
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3
Reset RT Counter

Figure 21-10. Start Bit Search Example 1

In **Figure 21-11**, the verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 of the next bit are within the bit time and data recovery is successful.

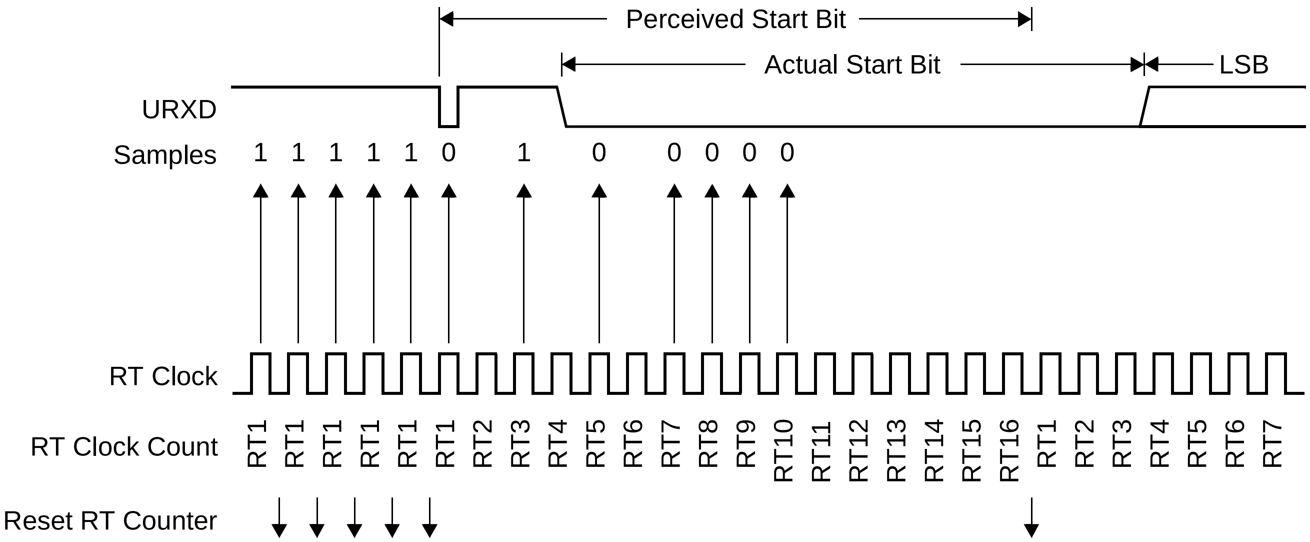

Perceived Start Bit
Actual Start Bit
LSB
URXD
Samples
1 1 1 1 1 0 1 0 0 0 0 0
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3 RT4 RT5 RT6 RT7
Reset RT Counter

Figure 21-11. Start Bit Search Example 2

In **Figure 21-12** the first start bit verification is a case similar to that in **Figure 21-10**, *Start Bit Search Example 1*, but this time the first logic 0 detected is the real beginning of start bit. The noise is at samples R3 and R5 which causes the RT counter to reset and the start bit search begins again. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 of the next bit are within the bit time and data recovery is successful. For this case the noise and framing error flags are not set.

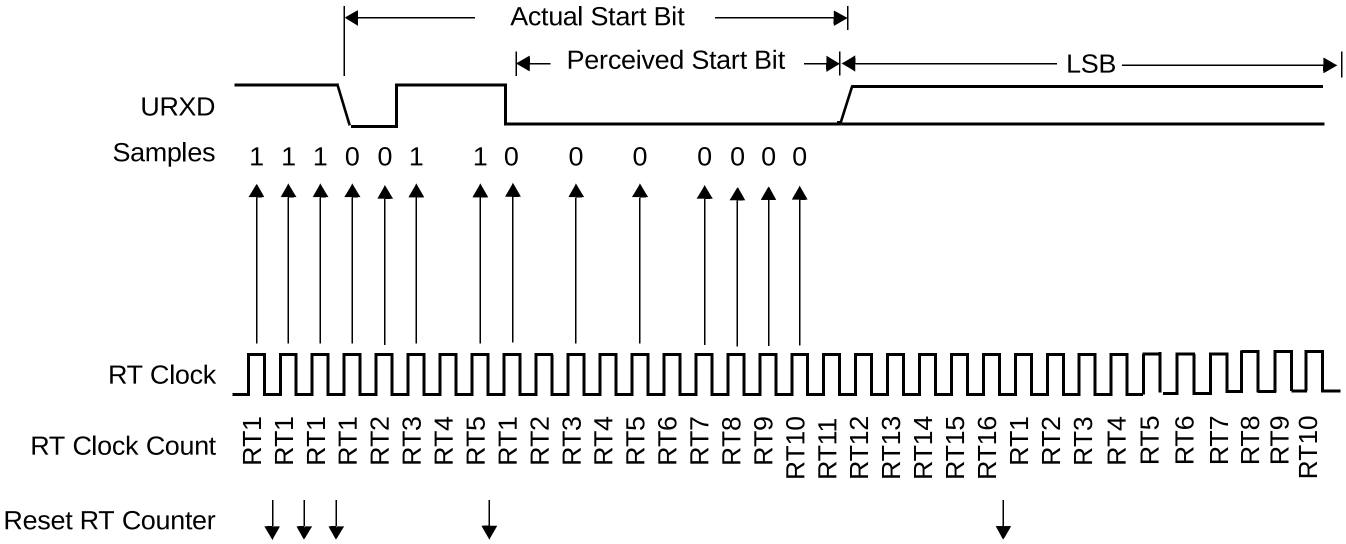
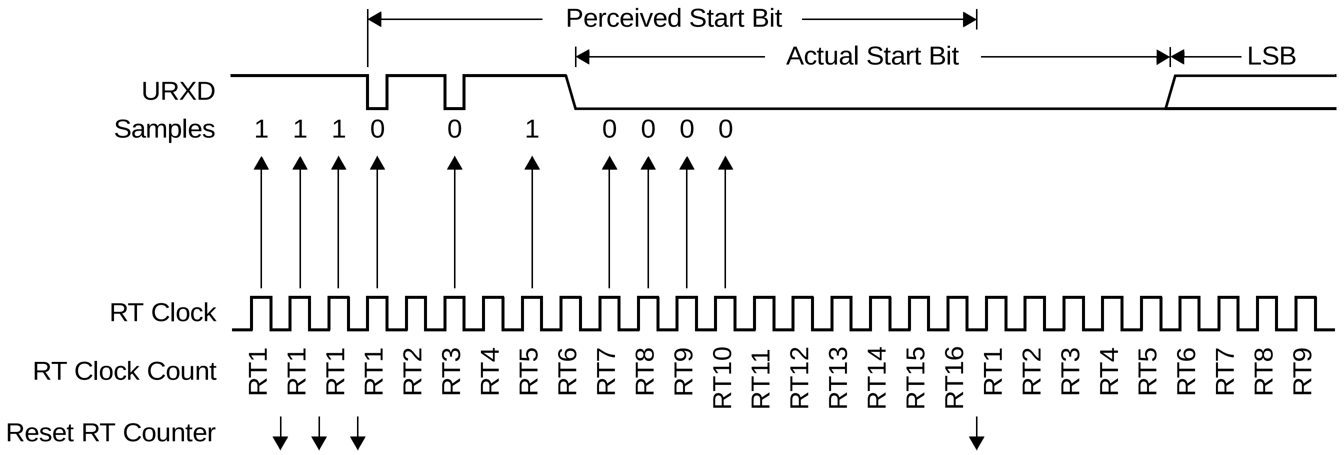

Actual Start Bit
Perceived Start Bit
LSB
URXD
Samples
1 1 1 0 0 1 1 0 0 0 0 0 0 0
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10
Reset RT Counter

Figure 21-12. Start Bit Search Example 3

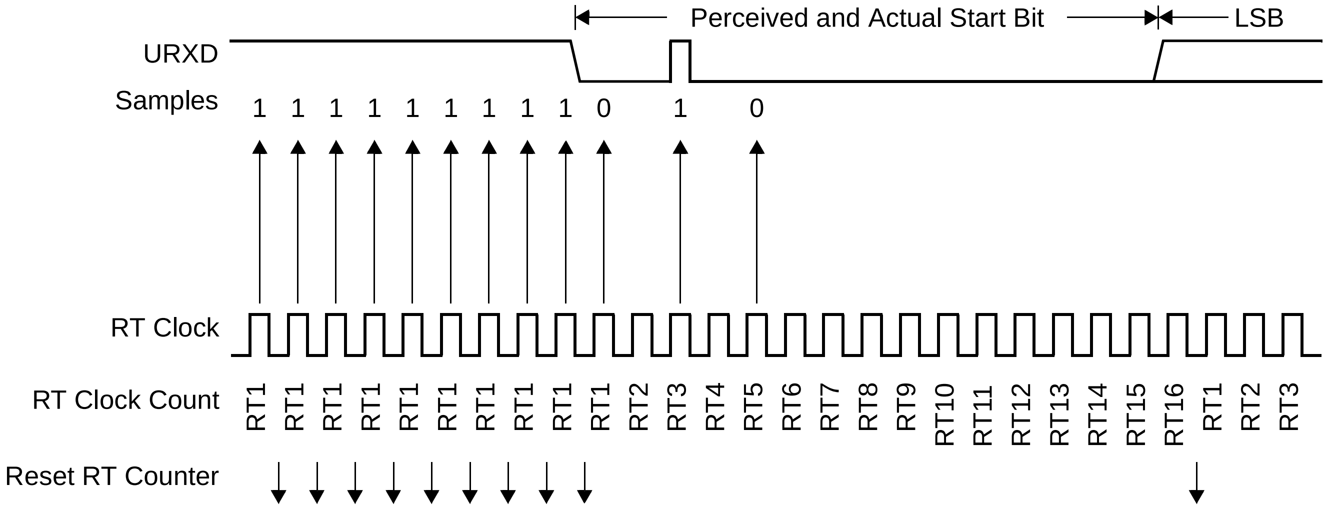
In **Figure 21-13**, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the RT8, RT9, and RT10 data samples of the next bit are within the bit time, and data recovery is successful.



Perceived Start Bit
Actual Start Bit
LSB
URXD
Samples
1 1 1 0 0 1 0 0 0 0
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9
Reset RT Counter

Figure 21-13. Start Bit Search Example 4

Figure 21-14 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.



Perceived and Actual Start Bit
LSB
URXD
Samples
1 1 1 1 1 1 1 1 1 0 1 0
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3
Reset RT Counter

Figure 21-14. Start Bit Search Example 5

Figure 21-15 shows a burst of noise near the beginning of the start bit that causes the start bit not to be found and resets the RT counter. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

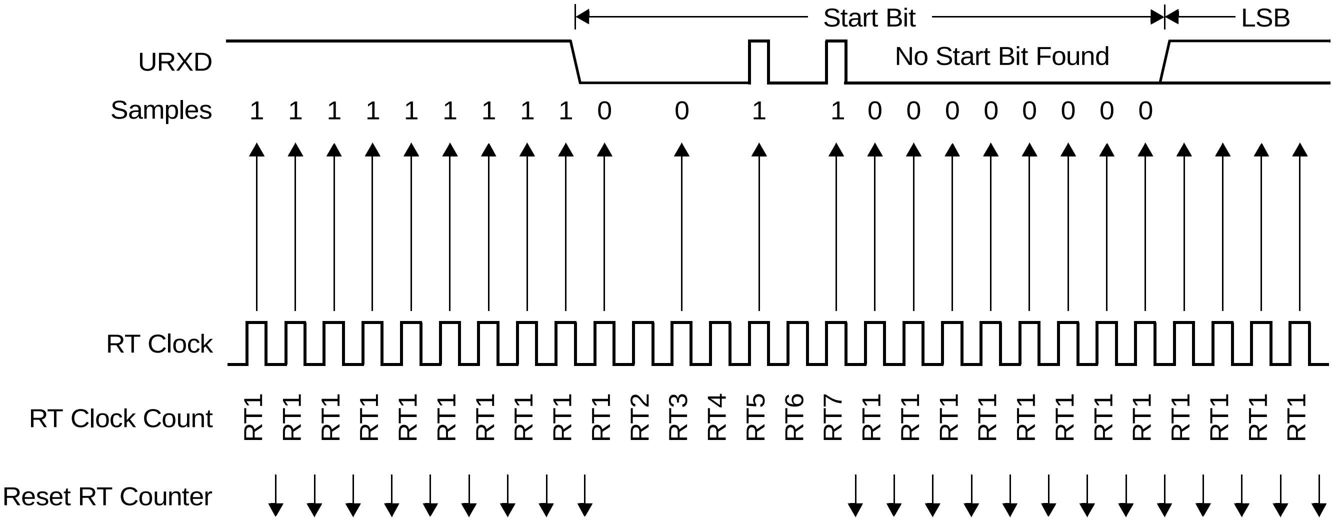


Figure 21-15. Start Bit Search Example 6

In **Figure 21-16**, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT counter. In the start bits only, the RT8, RT9, and RT10 data samples are not used for determining bit value.

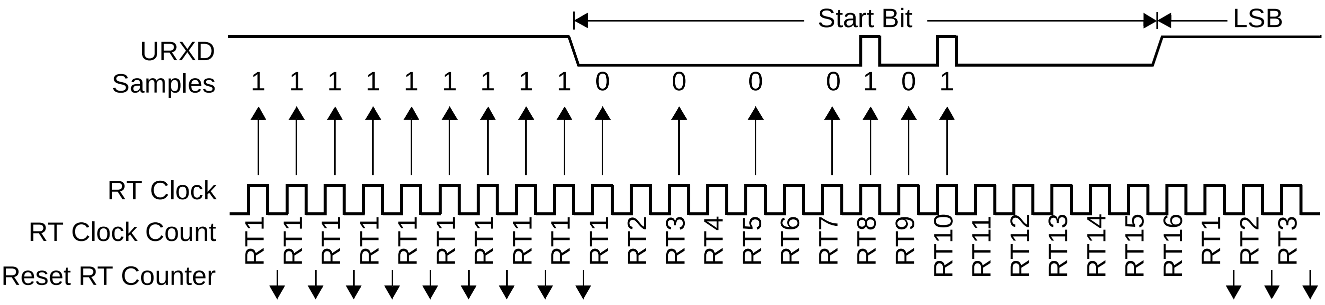

Start Bit
LSB
URXD
Samples
1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1
RT Clock
RT Clock Count
RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3
Reset RT Counter

Figure 21-16. Start Bit Search Example 7

21.2.3 Framing Error

If the data sampling logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, SCISR[FE]. A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set. FE inhibits further data reception until it is cleared. Clear SCISR[FE] by reading SCISR and then reading the SCIDR.

21.2.4 Parity Error

The UART can be configured to enable parity check via the Parity Enable (PE) bit in the SCICR. The parity type (SCICR[PT]) determines whether to check for even or odd parity. The Parity Error Flag, SCISR[PF], is set when the parity enable bit is set and the parity of the received character does not match the PT bit. Clear SCISR[PF] by reading the SCISR and then reading SCIDR.

21.2.5 Break Characters

The UART recognizes a break character as a start bit followed by 8 or 9logic 0 data bits and a logic 0 stop bit. Receiving a break character has the following effects on UART registers:

1. The framing error flag (SCISR[FE]) is set.
2. The receive data register full flag (SCISR[RDRF]) is set.

Note: Once the RDRF flag is cleared after being set by a break character, a valid frame must set the RDRF flag again before another break character can set it again.

3. The SCIDR is cleared.
4. The overrun flag (OR), noise flag (NF), parity error flag (PF), or the receiver active flag (RAF) is set (see the discussion in **Section 21.6**).

21.2.6 Baud-Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error occurs if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error occurs if the receiver clock is misaligned so that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero. In most applications, the baud-rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects a misalignment between transmitter bit times and receiver bit times.

21.2.6.1 Slow Data Tolerance

Figure 21-17 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

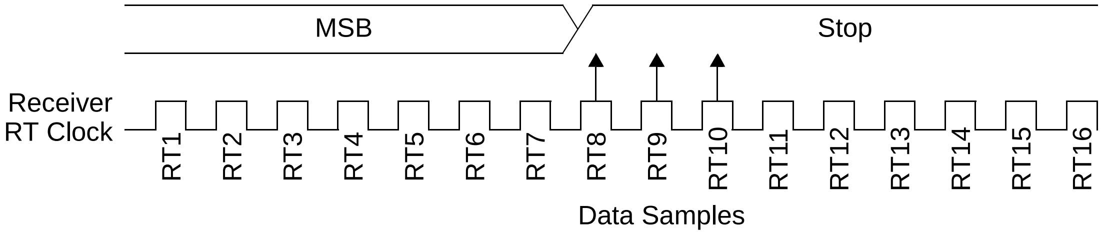


Figure 21-17. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver 9 bit × 16 RT cycles +10 RT cycles =154 RT cycles. With the misaligned character shown in **Figure 21-17**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$((154 - 147) / 154) \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles. With the misaligned character, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit × 16 RT cycles + 3 RT cycles = 163 RT cycles. The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) / 170) \times 100 = 4.12\%$$

21.2.6.2 Fast Data Tolerance

Figure 21-18 shows how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16, but it is still sampled at RT8, RT9, and RT10.

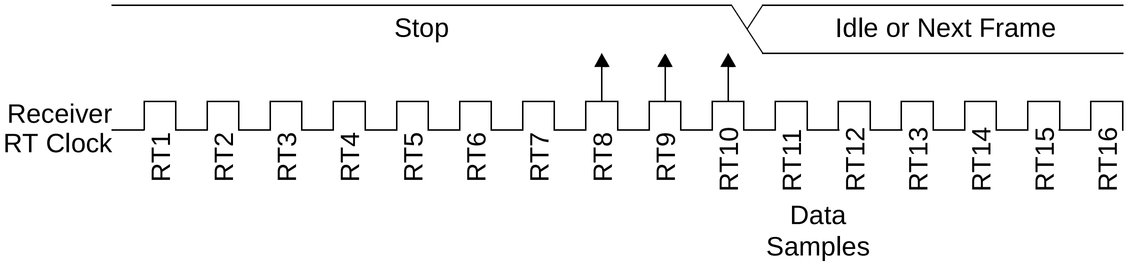


Figure 21-18. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver 9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles. With the misaligned character shown in Figure 21-18, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) / 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles. With the misaligned character, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit × 16 RT cycles = 176 RT cycles. The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) / 170) \times 100 = 3.53\%$$

21.2.7 Receiver Wake-Up

The receiver can be put into a standby state, so that the UART (SCI) can ignore transmissions intended only for other receivers in multiple-receiver systems. This is sometimes called putting the receiver to sleep. Setting the receiver wake-up (RWU) bit in the SCICR puts the receiver into a standby state during which receiver interrupts are disabled. The SCI still loads the receive data into the SCIDR, but it does not set the SCISR[RDRF] flag or any other flag. The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message. Once the receiver is asleep, there must be a wake-up procedure to allow it to respond to messages addressed to it. The SCICR[WAKE] bit determines how the SCI is brought out of the standby state to process an incoming message. This wake bit enables either idle line wake-up or address mark wake-up.

21.2.7.1 Idle Input Line Wake-Up (WAKE = 0)

In idle input line wake-up, an idle condition on URXD (all logic 1s) clears the SCICR[RWU] bit and wakes up the SCI. The initial frame or frames of every message contain addressing information. All receiver software evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its SCICR[RWU] bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on URXD.

Idle line wake-up requires that messages be separated by at least one idle character and that no message contain idle characters. The idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, SCISR[RDRF]. The idle line type bit, SCICR[ILT], determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

Note: With the WAKE bit clear, setting the SCICR[RWU] bit after URXD has been idle can cause the receiver to wake up immediately.

21.2.7.2 Address Mark Wake-Up (WAKE = 1)

In address mark wake-up, a logic 1 in the MSB position of a frame clears the SCICR[RWU] bit and wakes up the SCI. This frame is considered to contain an address character. Hence, all data characters should have their MSB at zero. Each receiver’s software evaluates the addressing information when awakened and compares it to its own address. If the addresses match, the receiver(s) process the frames that follow. If the addresses do not match, the receiver software puts the receiver to sleep by setting the SCICR[RWU] bit. The RWU bit remains set and the receiver remains on standby until another address frame appears on URXD.

The logic 1 in the MSB of an address character clears the receiver RWU bit before the stop bit is received and sets the SCISR[RDRF] interrupt flag. Address mark wake-up allows messages to contain idle characters but requires that the MSB be reserved for use in address frames.

21.3 Reset Initialization

After reset the UART transmitter and receiver are disabled and UTXD and URXD are not driven. For information on initializing the transmitter, refer to **Section 21.1.1**. For information on initializing the receiver, refer to **Section 21.2.1**.

21.4 Modes of Operation

The following sections summarize the UART modes of operation.

21.4.1 Run Mode

Run mode is the normal mode of operation.

21.4.2 Single-Wire Operation

Normally, the UART (SCI) uses two signals for transmitting and receiving data. In single-wire operation, URXD is disconnected from the UART and is available as a GPIO signal (see **Chapter 23**, *GPIO*). The UART uses UTXD for both receiving and transmitting data. Setting the data direction bit for UTXD, SCIDDR[22], configures UTXD as the output for transmitted data. Clearing the data direction bit, SCIDDR[22], disables the transmitter to drive UTXD.

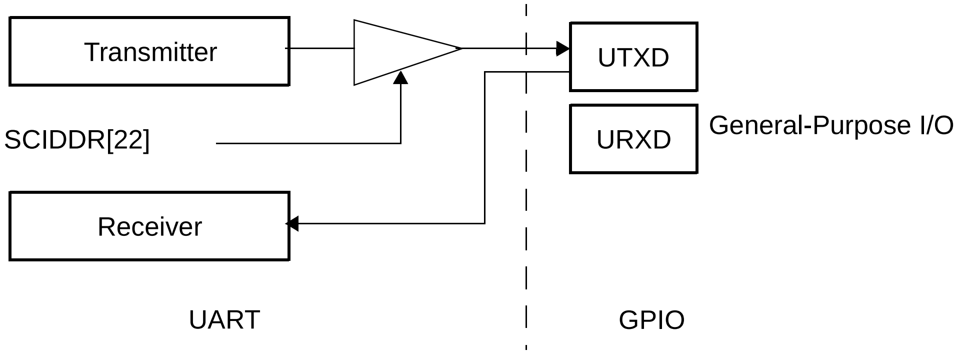


Figure 21-19. Single-Wire Operation (LOOPS = 1, RSRC = 1)

Enable single-wire operation by setting the SCICR[LOOPS] bit and the receiver source bit, SCICR[RSRC]. Setting the SCICR[LOOPS] bit disables the path from URXD to the receiver. Setting the SCICR[RSRC] bit connects the receiver input to the output of UTXD. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1). The PODR bit, which corresponds to UTXD at the PODR of GPIO (see **Chapter 23**, *GPIO*), configures UTXD for full CMOS drive or for open-drain drive. The PODR bit controls UTXD in both normal operation and single-wire operation. The PODR bit allows the UTXD outputs to be tied together in a multiple-transmitter system. Then the UTXD signals of nonactive transmitters follow the logic level of an active one. External pull-up resistors are necessary on open-drain outputs.

21.4.3 Loop Operation

To help isolate system problems, the Loop mode is sometimes used to check software without changing the physical connections in the external system. In Loop mode, the transmitter output is connected to the receiver input internally. The URXD signal is disconnected from the external connection which then becomes available for use as a GPIO signal. Clearing the data direction bit of UTXD disconnects the transmitter output from the external connection.

To enable loop operation, set the SCICR[LOOPS] bit and clear SCICR[RSRC]. Setting the SCICR[LOOPS] bit disables the path from URXD to the external output connection. Clearing the SCICR[RSRC] bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1) for loop operation.

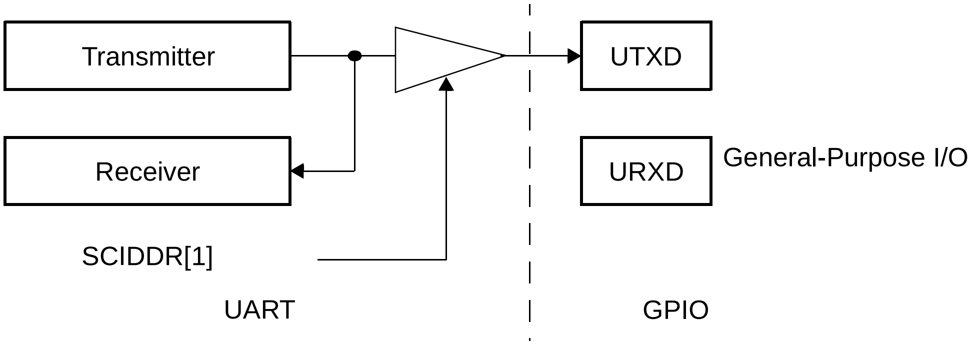


Figure 21-20. Loop Operation (LOOPS = 1, RSRC = 0)

21.4.4 Stop Mode

The UART stops its clock to provide reduced power consumption when the UART_STC bit is set in the Stop Control Register (see **page 19-6**). When the UART enters Stop mode, the states of the UART registers are unaffected. The UART registers cannot be accessed during Stop mode. When the UART_STC bit is cleared, UART operation resumes. Entering Stop mode during a transmission or reception results in invalid data. Therefore, disable the receiver and transmitter (SCICR[TE] = 0, SCICR[RE] = 0) before entering Stop mode.

21.4.5 Receiver Standby Mode

Refer to **Section 21.2.7**, *Receiver Wake-Up*.

21.5 Interrupt Operation

Table 21-7 lists the five interrupts generated by the UART to communicate with an SC140 core or external host.The UART outputs only one signal, which can be activated by each of the five interrupt sources (refer to **Figure 21-1**, *UART Interface*, on page 21-1). Receiver interrupts are disabled when the receiver is in standby state (RWU is set).

Table 21-7. UART Interrupt Sources

Source	Transmitter/ Receiver	Interrupt Enable Bit	Flag at Status Register	Description
TDRE	T	TIE:SCICR[24]	SCISR[16]	Indicates that a character was transferred from SCIDR to the transmit shift register.
TC	T	TCIE:SCICR[25]	SCISR[17]	Indicates that a transmit is complete.
RDRF	R	RIE:SCICR[26]	SCISR[18]	Indicates that received data is available in SCIDR.
OR	R	RIE:SCICR[26]	SCISR[20]	Indicates an overrun condition.
IDLE	R	ILIE:SCICR[27]	SCISR[19]	Indicates that receiver input has become idle.
Note: For details, refer to SCI Status Register (SCISR), on **page 21-28**				

The UART (SCI) only originates interrupt requests. An interrupt source flag (see **Table 21-7**) generates interrupt request if its associated interrupt enable bit is set. The interrupt vector offset and interrupt number are chip dependent.

21.6 UART Programming Model

All UART registers are mapped into the IPBus address space. Refer to **Section 8.5**, *IPBus Address Space* for the UART Base address. This section describes the UART (SCI) module registers, which are listed as follows:

- SCI Baud-Rate Register (SCIBR), on **page 21-25**.
- SCI Control Register (SCICR), on **page 21-25**.
- SCI Status Register (SCISR), on **page 21-28**.
- SCI Data Register (SCIDR), on **page 21-30**.
- SCI Data Direction Register (SCIDDR), on **page 21-31**.

SCIBR SCI Baud-Rate Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—			SBR12	SBR11	SBR10	SBR9	SBR8	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SCIBR determines the SCI baud rate. A write to SCIBR[19–23] has no effect without a write to SCIBR[24–31], since writing to SCIBR[19–23] puts the data in a temporary location until SCIBR[24–31] is written.

Note: The formula for calculating the baud rate is: SCI baud rate = SCI system clock/(16 × SBR).

Note: The baud-rate generator is disabled until the SCICR[TE] bit or the SCICR[RE] bit is set for the first time after reset. The baud-rate generator is disabled when SBR = 0.

Table 21-8. SCIBR Bit Descriptions

Name	Reset	Description	Settings
— 0-18	0	Reserved. Write to zero for future compatibility.	
SBR[12–0] 19-31	4	**SCI Baud Rate** The baud-rate register used by the counter to determine the baud rate of the SCI.	Can contain a value from 1 to 8191.

SCICR SCI Control Register 1

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	LOOPS	—	RSRC	M	WAKE	ILT	PE	PT	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-9. SCICR Bit Descriptions

Name	Reset	Description	Settings
— 0-15	0	Reserved. Write to zero for future compatibility.	
LOOPS 16	0	**Loop Select Bit** Disables the path from URXD to the receiver input for loop (RSRC = 0) or single-wire mode (RSRC =1). See **Table 21-10**. The transmitter and the receiver must be enabled to use the loop functions. The receiver input is determined by the RSRC bit. The transmitter output is controlled by SCIDDR[22] bit. If SCIDDR[22], the data direction bit for UTXD is set and LOOPS = 1, the transmitter output drives UTXD. If the data direction bit is clear and LOOPS =1, the SCI transmitter does not drive UTXD.	1 Loop operation enabled. 0 Normal operation enabled.
— 17	0	Reserved. Write to zero for future compatibility.	
RSRC 18	0	**Receiver Source Bit** When LOOPS = 1, determines the internal feedback path for the receiver.	1 Receiver input connects to UTXD. 0 Receiver input internally connected to transmitter output.
M 19	0	**Data Format Mode Bit** Determines whether data characters are eight or nine bits long.	1 One start bit, nine data bits, one stop bit. 0 One start bit, eight data bits, one stop bit.
WAKE 20	0	**Wake** Determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on URXD (10 consecutive logic 1s if M = 0 or 11 consecutive logic 1s if M=1).	1 Address mark wake-up. 0 Idle line wake-up.
ILT 21	0	**Idle Line Type Bit** Determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.	1 Idle character bit count begins after stop bit. 0 Idle character bit count begins after start bit.
PE 22	0	**Parity Enable Bit** Enables the parity function. When enabled, the parity function inserts (when transmitter enabled) and checks (when receiver enabled) a parity bit at the most significant bit position.	1 Parity function enabled. 0 Parity function disabled.
PT 23	0	**Parity Type Bit** Determines whether the SCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit.	1 Odd parity. 0 Even parity.

Table 21-9. SCICR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TIE 24	0	**Transmitter Interrupt Enable** Enables the transmit data register empty flag, TDRE, to generate interrupt requests. **Note:** Since SCISR[TDRE] reset value is 1, setting TIE immediately after reset results in a UART interrupt request, regardless of SCICR[TE].	1 TDRE interrupt source enabled. 0 TDRE interrupt source disabled.
TCIE 25	0	**Transmission Complete Interrupt Enable** Enables the transmission complete flag, TC, to generate interrupt requests. **Note:** Since the SCISR[TC] reset value is 1, setting TCIE immediately after reset results in a UART interrupt request, regardless of SCICR[TE].	1 TC interrupt source enabled. 0 TC interrupt source disabled.
RIE 26	0	**Receiver Full Interrupt Enable** Enables the receive data register full flag, RDRF, and the overrun flag, OR, to generate interrupt requests.	1 RDRF and OR interrupt sources enabled. 0 RDRF and OR interrupt sources disabled.
ILIE 27	0	**Idle Line Interrupt Enable** Enables the idle line flag, IDLE, to generate interrupt requests.	1 IDLE interrupt source enabled. 0 IDLE interrupt source disabled.
TE 28	0	**Transmitter Enable** Enables the SCI transmitter. The TE bit can be used to queue an idle preamble.	1 Transmitter enabled. 0 Transmitter disabled.
RE 29	0	**Receiver Enable** Enables the SCI receiver.	1 Receiver enabled. 0 Receiver disabled.
RWU 30	0	**Receiver Wake-Up** Enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.	1 RWU, Standby state. 0 Normal operation.
SBK 31	0	**Send Break** Toggling this bit sends one break character (10 or 11 logic 0s). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send break characters.	1 Transmit break characters. 0 No break characters.

Table 21-10. Loop Functions

LOOPS	RSRC	SCIDDR[22]	Function
0	x	x	Normal operation
1	0	0	Loop mode, UTXD is not driven by the SCI transmitter
1	0	1	Loop mode, UTXD is driven by the SCI transmitter
1	1	0	Single-wire mode UTXD acting as an input for the received data. The external connection that URXD shares can be configured as a GPIO.
1	1	1	Single-wire mode with UTXD acting as an output for the transmitted data. The transmitted data is also internally connected to the receiver input. The external connection that URXD shares can be configured as a GPIO.
Note: The function is described for the case in which the PAR[DD28] and PSOR[SO28] are both set (see **Chapter 23**, *GPIO*).			

SCISR SCI Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	—							RAF
Type	R								R/W							R
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCISR can be read any time. A write has no meaning or effect.

Table 21-11. SCISR Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
TDRE 16	1	**Transmit Data Register Empty Flag** Set when the transmit shift register receives a character from the SCI data register. When TDRE is 1, the transmit data register (SCIDR) is empty and can receive a new value to transmit. This flag can generate an interrupt request (refer to **Section 21.5**). Clear TDRE by reading TDRE and then writing to T[7–0] in the SCIDR.	1 Character transferred to transmit shift register; transmit data register empty. 0 No character transferred to transmit shift register.
TC 17	1	**Transmit Complete Flag** Set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, UTXD becomes idle (logic 1). This flag can generate an interrupt request (refer to **Section 21.5**). Clear TC by reading TC and then writing to T[7–0] in the SCIDR. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. Also, TC is cleared in the event of a simultaneous set and clear of the TC flag (transmission not complete).	1 No transmission in progress. 0 Transmission in progress.
RDRF 18	0	**Receive Data Register Full Flag** Set when the data in the receive shift register transfers to the SCI data register. This flag can generate an interrupt request (refer to **Section 21.5**). Clear RDRF by reading RDRF bit at SCISR and then reading R[7–0] in the SCIDR. **Note:** Once the RDRF flag is cleared, after it is set by a break or idle character, a valid frame must set the RDRF flag before another break or idle character can set it again.	1 Received data available in SCI data register. 0 Data not available in SCI data register.

Table 21-11. SCISR Bit Descriptions (Continued)

Name	Reset	Description	Settings
IDLE 19	0	**Idle Line Flag** Set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag.This flag can generate an interrupt request (refer to **Section 21.5**). Clear IDLE by reading IDLE and then reading R[7–0] in the SCIDR. **Note:** When the receiver wake-up bit (RWU) is set, an idle line condition does not set the IDLE flag.	1 Receiver input has become idle. 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared.
OR 20	0	**Overrun Flag** Set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. This flag can generate an interrupt request (refer to **Section 21.5**). Clear OR by reading OR then reading R[7–0] in the SCIDR.	1 Overrun. 0 No overrun.
NF 21	0	**Noise Flag** Set when the SCI detects noise on the receiver input. NF is set during the same cycle as the RDRF flag but is not set for an overrun. Clear NF by reading NF and then reading R[7–0] in the SCIDR.	1 Noise. 0 No noise.
FE 22	0	**Framing Error Flag** Set when a logic 0 is accepted as the stop bit. FE is set during the same cycle as the RDRF flag but is not set for an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading FE and then reading R[7–0] in the SCIDR.	1 Framing error. 0 No framing error.
PF 23	0	**Parity Error Flag** Set when the parity enable bit, PE, is set and the parity of the received data does not match its parity bit. Clear PF by reading PF and then reading R[7–0] in the SCIDR.	1 Parity error. 0 No parity error.
— 24–30	0	Reserved. Write to zero for future compatibility.	
RAF 31	0	**Receiver Active Flag** Set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character.	1 Reception in progress. 0 No reception in progress.

SCIDR SCI Data Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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	
	R8	T8	—						R7	T7	R6	T6	R5	T5	R4	T4
Type	R	R/W	R/W						R	W	R	W	R	W	R	W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: In the SCIDR, writing affects only T[8–0]; writing to R[8–0] has no effect.

Table 21-12. SCIDR Bit Descriptions

Name		Reset	Description	Settings
— 0–15		0	Reserved. Write to zero for future compatibility.	
R8 16		0	**Received Bit 8** The ninth data bit received when the SCI is configured for 9-bit data format (M = 1).	
T8 17		0	**Transmit Bit 8** The ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1).	
— 18–23		0	Reserved. Write to zero for future compatibility.	
24–31	**R[7–0]**	0	**Received Bits 7–0** Received bits seven through zero for 9-bit or 8-bit data formats.	
	T[7–0]		**Transmit Bits 7–0** Transmit bits seven through zero for 9-bit or 8-bit formats.	

Notes: **1.** If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten. The same value is transmitted until T8 is rewritten.

2. In 8-bit data format, only SCIDR[24–31] needs to be accessed.

3. When transmitting in 9-bit data format, write to SCIDR[16–31] (one access). Otherwise, write first to T8 and then to the low byte (SCIDR[24–31]).

SCIDDR SCI Data Direction Register

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

When LOOPS is cleared and TE is set, UTXD is an output regardless of the state of SCIDDR[DDRTX].

Table 21-13. SCIDDR Bit Descriptions

Name	Reset	Description	Settings
— 0–21	0	Reserved. Write to zero for future compatibility.	
DDRTX 22	0	**Data Direction Bit TX** Controls the TX signal direction in single-wire mode (refer to **Section 21.4.2**).	1 If TE=1, TX is driven by the transmitter. Otherwise, if TE=0, UTXD is driven by logic 0. 0 UTXD is not driven when the transmitter is disabled (TE=0) or when LOOPS=1.
— 23–31	0	Reserved. Write to zero for future compatibility.	
Note: The setting descriptions assume that the UTXD signal is configured for UART operation.			

Timers 22

The MSC8113 device contains 32 timers of 16 bits each that serve as frequency dividers, watchdog timers, clock generators, and event counters. Each timer receives input from one of 15 sources: six external input signals, eight timer outputs, or the local bus clock (BUSES_CLOCK).

The timers are divided into two groups: Timers Module A and Timers Module B. For Timers Module A, each timer receives an input clock from TIMER0, TIMER1, TDM0RCLK, TDM1RCLK, TDM0TCL and TDM1TCLK. Each timer can also receive input from Timers A[8–15], thus supporting various structures such as using one timer as a prescaler of another timer. TIMER0 and TIMER1 can also be driven by Timer A0 and Timer A4, respectively. For Timers Module B, each timer receives input from TIMER2, TIMER3, TDM2RCLK, TDM3RCLK, TDM2TCLK, and TDM3TCLK. Each timer also receives input from Timers B8 through B15, thus supporting the same structures as Timers Module A. TIMER2 and TIMER3 are also driven by Timer B0 and Timer B4, respectively (see **Figure 22-1**). Each timer generates interrupts. Timer A6 in the Timers Module A connects to the SIU and serves as the internal SIU timer. For details, see **Chapter 4**, *System Interface Unit (SIU)*.

Two timers in each timers module can drive two outputs (see **Figure 22-1**). When the timers serve as frequency dividers, the output can be configured in one of two ways:

- Pulse. The output frequency is: out = in/(compare register value)
- Toggle. The output frequency is: out = in/(compare register value × 2)

The timer modules are accessible and configured through the IPBus (see **Chapter 19**, *Internal Peripheral Bus (IPBus)*). Each timer that reaches the compare value (TCMP reg) can generate one interrupt signal that connects to one of the SC140 core LICs, so this SC140 core services the timer interrupts. When accepting an interrupt request, an SC140 core should read the Timer Event Register (TER) to identify the interrupt source and service it accordingly. Afterwards, the SC140 core should clear the flag in the TER by writing a value of 1 to the associated flag. Each SC140 core LIC connects to eight timers (see **Chapter 17**, *Interrupt Processing*). **Table 22-1** and **Table 22-2** show the timer frequency ranges.

Concatenating more than two timers is not allowed. For example, if the output of Timer A8 is configured as an input of Timer A9, then the output of Timer A9 cannot be used as an input to another timer. Also, connecting the output of a timer to its input is not allowed. However, the output of Timer A8 can be configured as an input to multiple timers.

Table 22-1. Input/Output Frequency Range

Timer Combination	Input		Output	
	Minimum	Maximum	Minimum	Maximum
One timer for the bus clock	0	Bus Clock	0	Bus Clock/2
One timer for the external clock	0	Bus Clock	0	(External Clock)/2
Concatenation of two timers for the bus clock	0	Bus Clock	0	Bus Clock/4
Concatenation of two timers for the external clock	0	Bus Clock	0	(External clock)/4

Table 22-2. Output Frequency Range as a Function of the Input Frequency

Timer Combination	Output			
	Minimum		Maximum	
One timer	Input / 0x10000	TCMP reg = 0xFFFF (toggle mode)	Input/2	TCMP reg = 0x0002(pulse mode)
Concatenation of two timers	Input / 0x100000000	TCMP reg = 0xFFFF (for both timers)	Input/4	TCMP reg = 0x0002 (for both timers)

Note: If a timer is not active, do not configure the associated GPIO to select that timer.

Note: When two timers are concatenated, the slave timer can be configured as a one-shot timer only if it is not already configured as a slave in concatenation mode. If the slave timer is configured in cyclic mode after it is used, the first interrupt may occur at the wrong time. All subsequent interrupts occur at the correct time. If timer 0 and timer 4 are used and the output is configured to the GPIO, the first output signal may also occur at the wrong time, but subsequent outputs occur at the correct time.

Note: Before you reenable a timer, set the compare value to a relatively high number and allow the counter to advance for at least three input clocks to flush the cache. Then reenable the timer.

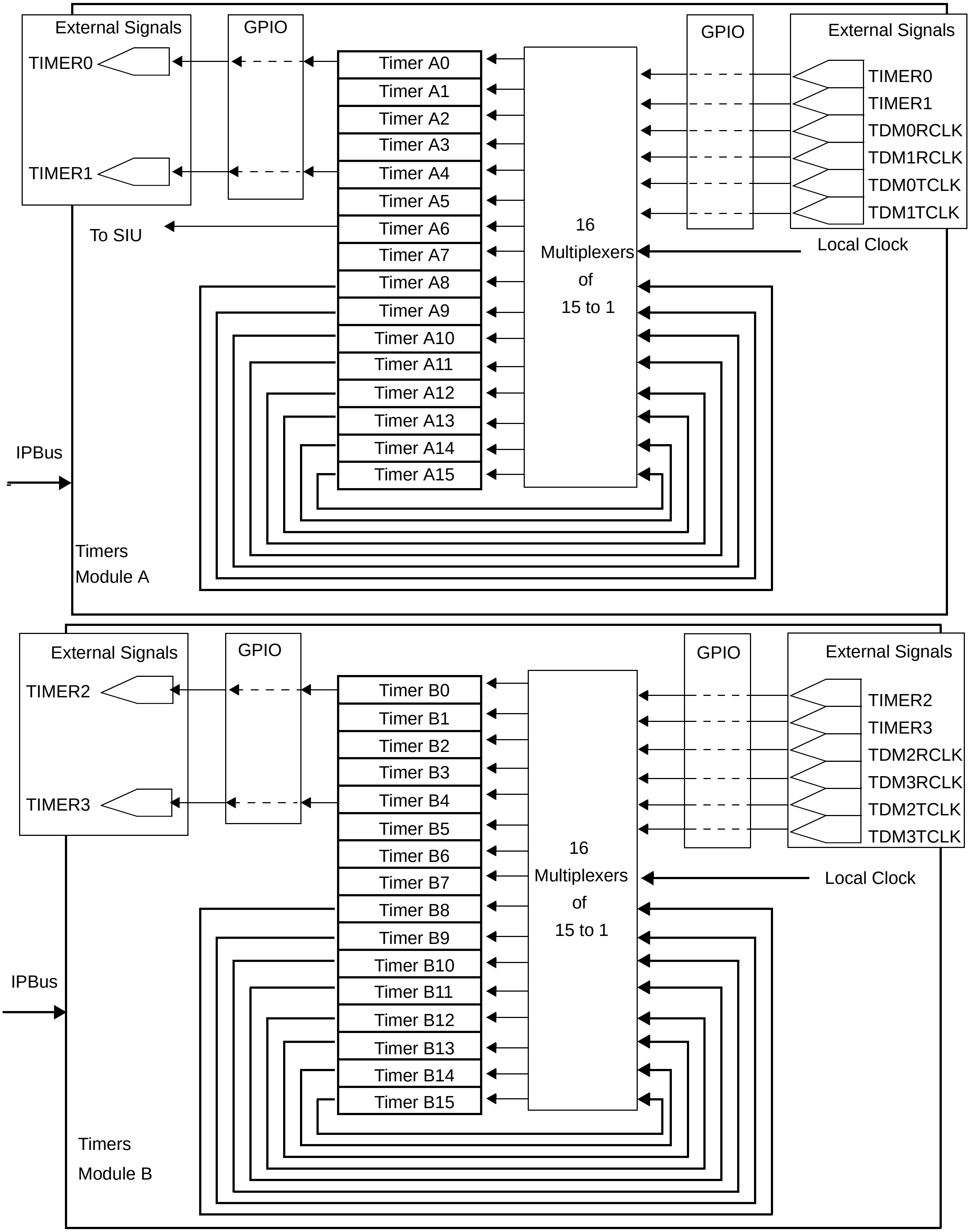


Figure 22-1. MSC8113 Timers

Figure 22-2 illustrates the structure of a timer in Timers Module A and Timers Module B. The timer is accessed through the IPBus, which is accessed through the local bus or the SQBus.

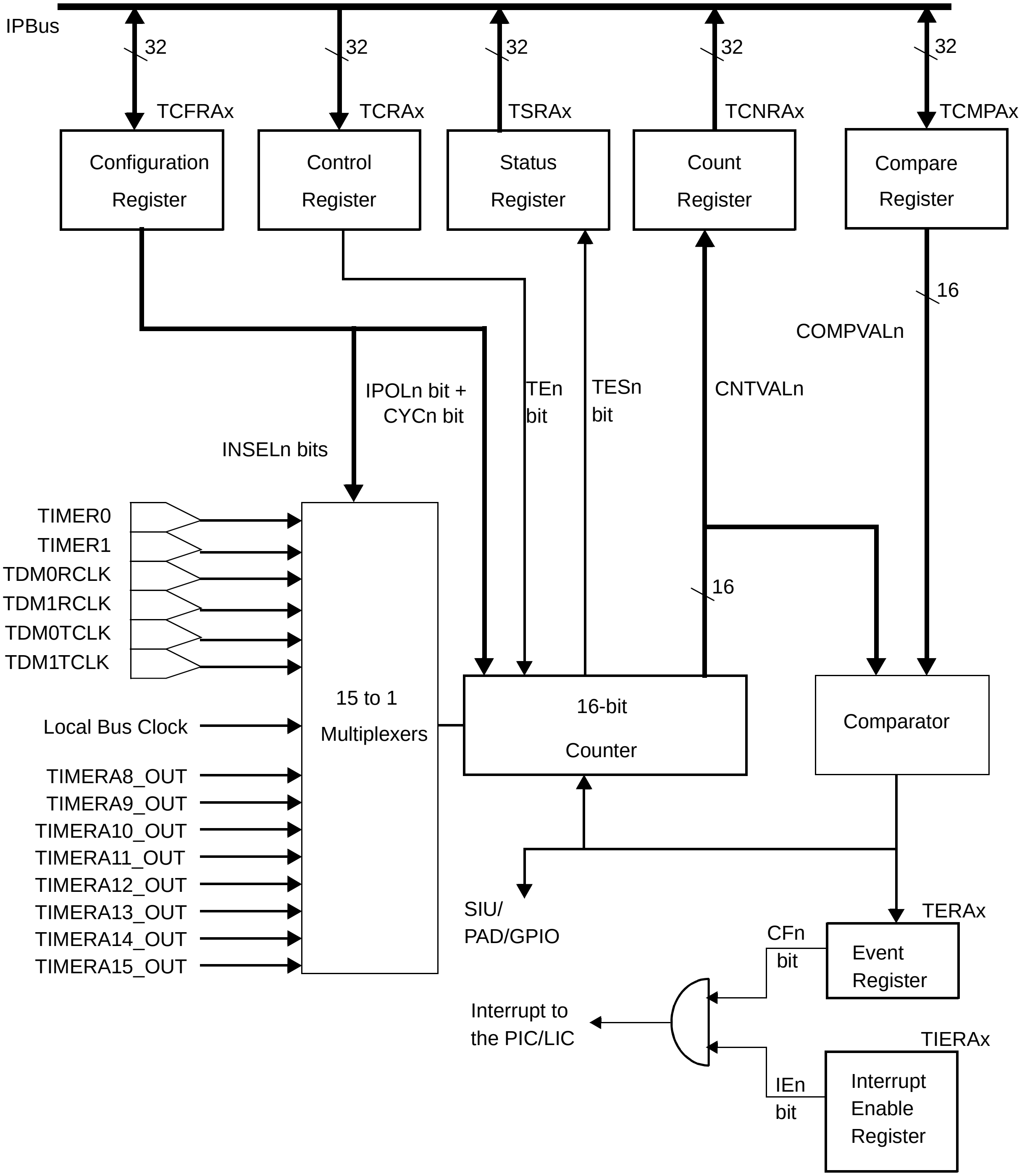


Figure 22-2. Timer *x* Module Block Diagram

When a timer is enabled, there is a delay of not more than eight timer clock cycles from when the TE bit is set in the Timer Control Register (TCRA*x*/TCRB*x*) until the timer counter starts counting and the appropriate TES bit in the Timer Status Register (TSRA/TSRB) is set. There is also a delay when TE = 1 is written to clear the timer counter while the timer is operating. The timer counter continues to count for not more than four timer clock cycles and only then resets to 0. It stays on the value 0 for not more than four timer clock cycles and then resumes counting.

The timer counter gets the clock from one of fifteen different inputs. When the counter reaches the value in the compare register, the timer generates a pulse or level value (configured in the Configuration Register), sets the relevant bit in the Event Register, and generates an interrupt if it is enabled by the Interrupt Enable Register. The clockout from Timer A0, TimerA4, Timer B0, and Timer B4 also drives the PAD/GPIO, if these timers are configured as outputs in the General Configuration Register. The clockout from Timer A6 goes to the SIU. You can read the value of the counter (Count Register) and verify whether the counter is enabled or disabled via the status register. If the timer counter generates a level interrupt and you write a 0 to TCRA*x* to disable the timer, the interrupt does not clear unless a 1 is written to the relevant bit in the TER. When timer *n* reaches the compare (TCMP) value, the TER[CFn] bit is set. The CFn bit is cleared only when a 1 is written to the associated TER[CFn] bit. The CFn bit is unaffected by Stop mode or timer restarts.

Note: The TDM can use each of the TDM0RCLK, TDM1RCLK, TDM2RCLK and TDM3RCLK signals as input clocks or data outputs. If the timer uses one of these clocks as an input clock, the TDM should configure the corresponding signal as an input clock via the GPIO, (see **Chapter 20**, *TDM Interface* and **Chapter 23**, *GPIO*).

To save power, the timer modules automatically shut down their clocks when they are not in use.When each of the 16 timers is disabled, its clock stops. Also, when all the timers are disabled and there is no access to the module, the main module clock stops, and the module goes into Stop mode. Each timer module has a status bit in the Stop Acknowledge Status Register (SASR) (see **Section 18.4**). The clocks automatically restart when they are required.

Each of the three MSC8113 SC140 cores can configure the timer modules. The configuration route is from the QBus via the SQBus to the IPBus to the timer modules. The arbitration of the SQBus occurs on the QBus. The external host and the DSI can also configure the timer modules. The external host configures the timers via the system bus to the local bus to the IPM via the IPBus to the timer modules.The DSI configures the timers via the local bus to the IPM via the IPBus to the timer modules.

Table 22-3 lists the timer actions and the registers used to perform them (per one timer).

Table 22-3. Timer Actions (Continued)

Actions	Register	Page
Programmer Actions Before the Timer is Enabled:		
Verify that the enable status bit is clear.	Timer Status Register A (TSRA) TESx = 0	**page 22-17**
Verify whether there are any level interrupts. If there are, check them or clear them by writing 1 to the relevant bit in the TER	Timer Event Register A (TERA) CFx = 0	**page 22-18**
Configure the timer input clock to one of 15 sources. If the source is one of TDM*_CLK, check the configuration of the signal by the TDM via the GPIO (see **Chapter 20**, *TDM Interface*)	Timer Configuration Register A (TCFRA) INSEL field	**page 22-12**
Configure the timer to count on the rising or falling edge of the selected source.	Timer Configuration Register A (TCFRA) IPOL bit: 0 Rising edge. 1 Falling edge.	**page 22-12**
Program a timer to operate in cyclic or one-shot mode When the counter reaches the compare register value, it wraps to 0 and continues counting if it is cyclic or it stays at the high value if it is a one-shot timer.	Timer Configuration Register A (TCFRA) CYC bit: 1 Cyclic mode. 0 One-shot mode.	**page 22-12**
If the selected source is TIMER0 or TIMER1, configure it as an input (its reset value). If the timerA0 output connects to TIMER0 or the timer A4 output connects toTIMER1, configure the signal lines as outputs through the GPIO registers.	Timer General Configuration Register A (TGCRA) DIR0 or DIR4 bit: 0 Input. 1 Output.	**page 22-9**
Designate the signal as a pulse so that it is asserted for one clock cycle (the input clock of the timer), or the signal value is toggled for every compare signal.	Timer General Configuration Register A (TGCRA) TOG0 or TOG4 bit: 0 Pulse. 1 Toggle.	**page 22-9**
In pulse mode, specify the pulse polarity as not changed or changed.	Timer General Configuration Register A (TGCRA) POL0 or POL4 bit: 0 Polarity is not changed. 1 Polarity is changed.	**page 22-9**
Designate the Interrupt as a pulse so that it is asserted for one clock cycle, or as a level.	Timer General Configuration Register A (TGCRA) INTP bit: 0 The Interrupt is a pulse. 1 The Interrupt is a level.	**page 22-9**
Configure which interrupt to mask.	Timer Interrupt Mask Register A (TIERA) IEx = 1.	**page 22-9**
Set the compare value.	Timer Compare Register A (TCMPA) COMPVAL field.	**page 22-15**
Enable a timer. If two timers are concatenated, first enable the prescaler and then enable the timers. If a timer is configured to receive its input clock from another signal, the signal must be driven before the timer is enabled.	Timer Control Register A (TCRA) TE field: 0 Timer disable. 1 Timer enable.	**page 22-16**

Table 22-3. Timer Actions (Continued)

Actions	Register	Page
Timer Actions:		
Once enabled, the timer starts counting from 0 to the value specified in the timer compare register. The first two cycles of the counter clock (can be checked by polling the TSR*x*) are lost after the bit is set to 1.	Timer Compare Register A (TCMPA) COMPVAL field.	**page 22-15**
When it reaches the value in the compare register, the timer sets a bit in the event register.	Timer Event Register A (TERA) CFx = 1.	**page 22-18**
An interrupt is generated.	Timer Interrupt Mask Register A (TIERA) IEx = 1.	**page 22-11**
Programmer Actions While Running:		
Restart the counter at any time by writing a 1 to the enable bit in the TCRA*x* (see the note on **page 22-16**).	Timer Control Register A (TCRA).	**page 22-16**
Read the value of the timer internal counter at any time.	Timer Count Register A (TCNRA) CNTVAL field.	**page 22-19**
Disable a timer by clearing the TE bit in the TCRA*x* register. To reconfigure the timer, write new value to the configuration registers and set the TE bit. The timer counter resets and starts counting from 0 (see the note on **page 22-16**).	Timer Control Register A (TCRA) TE bit = 0.	**page 22-16**

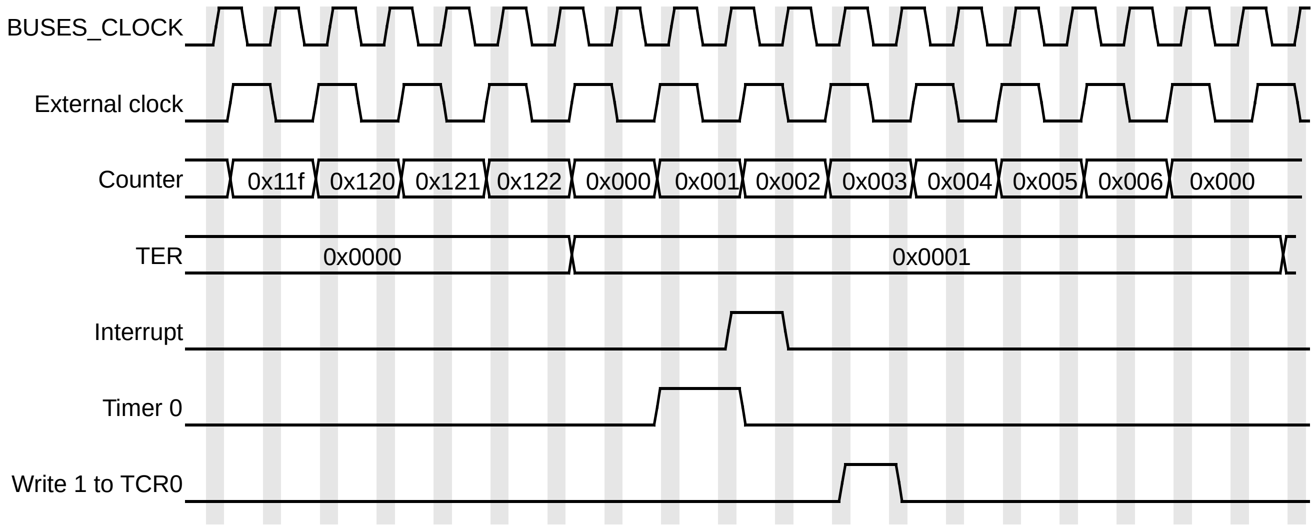


Figure 22-3. Counting Example

Figure 22-3 illustrates an example of simple counting using one timer in which the timer counter receives its clock from an external clock. It also illustrates how to restart the timer by writing 1 to the TCR. The configuration registers are:

- TGCRA[INTP] = 0, TGCRA[TOG] = 1
- TCFRA0[CYC] = 1, TCFRA0[IPOL] = 0
- TIER[IE0] = 1
- TCMPA0[COMPVAL] = 0x0123

22.1 Timers Programming Model

All timer registers are mapped into the IPBus address space. Refer to **Chapter 8**, *Memory Map*. The timer registers are divided into configuration, control, and status registers as follows:

- *Configuration registers*. Configure the timer input and output forms, the timer operation mode, and the compare values. Program each register before the timer is enabled and do not change the values while the timer is enabled. There are both global configuration registers and configuration registers for individual timers. Writing to configuration registers is not allowed when the timers are enabled.
- *Control registers*. Enable the timers and the interrupts. These registers can be changed during timer operation. There are both global control registers and control registers for individual timers.
- *Status registers*. Can be accessed at any time.

Note: The global configuration and control registers configure all 16 timers in a module, including the registers that define the I/O.

This section describes the timer module registers, which are listed as follows:

- Timer General Configuration Register A (TGCRA), **page 22-9**
- Timer General Configuration Register B (TGCRB), **page 22-10**
- Timer Interrupt Enable Register A (TIERA), **page 22-11**
- Timer Interrupt Enable Register B (TIERB), **page 22-11**
- Timer Configuration Register A (TCFRA[0–15]), **page 22-12**
- Timer Configuration Register B (TCFRB[0–15]), **page 22-14**
- Timer Compare Register A (TCMPA[0–15]), **page 22-15**
- Timer Compare Register B (TCMPB[0–15]), **page 22-15**
- Timer Control Register A (TCRA[0–15]), **page 22-16**
- Timer Control Register B (TCRB[0–15]), **page 22-16**
- Timer Status Register A (TSRA), **page 22-17**
- Timer Status Register B(TSRB), **page 22-17**
- Timer Event Register A (TERA), **page 22-18**
- Timer Event Register B (TERB), **page 22-18**
- Timer Count Register A (TCNRA[0–15]), **page 22-19**
- Timer Count Register B (TCNRB[0–15]), **page 22-20**

22.1.1 Configuration Registers

TGCRA Timer General Configuration Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
					----				POL4	TOG4	DIR4	—	INTP	POL0	TOG0	DIR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-4. TGCRA Bit Descriptions

Name	Reset	Description	Settings
— 0–23	0	Reserved. Write to zero for future compatibility.	
POL4 24	0	**Polarity** Defines the polarity of the TIMER1 pin, which is driven by Timer A4, when TIMER1 is an output. POL4 is typically used in pulse mode (TOG4) = 0.	0 The output signal polarity is not changed. 1 The output signal is inverted.
TOG4 25	0	**Pulse/Toggle** Defines whether TIMER1 is toggled or pulsed when the timer counter of Timer A1 reaches the value of the associated COMPVAL field. This bit is valid only when (DIR4 = 1).	0 The signal is asserted for one clock. 1 The signal is toggled.
DIR4 26	0	**Signal Direction** Defines the direction of TIMER1.	0 TIMER1 is an input. 1 TIMER1 is an output.
— 27	0	Reserved. Write to zero for future compatibility.	
INTP 28	0	Interrupt Pulse/Level Defines whether the Interrupts of Timers Module A are pulse or level. If the timer is disabled, the interrupt may remain asserted (even if it in pulse mode) until the timer is enabled again or write a value of ‘1’ to the associated flag in the TER.	0 Interrupts are asserted for one clock (pulse). 1 Interrupts are level form.
POL0 29	0	**Polarity** Defines the polarity of TIMER0, which is driven by Timer A0, when TIMER0 is an output. This bit is typically used in pulse mode (TOG0 = 0).	0 The output signal polarity is not changed. 1 The output signal is inverted.
TOG0 30	0	**Pulse/Toggle** Defines whether TIMER0 is toggled or pulsed when the timer counter of Timer A0 reaches the value of the associated COMPVAL field. This bit is valid only when (DIR0 = 1).	0 The signal is asserted for one clock. 1 The signal is toggled.
DIR0 31	0	**Signal Direction** Defines the direction of the TIMER0.	0 TIMER0 is an input. 1 TIMER0 is an output.

TGCRB Timer General Configuration Register B

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	—								POL4	TOG4	—		INTP	POL0	TOG0	—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Timer General Configuration Register B (TGCRB) does not contain direction (DIR) bits because the TIMER2 and TIMER3 are GPIO signals. See **Chapter 23**, *GPIO*.

Table 22-5. TGCRB Bit Definitions

Name	Reset	Description	Settings
— 0–23	0	Reserved. Write to zero for future compatibility.	
POL4 24	0	**Polarity** Defines the polarity of TIMER3, which is driven by Timer B4, when TIMER3 is an output. This bit is typically used in pulse mode (TOG4 = 0).	0 The output signal polarity is not changed. 1 The output signal is inverted.
TOG4 25	0	**Pulse/Toggle** Defines whether TIMER3 is toggled or pulsed when the timer counter of Timer B4 reaches the value of the associated COMPVAL field. This bit is valid only when then TIMER3 is configured as output.	0 The signal is asserted for one clock. 1 The signal is toggled.
— 26–27	0	Reserved. Write to zero for future compatibility.	
INTP 28	0	Interrupt Pulse/Level Defines whether the interrupts of Timer Module B are pulse or level. If the timer is disabled, the interrupt may remain asserted (even if it in pulse mode) until the timer is enabled again or write a value of ‘1’ to the associated flag in the TER.	0 Interrupts are asserted for one clock (pulse). 1 Interrupts are level form.
POL0 29	0	**Polarity** Defines the polarity of TIMER2, which is driven by Timer B0 when TIMER2 is an output. This bit is typically used in pulse mode (TOG0 = 0).	0 The output signal polarity is not changed. 1 The output signal is inverted.
TOG0 30	0	**Pulse/Toggle** Defines whether TIMER2 is toggled or pulsed when the timer counter of Timer B0 reaches the value of the associated COMPVAL field. This bit is valid only when then TIMER2 is configured as output.	0 The signal is asserted for one clock. 1 The signal is toggled.
— 31	0	Reserved. Write to zero for future compatibility.	

TIERA Timer Interrupt Enable Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The MSC8113 device generates an interrupt when both the TERA bit and its corresponding enable bit are set.

Table 22-6. TIERA Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
IE[15–0] 16–31	0	**Timer A[15–0] Interrupt Enable** When IEx and the corresponding TERA[CFx] are both set, an interrupt is generated.	0 Interrupt disabled. 1 Interrupt enabled.

TIERB Timer Interrupt Enable Register B

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The MSC8113 device generates an interrupt when both the TERB bit and its corresponding enable bit are set.

Table 22-7. TIERB Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
IE[15–0] 16–31	0	**Timer B[15–0] Interrupt Enable** When IEx and the corresponding TERB[CFx] are both set, an interrupt is generated.	0 Interrupt disabled. 1 Interrupt enabled.

TCFRA[0–15] Timer Configuration Register A[0–15]

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="9">—</td><td colspan="4">INSEL</td><td>—</td><td>IPOL</td><td>CYC</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Table 22-8. TCFRA[0–15] Bit Descriptions

Name	Reset	Description	Settings
— 0–24	0	Reserved. Write to zero for future compatibility.	
INSEL 25–28	6	**Input Select** Defines the source of the timer A*x* clock.	0000 TIMER0. 0001 TIMER1. 0010 TDM0RCLK. 0011 TDM1RCLK. 0100 TDM0TCLK. 0101 TDM1TCLK. 0110 Internal Local BUSES_CLOCK. 0111 Reserved. 1000 Timer A8 output. 1001 Timer A9 output. 1010 Timer A10 output. 1011 Timer A11 output. 1100 Timer A12 output. 1101 Timer A13 output. 1110 Timer A14 output. 1111 Timer A15 output.
— 29	0	Reserved. Write to zero for future compatibility.	
IPOL 30	0	**Input clock polarity** Defines the polarity of the input clock for Timer A*x*. IPOL has no effect if the bus clock is selected.	0 Counter changes at the clock rising edge. 1 Counter changes at the clock falling edge.

Table 22-8. TCFRA[0–15] Bit Descriptions (Continued)

Name	Reset	Description	Settings
CYC 31	0	**Cyclic/One-Shot** Defines whether the Timer A*x* mode of operation is cyclic or one-shot. In One-Shot mode, the counter of Timer-n counts up until it equals the TCMPA*x*[COMPVAL] field and then stops counting. **Note:** There are only two ways to use a timer properly in one-shot mode: **1.** Set the one-shot timer input clock to be the bus clock. **2.** The one-shot timer input clock must be higher than the bus clock frequency divided by 4. In Cyclic mode, the counter of Timer-n counts from 0 until TCMPA[COMPVAL], wraps back to 0 and continues counting.	0 One-Shot mode. 1 Cyclic mode.

Notes: **1.** IPOL in TCFRA*x* has no effect if the bus clock is used. The counter counts on the positive edge.

2. If an external clock is selected, you configure the GPIO registers. If an external input clock, TDM0RCLK or TDM1RCLK, is selected as an input clock, the corresponding signal line should be configured by the TDM registers as an input clock via the GPIO, in addition to configuring the GPIO registers (see **Chapter 20**, *TDM Interface* and **Chapter 23**, *GPIO*).

TCFRB[0–15] Timer Configuration Register B[0–15]

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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
	—									INSEL				—	IPOL	CYC
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Table 22-9. TCFRB[0–15] Bit Descriptions

Name	Reset	Description	Settings
— 0–24	0	Reserved. Write to zero for future compatibility.	
INSEL 25–28	6	**Input Select** Defines the source of Timer B*x* clock.	0000 TIMER2. 0001 TIMER3. 0010 TDM2RCLK. 0011 TDM3RCLK. 0100 TDM2TCLK. 0101 TDM3TCLK. 0110 Internal local BUSES_CLOCK. 0111 Reserved. 1000 Timer B8 output. 1001 Timer B9 output. 1010 Timer B10 output. 1011 Timer B11 output. 1100 Timer B12 output. 1101 Timer B13 output. 1110 Timer B14 output. 1111 Timer B15 output.
— 29	0	Reserved. Write to zero for future compatibility.	
IPOL 30	0	**Input Clock Polarity** Defines the polarity of the input clock for Timer B*x*. IPOL has no effect if the bus clock is selected.	0 Counter changes at the clock rising edge. 1 Counter changes at the clock falling edge.
CYC 31	0	**Cyclic/One-Shot** Defines whether the Timer B*x* operating mode is cyclic or one-shot. In Cyclic mode, the timer counter counts from 0 to TCMPB[COMPVAL], wraps back to 0, and continues counting. In One-Shot mode, the timer counter counts up until it equals the TCMPB*x*[COMPVAL] field and then stops counting.	0 One-Shot mode. 1 Cyclic mode.
Notes:	**1.**	IPOL in TCFRB*x* has no effect if the bus clock is used. The counter counts on the positive edge.	
	2.	For an external clock, configure the GPIO registers. For TDM0RCLK or TDM1RCLK selected as an input clock, configure the corresponding signal line through the TDM registers as an input clock via the GPIO, in addition to configuring the GPIO registers (see **Chapter 20**, *TDM Interface* and **Chapter 23**, *GPIO*)	

TCMPA[0–15] Timer Compare Register A[0–15]

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

Table 22-10. TCMPA[0–15] Bit Descriptions

Name	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
COMPVAL 16–31	0	**Compare Value** Contains the value that is compared to the counter value. When the counter value of Timer A*x* is equal to this field in the associated TCMPA*x* register, the comparator output is asserted, the CF bit in the associated TERA is set, and an interrupt is generated, if enabled. In Cyclic mode (TCFRA*x*[CYC] = 1), the counter is cleared and the counting continues. In One-Shot mode (TCFRA*x*[CYC] = 0), the counter is frozen and its output remains asserted until it is cleared.

TCMPB[0–15] Timer Compare Register B[0–15]

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

Table 22-11. TCMPB[0–15] Bit Descriptions

Name	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
COMPVAL 16–31	0	**Compare Value** Contains the value that is compared to the counter value. When the counter value and the value in this field are equal the comparator output is asserted, the associated TERB:CF bit is set, and an interrupt is generated, if enabled. In Cyclic mode (TCFRB*x*[CYC] = 1), the counter is cleared and the counting continues. In One-Shot mode (TCFRB*x*[CYC] = 0), the counter is frozen and its output remains asserted until it is cleared.

22.1.2 Control Registers

TCRA[0–15] Timer Control Register A[0–15]

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	—															TE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-12. TCRA[0–15] Bit Descriptions

Name	Reset	Description	Settings
— 0–30	0	Reserved. Write to zero for future compatibility.	
TE 31	0	**Timer Enable** Enables the timer and clears the timer counter. Writing a value of “1” to this bit during timer operation resets the timer counter. This reset operation is useful mostly for watchdog mode. **Note:** Writing a value of 1 to the enable bit causes a one-shot configured timer to count again, regardless of whether its value is 0 or 1.	0 The timer is disabled. 1 The timer is enabled.

TCRB[0–15] Timer Control Register B[0–15]

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	—															TE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-13. TCRB[0–15] Bit Descriptions

Name	Reset	Description	Settings
— 0–30	0	Reserved. Write to zero for future compatibility.	
TE 31	0	Timer Enable Enables the timer and clears the timer counter. Writing a value of “1” to this bit during timer operation resets the timer counter. This reset operation is useful mostly for watchdog mode. **Note:** Writing a value of 1 to the enable bit causes a one-shot configured timer to count again, regardless of whether its value is 0 or 1.	0 The timer is disabled. 1 The timer is enabled.

22.1.3 Status Registers

The MSC8113 device has two groups of status registers:

- *Global registers*, which hold the compare flags and the Status enable bits. The MSC8113 sets the status flag when an event occurs, and it is cleared by writing a 1 to the associated location.
- *Per Timer registers*, which hold the values of each timer.

TSRA Timer Status Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	TES15	TES14	TES13	TES12	TES11	TES10	TES9	TES8	TES7	TES6	TES5	TES4	TES3	TES2	TES1	TES0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-14. TSRA Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
TES[15–0] 16–31	0	**Timer A[15–0] Enable Status** Real status enable of Timer A[15–0].	0 Timer status is disabled. 1 Timer status is enabled.

TSRB Timer Status Register B

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	TES15	TES14	TES13	TES12	TES11	TES10	TES9	TES8	TES7	TES6	TES5	TES4	TES3	TES2	TES1	TES0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-15. TSRB Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
TES[15–0] 16–31	0	**Timer B[15–0] Enable Status** Real status enable of Timer B[15–0].	0 Timer status is disabled. 1 Timer status is enabled.

TERA Timer Event Register A

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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**
	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CFn bit is set when timer *n* reaches the TCMP value. To clear CFn, write a 1 to the associated CFn flag, which also clears the associated interrupt. If timer *n* reaches the TCMPA*x* value while a 1 is written to the associated CFn flag, the associated CFn flag stays set and is not cleared.

Table 22-16. TERA Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
CF[15–0] 16–31	0	**Timer Compare Flag A[15–0]** Set when Timer A[15–0] reaches the COMPVAL value.	0 The timer has not yet reached the COMPVAL value. 1 The timer has reached the COMPVAL value.

TERB Timer Event Register B

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
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**
	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CFn bit is set when timer *n* reaches the TCMP value. To clear CFn, write a 1 to the associated CFn flag, which also clears the associated interrupt. If timer *n* reaches the TCMPA*x* value while a 1 is written to the associated CFn flag, the associated CFn flag stays set and is not cleared.

Table 22-17. TERB Bit Descriptions

Name	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
CF[15–0] 16–31	0	**Timer Compare Flag B[15–0]** Set when Timer B[15–0] reaches the COMPVALvalue.	0 The timer has not yet reached the COMPVAL value. 1 The timer has reached the COMPVALvalue.

TCNRA[0–15] Timer Count Register A[0–15]

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	CNTVAL															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-18. TCNRA[0–15] Bit Descriptions

Name	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
CNTVAL 16–31	0	**Counter Value** Contains the counter value of Timer A*x* (from the Timers Module A).
Notes:	**1.**	When two timers are concatenated, you can read only one counter each time. Therefore, the value that is read cannot be synchronized from the two counters, as if there was a counter of 32 bits, unless the two counters reach their TCMPA*x* values and stop.
	2.	When the counter value is read, the maximum value can be TCMPA*x* – 1.

TCNRB[0–15] Timer Count Register B[0–15]

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">CNTVAL</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Table 22-19. TCNRB[0–15] Bit Descriptions

<table>
<tr><th>Name</th><th>Reset</th><th>Description</th></tr>
<tr><td>—
0–15</td><td>0</td><td>Reserved. Write to zero for future compatibility.</td></tr>
<tr><td><b>CNTVAL</b>
16–31</td><td>0</td><td><b>Counter Value</b>
Contains the counter value of Timer B<i>x</i> (from the Timers Module B).</td></tr>
<tr><td colspan="3"><b>Notes:</b> <b>1.</b> When two timers are concatenated, you can read only one counter each time. Therefore, the value that is read cannot be synchronized from the two counters, as if there were a counter of 32 bits, unless the two counters reach their TCMPB<i>x</i> values and stop.
<b>2.</b> When the counter value is read, the maximum value can be TCMPB<i>x</i> – 1.</td></tr>
</table>

GPIO 23

The MSC8113 device has 32 general-purpose I/O (GPIO) ports that are multiplexed as either GPIO ports or dedicated peripheral interface ports. In addition, fifteen of the GPIOs can be configured to serve as Ethernet interface ports, and fifteen of the GPIOs can be configured to generate interrupts to the Global Interrupt Controller (GIC). As GPIOs, each port is configured as an input or output (with a register for data output that is read or written at any time). If configured as output, the GPIO ports can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, an output drives a zero voltage but goes to tri-state when driving a high voltage. GPIO ports do not have internal pull-up resistors. The dedicated MSC8113 peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8113 applications.

Note: To understand the port assignment capability described in this chapter, you must first understand the Ethernet, Time-Division Multiplexing (TDM), timers, and UART peripherals.

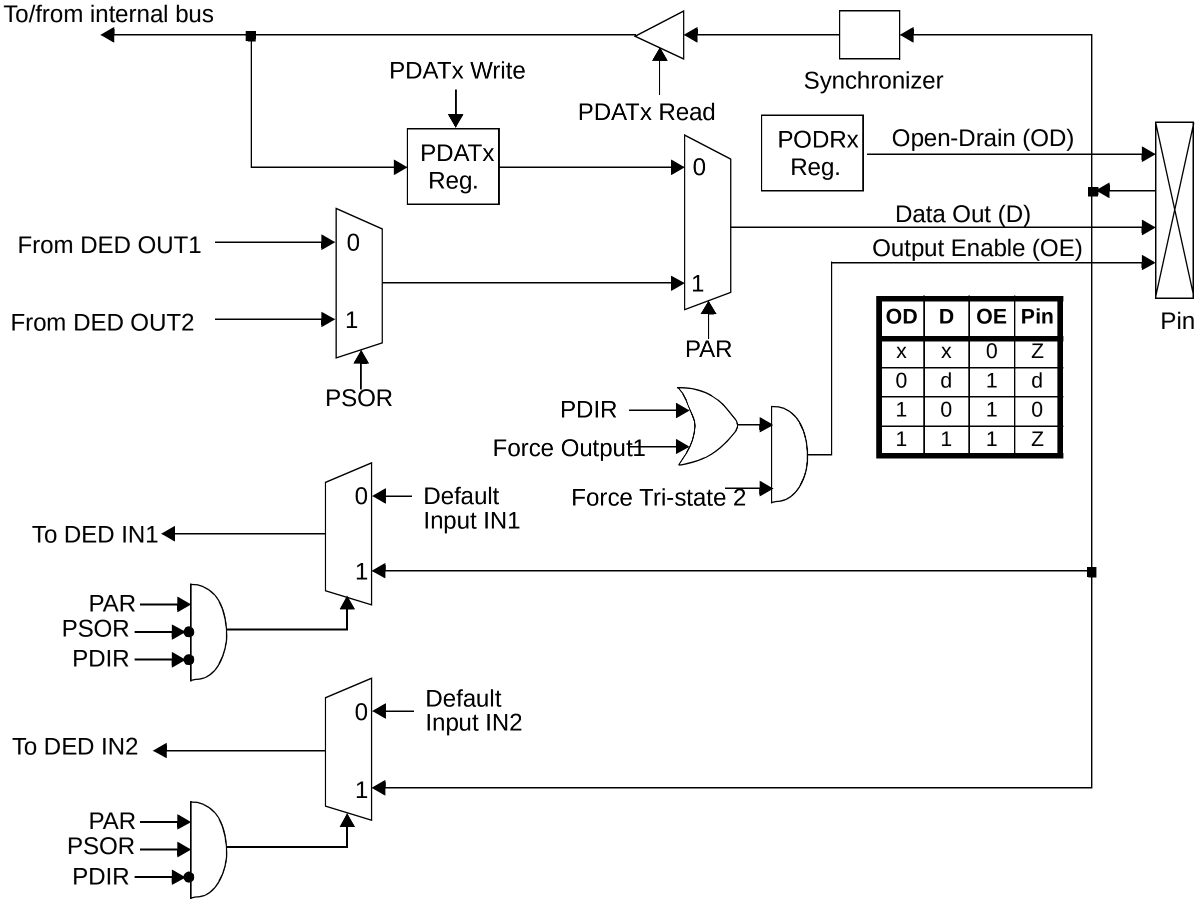
23.1 Features

Following are the key features of the GPIO ports:

- 32 GPIO ports.
- All ports are bidirectional.
- All ports have alternate on-device peripheral functions.
- All ports are set as GPIO inputs at system reset.
- All port values can be read while the port is connected to an internal peripheral.
- All ports have open-drain output capability.
- Fifteen of the GPIOs can be used as interrupt inputs.
- Fifteen of the GPIOs can be used as Ethernet interface ports

23.2 GPIO Block Diagram

Figure 23-2 shows a GPIO functional block diagram.



OD	D	OE	Pin
x	x	0	Z
0	d	1	d
1	0	1	0
1	1	1	Z

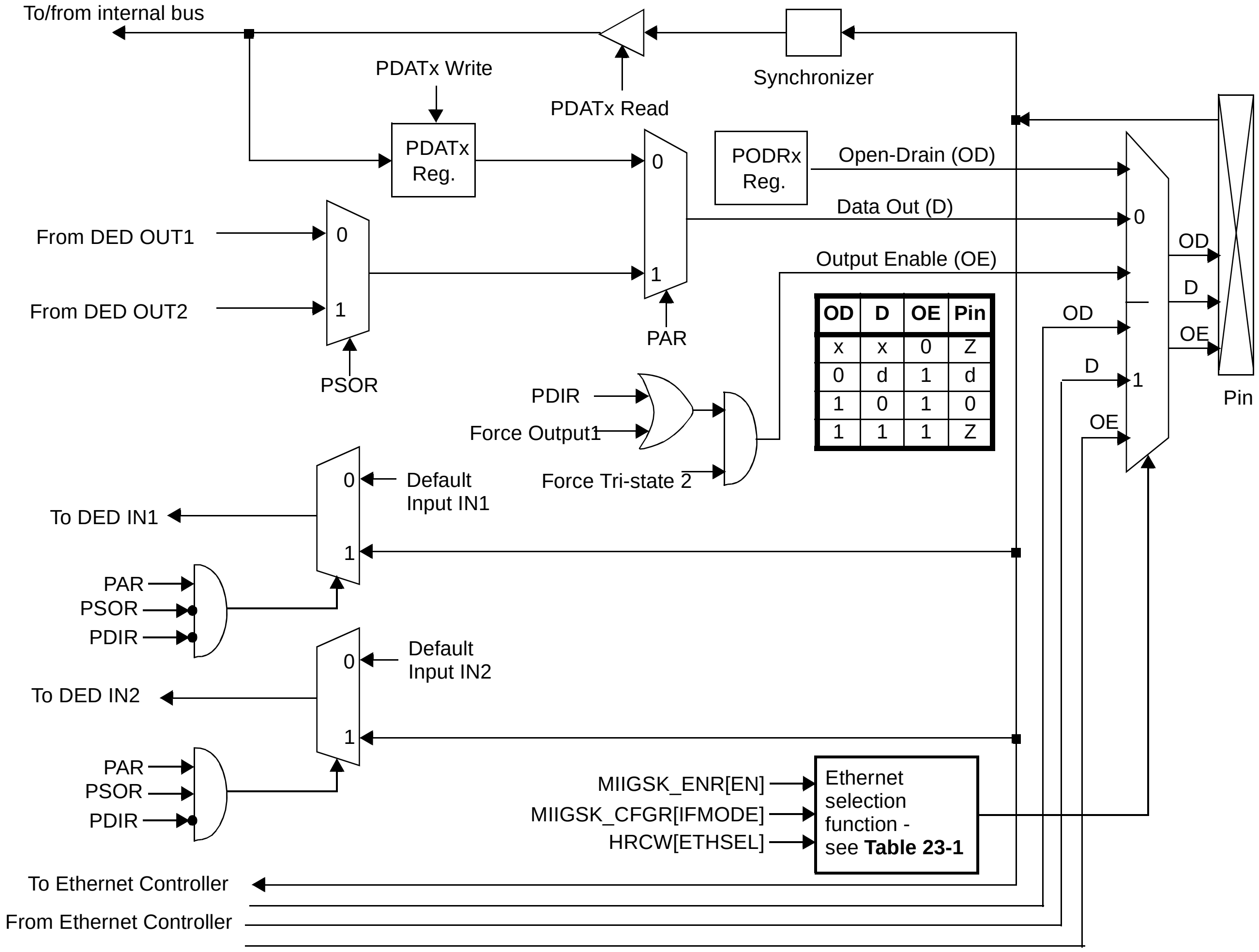
Notes: 1. Force Output may be asserted high by dedicated peripheral 2 direction control, only when PAR = 1 and PSOR = 1 (selects this peripheral) and PDIR = 0 (input). It is used for bidirectional operation allowing the peripheral to dynamically control the port direction.

2. Force Tri-state may be asserted low by dedicated peripheral 1 output enable, only when PAR = 1 and PSOR = 0 (selects this peripheral) and PDIR = 1 (output). It is used for tri-state output operation, allowing the peripheral to control the port drive dynamically.

Port Control Register Bits

Register Name	0	1	Description
PARx	General-purpose	Dedicated	Port Assignment Registers
PSORx	Dedicated 1	Dedicated 2	Port Special Options Registers
PDIRx	Input	Output	Port Data Direction Registers
PODRx	Regular	Open drain	Port Open-Drain Registers
PDATx	0 (data)	1 (data)	Port Data Registers

Figure 23-1. Port Functional Operation



OD	D	OE	Pin
x	x	0	Z
0	d	1	d
1	0	1	0
1	1	1	Z

Notes: 1. Force Output may be asserted high by dedicated peripheral 2 direction control, only when PAR = 1 and PSOR = 1 (selects this peripheral) and PDIR = 0 (input). It is used for bidirectional operation allowing the peripheral to dynamically control the port direction.

2. Force Tri-state may be asserted low by dedicated peripheral 1 output enable, only when PAR = 1 and PSOR = 0 (selects this peripheral) and PDIR = 1 (output). It is used for tri-state output operation, allowing the peripheral to control the port drive dynamically.

Port Control Register Bits

Register Name	0	1	Description
PARx	General-purpose	Dedicated	Port Assignment Registers
PSORx	Dedicated 1	Dedicated 2	Port Special Options Registers
PDIRx	Input	Output	Port Data Direction Registers
PODRx	Regular	Open drain	Port Open-Drain Registers
PDATx	0 (data)	1 (data)	Port Data Registers

Figure 23-2. Port Functional Operation

23.3 Ethernet Functionality of GPIO

Whether a port operates as a GPIO/dedicated input/output or Ethernet Controller input/output depends on the settings of HRCW[ETHSEL], as described in **Table 23-1**.

Table 23-1. GPIO/Dedicated Functionality Versus Ethernet Functionality

GPIO	Port Function			

	HRCW[ETHSEL] = 0 and MIIGSK_ENR[EN] = 1			HRCW[ETHSEL] = 1
				or
				MIIGSK_ENR[EN] = 0

	MIIGSK_CFGR[IFMODE] = 00 (MII mode)	MIIGSK_CFGR[IFMODE] = 01 (RMII mode)	MIIGSK_CFGR[IFMODE] = 10 (SMII mode)	
0	ETHTXD0	ETHTXD0	GPIO or Dedicated functionality according to PAR and PSOR	
1	ETHTXD1	ETHTXD1	GPIO or Dedicated functionality according to PAR and PSOR	
2	GPIO or Dedicated functionality according to PAR and PSOR			
3	ETHTXD2	GPIO or Dedicated functionality according to PAR and PSOR		
4	ETHTX_ER	GPIO or Dedicated functionality according to PAR and PSOR		
5	ETHRXD3	GPIO or Dedicated functionality according to PAR and PSOR		
6	ETHRXD2	GPIO or Dedicated functionality according to PAR and PSOR		
7	ETHTXD3	GPIO or Dedicated functionality according to PAR and PSOR		
8	ETHCOL	GPIO or Dedicated functionality according to PAR and PSOR		
9	ETHMDIO	ETHMDIO	ETHMDIO	GPIO or Dedicated functionality according to PAR and PSOR
10	ETHRX_DV	ETHCRS_DV	NC (leave unconnected)	GPIO or Dedicated functionality according to PAR and PSOR
11	ETHRX_ER	ETHRX_ER	ETHTXD	GPIO or Dedicated functionality according to PAR and PSOR
12	ETHRXD1	ETHRXD1	ETHSYNC	GPIO or Dedicated functionality according to PAR and PSOR
13	ETHMDC	ETHMDC	ETHMDC	GPIO or Dedicated functionality according to PAR and PSOR
14	ETHRXD0	ETHRXD0	NC (leave unconnected)	GPIO or Dedicated functionality according to PAR and PSOR
15	GPIO or Dedicated functionality according to PAR and PSOR			
16	GPIO or Dedicated functionality according to PAR and PSOR			
17	GPIO or Dedicated functionality according to PAR and PSOR			
18	GPIO or Dedicated functionality according to PAR and PSOR			

Table 23-1. GPIO/Dedicated Functionality Versus Ethernet Functionality (Continued)

GPIO	Port Function			

	HRCW[ETHSEL] = 0 and MIIGSK_ENR[EN] = 1			HRCW[ETHSEL] = 1
				or
				MIIGSK_ENR[EN] = 0

	MIIGSK_CFGR[IFMODE] = 00 (MII mode)	MIIGSK_CFGR[IFMODE] = 01 (RMII mode)	MIIGSK_CFGR[IFMODE] = 10 (SMII mode)	
19	GPIO or Dedicated functionality according to PAR and PSOR			
20	GPIO or Dedicated functionality according to PAR and PSOR			
21	GPIO or Dedicated functionality according to PAR and PSOR			
22	GPIO or Dedicated functionality according to PAR and PSOR			
23	GPIO or Dedicated functionality according to PAR and PSOR			
24	GPIO or Dedicated functionality according to PAR and PSOR			
25	GPIO or Dedicated functionality according to PAR and PSOR			
26	GPIO or Dedicated functionality according to PAR and PSOR			
27	GPIO or Dedicated functionality according to PAR and PSOR			
28	GPIO or Dedicated functionality according to PAR and PSOR			
29	TX_EN	TX_EN	GPIO or dedicated functionality according to PAR and PSOR	
30	GPIO or Dedicated functionality according to PAR and PSOR			
31	GPIO or Dedicated functionality according to PAR and PSOR			
The ETHMDIO is selected as MDO when the 10/100 fast Ethernet controller is transmitting management frames and as MDI when received frames are transmitted toward the Ethernet controller.				

When GPIO port has Ethernet functionality:

- Its direction and driving mode are controlled by the Ethernet controller regardless of PDIR*x* and PODR*x* values. See *Ethernet Controller*.
- Data written to PDAT*x* is stored in the output register, but it is prevented from reaching the external port.
- A read of PDAT*x* returns the data at the external port, independently of whether the port is defined as input or output in Ethernet controller.
- Default values are supplied to internal peripheral inputs connected to this GPIO port.

Note: PAR*x* of a GPIO port that has Ethernet functionality should be cleared (0).

23.4 GPIO Connection Functions

This section describes the GPIO port when it has GPIO or dedicated functionality, which depends on the settings in the Pin Assignment Register (PAR), as follows:

- Each port is independently configured as a GPIO if the corresponding PAR bit is cleared. A port is configured as an input if the corresponding control bit in the Pin Data Direction Register (PDIR) is cleared; it is configured as an output if the corresponding PDIR bit is set.
- Each port is configured as a dedicated on-device peripheral port if the corresponding PAR bit is set.

All PAR and PDIR bits are cleared on total system reset, configuring all ports as GPIO inputs.

Data transfer is done through the Pin Data Register (PDAT*x*). Data written to the PDAT*x* is stored in an output register. If a GPIO is configured as an output, the output register data is gated onto the GPIO port. If a GPIO is configured as an input, a read of PDAT*x* is actually a read of the GPIO port itself. Data written to PDAT*x* when the GPIO is configured as an input is still stored in the output register, but it is prevented from reaching the external port.

When a multiplexed GPIO port is not configured as a GPIO, it has a dedicated functionality, as described in **Table 23-2**. If an input to a peripheral is not supplied externally, a default value is supplied to the internal peripheral as listed in the right-most column.

Note: Some functions can be output on two different ports. You can freely configure such functions to be output on the two ports at once. However, there is typically no advantage in doing so unless there is a large fanout in which it is advantageous to share the load between two ports.

Table 23-2 describes the functionality of the GPIO ports according to the configuration of the port registers (PAR*x*, PSOR*x*, and PDIR*x*). Each port can be configured as a GPIO (input, regular output, or open-drain output), one of two dedicated outputs, or one of two dedicated inputs. A route of one GPIO-dedicated output to another GPIO-dedicated input gives even more flexibility. The implemented routing is described in **Table 23-2** as primary and secondary input.

Figure 23-3 shows how the default value of SIU TMCLK is used to route TimerA6 output to the SIU TMCNT and PIT logic (either when this connection is a GPIO or when it serves the timer blocks). When this connection is programmed as an input to serve the SIU TMCLK, it maps default GND to the TIMER2 clock input of the timers block.

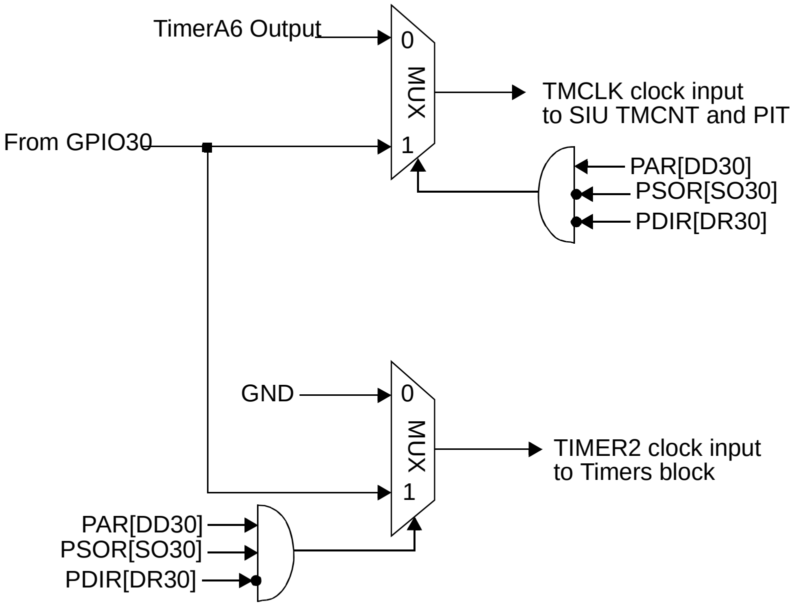


Figure 23-3. Using the Default Value to Select SIU TMCNT Clock Source

Figure 23-4 shows how unused GPIO connects to the default value of another GPIO. The GPIO connects to dedicated peripheral 2 as input at the same time dedicated peripheral 1 of that GPIO connects by its default value to another unused GPIO. For example, **Figure 23-4** shows GPIO6 as primary for $\overline{\text{IRQ4}}$ and GPIO0 connected as secondary for $\overline{\text{IRQ4}}$. If the combination of PAR, PSOR, and PDIR for GPIO6 does not select the primary $\overline{\text{IRQ4}}$ input, the secondary $\overline{\text{IRQ4}}$ input can be used if the combination of PAR, PSOR, and PDIR for GPIO0 selects it.

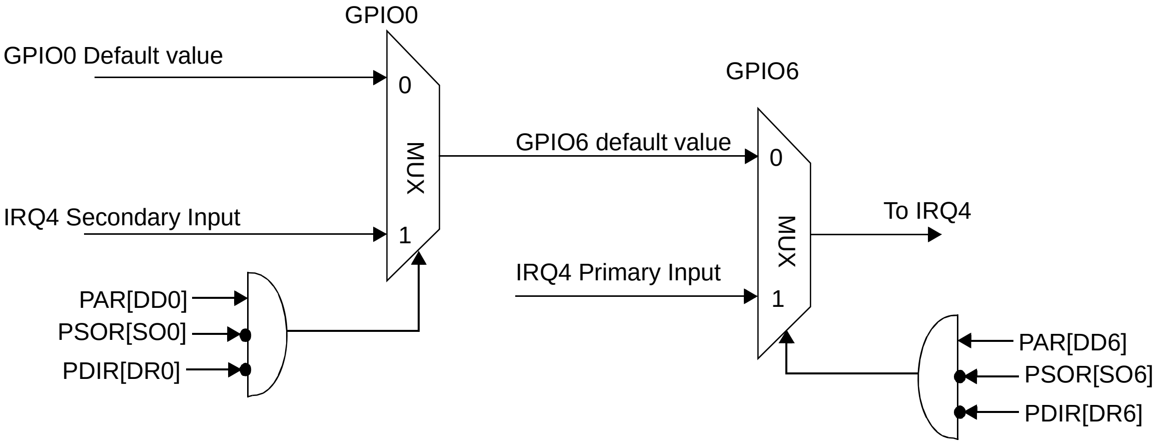


Figure 23-4. Using the Default Value to Connect the $\overline{\text{IRQ4}}$ Secondary to Primary Input Source

Table 23-2. GPIO Dedicated Assignment (PARx = 1)

GPIO	Port Function					
	PSORx = 0			PSORx = 1		
	PDIRx = 1 (Output, Unless Tri-State Option is Specified)	PDIRx = 0 (Input)	Default Input	PDIRx = 1 (Output)	PDIRx = 0 (Input, Unless Inout Option is Specified)	Default Input
0	—	$\overline{\text{IRQ4}}$ (secondary)	1	—	—	0
1	—	$\overline{\text{IRQ5}}$ (secondary)	1	—	TIMER0 (Inout)	0
2	—	$\overline{\text{IRQ6}}$ (secondary)	1	—	TIMER1 (Inout)	0
3	—	$\overline{\text{IRQ1}}$	1	—	TDM3TSYN (Inout)	0
4	—	$\overline{\text{IRQ2}}$	1	—	TDM3TCLK	0
5	—	$\overline{\text{IRQ3}}$	1	—	TDM3TDAT (Inout)	0
6	—	$\overline{\text{IRQ4}}$ (primary)	by GPIO0	—	TDM3RSYN (Inout)	0
7	—	$\overline{\text{IRQ5}}$ (primary)	by GPIO1	—	TDM3RCLK (Inout)	0
8	—	$\overline{\text{IRQ6}}$ (primary)	by GPIO2	—	TDM3RDAT (Inout)	0
9	—	$\overline{\text{IRQ7}}$	1	—	TDM2TSYN (Inout)	0
10	—	$\overline{\text{IRQ8}}$	1	—	TDM2TCLK	0
11	—	$\overline{\text{IRQ9}}$	1	—	TDM2TDAT (Inout)	0
12	—	$\overline{\text{IRQ10}}$	1	—	TDM2RSYN (Inout)	0
13	—	$\overline{\text{IRQ11}}$	1	—	TDM2RCLK (Inout)	0
14	—	$\overline{\text{IRQ12}}$	1	—	TDM2RDAT (Inout)	0
15	—	DREQ1 (primary)	by GPIO27	—	TDM1TSYN (Inout)	0
16	—	TDM1TCLK	0	$\overline{\text{DRACK1}}$	$\overline{\text{DONE1}}$ (Inout)	1
17	$\overline{\text{DACK1}}$	—	1	—	TDM1TDAT (Inout)	0
18	—	DREQ2 (primary)	by GPIO28	—	TDM1RSYN (Inout)	0
19	$\overline{\text{DACK2}}$	—	0	—	TDM1RCLK (Inout)	0
20	—	—	1	—	TDM1RDAT (Inout)	0

Table 23-2. GPIO Dedicated Assignment (PARx = 1) (Continued)

GPIO	Port Function					
	PSORx = 0			PSORx = 1		
	PDIRx = 1 (Output, Unless Tri-State Option is Specified)	PDIRx = 0 (Input)	Default Input	PDIRx = 1 (Output)	PDIRx = 0 (Input, Unless Inout Option is Specified)	Default Input
21	—	—	0	—	TDM0TSYN (Inout)	0
22	—	TDM0TCLK	0	$\overline{\text{DRACK2}}$	$\overline{\text{DONE2}}$ (Inout)	1
23	—	$\overline{\text{IRQ13}}$	1	—	TDM0TDAT (Inout)	0
24	—	$\overline{\text{IRQ14}}$	1	—	TDM0RSYN (Inout)	0
25	—	$\overline{\text{IRQ15}}$	1	—	TDM0RCLK (Inout)	0
26	—	—	0	—	TDM0RDAT (Inout)	0
27	—	DREQ1 (secondary)	0	—	URXD	1
28	—	DREQ2 (secondary)	0	—	UTXD (Inout)	0
29	—	—	0	—	—	0
30	—	TMCLK	TimerA6 output	—	TIMER2 (Inout)	0
31	—	—	0	—	TIMER3 (Inout)	0
Note: Ports designated as secondary are available only when the primary option for this function is not used.						

23.5 GPIO Programming Model

The GPIO registers reside on the 256 KB address space of the IPBus. They are accessed through the SQBus, the system bus, and the DSI. The addresses of the GPIO registers for accesses through the SQBus are presented in **Section 8.5**, *IPBus Address Space*, on page 8-12. The addresses of the GPIO registers for accesses through the system bus are presented in **Section 8.7**, *System Bus Address Space*, on page 8-55. The addresses of the GPIO registers for accesses through the DSI are presented in **Section 8.8**, *DSI Address Map*, on page 8-61. The GPIO block has five memory-mapped, read/write, 32-bit control registers. This section describes these registers in detail. Following is a list of the registers:

- Pin Open-Drain Register (PODR), **page 23-10**
- Pin Data Register (PDAT), **page 23-10**
- Pin Data Direction Registers (PDIR), **page 23-11**
- Pin Assignment Register (PAR), **page 23-11**
- Pin Special Options Registers (PSOR), **page 23-12**

PODR Pin Open-Drain Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	OD31	OD30	OD29	OD28	OD27	OD26	OD25	OD24	OD23	OD22	OD21	OD20	OD19	OD18	OD17	OD16
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
	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PODR indicates a normal or active low open drain mode for wired-OR configuration of the outputs. When a GPIO port has Ethernet functionality (see **Table 23-1**), PODR*x* does not influence its driving mode.

Table 23-3. PODR Bit Descriptions

Name	Reset	Description	Settings
OD[31–0] 0–31	0	**Open-Drain Configuration** Determines whether the corresponding port is actively driven as an output or is an open-drain driver. As an open-drain driver, the port is driven active-low. Otherwise, it is tri-stated (high impedance).	0 The I/O port is actively driven as an output. 1 The I/O port is an open-drain driver.

PDAT Pin Data Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
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
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A read of a PDAT register returns the data at the pin, independently of whether the port is defined as an input or output. Thus, output conflicts at the pin can be detected by comparing the written data with the data on the pin. A write to the PDAT*x* is sampled in a register bit, and if the equivalent PDIR bit is configured as an output, the value sampled for that bit is driven onto its respective pin. PDAT can be read or written at any time.

If a pin is selected as GPIO, it is accessed through the PDAT. Data written to the PDAT register is stored in an output register. If a port is configured as an output, the output register data is gated onto the pin. When PDAT is read, the GPIO pin itself is read. If a GPIO port is configured as an input, data written to the PDAT register is still stored in the output register, but it is prevented from reaching the actual pin. When the PDAT register is read, the state of the actual pin is read.

When a GPIO port has Ethernet functionality (see **Table 23-1**), data written to PDAT*x* is stored in the output register, but it is prevented from reaching the external port. Read of PDAT*x* returns the data at the external port, independently of whether the port is defined as input or output in Ethernet Controller.

PDIR Pin Data Direction Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	DR31	DR30	DR29	DR28	DR27	DR26	DR25	DR24	DR23	DR22	DR21	DR20	DR19	DR18	DR17	DR16
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
	DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDIR is cleared at system reset.

Table 23-4. PDIR Bit Descriptions

Name	Reset	Description	Settings
DR 0–31	0	**Direction** Indicates whether a port is an input or an output.	0 The corresponding port is an input. 1 The corresponding port is an output.

PAR Pin Assignment Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	DD31	DD30	DD29	DD28	DD27	DD26	DD25	DD24	DD23	DD22	DD21	DD20	DD19	DD18	DD17	DD16
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
	DD15	DD14	DD13	DD12	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAR is cleared at system reset.

Table 23-5. PAR Bit Descriptions

Name	Reset	Description	Bit Settings
DD[31–0] 0–31	0	**Dedicated Enable** Indicates whether a pin is a GPIO or a dedicated peripheral port. As a dedicated peripheral function, the pin is used by the internal module. The internal peripheral function to which it is dedicated can be determined by other bits, such as those in the PSOR. When a GPIO port has Ethernet functionality (see **Table 23-1**), its PAR bit should be set to 0.	0 GPIO. The peripheral functions of the pin are not used. 1 Dedicated peripheral function.

PSOR Pin Special Options Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SO31	SO30	SO29	SO28	SO27	SO26	SO25	SO24	SO23	SO22	SO21	SO20	SO19	SO18	SO17	SO16
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
	SO15	SO14	SO13	SO12	SO11	SO10	SO9	SO8	SO7	SO6	SO5	SO4	SO3	SO2	SO1	SO0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSOR bits are effective only if the corresponding PAR[DD*x*] bit is 1 (a dedicated peripheral function).

Table 23-6. PSOR Bit Descriptions

Name	Reset	Description	Settings
SO[31–0] 0–31	0	**Special-Option** Determines whether an external connection configured for a dedicated function (PAR[DD] = 1) uses option 1 or option 2. See **Table 23-2**.	0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

If the corresponding PAR[DD*x*] bit is 1 (configured as a dedicated peripheral function port) before a PSOR or PDIR bit is programmed, an external connection may function for a short period as an unwanted dedicated function and cause unpredictable behavior. Thus, it is recommended that you program the GPIO in the following sequence: PSOR, PODR, PDIR, PAR.

I²C Software Module 24

The I^{2}C software module implements the I^{2}C master bus protocol through two signal lines in the GPIO port. It allows the SC140 core to transfer sequential bytes of code or data to internal memory from any I^{2}C slave memory device, such as a serial EEPROM.

Note: This chapter assumes that you are already familiar with the I^{2}C bus specification. Note that the MSC8113 GPIO specifications do not fully conform to the I^{2}C bus electrical specification. Refer to the *MSC8113 Technical Data* sheet for details.

The software module uses the following:

- *Low level routines*. See **Table 24-1**.
- *Global symbols*. See **Table 24-2**.
- *Global core registers*. See **Table 24-3**.

The I^{2}C software module is invoked by calling the *i2c_read_SequentialData* routine using the falling routine:

```
; d12- size of data bytes to read
; r3 - address at the serial EEPROM
; r4 - address at memory
move.l #DATA_SIZE,d12
move.l #SerMem_ADDR,r3
move.l #IntMem_ADDR,r4
bsr i2c_read_SequentialData
```

The *i2c_read_SequentialData* routine sets the test (T) bit in the SC140 core executing the software module and returns from the routine in the following error cases:

- Assertion of start or stop condition in a byte read or write session.
- I^{2}C arbitration is lost when the I^{2}C master transmits a bit on SDA when SCL is low and does not receive an equal bit on SDA when SCL is high.
- The ACK bit does not have the expected value. On a read session, it should be low for all bytes except the last byte.

In a Multi-Master environment in which all masters use the same **`DATA_SIZE`**, **`SerMem_ADDR`**, and **`IntMem_ADDR`** parameters, the *i2c_WaitFor_StartCond_BusFreeTime* routine enables a concurrent loading session by synchronizing the Start Conditions and using the I^{2}C bus loose arbitration scheme. **Figure 24-1** and **Figure 6-10** on page 6-16 provide sample scenarios.

Table 24-1. Low-Level Routines

Routine Name	Description
i2c_txrx_bit	Refer to **Section 24.1**, *i2c_txrx_bit Routine,* on page 24-3
i2c_txrx_byte	Refer to **Section 24.2**, *i2c_txrx_byte Routine,* on page 24-5
i2c_read_SequentialData	Refer to **Section 24.3**, *i2c_read_SequentialData Routine,* on page 24-7
i2c_sample_gpio	Refer to **Section 24.4**, *i2c_sample_gpio Routine,* on page 24-9
i2c_assert_start	Refer to **Section 24.5**, *i2c_assert_start Routine,* on page 24-10
i2c_assert_stop	Refer to **Section 24.6**, *i2c_assert_stop Routine,* on page 24-11
i2c_WaitFor_StartCond_BusFreeTime	Refer to **Section 24.7**, *i2c_WaitFor_StartCond_BusFreeTime Routine,* on page 24-12
i2c_write_SequentialData	Refer to **Section 24.8**, *i2c_write_SequentialData Routine,* on page 24-13

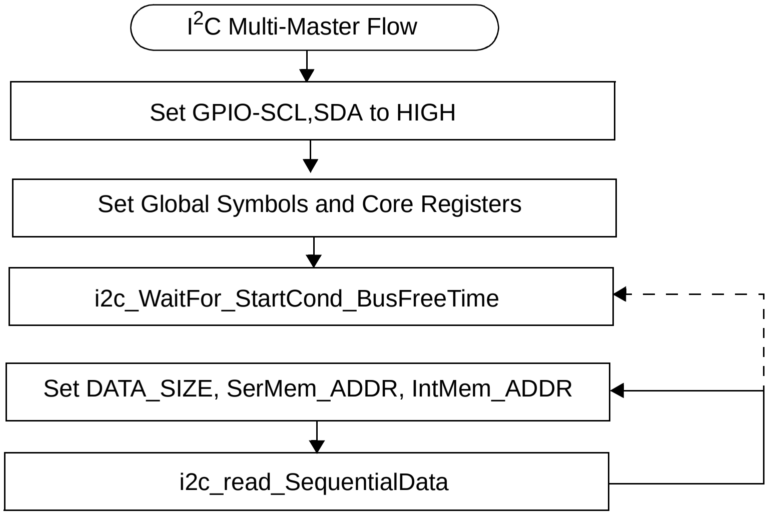

I2C Multi-Master Flow
Set GPIO-SCL,SDA to HIGH
Set Global Symbols and Core Registers
i2c_WaitFor_StartCond_BusFreeTime
Set DATA_SIZE, SerMem_ADDR, IntMem_ADDR
i2c_read_SequentialData

Figure 24-1. I^{2}C Multi-Master Procedure Flow

Table 24-2. Global Symbols

Symbol Name	Value	Description
SCL_SDA_01	0x4000	01 mask for SCL/SDA GPIO31/GPIO30.
SCL_SDA_10	0x8000	10 mask for SCL/SDA GPIO31/GPIO30.
SCL_SDA_11	0xc000	11 mask for SCL/SDA GPIO31/GPIO30.
HIGH_PERIOD	0x00000040	High Period time for 50 KHz SCL at 500 Mhz core frequency and Core/Bus frequency ratio of 3. Refer to **Table 24-4**.
HALF_LOW_PERIOD	0x00000200	Half time of the Low Period for 50 KHz SCL at 500 Mhz core frequency. Refer to **Table 24-4**.

Table 24-2. Global Symbols

Symbol Name	Value	Description
BUS_FREE_TIME	0x0000190B	Waiting time for WaitFor_BusFreeTime routine 1 ms at 500 MHz. Refer to **Table 24-7**.
HALF_BUS_FREE_TIME	0x00000C85	Waiting time at WaitFor_BusFreeTime routine for stable SCL = SDA = 1 for 500 µs at 500MHz. Refer to **Table 24-7**.
HD_STA_TIME	0x00000021	Hold time before asserting Start condition: 5 µs at 500 Mhz. Refer to **Table 24-5**.
BUF_TIME	0x00000042	Delay time after stop condition assertion 10 µs at 500 MHz. Refer to **Table 24-6**.
BASE_IP_B	0x01FBC000	IPBus memory.
PDAT_ADDR	0x01FBC208	GPIO PDAT register.

Table 24-3. Global Registers

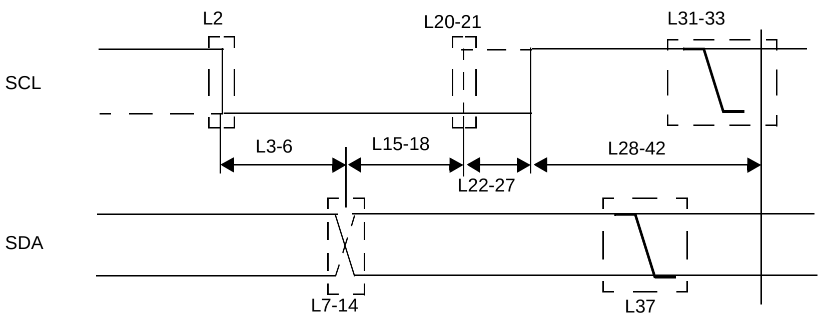
Register	Value	Description
D4		Bit position.
D5		Transmit byte.
D6		Receive byte.
D7		Control byte, bit 1 determines transmit or receive bit session.
D8	SCL_SDA_11	11 mask for SCL/SDA GPIO31/GPIO30.
D9		CHIPID value.
D10		Checksum calculation.
D12		Size of data to read.
D14	HIGH_PERIOD	High Period time for 50KHz SCL at 500Mhz core frequency.
D15	HALF_LOW_PERIOD	Half time of the Low Period for 50KHz SCL at 500Mhz core frequency.
R2	BASE_IP_B	IP Bus memory.
R3		Address at the serial EEPROM.
R4		Address of internal memory.
R9	PDAT_ADDR	GPIO PDAT register.

24.1 i2c_txrx_bit Routine

The *i2c_txrx_bit* routine transmits or receives a bit on the I^{2}C Bus. It also detects a start or stop condition during a write session and loss of arbitration during a read session.

Global Register Use	D4, D5,D6, D7, D8, D14. R9.
Local Register Use	D0 Timing of Low Period, Bit value extraction. D1 Bit value to set the SDA line.
Routine call	`i2c_sample_gpio`

Figure 24-2. i2c_txrx_bit Routine

Signal Diagram	 L2 L20-21 L31-33 SCL L3-6 L15-18 L28-42 L22-27 SDA L7-14 L37

Line	Code	Description
1	i2c_txrx_bit	
2	bmclr.w #SCL_SDA_10,(r9)	SCL low, no change of SDA.
3	move.l d15,d0	
4	lperiod_loop_1	Wait HALF_LOW_PERIOD.
5	deceq d0	
6	bf lperiod_loop_1	
7	tfr d5,d0	Byte to send.
8	and d4,d0	Bit to send.
9	or d7,d0	If receive session set output bit to 1.
10	tsteq d0	
11	clr d1	SCL_SDA_00.
12	iff move.w #SCL_SDA_01,d1	SCL_SDA_01.
13	move.w d1,(r9)	Set SDA to: 1 if read bit or write of 1 bit. 0 if write of 0 bit.
14	tfr d15,d0	Wait HALF_LOW_PERIOD. See **Table 24-4**.
15	lperiod_data_2	
16	deceq d0	
17	bf lperiod_loop_2	
18	or #SCL_SDA_10,d1.l	
19	move.l d1,r0	
20	move.w d1,(r9)	Set SCL to 1.
21	wait_scl_high	Wait for SCL line to be 1.
22	bsr i2c_sample_gpio	GPIO at D2.
23	and #SCL_SDA_11,d2,d3	
24	and #SCL_SDA_10,d2	
25	tsteq d2	
26	bt wait_scl_high	
27	bmchg #SCL_SDA_01,d3.l	
28	tfr d14,d0	Wait HIGH_PERIOD. Refer to **Table 24-4**.
29	hperiod_loop	Check for SDA to be the same, and for start/stop condition.
30	bsr i2c_sample_gpio	GPIO at D2.
31	bmtsts #SCL_SDA_10,d2.l	

Figure 24-2. i2c_txrx_bit Routine (Continued)

32	and d8,d2	
33	iff rts	Exit on SCl low.
34	cmpeq d3,d2	Check for SDA change while in HIGH_PERIOD (SCL = 1). T bit set for arbitration lost start/stop condition.
35	nop	
36	ift rts	Start/stop condition indication to higher routine.
37	cmpeq d8,d2	Check for SDA value.
38	nop	
39	ift bmset #1,d6.l	Set D6 according to receive/write bit value.
40	deceq d0	
41	bf hperiod_loop	
42	bmtstc #$1,d7.l	
43	nop	
44	iff rts	Write session, no need to check for arbitration lost.
45	bmtstc #$1,d6.l	Check for arbitration lost in write session.
46	move.l r0,d1	
47	iff move.w #SCL_SDA_10,d0	
48	ift tfr d8,d0	
49	cmpeq d0,d1	Set T bit for arbitration lost indication for higher routines.
50	rts	

Figure 24-2. i2c_txrx_bit Routine (Continued)

Table 24-4. HIGH_PERIOD and HALF_LOW_PERIOD Timing

Parameter	Core_Clock/Unit at Core/Bus = 3	Core_Clock/Unit at Core/Bus = 4	Core_Clock/Unit at Core/Bus = 5	Core_Clock/Unit at Core/Bus = 6	Initial value (Set at boot code)
HIGH_PERIOD	82	95	105	115	64
HALF_LOW_PERIOD	5	5	5	5	512

24.2 i2c_txrx_byte Routine

The *i2c_txrx_byte* routine transmits or receives a byte on the I^{2}C Bus. It also checks the ACK bit (for a write) or asserts the ACK bit (for a read). The routine returns to a higher routine for a lost arbitration or a start/stop condition, indicated when the txrx_bit routine sets the T bit.

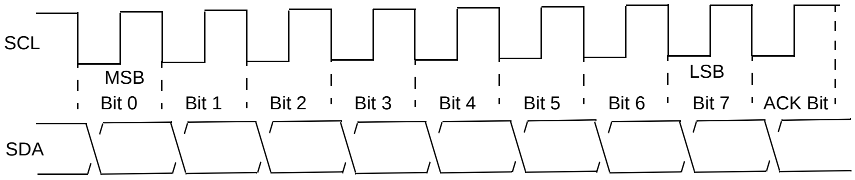
Global Register Use	D4, D6, D7.
Local Register Use	
Routine call	i2c_txrx_bit
Signal Diagram	

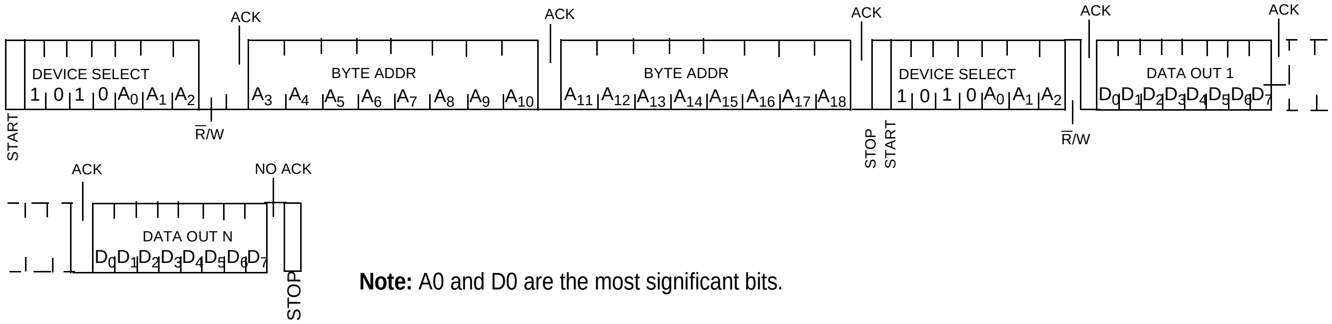
Figure 24-3. i2c_txrx_byte Routine

Line	Code	Description
1	i2c_txrx_byte	
2	clr d6	Receive byte.
3	move.w #$80,d4	Bit position.
4	byte_loop	
5	bsr i2c_txrx_bit	
6	ift rts	Return if arbitration lost or start/stop condition indication.
7	asr d4,d4	
8	asl d6,d6	Next bit read.
9	tsteq d4	
10	bf byte_loop	
11	bmchg #$1,d7.l	
12	bsr i2c_txrx_bit	
13	asr d6,d6	
14	bmchg #$1,d7.l	
15	rts	

Figure 24-3. i2c_txrx_byte Routine

24.3 i2c_read_SequentialData Routine

The *i2c_read_SequentialData* routine reads sequential bytes from the serial memory according to the protocol shown in **Figure 24-4**. It also returns to a higher routine for a lost arbitration, a wrong ACK bit value, or a start/stop condition, indicated when a lower routine sets the T bit.



Note: A0 and D0 are the most significant bits.

Figure 24-4. I^{2}C Serial Memory Sequential Read

Global Register Use	D5, D6, D7, D12. R3,R4.
Local Register Use	D0 manipulate slave address value. R7 keep slave address value.

Routine call	`i2c_txrx_byte.`
	`i2c_assert_start.`
	`i2c_assert_stop.`

Signal Diagram	Refer to **Figure 24-4.**

Line	Code	Description
1	i2c_read_SequentialData	
2	clr d7	
3	moveu.l #$00ffffff,d0	
4	and d0,d12	
5	bsr i2c_assert_start	
6	clr d5	
7	move.l r3,d0	Serial memory address.
8	extractu #$3,#$10,d0,d5	Extract A_0,A_1,A_2.
9	asl d5,d5	
10	bmset #$a0,d5	Write session.
11	move.l d5,r7	
12	bsr i2c_txrx_byte	Transmit slave address and A_0,A_1,A_2.
13	ift rts	Return for any arbitration lost, wrong ACK, or start/stop condition indication.
14	clr d5	Transmit Memory address A_3 to A_{19}.
15	move.l r3,d0	

Figure 24-5. i2c_read_SequentialData Routine

16	extractu #$8,#$0,d0,d5	
17	bsr i2c_txrx_byte	
18	ift rts	Return for any arbitration lost, wrong ACK, or start/stop condition indication.
19	move.l r3,d0	
20	extractu #$8,#$8,d0,d5	
21	bsr i2c_txrx_byte	
22	ift rts	Return for any arbitration lost, wrong ACK, or start/stop condition indication.
23	bsr assert_stop	
24	bsr assert_start	
25	move.l r7,d5	
26	bmset #$1,d5	Read session.
27	bsr i2c_txrx_byte	
28	ift rts	Return for any arbitration lost, Wrong ACK, start/stop condition indication.
29	read_byte_loop	
30	deceq d12	
31	move.w #$1,d7	Read byte indication.
32	ift bmset #$2,d7.l	Last byte indication.
33	bsr i2c_txrx_byte	
34	ift rts	
35	move.l #$1,d7	
36	bsr txrx_byte	
37	ift rts	Return for any arbitration lost, Wrong ACK, start/stop condition indication.
38	move.b d6,(r4)	
39	inca r4	
40	inca r3	
41	bmtstc #$0001,d11.h	Checksum calculation.
42	nop	
43	ift asll #$8,d6	
44	eor d6,d10	
45	bmchg #$0001,d11.h	
46	tsteq d12	
47	bf read_byte_loop	
48	bsr i2c_assert_stop	
49	bmtsts #$f,d11.l	Clear T bit.
50	rts	

Figure 24-5. i2c_read_SequentialData Routine (Continued)

24.4 i2c_sample_gpio Routine

The *i2c_sample_gpio* routine samples the GPIO-SCL/SDA pins. It allows only two successive stabilized samples to be acknowledged.

Global Register Use	D2, D9.
Local Register Use	D1.

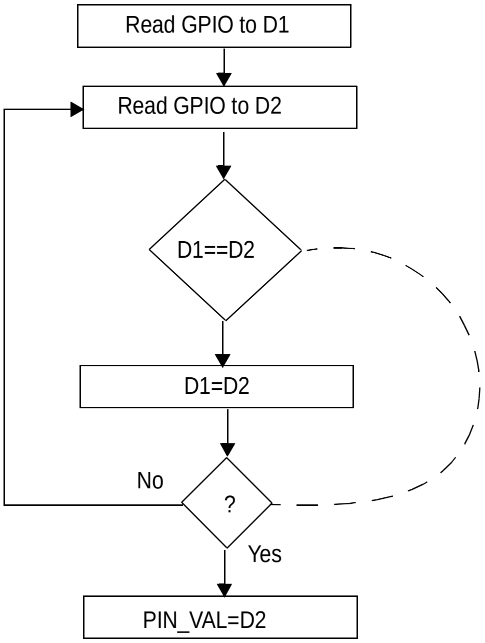
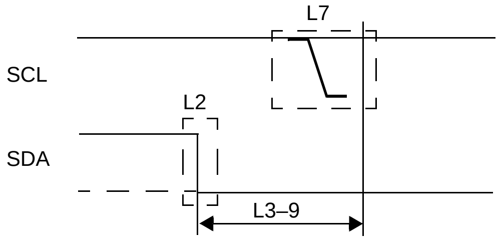
Line	Code	Description
1	i2c_sample_gpio	 Read GPIO to D1 Read GPIO to D2 D1==D2 D1=D2 No ? Yes PIN_VAL=D2
2	moveu.w PDAT_ADDR,d1	
3	and d8,d1	
4	loop_sample	
5	moveu.w PDAT_ADDR,d2	
6	and d8,d2	
7	cmpeq d1,d2	
8	tfr d2,d1	
9	bf loop_sample	
10	rts	

Figure 24-6. i2c_sample_gpio Routine

24.5 i2c_assert_start Routine

The *i2c_assert_start* routine asserts the start condition according to the I^{2}C bus specification.

Global Register Use	D2. R9.
Local Register Use	D0 Loop counting.
Routine call	`i2c_sample_gpio`.
Signal Diagram	

Line	Code	Description
1	i2c_assert_start	
2	move.w #SCL_SDA_10,(r9)	Set SDA to 0.
3	move.l #HD_STA_TIME,d0	Wait HD_STA_TIME. Refer to **Table 24-5**.
4	start_loop	
5	bsr i2c_sample_gpio	
6	bmtstc #SCL_SDA_10,d2.l	
7	bt start_rts	Exit loop on SCL = 0.
8	deceq d0	
9	bf start_loop	
10	start_rts	
11	rts	

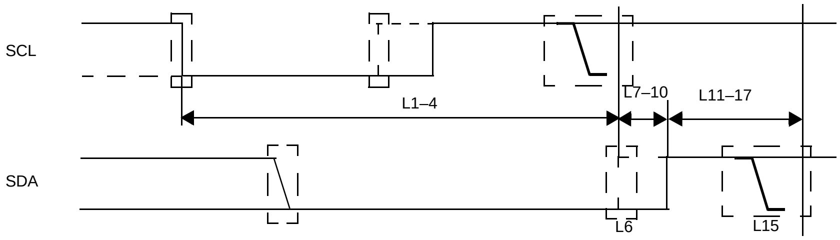
Figure 24-7. i2c_assert_start Routine

Table 24-5. HD_STA_TIME Timing

Parameter	Core_Clock at Core/Bus=3	Core_Clock at Core/Bus=4	Core_Clock at Core/Bus=5	Core_Clock at Core/Bus=6
HD_STA_TIME	0x21			

24.6 i2c_assert_stop Routine

The *i2c_assert_stop* routine asserts the stop condition according to the I^{2}C bus specification.

Global Register Use	D2,D4,D6, D7,D8. R9.
Local Register Use	D0 Counting loop.
Routine call	`i2c_txrx_bit`. `i2c_sample_gpio`.
Signal Diagram	

Line	Code	Description
1	i2c_assert_stop	
2	clr d4	
3	clr d6	
3	clr d7	
4	bsr i2c_txrx_bit	Write bit 0.
5	ift rts	Return for any arbitration lost, wrong ACK, or start/stop condition indication.
6	move.w d8,(r9)	Set SDA to 1.
7	wait_sda_high	Wait for SDA to be 1.
8	bsr i2c_sample_gpio	
9	bmtstc #SCL_SDA_01,d2.l	
10	bt wait_sda_high	
11	move.l #BUF_TIME,d0	
12	stop_loop	Wait BUF_TIME. Refer to **Table 24-6**.
13	bsr i2c_sample_gpio	
14	bmtstc #SCL_SDA_01,d2.l	
15	bt stop_rts	Exit loop on SDA = 0.
16	deceq d0	
17	bf stop_loop	
18	stop_rts	
19	rts	

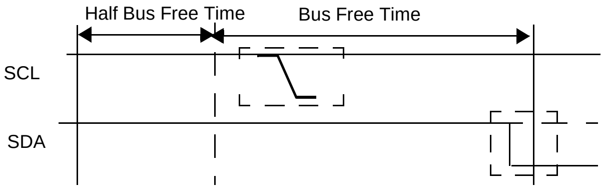
Figure 24-8. i2c_assert_stop Routine

Table 24-6. BUF_TIME Timing

Parameter	Core_Clock at Core/Bus = 3	Core_Clock at Core/Bus = 4	Core_Clock at Core/Bus = 5	Core_Clock at Core/Bus = 6
BUF_TIME	0x42			

24.7 i2c_WaitFor_StartCond_BusFreeTime Routine

The *i2c_WaitFor_StartCond_BusFreeTime* routine waits for either a start condition assertion according to the I^2C bus specification, or the time interval defined by the **BUS_FREE** parameter.

Global Register Use	D2.
Local Register Use	D0 Counting loop.
Routine call	**i2c_sample_gpio**.
Signal Diagram	

Line	Code	Description
1	i2c_WaitFor_StartCond_BusFreeTime	
2	move.l #HALF_BUS_FREE_TIME,d0	Wait **HALF_BUS_FREE_TIME**. Refer to **Table 24-6**.
3	busfree_loop	
	bsr i2c_sample_gpio	
	bmtsts #SCL_SDA_11,d2.l	
	bf i2c_WaitFor_StartCond_BusFreeTime	
	deceq d0	
	bf busfree_loop	
	move.l #BUS_FREE_TIME,d0	Wait **BUS_FREE_TIME**. Refer to **Table 24-7**.
3	wait_loop	
4	bsr i2c_sample_gpio	
5	bmtstc #SCL_SDA_10,d2.l	
6	bt i2c_WaitFor_StartCond_BusFreeTime	If SCL = 0, rerun **BusFree** wait loop.
7	bmtstc #SCL_SDA_01,d2.l	
8	bt wait_rts	
9	deceq d0	
10	bf wait_loop	
11	wait_rts	
12	rts	

Figure 24-9. i2c_WaitFor_StartCond_BusFreeTime Routine

Table 24-7. BUS_FREE_TIME Timing

Parameter	Core_Clock at Core/Bus = 3	Core_Clock at Core/Bus = 4	Core_Clock at Core/Bus = 5	Core_Clock at Core/Bus = 6
BUS_FREE_TIME (1 ms at 500 MHz)	500000	580000	630000	692400
HALF_BUS_FREE_TIME (500 µs at 500 MHz)	250000	290000	3150000	346200

24.8 i2c_write_SequentialData Routine

The *i2c_write_SequentialData* routine writes sequential bytes to the serial memory according to the protocol described in **Figure 24-10**. It also returns to a higher routine for a lost arbitration, a wrong ACK bit, or a start/stop condition, indicated when a lower routine sets the T bit.

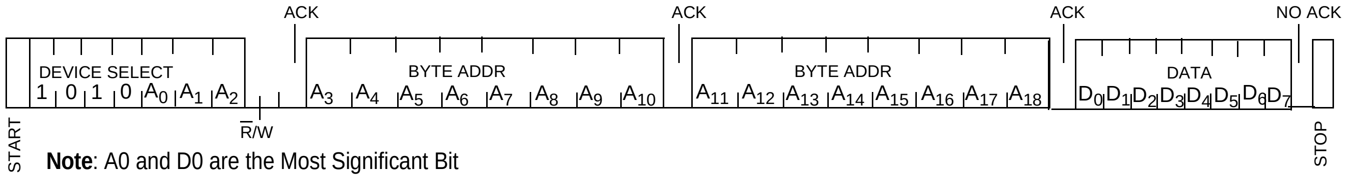


Figure 24-10. I^{2}C Serial Memory Sequential Write

Global Register Use	D2,D7. R3.
Local Register Use	D0 Counting loop. D5 Slave address.
Routine call	`i2c_assert_start`. `i2c_assert_stop`.
Signal Diagram	Refer to **Section 24-10**.

Line	Code	Description
1	i2c_write_SequentialData	
2	clr d7	
3	bsr i2c_assert_start	
4	clr d5	Slave address
5	move.l r3,d0	
6	extractu #$3,#$10,d0,d5	
7	asl d5,d5	
8	bmset #$a0,d5.l	
9	bsr i2c_txrx_byte	
10	ift rts	
11	clr d5	Memory address
12	move.l r3,d0	
13	extractu #$8,#$8,d0,d5	
14	bsr i2c_txrx_byte	

Figure 24-11. i2c_write_SequentialData Routine

15	ift rts	
16	move.l r3,d0	
17	extractu #$8,#$0,d0,d5	
18	bsr i2c_txrx_byte	
19	ift rts	
20	move.b (r4)+,d5	
21	bsr i2c_txrx_byte	
22	ift rts	
23	bsr i2c_assert_stop	
24	inca r3	
25	move.l #$100000,d0	Burn waiting time.
26	write_loop	
27	deceq d0	
28	bf write_loop	
29	deceq d12	
30	bf i2c_write_SequentialData	
31	rts	

Figure 24-11. i2c_write_SequentialData Routine (Continued)

Ethernet Controller 25

The **IEEE** Std. 802.3 standard specifies a local area network (LAN) protocol that uses carrier-sense multiple access with collision detection (CSMA/CD). Ethernet is one of the common implementations of this standard and is a simple, cost-effective option for backbone and server connectivity. Because Ethernet is very close to other **IEEE** Std. 802.3 protocols and because they can all coexist on the same LAN, this chapter uses the term Ethernet to indicate the **IEEE** Std. 802.3 interface unless otherwise noted. The 10/100 Ethernet supplement to the standard increases Ethernet speed from 10 to 100 megabits per second (Mbps). Another supplement defines the requirements for a media-independent interface (MII) that can support various physical implementations. Other protocols developed modifications to this basic interface, including the reduced media-independent interface (RMII) and the serial media-independent interface (SMII). The MSC8113 Ethernet controller supports MII, RMII, and SMII for the 10/100 Ethernet rate.

25.1 Ethernet Basics

The Ethernet protocol implements the bottom two layers of the open systems interconnection (OSI) 7-layer model, which are the data link and physical sublayers. **Figure 25-1** depicts the typical Ethernet protocol stack and the relationship to the OSI model.

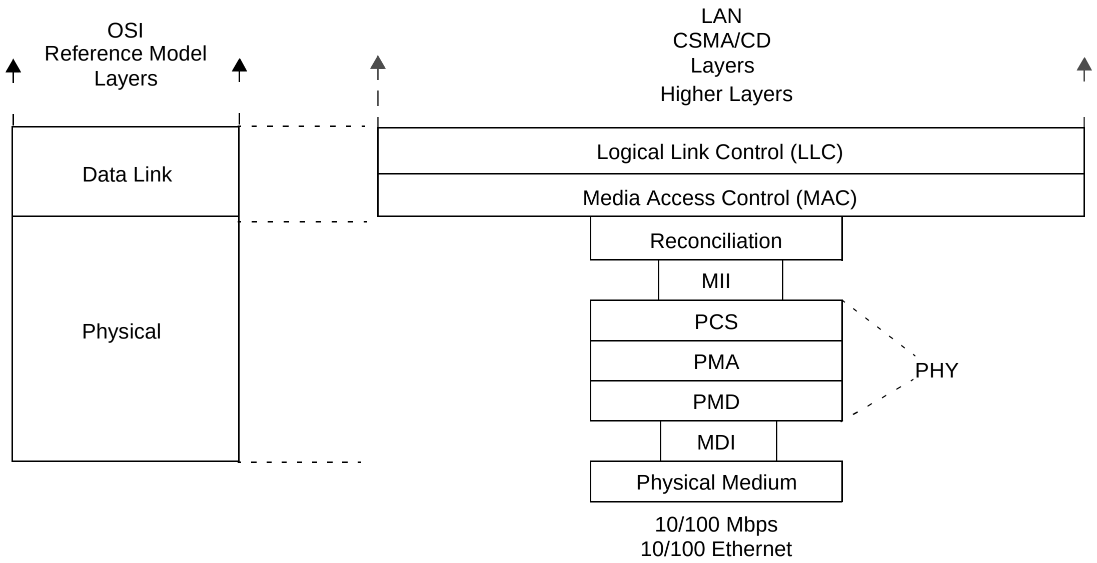


Figure 25-1. Ethernet Protocol in Relation to the OSI Protocol Stack

The 10/100 Mbps baseband Ethernet provides the sublayers listed in **Table 25-1**.

Table 25-1. Sublayers of the 10/100 Mbps Baseband Ethernet

Sublayer	Description
Media Access Control (MAC)	A logical connection between the MAC and its peer station. It initializes, controls, and manages the connection with the peer station.
Reconciliation	A command translator that maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.
Media-Independent Interface (MII)	A standard (**IEEE** Std. 802.3) interface between the MAC layer and the physical layer for 10/100 Mbps operations. It isolates the MAC layer and the physical layer so that the MAC layer can be used with various implementations of the physical layer.
PCS	Contains the functions to encode the data bits into code groups that can be transmitted over the physical medium. Three PCS structures are defined for 100Base-T: one for 100Base-X, one for 100Base-T4, and one for 100Base-T2. (see **IEEE** Std. 802.3, Clauses 23, 24, and 32.) The PCS transmit function accepts data nibbles from the MII, encodes these nibbles using an 8B6T coding scheme, and passes the resulting ternary symbols to the PMA. In the reverse direction, the PMA conveys received ternary symbols to the PCS receive function. The PCS receive function decodes them into octets and then passes the octets one nibble at a time up to the MII. The PCS also contains a PCS Carrier Sense function, a PCS Error Sense function, a PCS Collision Presence function, and a management interface.
Physical Medium Attachment (PMA)	Within 802.3, the portion of the physical layer that contains the functions for transmission, reception, and (depending on the PHY) collision detection, clock recovery, and skew alignment.
Physical Medium Dependent (PMD)	Handles signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.
Medium-Dependent Interface (MDI)	A connector that defines different connector types for different physical media and PMD devices.

Ethernet/**IEEE** Std. 802.3 frames are based on the frame structure shown in **Figure 25-2**. The term *packet* sometimes refers to the frame plus the preamble and start frame delimiter (SFD).

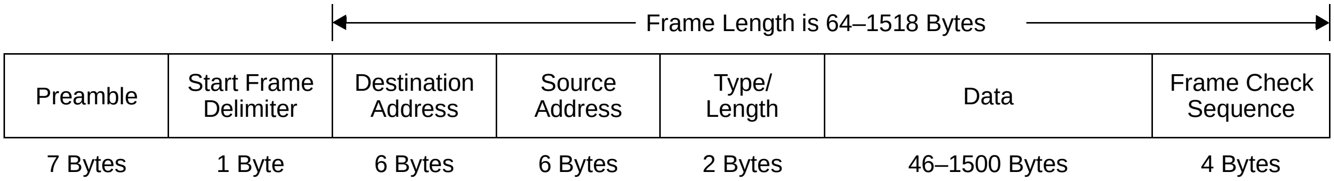


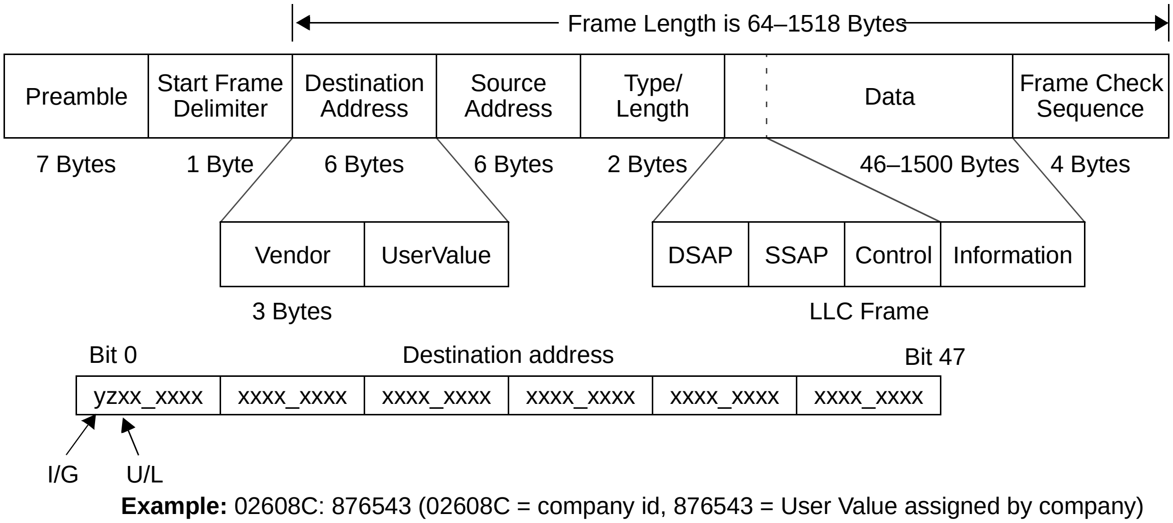
Figure 25-2. Ethernet/**IEEE** Std. 802.3 Frame Structure

The elements of an Ethernet frame are as follows:

- *Preamble*. Alternating ones and zeros for receiver timing synchronization (each byte containing the value 0x55).
- *Start frame delimiter (SFD)*. A sequence of 0xD5 (10101011 because the bit ordering is lsb first) indicates the beginning of the frame.
- *Destination address (DA)*. The first bit of a total of 48 bits identifies the address as an individual address (0) or a group address (1). The second bit indicates whether the address is locally-defined (1) or globally-defined (0).

- *Source address (SA)*. A total of 48 bits. Original versions of the **IEEE** Std. 802.3 specification allowed 16-bit addressing, which has never been widely used.
- *Ethernet type field/**IEEE** Std. 802.3 length field*. Signifies the protocol used in the rest of the frame (for example, TCP/IP). The length field specifies the length of the data portion of the frame. For both Ethernet and **IEEE** Std. 802.3 frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This limitation requires that a type field be identified by a decimal number equal to or greater than 1536 (0x0600) but less than 65535 (0xFFFF). However, if the number is between 0 and 1500 (0x0000 through 0x05DC), this field indicates the length of the MAC client data. The range 1501–1536 (0x5DD–0x600) is intentionally undefined.
- *Data and padding*. Padding is optional. It is needed only if the data is smaller than 64 octets (one octet = one byte) to ensure the minimum frame size of 64 octets as specified in the standard. In **IEEE** Std. 802.3x, the first two octets of the data field contain an opcode (OP) (pause = 0x0001) and the second two octets transmit a pause time (PT) parameter (pausetime = 0x0000 for on and 0xFFFF for off). In addition, a third two-octet field can be used for an extended pause control parameter (PTE). Because the use of these fields varies with the protocol, the ability to examine them and report their content can significantly accelerate Ethernet frame processing.
- *Frame-check sequence (FCS)*. Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD, and CRC.

Figure 25-3 shows additional details on the Ethernet/**IEEE** Std. 802.3 frame structure.



Example: 02608C: 876543 (02608C = company id, 876543 = User Value assigned by company)

Figure 25-3. Ethernet/**IEEE** Std. 802.3 Frame Structure With Details

As **Figure 25-3** shows, the **IEEE** Std. 802.3 section 3.11 (MAC frame format) defines the frame format so that the octets of a frame are transmitted from left to right with the preamble first and the FCS last. The bits of each octet are transmitted least-significant bit (lsb) first.

Table 25-2 illustrates how the destination address example (02608C:876543) shown in **Figure 25-3** is normally written and how the octets are actually transmitted.

Table 25-2. Data Transmission Example Using a Destination Address

Bit Order	Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6
Normal bit order	0000 0010	0110 0000	1000 1100	1000 0111	0110 0101	0100 0011
Transmitted bit order	0100 0000	0000 0110	0011 0001	1110 0001	1010 0110	1100 0010
Note: The example is an individual address because the lsb is cleared, but it is locally-defined because the second least-significant bit is set.						

Originally, a type field was used for protocol identification. The **IEEE** Std. 802.3 specification eliminated the type field, replacing it with the length field, which identifies the length of the data field, in bytes. The protocol type in 802.3 frames is held within the data portion of the packet. The logical link control (LLC) provides services to the network layer, regardless of media type, such as FDDI, Ethernet, and token ring. The LLC layer uses LLC protocol data units (PDUs) to communicate between the MAC layer and the upper layers of the protocol stack. Three variables determine access into the upper layers via the LLC-PDU:

- *Destination service access point (DSAP)*. Specifies a unique identifier within the station providing protocol information for the upper layer.
- *Source service access point (SSAP)*. Provides the same information for the source address.
- *Control*. The last two bits specify the type of control variable (unnumbered, information transfer, or supervisory) and the rest of the 8- or 16-bit field defines the rest of the control parameters.

The LLC defines service access for protocols that conform to the open system interconnection (OSI) model for network protocols. However, many protocols (including IP and IPX) do not obey the rules for those layers, and information on these protocols must be added to the LLC via a method called a subnetwork access protocol (SNAP) frame. A SNAP encapsulation is indicated when the DSAP and SSAP addresses are set to 0xAA and the LLC Control field is set to 0x03. The SNAP header is five bytes long. The first three bytes contain the organization code (SNAP OUI), which is assigned by the **IEEE**. The last two bytes contain the type value set from the original Ethernet specifications if SNAP OUI = 00-00-00, or they become a SNAP protocol identifier if SNAP OUI is non zero.

25.2 Media-Independent Interfaces

The Ethernet protocol can support a variety of physical interfaces to transfer information. The MSC8113 Ethernet controller supports three interfaces: MII, RMII, and SMII. The major difference between them is the number of signals used for data transfer:

- *MII*. Supports the full set of 18 Ethernet signals (four receive data lines, four transmit data lines, transmit and receive clocks, six control signals, and two data management signals).
- *RMII*. Supports a reduced set of 10 Ethernet signals (two receive data lines, two transmit data lines, reference clock, three control signals, and two data management signals).
- *SMII*. Supports a serial interface with 6 Ethernet signals (a receive data line, a transmit data line, a sync signal, a clock, and two data management signals). In SMII MAC-to-MAC mode, instead of driving the SYNC signal output, the Ethernet controller uses the SYNC_IN signal input; in this mode, the ETHMDC and ETHMDIO signals are not used.

Section 25.5 discusses the signals used by each of these interface types.

25.3 MSC8113 Ethernet Controller

Figure 25-4 shows the Ethernet controller block diagram.

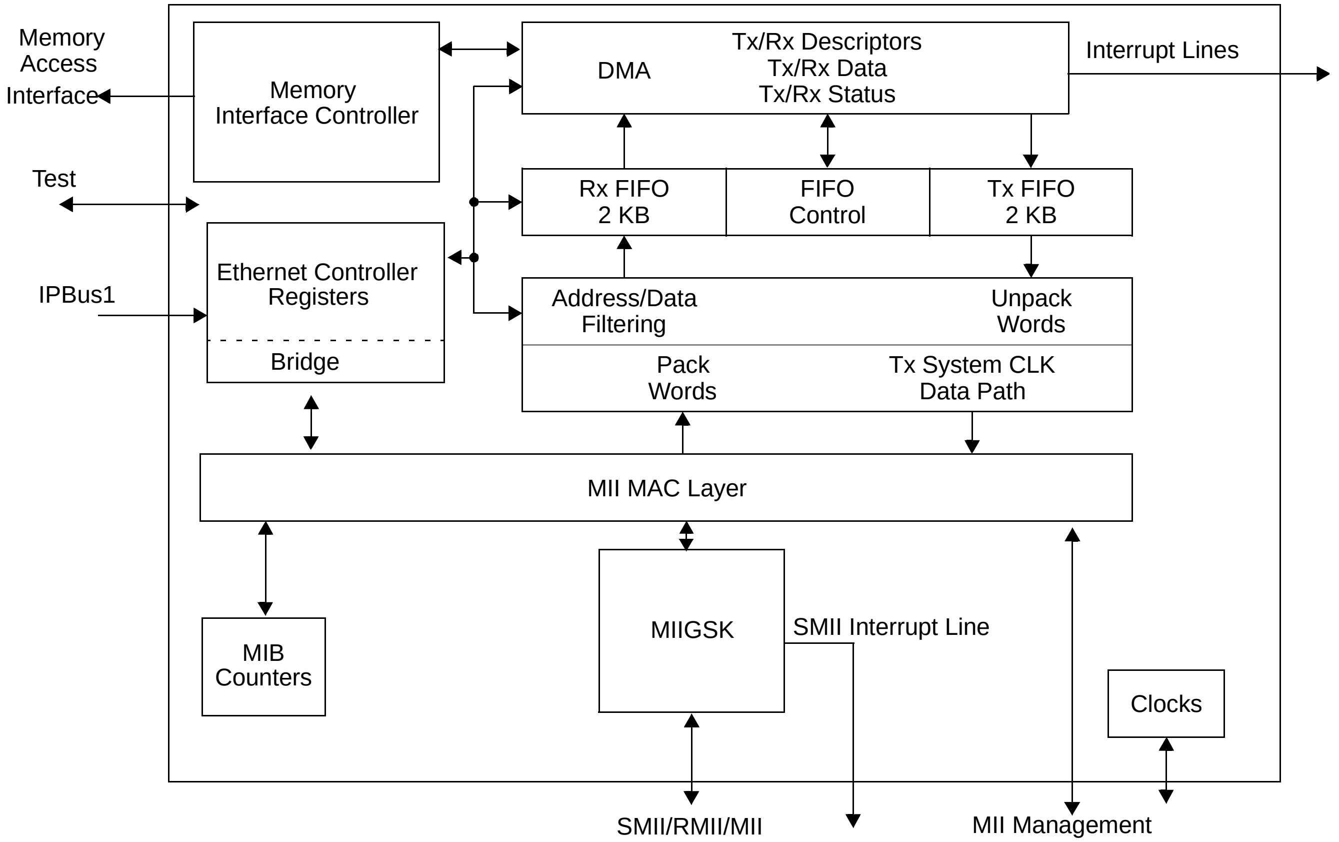


Figure 25-4. Ethernet Controller Overview

The Ethernet controller supports 10 Mbps and 100 Mbps Ethernet/802.3 networks and contains the following components:

- Ethernet media access controller (MAC)
- First-in first-out (FIFO) controller and direct memory access (DMA) controller
- Register-based statistical module that supports management information base (MIB) remote monitoring (RMON)
- MII Bridge (MIIGSK)

The most significant byte of data in a receive or transmit data buffer corresponds to the most significant byte of a frame, respectively.

25.4 Modes of Operation

Table 25-3 summarizes the available Ethernet controller operating modes. For detailed register information, see **Section 25.17**, *Ethernet Controller Programming Model,* on page 25-49.

Table 25-3. Selecting the Ethernet Controller Operating Modes

Interface Mode	Register Configurations	Operating Mode
MII mode	MIIGSK_CFGR[IFMODE] = 00	Operating speed is determined by the ETHTX_CLK and ETHRX_CLK signals, which are driven by the transceiver.
RMII mode	MIIGSK_CFGR[IFMODE] = 01	For RMII and SMII modes: 100 Mbps: • MIIGSK_CFGR[FRCONT] = 0 10 Mbps: • MIIGSK_CFGR[FRCONT] = 1
SMII mode (MAC-to-PHY or MAC-to-MAC)	MAC-to-PHY: • MIIGSK_CFGR[IFMODE] = 10 • SMII_SYNC_DIR[SYNC_IN] = 0 • SMII_SYNC_DIR[SYNC] = 1 MAC-to-MAC: • MIIGSK_CFGR[IFMODE] = 10 • SMII_SYNC_DIR[SYNC_IN] = 1 • SMII_SYNC_DIR[SYNC] = 0	
Loopback mode	MII: • MIIGSK_CFGR[IFMODE] = 00 • MIIGSK_CFGR[LBMODE] = 0 • MACCFG1R[MIILB] = 1 RMII: • MIIGSK_CFGR[IFMODE] = 01 • MIIGSK_CFGR[LBMODE] = 1 • MACCFG1R[MIILB] = 0 SMII Sync Out: • MIIGSK_CFGR[IFMODE] = 10 • MIIGSK_CFGR[LBMODE] = 1 • SMII_SYNC_DIR[SYNC_IN] = 0 • SMII_SYNC_DIR[SYNC] = 1 • MACCFG1R[MIILB] = 0	

Echo mode (MII mode only)	MIIGSK_CFGR[IFMODE] = 00
	MIIGSK_CFGR[EMODE] = 1
	MIIGSK_CFGR[LBMODE} = 0	

25.4.1 MII Mode

MII is the media-independent interface defined by the **IEEE** Std. 802.3 standard for 10/100 Mbps operation. To operate the Ethernet controller in MII mode, write a value of 00 to MIIGSK_CFGR[IFMODE] (see **Table 25-72**). The actual transfer speed is determined by the ETHTX_CLK and ETHRX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed, or software controls it via the transceiver serial management interface (ETHMDC/ETHMDIO).

25.4.2 RMII Mode

RMII is a reduced pin-count MII. To operate the Ethernet controller in RMII mode, write a value of 01 to MIIGSK_CFGR[IFMODE] (see **Table 25-72**). The operating mode is determined by the Frequency Control bit (MIIGSK_CFGR[FRCONT]); the default value of 0 selects 100 Mbps operation.

For 10 Mbps operation, write a 1 to MIIGSK_CFGR[FRCONT]. The RMII reference clock generates both transmit and receive clocks for the RMII.

25.4.3 SMII Mode

SMII is a serial MII. The SMII can operate as a MAC-to-PHY or a MAC-to-MAC connection:

- A MAC-to-PHY conveys complete MII information between a 10/100 PHY and MAC using two signals per port, and generates the output SYNC signal to allow a MAC-to-PHY connection. The SMII reference clock generates both transmit and receive clocks for the MII clocks. You can configure the Ethernet controller for SMII MAC-to-PHY mode by writing 10 to MIIGSK_CFGR[IFMODE] and selecting the SYNC output signal by writing a 0 to MIIGSK_SMII_SYNCDIR[SYNC_IN] and a 1 to MIIGSK_SMII_SYNCDIR[SYNC] (see **Table 25-72** and **Table 25-75**). The operating mode is determined by the Frequency Control bit (MIIGSK_CFGR[FRCONT]); the default value 0 selects 100 Mbps operation. For 10 Mbps operation, set the frequency by writing a 1 to MIIGSK_CFGR[FRCONT].
- For a MAC-to-MAC connection, the SMII uses a SYNC input signal to support data synchronization and disables the typical output SYNC signal generation. Select this mode by writing 10 to MIIGSK_CFGR[IFMODE] and selecting the ETHSYNC_IN input by writing a 1 to MIIGSK_SMII_SYNCDIR[SYNC_IN] and a 0 to MIIGSK_SMII_DYNCDIR[SYNC] (see **Table 25-72** and **Table 25-75**). The operating mode is determined by the Frequency Control bit (MIIGSK_CFGR[FRCONT]); the default value 0 selects 100 Mbps operation. For 10 Mbps operation, set the frequency by writing a 1 to MIIGSKCFGR[FRCONT]. The SMII reference clock generates both transmit and receive clocks for the MII.

25.4.4 Special Modes

The Ethernet controller operates in three special modes:

- Loopback mode
- Echo mode
- Low-Power Stop mode

25.4.4.1 Loopback Mode

Loopback mode operation is determined by the type of interface selected, as follows:

- *Internal Loopback in MII Domain*:
  - — Select MII mode (MIIGSK_CFGR[IFMODE] = 00).
  - — Set MACCFG1R[MIILB].
  - — Clear MIIGSK_CFGR[LBMODE].
  - — Set MACCFG2R[FDUP] to select Full Duplex mode.

  This configuration causes the MII MAC transmit outputs to be looped back to the MAC receive inputs. Clearing MACCFG1R[MIILB] results in a return to normal operation in MII mode.
- *Internal Loopback in RMII Domain*:
  - — Select RMII mode (MIIGSK_CFGR[IFMODE] = 01]
  - — Clear MACCFG1R[MIILB]
  - — Set RMIICFGR[LBMODE]
  - — Set MACCFG2R[FDUP] to select Full Duplex mode.

  This configuration causes the RMII MAC transmit outputs to be looped back to the MAC receive inputs. Clearing RMIICFGR[LBMODE] results in a return to normal operation in RMII mode.
- *Internal Loopback in SMII SYNC OUT Domain*.
  - — Select SMII (MIIGSK_CFGR[IFMODE] = 10)
  - — Set MIIGSK_SMII_SYNCDIR[SYNC]
  - — Clear MIIGSK_SMII_SYNCDIR[SYNC_IN]
  - — Clearing MACCFG1R[MIILB]
  - — Set MIIGSK_CFGR[LBMODE]
  - — Set MACCFG2R[FDUP] to select Full Duplex mode.

  This configuration causes the SMII MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in a return to normal operation in SMII SYNC Out mode.

25.4.4.2 Echo Mode

The Ethernet controller allows the MII to operate in Echo mode using the following configuration:

- Select MII mode (MIIGSK_CFGR[IFMODE] = 00)
- Clear MACCFG1R[MIILB]
- Clear MIIGSK_CFGR[LBMODE]
- Set MIIGSK_CFGR[EMODE]

Selecting this mode causes the Ethernet controller MII to receive inputs from the MII PHY that are looped back to the Ethernet controller MII transmit outputs of the MII PHY. In this mode, the MII PHY receives the frame.

25.4.4.3 Low-Power Stop Mode

The Ethernet controller enters Low-Power Stop mode when the following conditions are met; the Ethernet controller responds with a stop acknowledgement:

- Set SCR1[ETH_STC].
- Clear all the IMASK register Interrupt Events Enable bits.
- No Ethernet controller access is pending on the Internal Peripheral Interface (IPI) line.

In Low Power Stop mode, the Ethernet controller conversion operation is still enabled, but the Ethernet controller registers cannot be accessed, and no new interrupts are captured because the Ethernet controller output interrupt line is deasserted. To exit from Low Power Stop mode, clear SCR1[ETH_STC] in the IP master block. To clear all pending interrupts when in MII or RMII mode, write a value of 0xFFFFFFF to the IEVENT register. You can reenable the interrupt sources in the IMASK register as required. To clear all pending interrupts when in SMII mode, write a value of 0xFFFFFFFF to the MIIGSK_IEVENT and the IEVENT registers. You can reenable the interrupt sources in MIIGSK_IMASK and IMASK registers as required.

Note: The MIIGSK_IMASK and MIIGSK_IEVENT registers are valid only in SMII mode.

25.4.5 Management Interface

The management interface (ETHMDIO/ETHMDC) is identical to that defined in **IEEE** Std. 802.3u™ for all normal operating modes (MII/RMII/SMII).

25.5 External Signals

This section defines the Ethernet controller-to-chip signal I/O. The network interface supports three options:

- MII option requires 16 I/O signals (see **Table 25-6**).
- RMII option requires 8 I/O signals (see **Table 25-7**).
- SMII requires 4 I/O signals (see **Table 25-8**).

Note: The Ethernet controller ETHMDC and ETHMDIO management interface signals are common to all modes

All options support 10 and 100 Mbps Ethernet rates. Fifteen of these signals are multiplexed with GPIO or DSI/system data bus signal lines. Three additional lines are dedicated signals. **Table 25-4** lists the multiplex options for the Ethernet signals.

Table 25-4. Ethernet Signals

DSI/System Data Bus Multiplexing[1]	GPIO/TDM Signal Multiplexing[2]	MII Standard Name	RMII Standard Name	SMII Standard Name
HD40/D40/ETHRXD0	GPIO14/TDM2RDAT/$\overline{\text{IRQ12}}$/ETHRXD0[3]	RXD0	RXD0	—
HD41/D41/ETHRXD1	GPIO12/TDM2RSYN/$\overline{\text{IRQ10}}$/ETHRXD1/ETHSYNC	RXD1	RXD1	SYNC
HD42/D42/ETHRXD2/NC	GPIO6/TDM3RSYN/$\overline{\text{IRQ4}}$/ETHRXD2/NC	RXD2	—	—
HD43/D43/ETHRXD3/NC	GPIO5/TDM3TDAT/$\overline{\text{IRQ3}}$/ETHRXD3/NC	RXD3	—	—
HD46/D46/ETHTXD0	GPIO0/CHIP_ID0/$\overline{\text{IRQ4}}$/ETHTXD0	TXD0	TXD0	—
HD47/D47/ETHTXD1	GPIO1/TIMER0/CHIP_ID1/$\overline{\text{IRQ5}}$/ETHTXD1	TXD1	TXD1	—
HD48/D48/ETHTXD2/NC	GPIO3/TDM3TSYN/IRQ1/ETHTXD2/NC	TXD2	—	—
HD49/D49/ETHTXD3/NC	GPIO7/TDM3RCLK/$\overline{\text{IRQ5}}$/ETHTXD3/NC	TXD3	—	—
HD54/D54/ETHTX_EN	GPIO29/CHIP_ID3/ETHTX_EN	TX_EN	TX_EN	—
HD55/D55/ETHTX_ER/NC	GPIO4/TDM3TCLK/$\overline{\text{IRQ2}}$/ETHTX_ER/NC	TX_ER	—	—
HD56/D56/ETHRX_DV/ ETHCRS_DV	GPIO10/TDM2TCLK/$\overline{\text{IRQ8}}$/ETHRX_DV/ ETHCRS_DV[3]	RX_DV	CRS_DV	—
HD57/D57/ETHRX_ER	GPIO11/TDM2TDAT/$\overline{\text{IRQ9}}$/ETHRX_ER/ETHTXD	RX_ER	RX_ER	TXD
HD58/D58/ETHMDC	GPIO13/TDM2RCLK/$\overline{\text{IRQ11}}$/ETHMDC	MDC	MDC	MDC
HD59/D59/ETHMDIO	GPIO9/TDM2SYN/$\overline{\text{IRQ7}}$/ETHMDIO	MDIO	MDIO	MDIO
HD60/D60/ETHCOL	GPIO8/TDM3RDAT/$\overline{\text{IRQ6}}$/ETHCOL/NC	COL	—	—
Non-Multiplexed/Dedicated Signals				
ETHRX_CLK/NC/ETHSYNC_IN		RX_CLK	—	SYNC_IN
ETHTX_CLK/ETHREF_CLK/ETHCLOCK		TX_CLK	REF_CLK	CLOCK

Table 25-4. Ethernet Signals

DSI/System Data Bus Multiplexing[1]	GPIO/TDM Signal Multiplexing[2]	MII Standard Name	RMII Standard Name	SMII Standard Name
ETHCRS/NC/ETHRXD		CRS	—	RXD

Note: 1 When the Ethernet signals are enabled for multiplexing with the DSI/system bus, the remaining signal lines (HD[32–39]/D[32–39]. HD[44–45]/D44–45], HD[50–53]/D[50–53], and HD[61–63]/D[61–63]) are reserved. Only MII and RMII signals can be multiplexed on the DSI/system bus.

2. When the Ethernet signals are enabled for multiplexing with the GPIO/TDM signals, the actual signals used depend on the type of Ethernet interface selected. An MII uses all the defined signals, leaving two of the TDM interfaces (TDM0 and TDM1) available. RMII and SMII use only signals from TDM2, leaving TDM0, TDM1, and TDM3 available.

3. This signal must not be connected when the device is in SMII mode.

Table 25-6. Ethernet Controller Interface Signals In MII Mode

Signal Name	Type	Description	Reset State
ETHTXD[3–0]	O	**MII Transmit Data Bits 3–0**	0
ETHTX_EN	O	**MII Transmit Enable**	0
ETHTX_CLK	I	**MII Transmit Clock** The ETHTX_CLK is a 25/2.5 MHz clock to support 100/10 Mbps data rate operations. ETHTX_CLK is a continuous transmit clock that provides the timing reference for the transfer of the ETHTX_EN, ETHTXD, and ETHTX_ER signals from the MAC to the PHY. In MII mode, the ETHTX_CLK is sourced by the PHY.	—
ETHTX_ER	O	**MII Transmit Error**	0
ETHCOL	I	**MII Collision Detect**	—
ETHRX_DV	I	**MII Receive Data Valid**	—
ETHRXD[3–0]	I	**MII Receive Data Bits 3–0**	—
ETHRX_ER	I	**MII Receive Error**	—
ETHRX_CLK	I	**MII Receive Clock** A continuous clock (2.5, 25 MHz) that provides a timing reference for ETHRX_DV, ETHRXD, and ETHRX_ER.	—
ETHRX_CRS	I	**MII Carrier Sense**	—
ETHMDC	O	**Management Clock** The clock of the management interface. ETHMDIO should be synchronized with this clock (typically 2.5 MHz) which is supplied by the MAC. The **IEEE** standard sets the minimum period at 400 ns or 2.5 MHz, but the device can be configured as fast as 12.5 MHz, if the PHY supports that speed.	—
ETHMDIO	I/O	**Management Data** ETHMDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during management write cycles.	—

Table 25-7. Ethernet Controller Interface Signals In RMII Mode

Signal Name	Type	Description	Reset State
ETHTXD[1–0]	O	**MII Transmit Data Bits 1–0**	0
ETHTX_EN	O	**MII Transmit Enable**	0
ETHREF_CLK	I	**RMII Synchronous Clock Reference, REF_CLK** The ETHREF_CLK is a 50 MHz Clock. and is defined as a synchronous clock reference for receive, transmit, and control.	—
CRS_DV	I	**Carrier Sense/Receive Data Valid**	—
ETHRXD[1–0]	I	**MII - Receive Data Bits 3–0**	—
ETHRX_ER	I	**MII Receive Error** Use of this signal is optional.	—
ETHMDC	O	**Management Clock** This output signal is the clock of the management interface. ETHMDIO should be synchronized with this clock (typically 2.5 MHz) which is supplied by the MAC. The **IEEE** standard sets the minimum period at 400 ns or 2.5 MHz, but the device can be configured as fast as 12.5 MHz, if the PHY supports that speed.	—
ETHMDIO	I/O	**Management Data** ETHMDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during management write cycles.	—

Table 25-8. Ethernet Controller Interface Signals In SMII Mode

Signal Name	Type	Description	Reset State
ETHSYNC	O	**Transmit Synchronization Control Signal** Used in SMII MAC-to-PHY (Sync Out) mode. This signal is not used in MAC-to-MAC (Sync In) mode.	0
ETHTXD	O	**Transmit Data**	0
ETHRXD	I	**Receive Data**	—
ETHSYNC_IN	I	**Receive Synchronization Control Signal** Used in SMII MAC-to-MAC Sync In mode. This signal is not used in Sync Out mode.	—
ETHCLOCK	I	**Clock** Global 125MHz reference clock.	—
ETHMDC	O	**Management Clock** The clock of the management interface. ETHMDIO should be synchronized with this clock (typically 2.5 MHz) which is supplied by the MAC. The **IEEE** standard sets the minimum period at 400 ns or 2.5 MHz, but the device can be configured as fast as 12.5 MHz, if the PHY supports that speed.	—
ETHMDIO	I/O	**Management Data** Inputs PHY-supplied status during management read cycles and outputs control during management write cycles.	—

25.6 Ethernet Controller Interfaces

The following sections give a detailed functional description of the supported interfaces.

25.6.1 MII

The MSC8113 Ethernet controller MII receive and transmit modules comply with the **IEEE** Std. 802.3.

25.6.1.1 MII Transmit Flow

The Ethernet controller drives the transmit enable (ETHTX_EN) output signal, the ethernet transmit (ETHTXD[3–0]) output data bus, and the ethernet transmit error (ETHTX_ER) output signal on the rising edge of the ethernet transmit (ETHTX_CLK) input clock. The Ethernet controller uses ETHTX_EN to indicate valid data on ETHTXD[3–0]. When ETHTX_EN is deasserted, the Ethernet controller drives 0s on ETHTXD[3–0]. When ETHTX_EN is asserted, the Ethernet controller drives the preamble (0b0101) nibbles on each rising edge of ETHTX_CLK. The number of driven preambles is configured by writing to MIICFG2R[PREAL] (see page 25-85). After driving the specified number of preambles, the Ethernet controller drives one byte of SFD (0b01011101) and then the data. Four CRC bytes are appended according to the values in MACCFG2R[PADCRC] and MACCFG2R[CRCEN] (see page 25-85). ETHTX_EN is deasserted at the rising edge of the ETHTX_CLK following the Ethernet controllers transmits the last nibble of the frame. In Full-Duplex mode, both the carrier sense (ETHCRS) input signal and collision (ETHCOL) input signal from the PHY are ignored. For transmission in Half Duplex Mode details, see **Section 25.7**. **Figure 25-5** shows the Ethernet controller MII transmit flow in Full Duplex Mode.

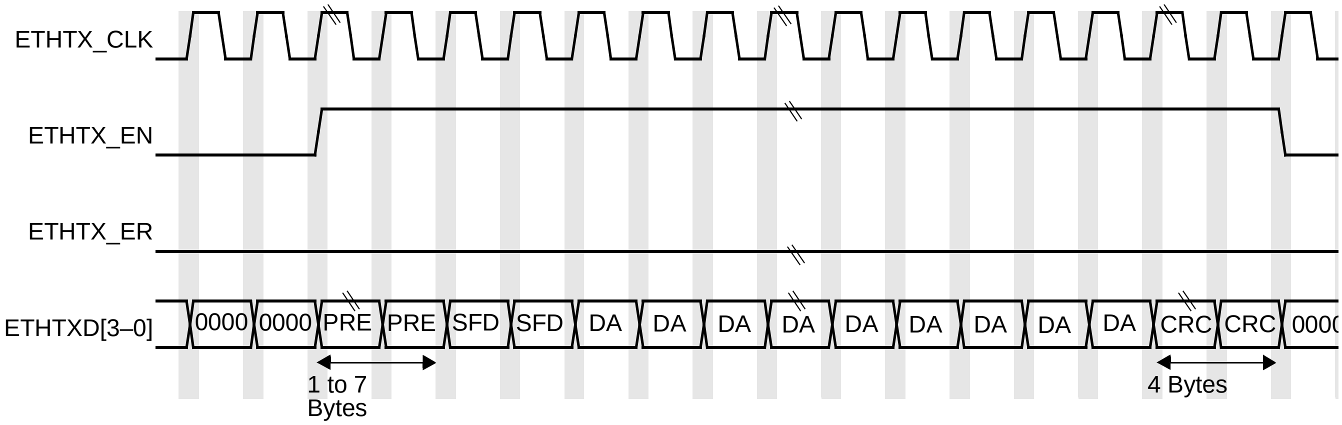


Figure 25-5. MII Transmit Flow in 100 Mbps Full Duplex Mode with No Error

25.6.1.2 MII Receive Flow

The Ethernet controller samples the receive data valid (ETHRX_DV), receive data bus (ETHRXD[3–0]) and receive error (ETHRX_ER) input signals at the rising edge of the receive clock (ETHRX_CLK). When a packet or frame is transmitted to the Ethernet controller, ETHRX_DV indicates that recovered and decoded nibbles are being transmitted on ETHRXD[3–0]. For a received frame to be correctly interpreted, ETHRX_DV must be asserted while the value on the ETHRXD[3–0] is 1 or more bytes of preamble (defined as 0b01010101) followed by 1 byte of start frame delimiter (SFD) (defined as 0b01011101). After the Ethernet controller detects SFD, ETHRX_DV must stay asserted until the Ethernet controller receives the last nibble of data. **Figure 25-6** shows the Ethernet controller MII transmit flow in Full Duplex Mode.

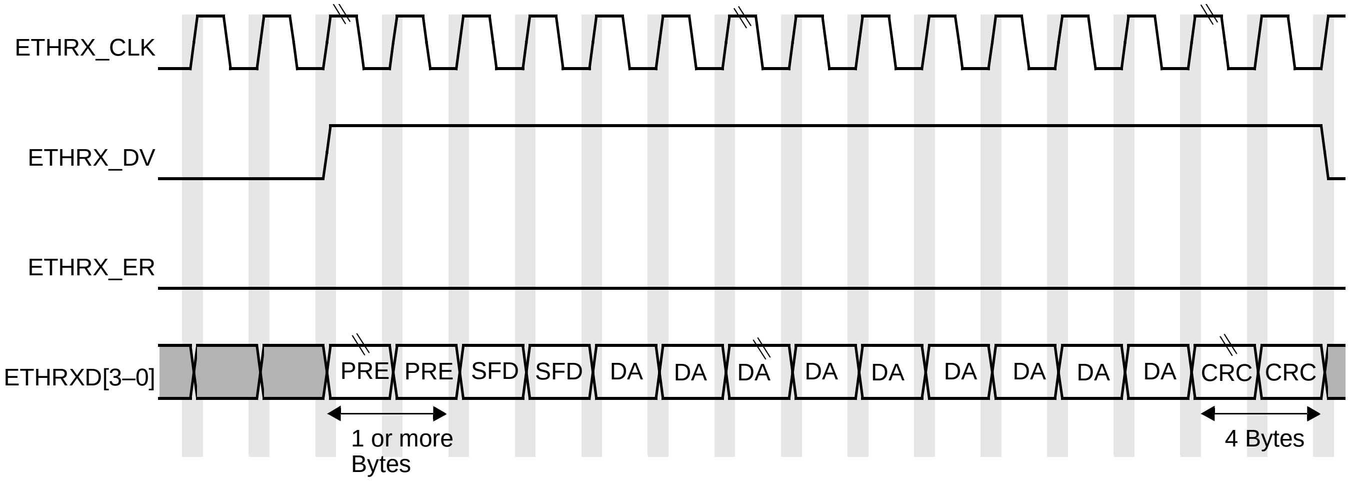


Figure 25-6. MII Receive Flow in 100 Mbps Full Duplex Mode with No Error

25.6.2 RMII

The MSC8113 Ethernet controller RMII receive and transmit interface complies with the RMII specification defined by the RMII consortium.

25.6.2.1 RMII Transmit Flow

The Ethernet controller drives the transmit enable (ETHTX_EN) output signal and the ethernet transmit (ETHTXD[1–0]) output data bus on the rising edge of the input ethernet transmit (ETHREF_CLK) input clock. The Ethernet controllers uses ETHTX_EN to indicate that it is driving valid data on ETHTXD[1–0]. When ETHTX_EN is deasserted, the Ethernet controller drives 0s on ETHTXD[1–0]. When ETHTX_EN is asserted, the Ethernet controller drives 2-bit preambles (0b01) on each rising of ETHREF_CLK. The number of preambles is configured by writing to MIICFG2R[PREAL] (see page 25-85). After the preambles, the Ethernet controller drives one SFD byte (0b01011101) followed by the data. Four CRC bytes are appended according to MACCFG2R[PADCRC] and MACCFG2R[CRCEN. ETHTX_EN is deasserted at the rising edge of ETHREF_CLK after the last two bits of the frame are transmitted.

In Half Duplex Mode, asserting the carrier sense receive data valid (CRS_DV) input signal together with ETHTX_EN indicates a collision. The behavior of the Ethernet controller in a collision state is the same as in MII mode (see **Section 25.7.1**). **Figure 25-7** and **Figure 25-8** show the Ethernet controller RMII transmit flow in Full Duplex mode.

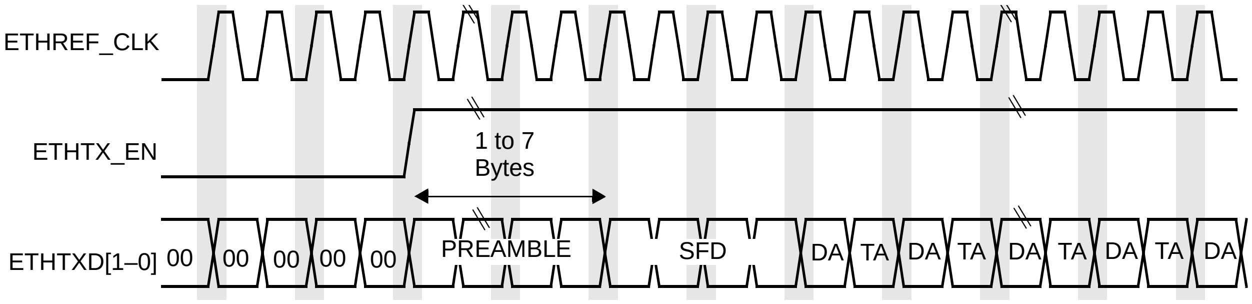


Figure 25-7. Start Of Frame In RMII Mode 100 MBPS Full Duplex

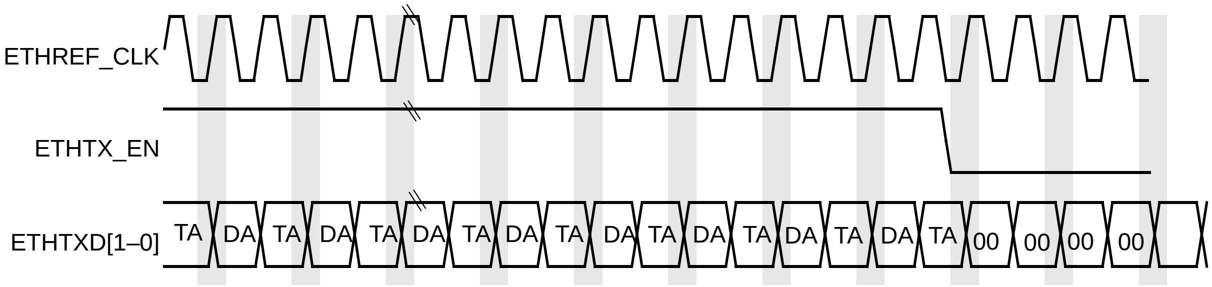


Figure 25-8. End Of Frame In RMII Mode 100 MBPS Full Duplex

25.6.2.2 RMII Receive Flow

The Ethernet controller samples receive data valid (CRS_DV), receive data bus (ETHRXD[1–0]), and receive error (ETHRX_ER) input signals on the rising edge of the receive clock (ETHREF_CLK). When a packet frame is transmitted, CRS_DV is used to indicate that the line is not idle. To allow the Ethernet controller to interpret a received frame correctly, CRS_DV must be asserted while ETHRXD[1–0] drive the specified number of preambles (0b01010101) followed by the start frame delimiter (SFD) (0b01011101). After the Ethernet controller detects SFD, ETHRX_DV must stay asserted until the last nibble of data is received by the Ethernet controller. end-of-frame is indicated by the deassertion of CRS_DV for two continuous ETHREF_CLK cycles. **Figure 25-7** and **Figure 25-8** show the Ethernet controller RMII transmit flow in Full Duplex mode.

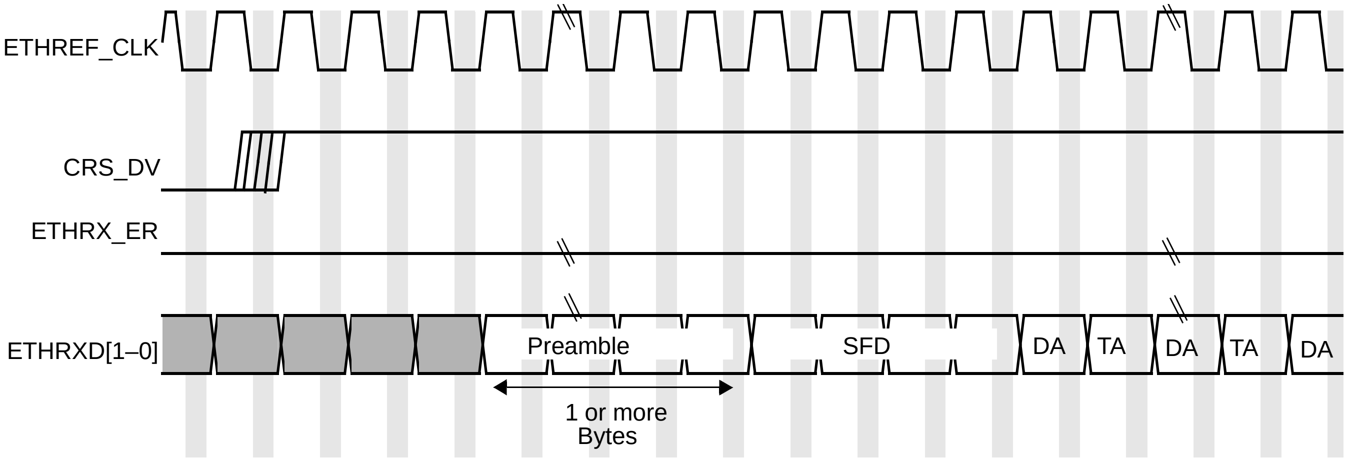


Figure 25-9. Start Of Receive Frame in RMII Mode

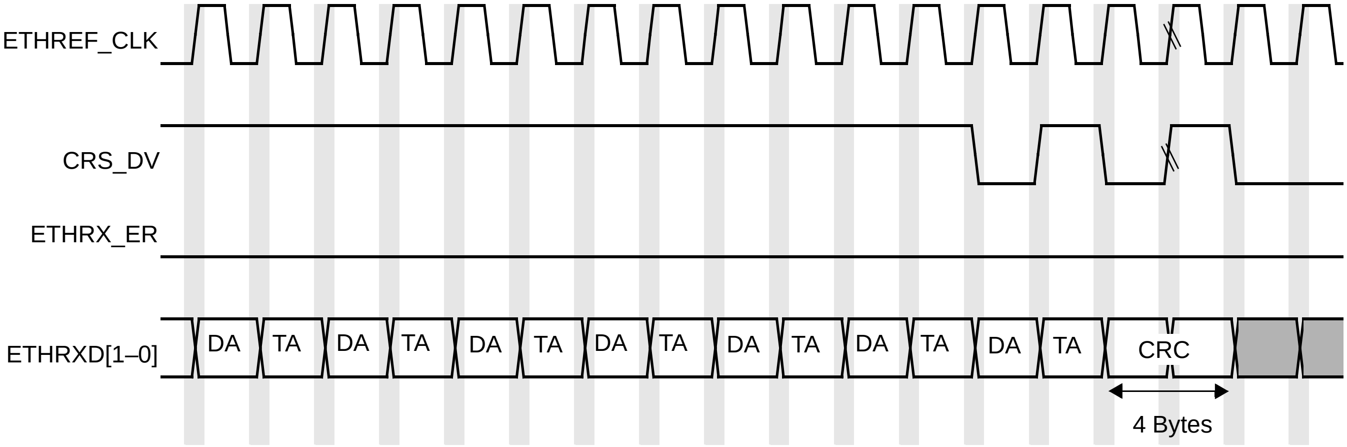


Figure 25-10. End Of Receive Frame in RMII Mode

25.6.3 SMII

The MSC8113 Ethernet controller SMII receive and transmit interface complies with the Cisco serial MII specification.

25.6.3.1 SMII Transmit Flow

The Ethernet controller drives the transmit data (ETHTXD) output signal and the SYNC output signal on the rising edge of the input clock (ETHCLOCK). It continuously transmits 10-bit segments and asserts SYNC when the first bit is driven on ETHTXD. The transmission error (TX_ER) bit is the first bit in the segment. The value of TX_ER is 1 if there a transmission error. The transmit enable (TX_EN) bit is the second bit in the segment. The value of TX_EN is 1 if the segment is a valid data transmission. Bits 3–10 of the segment are the data bits. When there is no valid frame transmission, the Ethernet controller transmits 10-bit segments with TX_EN equal to 0 and the 8 data bits equal to the inter-frame bit (IFB) value configured in the MIIGSK_TIFBR (see page 25-99).

When there is a valid frame transmission, the Ethernet controller transmits one or more 10-bit segments, as follows:

- Preamble: TX_ER, TX_EN = 1, 0b01010101
- SFD: TX_ER, TX_EN = 1, 0b01010111
- Multiple segments each convey one byte of data in the form: TX_ER, TX_EN = 1, data byte
- 4 CRC bytes can be appended according to MACCFG2R PADCRC/CRCEN fields (page 25-85).

In Half Duplex mode, if the carrier sense control bit (CRS) in the received segment and the TX_EN control bit are both 1, a collision has occurred. The behavior of the Ethernet controller in the collision state is the same as the MII mode (see **Section 25.7.1**).

Note: In SMII SYNC In mode, the Ethernet controller starts transmitting IFG segments on the third receive SYNC (ETHSYNC_IN) signal after it is enabled.

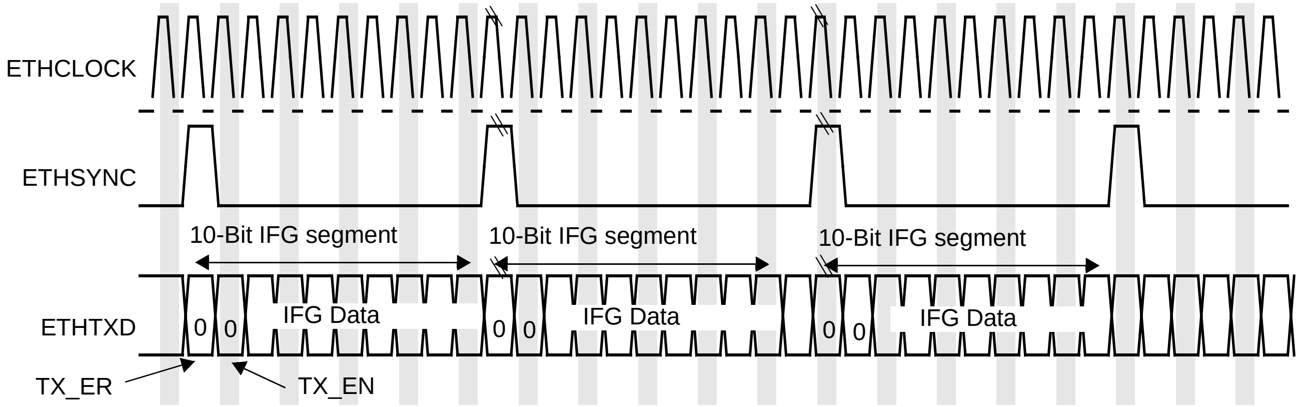


Figure 25-11. Transmission Flow with No Errors IFG state in SMII Mode (100 Mbps)

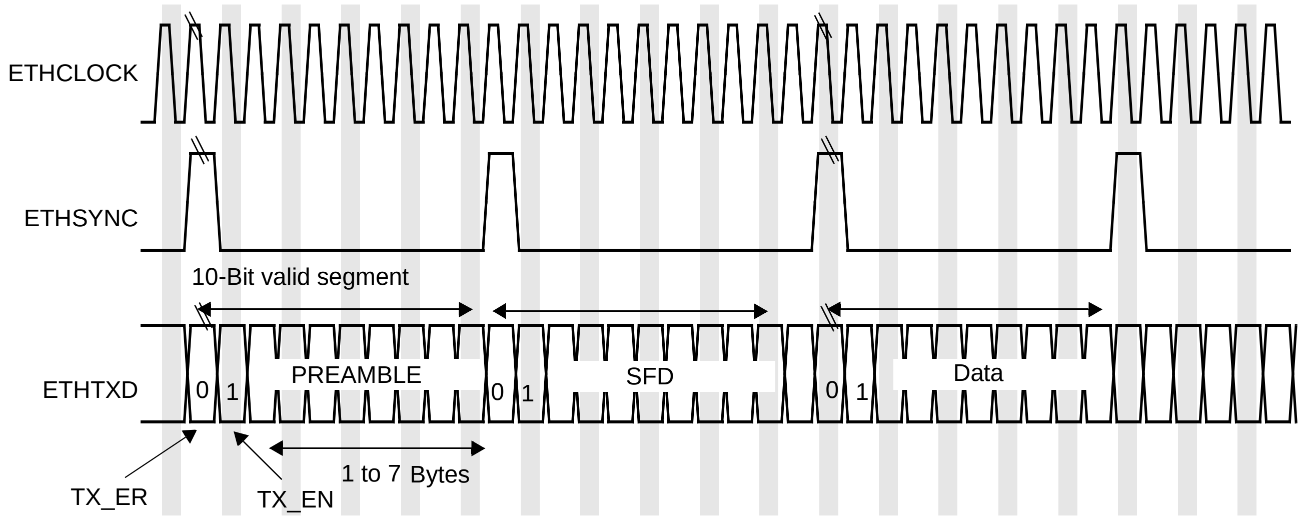


Figure 25-12. Transmission Flow with No Errors Start Of Valid Frame SMII SYNC Mode (Full Duplex 100 Mbps)

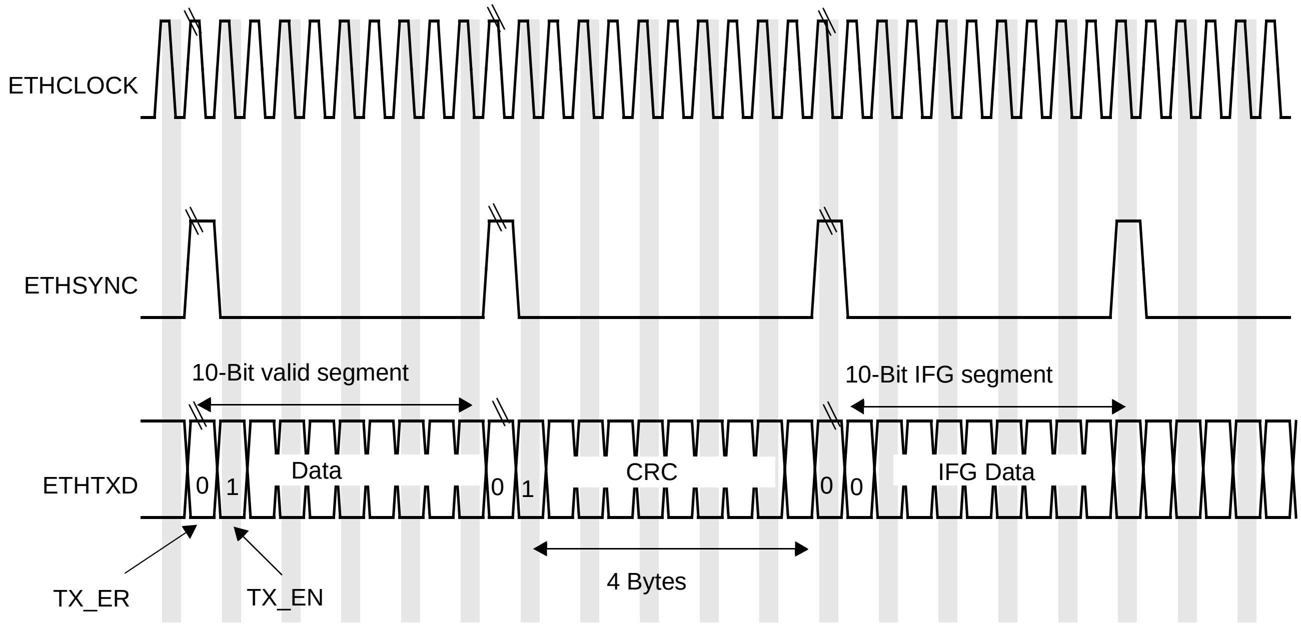


Figure 25-13. Transmission Flow with No Errors End Of Frame in SMII SYNC Mode (Full Duplex 100 Mbps

25.6.3.2 SMII Receive Flow Mode

The Ethernet controller samples the receive data ETHRXD input signal at the rising edge of the receive clock (ETHCLOCK). Each segment of received data must start synchronously to the SYNC output signal. The first bit of the received segment indicates that carrier sense is asserted (ETHCRS). The Ethernet controller uses the second bit in the received 10-bit segment to determine whether the received segment is a valid segment of data or an inter-frame segment. Bits 3–10 of the received segment are the data bits. To interpret a received frame correctly, the received 10 bit segment must have the following in order:

- Preamble: CRS, RX_DV = 1, 0b01010101
- SFD: CRS, RX_DV = 1, 0b01010111
- Multiple 1-byte data segments: CRS, RX_DV = 1, data byte

Note: In SMII SYNC In mode, the Ethernet controller is synchronized on the receive SYNC signal. The receive and transmit operation proceeds according to the input sync signal, ETHSYNC_IN, which indicates the start of a new segment.

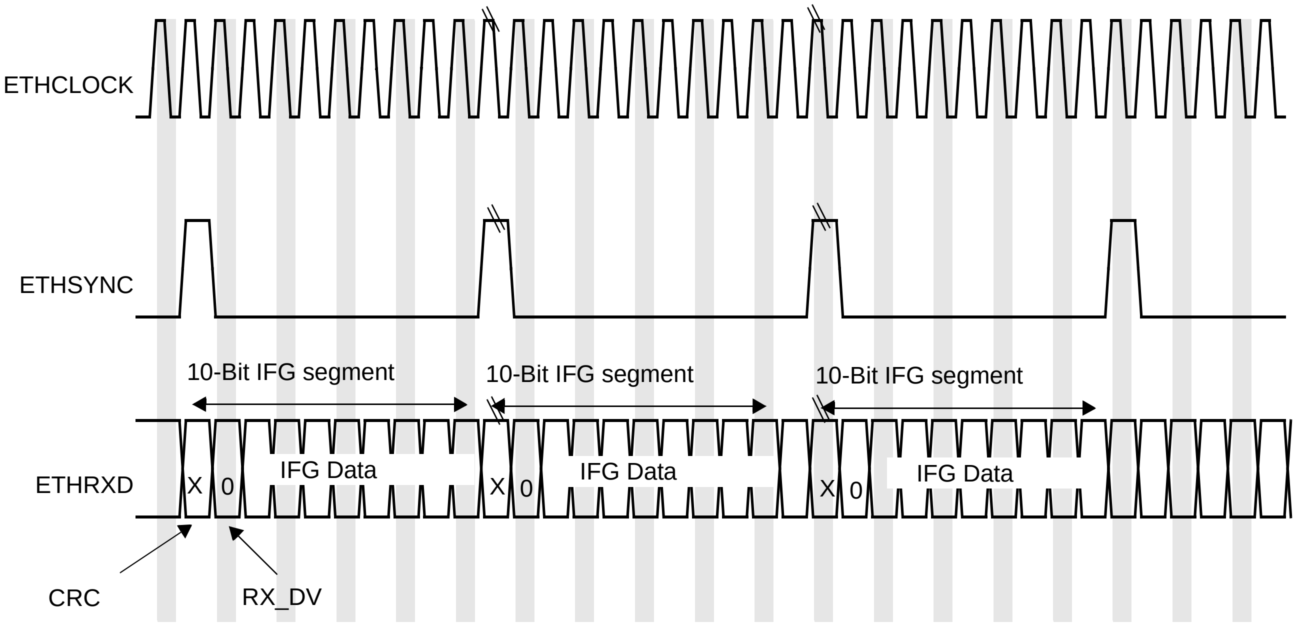


Figure 25-14. Receive Flow with No Errors IFG State in SMII Mode (100 Mbps)

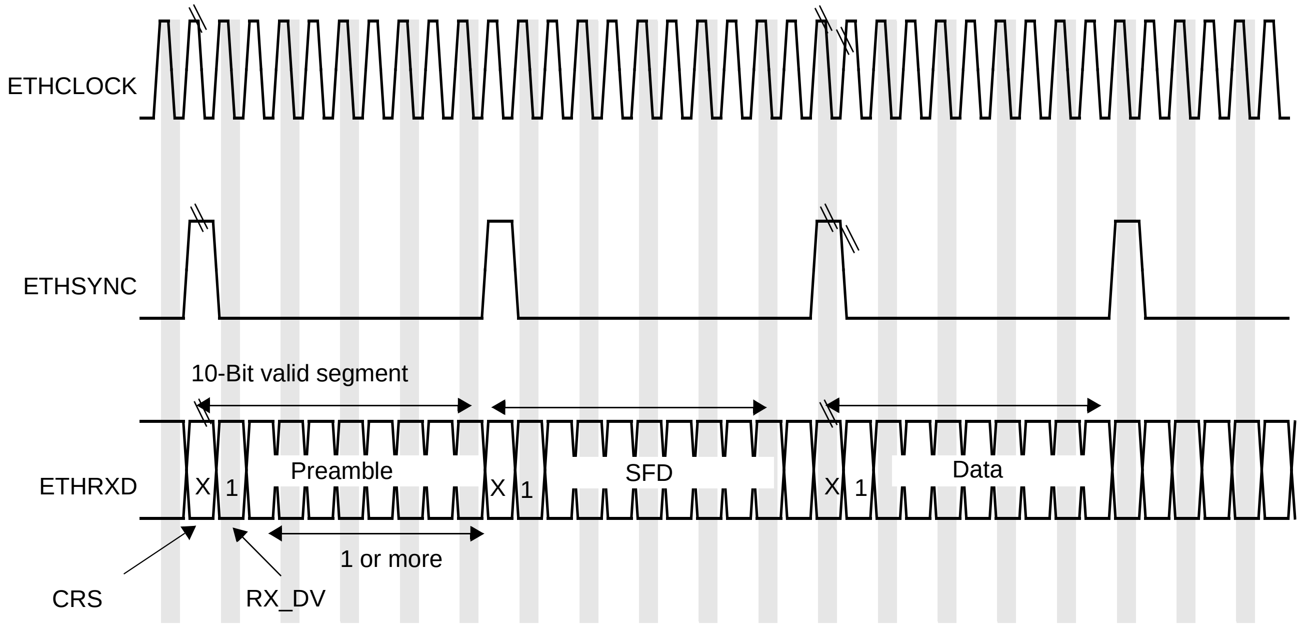


Figure 25-15. Receive Flow with No Errors Start Of Valid Frame SMII Mode (100 Mbps)

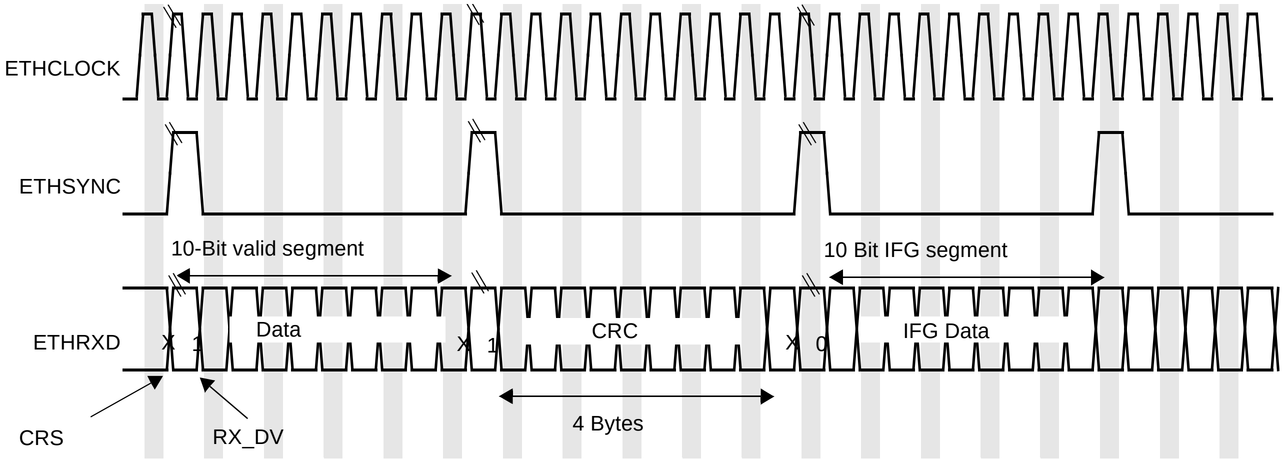


Figure 25-16. Receive Flow with No Errors End Of Valid Frame SMII Mode (100 Mbps)

25.7 MAC Control of CSMA/CD

The Half-Duplex Register (HAFDUPR) controls the carrier-sense multiple access/collision detection (CSMA/CD) logic (page 25-87). Half-duplex is supported for both 10 Mbps and 100 Mbps operation. After the packets are transmitted, the device begins timing the inter-packet gap (IPG) as programmed in the Inter-Packet Gap/Inter-Frame Gap Register (IPGIFGR) (see page 25-86). The system is then free to begin another frame transfer.

In Full-Duplex mode, both the carrier sense (ETHCRS) and collision (ETHCOL) signals from the PHY are ignored, but in Half-Duplex mode, the Ethernet controller defers to ETHCRS and, after a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third ETHCRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if ETHCRS is sensed. During the final one-third of the IPG, ETHCRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

25.7.1 Handling Packet Collisions

In Half-Duplex mode, the Ethernet controller is sensitive to ETHCOL. If a collision occurs, it aborts the packet and sends the 32-bit jam sequence. The jam sequence inverts several bits of the CRC to guarantee an invalid CRC upon reception. A collision signal is sent to the system to request retransmission of the start of the frame. The Ethernet controller then stops the transfer for a time determined by the truncated binary exponential back-off (BEB) algorithm. The delay time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly-distributed random integer *r* in the range:

$$0 < or = r < or = 2^k \text{ where } k = \min(n,10).$$

After this back-off time, the packet is retried.

Note: You can configure the Half-Duplex Register (HAFDUPR) to skip the back-off time. However, this is not a standard operation and it must be used carefully.

After the first collision, the Ethernet controller backs off either 0 or 1 slot times. After the fifth collision, the Ethernet controller backs off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back off is 1024. This can be adjusted via the HAFDUPR. An alternate truncation point, such as 7, can be programmed. On the average, the MAC is more aggressive after seven collisions than other stations on the network. If any packet has excessive collisions, it is aborted. The controller flushes the frame and moves to the next packet in line. If the system requests to send a packet while the Ethernet controller is deferring to a carrier, the Ethernet controller simply waits until the end of the carrier event plus the IPG timing before it fulfills the request.

25.7.2 Controlling Packet Flow

Packet flow can be handled in several ways. A default retransmit attempt limit of 15 can be reduced via the HAFDUPR. The slot time or collision window can gate the retry window and possibly reduce the transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time starts at the beginning of the packet, the end occurs around byte 56 of the frame data. Full-duplex flow control is covered in **IEEE** Std. 802.3x™. The standard does not address flow control in half-duplex environments. However, the concept of back pressure is common in the industry. The Ethernet controller implements the optional back pressure mechanism using the raise carrier method, if the system receive logic can stop the reception of packets in a network-friendly way by setting the transmit half-duplex flow control (TCTRL[THDF]). If the medium is idle, the Ethernet controller raises the carrier by transmitting the preamble. Other stations on the half-duplex network then defer to the carrier.

If the preamble transmission causes a collision, the Ethernet controller ensures the minimum 96-bit presence on the wire, drops the preamble, and waits a back-off time depending on the value of the (half-duplex) BPNB bit. These transmitting-preamble-for-back-pressure collisions are not counted. If BPNB is set, the Ethernet controller waits an inter-packet gap before resuming the preamble transmission and does not defer. If BPNB is cleared, the Ethernet controller adheres to the truncated BEB algorithm so that packets can be received. This also can be detrimental in that packets may now experience excessive collisions, causing them to be dropped in the stations from which they originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, BPNB must be set. The Ethernet controller periodically drops the carrier (ceases transmitting preamble) to avoid excessive defer conditions in other stations on the shared network. It does not defer when attempting to send packets in back pressure. Back pressure is nonstandard, yet it can be effective in reducing the receive packet flow.

25.7.3 Controlling PHY Links

Control and status to and from the PHY is provided via the two-wire MII management interface described in **IEEE** Std. 802.3u™. The MII management registers exercise this interface between a host processor and one or more PHY devices The Ethernet Controller MII registers provide the ability to perform continuous read cycles (called a scan cycle) even though scan cycles are not explicitly defined in the standard. When requested (by setting MIIMCOMR[SCYC]), the MSC8113 performs repetitive read cycles of the PHY status register, for example. This method allows link characteristics to be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) indicate availability of each read of the scan cycle to the host from MIIMSTATR[PHYS]. The length of the MII management interface preamble can also be modified through the MII registers. After establishing that a PHY supports preamble suppression, the host may configure the Ethernet controller accordingly. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface. See **Section 25.17.6**, *MII Management Registers*, on page 25-92 for details.

25.8 RMON Support

The Ethernet controller automatically gathers network statistics required for RMON without needing to receive all addresses. The following are supported:

- RMON MIB group1
- RMON MIB group2
- RMON MIB group 3
- RMON MIB group 9
- RMON MIB2
- 802.3 Ethernet MIB

For RMON statistics and their corresponding counters, see **Section 25.17.8**, *RMON Management Information Base (MIB)*, on page 25-105.

25.9 Frame Recognition

The Ethernet controller performs frame recognition in two ways: pattern matching or destination address. A frame can be rejected or accepted on the basis of the outcome of destination address filtering, pattern matching filtering, or both.

25.9.1 Pattern Matching Recognition

You can program the Ethernet controller to filter received frames based on pattern matches within the first 256 bytes of the frame. Pattern matches are used to accept or reject a frame, and/or to perform special processing of a frame if an address match is detected. When the controller matches a pattern, it examines the associated attributes and rejects, accepts, or conditionally accepts a frame. Conditional acceptance occurs for a pattern match in which the accept decode (PCNTRL*n*[PMC]) is 10 and the continue search enable bit, PCNTRL*n*[CSE], is also set. You can program the Ethernet controller to continue searching past a successful pattern match by setting PCNTRL*n*[CSE]. However, once a sample pattern is not matched, the frame is discarded immediately, regardless of the PCNTRL*n*[CSE] value.

Figure 25-17 shows a flowchart for the frame acceptance/rejection process based on pattern matching. For an incoming frame, the Ethernet controller first searches for a pattern match from preselected patterns. Upon a match, the logic determines whether the continue search enable PCNTRL*n*[CSE] bit is set. If this bit is not set, the frame is accepted and the Ethernet controller takes the action indicated by the corresponding attributes register (see the PATTRB*n* register on page 25-135). If this bit is set, the Ethernet controller continues to search for the next pattern match. If it does not match the pattern, the controller rejects the frame. If it finds a pattern match with CSE cleared, it accepts the frame without any further search. If a subsequent pattern match is found but CSE is set, the search continues. This process continues until the end of frame (EOF) or until the first 256 bytes are received.

If the end of a pattern search is reached and no acceptance/rejection is established, the Ethernet controller performs a test for reject all. This test allows some rejection capability on some non-Ethernet type frames. If RCTRL[RA] is not set, the Ethernet controller is left with destination address matching as the only criterion for filtering frames (see **Figure 25-17**). If the reject all bit is set, the frame is discarded.

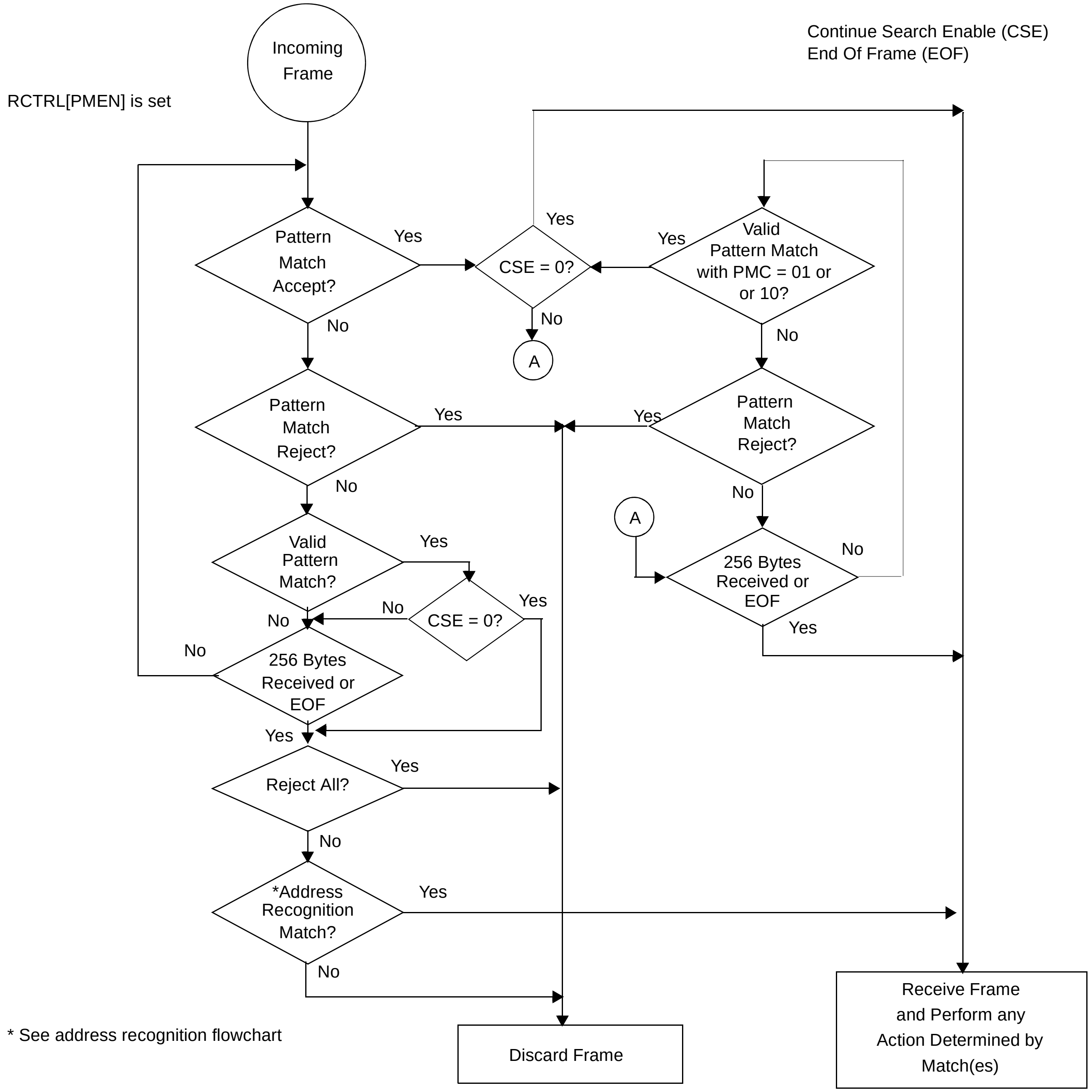

Continue Search Enable (CSE)
End Of Frame (EOF)
Incoming Frame
RCTRL[PMEN] is set
Pattern Match Accept?
Yes
No
CSE = 0?
Yes
No
A
Valid Pattern Match with PMC = 01 or or 10?
Yes
No
Pattern Match Reject?
Yes
No
Pattern Match Reject?
Yes
No
A
Valid Pattern Match?
Yes
CSE = 0?
No
Yes
No
256 Bytes Received or EOF
No
Yes
256 Bytes Received or EOF
No
Yes
Reject All?
Yes
No
*Address Recognition Match?
Yes
No
* See address recognition flowchart
Discard Frame
Receive Frame and Perform any Action Determined by Match(es)

Figure 25-17. Frame Acceptance/Rejection Flowchart

25.9.2 Destination Address Recognition

The Ethernet controller filters frame using the traditional destination address recognition methods, with the following requirements:

- Reject all must not be enabled (that is, RCTRL[RA] is cleared).
- There should be no pattern matches with a decode of (PCNTRL*n*[PMC] = 10 or 11).

If a specific pattern has PCNTRL*n*[PMC] = 10 or 11, destination address filtering still occurs if that specific pattern is not matched. With address recognition filtering, patterns are set up to identify only frames that need special processing (filing to a specified queue). If a frame is accepted due to destination address filtering and no pattern match occurs, the Ethernet controller uses the default queue specified in DATTR as the destination queue.

The Ethernet controller identifies the type of the address. Hash table filtering, for example, can be implemented with ease. The addresses are classified as physical (individual), group (multicast), broadcast (all-ones group address), and promiscuous. The difference between an individual address and a group address is determined by the I/G bit in the destination address field. Additional fields in the frame can be searched for a pattern match. **Figure 25-18** shows a flowchart for address recognition on received frames. In the actual implementation, most decision points shown in the figure occur simultaneously.

The Ethernet controller compares the destination address field of the received frame with the physical address that you program in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the destination address does not match the station address, the controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. You must write zeros to the hash to avoid a hash match and ones to the station address to avoid an individual address match, or you can turn on Promiscuous mode (see page 25-78).

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast, and broadcast addresses are enabled, the frame is conditionally accepted and further activity on the frame is determined by pattern matching. If the group address is not a broadcast address, you can perform address recognition on multiple group addresses using the GADDR*n* hash table. In Promiscuous mode, the Ethernet controller receives all of the incoming frames, regardless of their address, but performs further processing with pattern matching. Therefore, in Promiscuous mode the Ethernet controller allows all but a few specific addresses to be received.

Address recognition allows the Ethernet controller to use the traditional destination address selection criteria to determine which frames to keep and which to reject. Because pattern matching takes place in the background even in address recognition mode, the controller can use the pattern matching to implement features such as selectively storing some frames in one queue and others in a different queue even if they had the same destination address. The pattern matching capabilities are explained in detail in **Section 25.10.2**, *Receive Frame Processing with Pattern Matching*, on page 25-30.

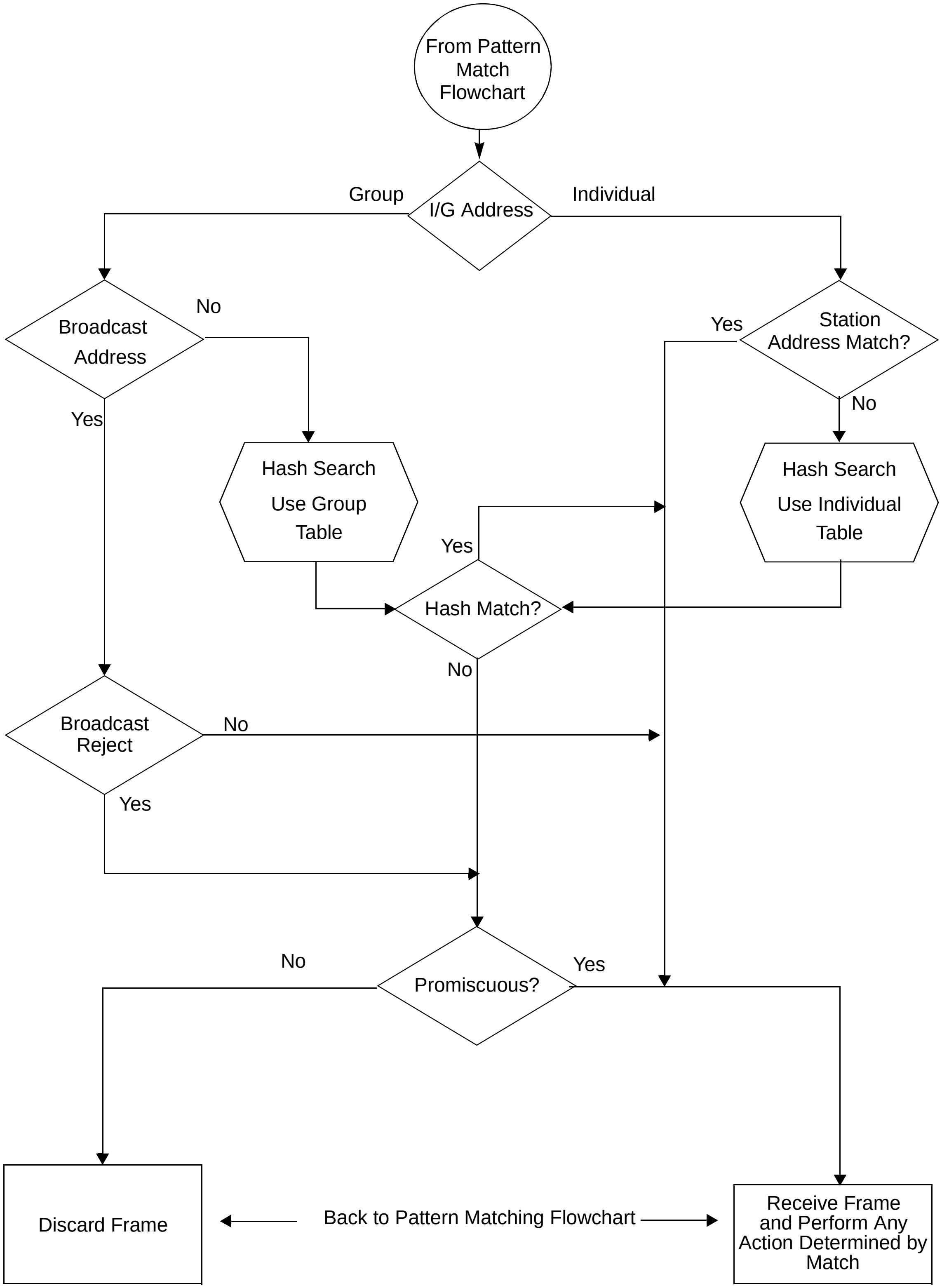

From Pattern Match Flowchart
I/G Address
Group
Individual
Broadcast Address
No
Yes
Station Address Match?
Yes
No
Hash Search Use Group Table
Hash Search Use Individual Table
Hash Match?
Yes
No
Broadcast Reject
No
Yes
Promiscuous?
No
Yes
Discard Frame
Back to Pattern Matching Flowchart
Receive Frame and Perform Any Action Determined by Match

Figure 25-18. Ethernet Address Recognition Flowchart

25.9.3 Hash Table Algorithm

Using the hash table process for individual and group hash filtering, the Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in the Group Address Registers (GADDR[0–7]) or Individual Address Registers (IADDR[0–7]). The eight high-order bits of a CRC checksum are used to index into the hash table. The high-order three bits of this 8-bit field are used to select one of the eight registers in either the individual or group hash table. The low-order five bits select a bit within the 32-bit register. A value of 0 in the high-order three bits selects IADDR0/GADDR0.

The same process is used if the Ethernet controller receives a frame. If the CRC checksum selects a bit that is set in the group/individual hash table, the frame is conditionally accepted pending the pattern match result. If 32 group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 224/256 (87.5 percent) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses. In addition, if pattern matching is enabled, the Ethernet controller can reduce the burden on the software and further accelerate the reception by performing additional filtering.

Better performance is achieved when the group and individual hash tables are used in combination. For instance, if 32 group and 32 physical addresses are stored in their respective hash tables, because 87.5 percent of all group addresses and 87.5 percent of all individual address are rejected, then 87.5 percent of all frames are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 256-bin hash table increases, most of the hash table bits are set, preventing only a small fraction of frames from reaching memory. Pattern matching can become extremely valuable because it can reduce the extra memory bus usage due to unintended hash table hits.

Note: The hash tables cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. Pattern matching can be used to reject frames with unintended address hits in the hash table.

25.10 Buffer Descriptors

The Ethernet controller stores data in memory regions defined by buffer descriptors (BDs), which are defined as transmit buffers (TxBDs) or receive buffers (RxBDs) that are organized and accessed through BD tables. The Ethernet controller BDs have the following features:

- Configurable BD size via ECNTRL[DBDS].
  - 8-byte BDs are compatible with other Ethernet interfaces, such as the MSC8101, MSC8103, MPC8260, and MPC8560 devices, permitting reusability of existing Ethernet applications from these devices. Features include:
    - Pattern matching reject
    - Filing capability
  - Extended BD mode attribute information includes:
    - Frame status
    - Extended address match results
    - Extended pattern match results
    - Insertion information on the transmitting side
    - Frame data length
    - Insertion with expansion or replacement

The 8-byte BD mode imposes some limitations primarily because of the lack of reporting means (no more fields in the BD to report events). Even with this limitation, significant pattern matching and filing is possible. However, insertions cannot be performed on the transmitting side. Pattern matching can be used to define up to eight more station addresses. Pattern match reject can also be performed in 8-byte mode.

Use the Default Attribute Register to change the default filing queue. These features can reduce the memory space (smaller BD rings) required and increase performance (less DMA read/write required for the 8-byte BD as compared to the 32-byte BD). Using the 32-byte BD, you can customize the automatic insertion of information received and transmitted based on Ethernet address, IP address, and so on, and use this information for filtering, security, or other custom applications. The RxBD contains a summary of the receive frame attributes; including pattern match and address match. The TxBD can be used to modify the transmitted information by inserting customized data.

25.10.1 Data Buffer Descriptor

Data BDs encapsulate all information necessary for the Ethernet controller to transmit or receive an Ethernet frame (see **Figure 25-19**). The BD centralizes status information for the data packet in its status field and contains a data BD pointer to the location of the data buffer. Software sets up the BDs in memory. Because of prefetching, a minimum of two BDs per ring are required. This applies to both the transmit and the receive descriptor rings. Software also points the data pointer to memory. Within the status field is an ownership bit that defines the current state of the buffer (to which the data pointer points). Other bits in the status field of the BD communicate status/control information between the Ethernet controller and the software driver. Because there is no next BD pointer in the transmit/receive BD (see **Figure 25-20**), all BDs must reside sequentially in memory. The Ethernet controller increments the current BD location appropriately to the next BD location to be processed. A wrap bit in the last BD informs the Ethernet controller to loop back to the beginning of the BD chain.

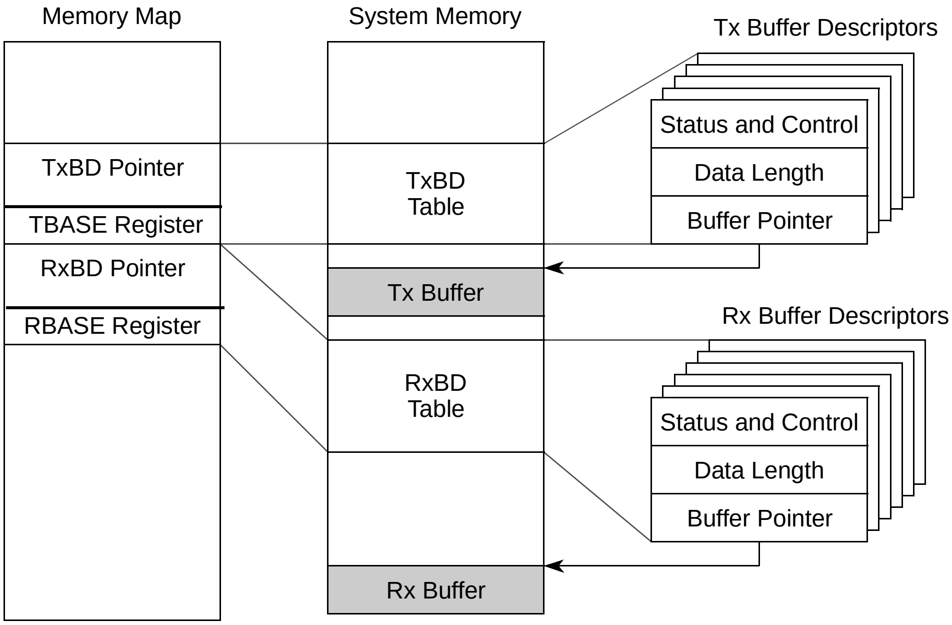


Figure 25-19. Example Memory Structure for an 8-Byte BD

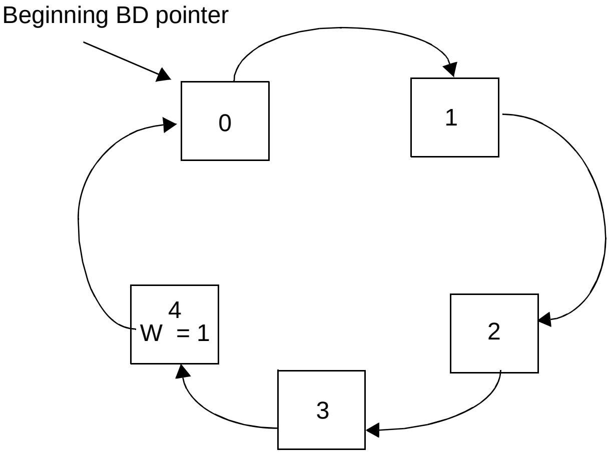


Figure 25-20. Buffer Descriptor Ring

25.10.2 Receive Frame Processing with Pattern Matching

The frame processing with pattern matching includes receive frame filing. The Ethernet controller provides the following powerful features in processing a receive frame:

- Up to eight exact unicast or multicast MAC address matches
- IP address filtering
- Frame filing based on pattern hits
- Frame rejection or acceptance based on pattern hits
- Pattern detection within the first 256 bytes of a frame
- Up to 16 unique 4-byte patterns
- Flexible programmable pattern size from four to 64 bytes
- Programmable match start offset (0 to 252 bytes within the frame depending on pattern size)
- Noncontiguous, concatenated patterns (each bit of a pattern can be individually masked)
- Multi-pattern hit detection

Pattern matching enables you to process receive frames with a set of tools to assist network applications. Features such as filing of frames in queues based on a pattern hit can accelerate post processing of data. You can further enhance address recognition filtering by applying additional processing to frames that pass the destination address check. Flexibility is built into the Ethernet controller pattern matching to give you more control in manipulating receive frames. **Table 25-9** shows an example of how to use the Ethernet controller frame processing features for specific applications.

Table 25-9. Frame Processing Table Example

Pattern Offsets	8-byte Patterns	Operation	Queue
16, 35	P4–P5	File	1
32,55	P8–P9	File	3
16,15	P10–P11	File	2
4,9	P12–P13	Reject	
8,9	P14–P15	Reject	

25.10.3 Receive Pattern Matching Filing

The 16-entry pattern matching table (4-byte patterns p[0–15]) consists of a group of four registers that define whether a pattern match search is to occur, what to search for, where to search, how to process the result of a match if one occurs, and what to do if one does not occur, as follows:

- PCNTRL*n*. The pattern control register that contains the matching index (MI), continue search enable (CSE), concatenated pattern (CP), as well as the pattern match control (PMC) fields.
- PMD*n* and PMASK*n*. 32-bit registers to define the 4-byte patterns the user wants to match. The combination of data and mask registers allows for patterns of any combinations of bits (that is, noncontiguous bit patterns) within the 4-byte patterns.
- Pattern Match Attribute Register (PATTRB*n*). Determines how to process the frame after a pattern match occurs. The pattern match file (PMF) and queue classification (QC) bits are used to select which of the four receive queues stores the frame.

Figure 25-22 shows these registers, their relationship to each other, and how the contents of the registers are mapped to a BD. Pattern matching table entries can be used in several ways:

- Concatenate two entries by setting the appropriate PCNTRL*n*[CP] to set up an exact address match. A maximum of eight additional exact addresses can be set up if all entries are used.
- Concatenate multiple entries to form a larger match pattern. The control and attribute registers (PCNTRL*n* and the PATTRB*n*) associated with the first entry of the concatenated entries is used if a successful match on all entries occurs. The MI field is an exception; it is unique for each 4-byte pattern. For example, for the eight 8-byte pattern configuration, the registers used are PCNTRL0/PATTRB0, PCNTRL2/PATTRB2, PCNTRL4/PATTRB4, PCNTRL6/PATTRB6, PCNTRL8/PATTRB8, PCNTRL10/PATTRB10, PCNTRL12/PATTRB12, and PCNTRL14/PATTRB14 (see **Figure 25-22**).
- Pattern matching can be the only criterion for accepting or rejecting a frame and filing data.
- Entries can be set up so that after an initial match occurs, the search for additional patterns continues, but only if the continue search enable (CSE) bit is set in the hit entry.

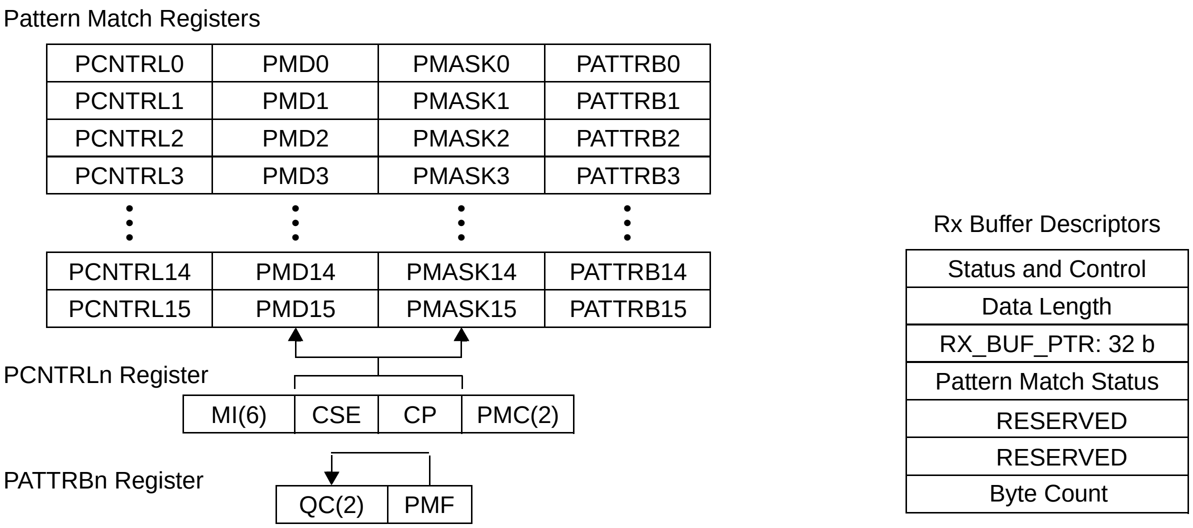


Figure 25-21. Pattern Matching Table

You must be aware of the hardware behavior if one of the following scenarios occurs:

- Pattern matches are processed chronologically in 4-byte increments and can continue until:
  - — 256 bytes or end of frame is reached.
  - — A pattern match hit occurs with its associated CSE bit cleared.
  - — A pattern match hit occurs with its associated pattern match reject bit set.
- If several entries hit within the same 4-byte data boundary, the controls and attributes associated with the lowest numbered entry are used. Thus, the order of the entries (PCNTRL*n*) can be an additional processing priority selector (PATTRB*n*).
- After a pattern match for an entry in which the PCNTRL*n*[PMC] decodes to pattern match reject, all other controls and attributes are ignored and the frame is discarded, regardless of the state of the CSE bit.
- Pattern matching continues if a pattern match hit occurs in which the CSE bit is set and the controls and attributes corresponding to this entry are used if no further match is found.
- Pattern matching stops if there is a pattern match with the CSE bit cleared, and the attributes of this entry are used to process the frame.
- The attributes of the last entry matched are used even when 256 bytes or EOF is reached.

- The following conditions result in the use of destination address recognition as the criterion for accepting or rejecting frames.
  - — RCTRL[PMEN] and RCTRL[RA] are cleared. If a destination address is recognized (individual address or hash table hit), the default attribute register is used to process the frames. No pattern match information is reported.
  - — PCNTRL*n*[PMC] = 01. Because PMC does not decode to accepting or rejecting, this pattern cannot be used for accepting or rejecting the frame. If a pattern match hit occurs and a destination address is recognized, the frame is processed using the attributes of the corresponding PATTRB*n*[PMF] or PATTRB*n*[PME] with the hit. Recall that a PMC = 01 applies only for filing data on a frame that is accepted on the basis of a previous pattern (one with PMC = 10) or on the basis of DA recognition.
  - — RCTRL[PMEN] is set and RCTRL[RA] is cleared. If no pattern match hit occurs but a destination address is recognized, the frame uses the information in the default attributes register (DATTR) to determine how to process the frame.
  - — When RCTRL[PMEN] is set, RCTRL[RA] is cleared, and the PCNTRL*n*[PMC] decodes to Entry Disabled for all entries, destination address recognition determines whether the frame is rejected or accepted (filed in default attribute register).

Figure 25-22 illustrates several examples of different concatenations. The matching search does not have to start at the beginning of the frame. You can begin matching within a 252-byte window from the start of the frame using a 4-byte aligned match index (or offset) in the corresponding PCNTRL*n* register. For example, with a 4-byte pattern, the Ethernet controller can be programmed with a match index of 16 bytes. The first 16 bytes of a frame are disregarded and bytes 17–20 are examined for a match. You can program the MI fields associated with individual 4-byte patterns that make up a concatenated pattern to point to different locations in the frame. In other words, each concatenated pattern can point its MI field anywhere within the 256 bytes of the frame.

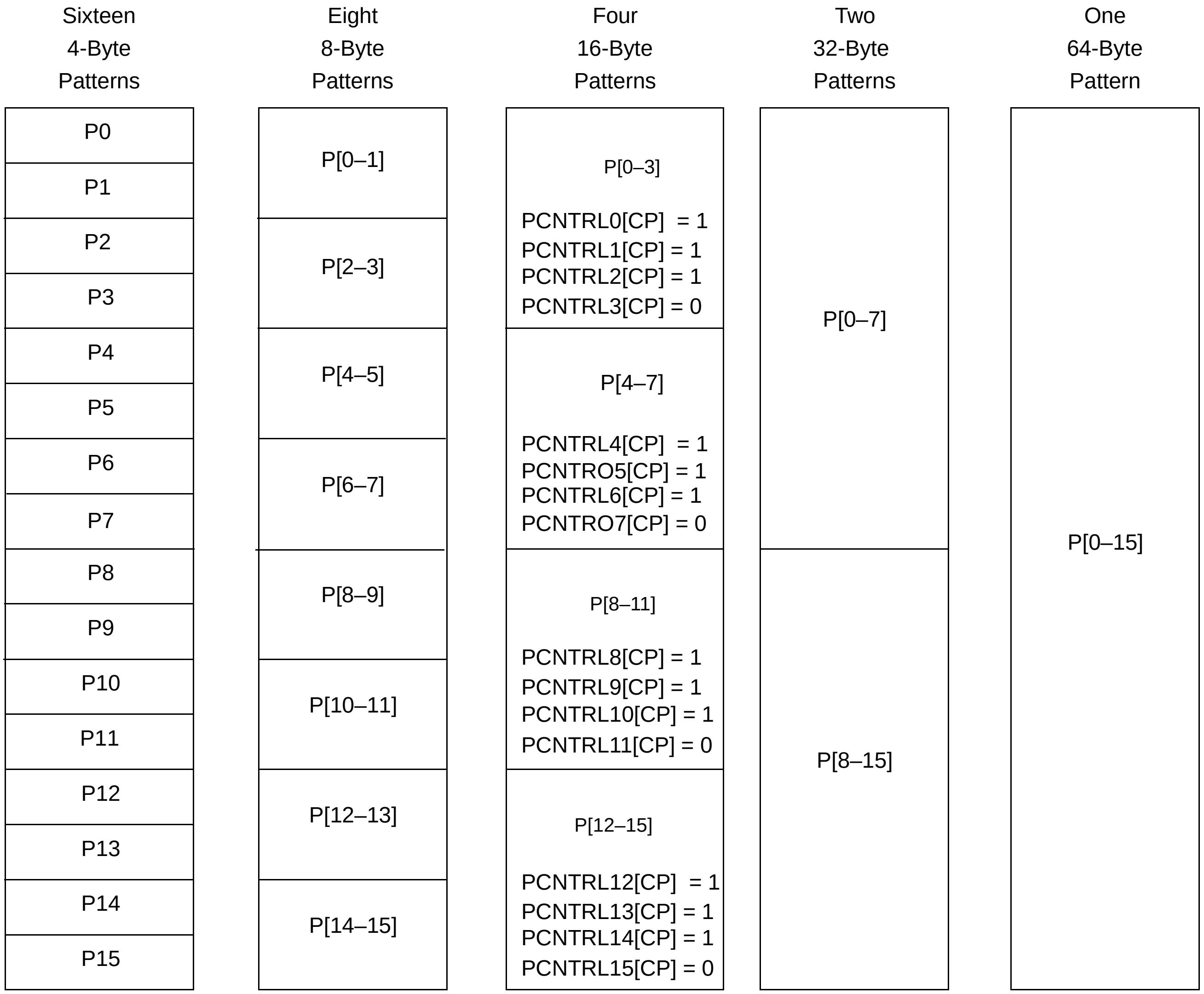


Figure 25-22. Example of Pattern Configurations

Table 25-10 describes the actions taken for different combinations of the pattern match reject (PCNTRL*n*[PMC] = 11), pattern match accept (PCNTRL*n*[PMC] = 10), pattern match file (PATTRB*n*[PMF] is set), reject all (RCTRL[RA]), short frame received, reject short frame (RCTRL[RSF]), unicast (UC) hit, multicast (MC) hit, broadcast (BC) hit, promiscuous (RCTRL[PROM]), broadcast (BC) reject (RCTRL[BC_REJ]), and pause frame received.

Table 25-10. Receive Frame Filtering Result

Pattern Match Reject	Pattern Match Accept	Pattern Match File	Reject All	Short Frame	Reject Short Frame	UC Hit	UC Hash Hit	MC Hash Hit	BC Hit	PROM = 1	BC_ REJ = 1	Pause Frame	Action
X	X	X	X	Yes	No	X	X	X	X	X	X	X	Reject
No	No	X	Yes	X	X	X	X	X	X	X	X	X	Reject
No	Yes	X	X	No	X	X	X	X	X	X	X	No	Accept
Yes	X	X	X	X	X	X	X	X	X	X	X	X	Reject
No	No	X	No	No	X	Yes	No	No	No	X	X	No	Accept
No	No	X	No	No	X	No	Yes	No	No	X	X	No	Accept
No	No	X	No	No	X	No	No	Yes	No	X	X	No	Accept
No	No	X	No	No	X	No	No	No	Yes	No	No	No	Accept
No	No	X	No	No	X	No	No	No	Yes	No	Yes	No	Reject
No	No	X	No	No	X	X	X	X	X	Yes	X	No	Accept
X	X	X	X	X	X	X	X	X	X	X	X	Yes	Reject
Note: In the “Action” column, Accept = Accept with possibility of file.													

25.10.4 Filing

On the basis of a pattern hit, the Ethernet controller can file or route a receive frame to any of four different RxBD queues. Software uses the PATTRB*n*[QC] bit to program the queue destination for each pattern. Different patterns can have the same destination queue. Receive frames without a pattern hit or with a pattern hit with the filing option disabled can use the default attribute register to determine how to process the frame. Software can set up patterns and their associated control information to search for specific destination addresses, IP addresses, or priority fields such as a VLAN frame class of service (COS) field. The identified frame can be filed to the proper queue destination. The attributes of the pattern match reported in the RxBD[MP] fields specify whether a frame is to be filed and the queue in which to file the frame. For example, suppose a receive frame has a pattern match hit with pattern p2 (RxBD[MP] = 2), and pattern p2 is set up for filing (PATTRB*n*[PMF] is set). The QC field of the pattern attribute register for p2, PATTRB2[QC] specifies the queue to store the frame. However, under the same conditions, if pattern p2 is not set up for filing (PATTRB*n*[PMF] is cleared), the queue selected in the default attribute register is used. **Figure 25-23** depicts an example of how to prepare multiple patterns to examine different fields within a frame. Depending on which patterns hit or win (for multiple pattern hits), the frame can be filed to a preselected queue.

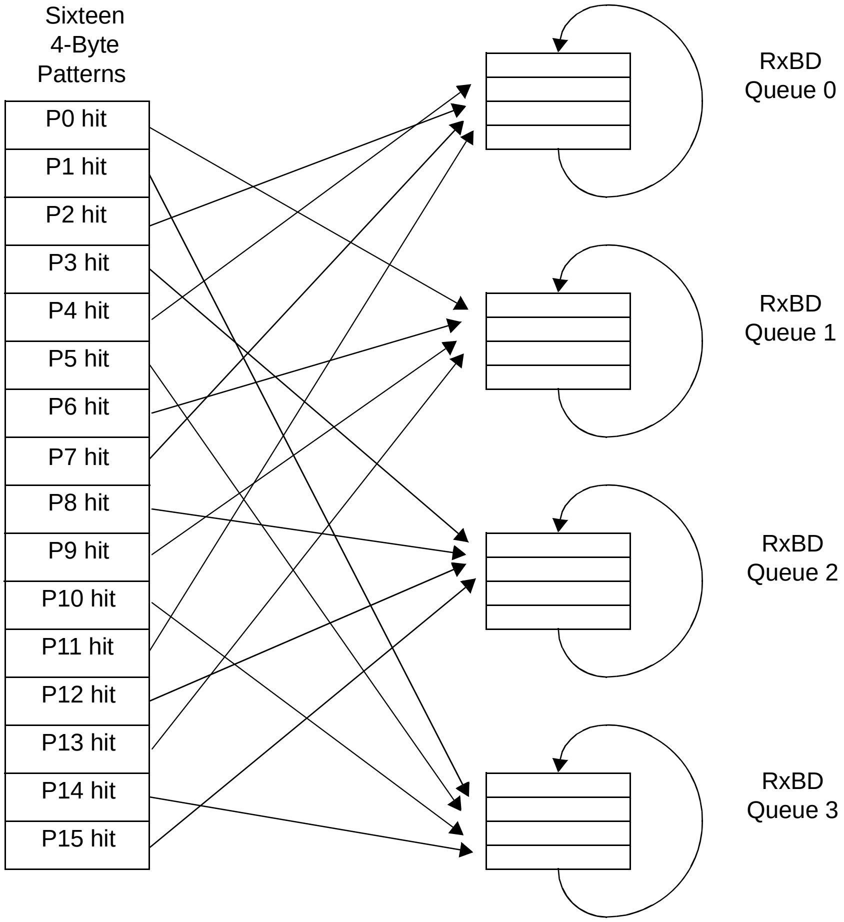


Figure 25-23. Frame Filing Based on Pattern Hit

25.10.5 Transmit Frame Processing with Insertion

Two insertion features for processing a transmit frame are insertion by replacement and insertion by expansion. Insertion is performed on a per BD basis. The mode of insertion is selected through the TxBD[IT] field. Insertion assists you in altering memory data buffers for transmit frames, which can be useful for applications requiring common repetitive insertion operations. The insertion data size can be any byte length consistent with creating a transmit frame of valid size. **Figure 25-24** shows how a TxBD can be used to process the transmit frame and perform insertion with replacement (BD0) and insertion with expansion (BD1). To insert data into a transmit frame, you must set up the following fields of the 32-byte BD for that frame:

- Status and control
- Data length and Tx data buffer pointer
- Insert buffer pointer, which points to the address of the buffer to be inserted
- Insert index, which is the number of bytes to jump within the transmit buffer before beginning to insert data
- Insert length, which is the number of bytes of data that are inserted into the buffer

- Insert type, which is the type of insertion to perform and must be set to TxBD[IT] = 01 for replacement

Insertion by expansion is similar to insertion by replacement, except that the insert type is set to TxBD[IT] = 10. While the insertion with expansion mode is selected, software must ensure that the data length in the type/length field is correct.

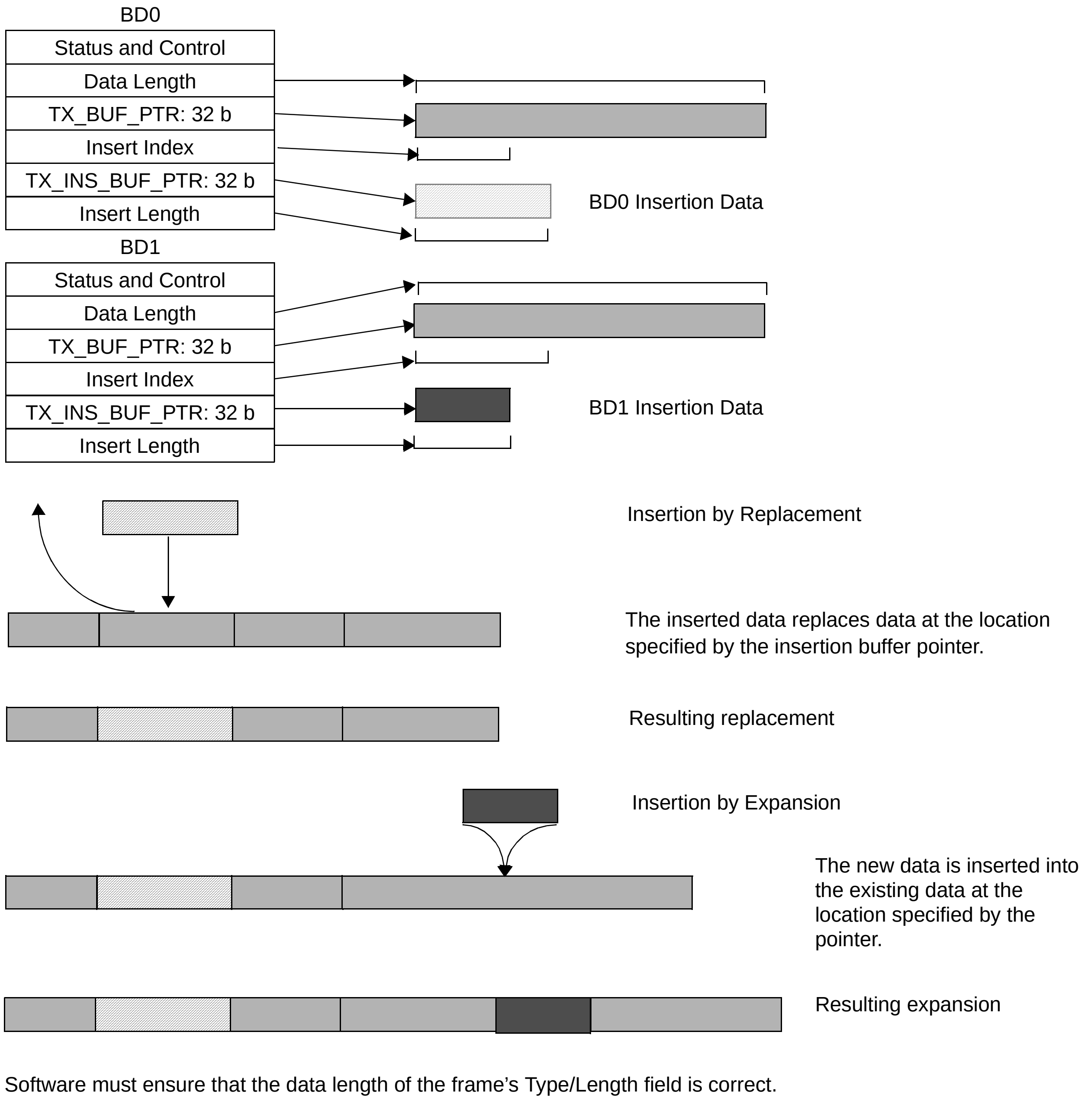


Figure 25-24. Insertion by Replacement and Insertion by Expansion

25.11 Flow Control

Because collisions cannot occur in Full-Duplex mode, the Ethernet controller can operate at the maximum rate. If the rate becomes too fast for a station receiver, the station transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. **Table 25-11** shows the flow-control frame structure.

If flow-control mode is enabled (MACCFG1R[RXFL] is set) and the Ethernet controller receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. During this pause, only a control frame can be sent by the Ethernet controller. Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

Table 25-11. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address		
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed independent constant of 512 bit-times (unlike slot time). The most-significant octet is transmitted first. with a two pause_quanta resolution. **Note:** Because the pause period has a resolution of two pause_quanta, the value programmed in this field is rounded up to the nearest even number before it is used, as follows: **MAC Parameter Value** — **Pause Period** 0 — none 1 or 2 — 2 × pause_quanta 3 or 4 — 4 × pause_quanta
2	Extended MAC parameter		Pause time extended as defined by the PTV[PTE] field. The most significant octet is transmitted first.
40	Reserved	—	
4	FCS		Frame check sequence (CRC)

Before transmitting flow control frames, poll the Used Entry Count Register in the FIFO. If the value is equal to or greater than the value in the Alarm Register, use the following recommended steps to transmit flow control frames:

1. Clear IEVENT[GTSC] (see *IEVENT Interrupt Event Register*, on page 25-53).
2. Set DMACTRL[GTS] (see *DMACTRL DMA Control Register*, on page 25-59)
3. Wait for IEVENT[GTSC] = 1 (see *IEVENT Interrupt Event Register*, on page 25-53).
4. Set TCTRL[TFC_PAUSE] (see *TCTRL Transmit Control Register*, on page 25-69).
5. Wait for IEVENT[TXC] = 1(see *IEVENT Interrupt Event Register*, on page 25-53).
6. Clear IEVENT[GTSC,TXC] (see *IEVENT Interrupt Event Register*, on page 25-53).
7. Clear DMACTRL[GTS] (see *DMACTRL DMA Control Register*, on page 25-59)

An additional method for sending flow control frames is to configure a PAUSE frame as an out-of-sequence frame.

25.12 Interrupt Handling

The Ethernet controller reports on events to the host and generates a maskable interrupt for each of the following events:

- Ethernet controller Ring 0–3 Receive Frame Event (RFE[0–3]). When IEVENT[RXF0–3]/RXB[0–3] is set.
- Ethernet controller Transmit Frame Event (TFE). When IEVENT[TXF]/[TXB] is set.
- Ethernet controller Another Event (AE). When one or more of the following IEVENT bits are set: RXC, BSY, EBERR, MSRO, GTSC, BABT, TXC, TXE, IE, LC, CRL, XFUN, GRSC. For details on IEVENT, see page 25-53.
- Receive inter frame bit status interrupt (RIFGSI) (Only In SMII Mode). For details on Receive Inter-Frame Gap Data Event, see page 25-86.

The maskable interrupts are routed to the global interrupt controllers (GIC) or periodic interrupt controller (PIC) of each SC140 core (see **Table 25-12**). The PIC can receive any of the five interrupt lines (one for each event) while the GIC receives one interrupt line (AE) that is an OR of interrupts from the rest of the IEVENT register bits. The interrupt request is level triggered, active low.

Table 25-12. Ethernet Controller Interrupt Routing

Mnemonic	Explanation	Routing
RFE0	Ethernet controller Ring 0 Receive Frame/Buffer Event	From Ethernet controller to all PIC $\overline{\text{IRQ0}}$
RFE1	Ethernet controller Ring 1 Receive Frame/Buffer Event	From Ethernet controller to all PIC $\overline{\text{IRQ1}}$
RFE2	Ethernet controller Ring 2 Receive Frame/Buffer Event	From Ethernet controller to all PIC $\overline{\text{IRQ2}}$
RFE3	Ethernet controller Ring 3 Receive Frame/Buffer Event	From Ethernet controller to all PIC $\overline{\text{IRQ3}}$
TFE	Ethernet controller Transmit Frame/Buffer Event	From Ethernet controller to all PIC $\overline{\text{IRQ4}}$
AE	Ethernet controller Another Event	From Ethernet controller to GIC bit 20 of GCIER
RIFGSI	Ethernet controller Receive Inter Frame Gap Status Interrupt (RIFGSI) **Note:** RIFGSI is used only in SMII mode.	From Ethernet controller to all PIC $\overline{\text{IRQ5}}$

Interrupt handler actions are as follows:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time.
- Process the TxBDs to reuse them if the IEVENT[TXB or TXF] is set. If the transmit speed is fast or the interrupt delay is long, the Ethernet controller may have sent more than one transmit buffer, so it is important to check more than one TxBD during interrupt handling. A common practice is to process TxBDs in the handler until one is found with R set. See **Table 25-13**.
- Obtain data from the RxBD if IEVENT[RXC, RXB*n*, or RXF*n*] is set. If the receive speed is fast or the interrupt delay is long, the Ethernet controller may have received more than one receive buffer, so it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the Ethernet controller prefetches BDs, the BD table must be big enough that there is always another empty BD to prefetch. See **Table 25-14**.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS] and continue normal execution.

Table 25-13. Non-Error Transmit Interrupts

Interrupt	Description
TXB	Transmit buffer: A TxBD that is not the last one in the frame was updated.
TXF	Transmit frame: A frame was transmitted and the last TxBD of that frame was updated.
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame being transmitted.(GTS can be asserted by setting DMACTRL[GTS] bit)
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.

Table 25-14. Non-Error Receive Interrupts

Interrupt	Description	Action Taken by Ethernet Controller
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed.	None.
RXB	Receive buffer: An RxBD that is not the last one of the frame was updated.	
RXF	Receive frame: A frame was received and the last RxBD of that frame was updated.	

25.13 Error-Handling

The Ethernet controller reports frame reception and transmission error conditions via the channel BDs, the error counters, and the IEVENT register. Transmission errors are described in **Table 25-15**. Reception errors are described in **Table 25-16**.

Table 25-15. Transmission Errors

Type Of Error	Ethernet Controller Operation
Transmitter underrun	The controller appends 32 bits that ensure a CRC error terminates buffer transmission, sets TxBD[UN], closes the buffer, IEVENT[XFUN], and IEVENT[TXE] are both set. The controller resumes transmission after TSTAT[THLT] is cleared and DMACTRL[GTS] is cleared.
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, IEVENT[CRL], and IEVENT[TXE] are both set. Transmission resumes after TSTAT[THLT] is cleared and DMACTRL[GTS] is cleared.
Excessive defer abort	The controller terminates buffer transmission, sets TxBD[DEF], closes the buffer, IEVENT[CRL/XDA], and IEVENT[TXE] conditions TBD. Transmission resumes after TSTAT[THLT] is cleared.
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, IEVENT[LC], and IEVENT[TXE] are both set. The controller resumes transmission after TSTAT[THLT] is cleared and DMACTRL[GTS] is cleared.
Memory Read Error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], the DMA controller stops sending data to the FIFO, which causes an underrun error but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits continue once TSTAT[THLT] is cleared.
Insertion Error	Insertion error. The controller sets TxBD[IE] and IEVENT[IE] and continues to send frames with possible incorrect insertion data.
Babbling Transmit Error	A frame is transmitted that exceeds the MAC Maximum Frame Length and. The controller sets IEVENT[BABT] and continues without interruption.

Table 25-16. Reception Errors

Type Of Error	Ethernet Controller Operation
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, and sets IEVENT[RXF*n*], The receiver then enters hunt mode (seeking start of a new frame). **Note:** In some situations, the overrun condition may be due to heavy accesses to external memory by devices competing with the Ethernet controller. It may take the controller longer to access the external memory and therefore increase the likelihood of an overrun. The system busy condition may not be reflected by the IEVENT[BSY} bit or by the IEVENT[GRSC] bit. The main symptom may be that the receiver drops frames with no dropped frame indication. In some cases, the receiver may stop operating and all frames will be dropped and logged as dropped frames. If this occurs, only a hard reset will clear the condition. To prevent this from occurring when there is a high Ethernet traffic, change the SIU priorities in the PPC_ALRH and PPC_ALRL so that all Ethernet accesses to the System Bus have a higher priority than the SC140 cores. This should be done by code running from internal memory while only one core is active and the DMA is not active. Alternately, move all the Ethernet buffers to M2 memory.
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY]. In addition, the proper RSTAT halt bit is set. (While using four queues, only the queue that encountered the error halts.) The halted queue resumes reception once its RSTAT halt bit is cleared.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF*n*] is set, and the alignment error counter increments. The Ethernet controller relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.

Table 25-16. Reception Errors (Continued)

Type Of Error	Ethernet Controller Operation
CRC error	If a CRC error occurs, the controller sets RxBD[CR], closes the buffer, and sets IEVENT[RXF*n*]. This Ethernet controller relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory Read Error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame and increments the discarded frame counter (DISFC). In addition, the proper RSTAT halt bit is set. (For four queues, only the queue that encountered the error halts.) The halted queue resumes reception once its RSTAT halt bit is cleared.
RIFGSI	While receiving IFG segments in SMII Mode the received IFG segments are sampled to the MIGSK_RIFGR. If there is a difference between a received inter-frame gap bit (or more) in the MIIGSK_RIFBR register and its corresponding bit in the MIIGSK_ERIFBR register then the corresponding bit in the MIIGSK_IEVENT register is set. If it is also enabled in the corresponding bit in the MIIGSK_IMASK register then RIFGSI interrupt is set.

25.14 Inter-Packet Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). After a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the packet is not successfully sent within a specified number of retries, an error is indicated. The minimum inter-packet gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back packets with this minimum spacing. In addition, after waiting a required number of clocks (based on the back-off algorithm), the transmitter waits for carrier sense to be deasserted before retransmitting the packet. Retransmission begins 36 serial clocks after carrier sense is deasserted for at least 60 serial clocks.

25.15 Connecting to Physical Interfaces

This section describes how to use the MIIGSK interface to connect the Ethernet controller to the PHY/MAC in MII, RMII, and SMII modes. In RMII and SMII mode, some part of the Ethernet controller signals are used. **Table 25-4** indicates which signals are reserved in these modes and which signals should not be connected.

Note: The MAC-to-MAC connection is not defined in the **IEEE** Std. 802.3. Care must be taken to ensure that the receive side has enough set-up and hold time.

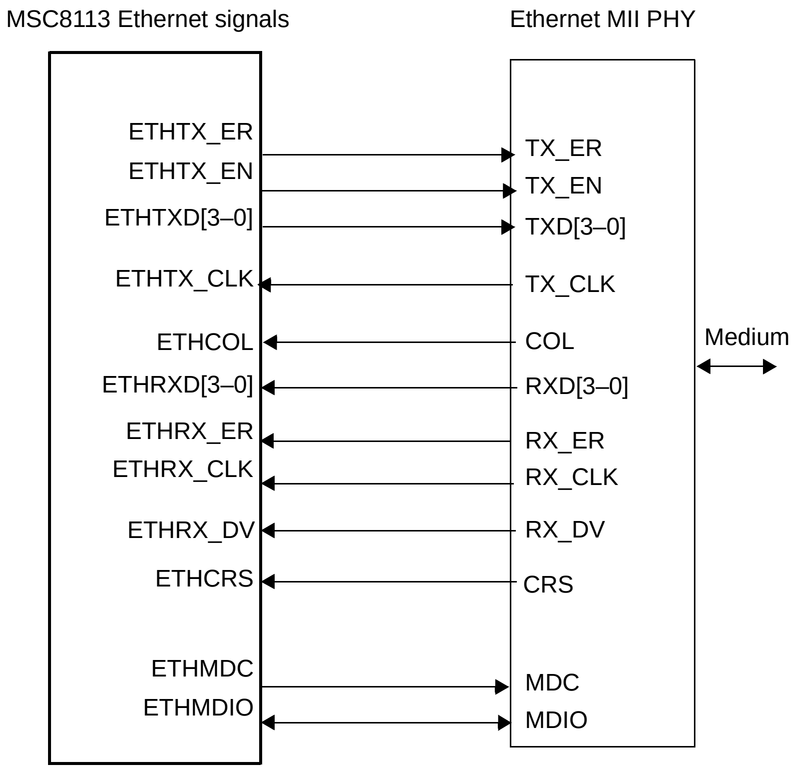


Figure 25-25. Ethernet Controller to PHY Connection, MII Mode (Transparent)

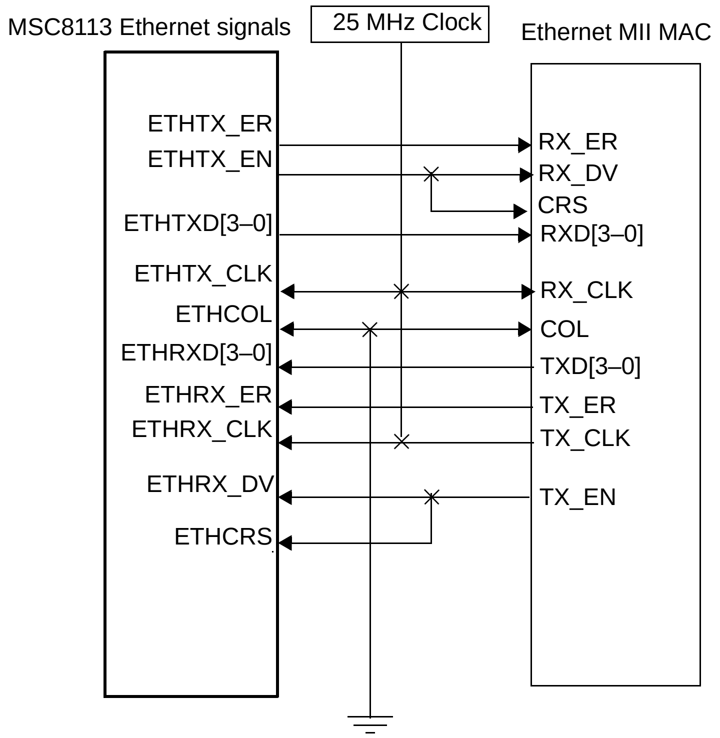


Figure 25-26. Full Duplex MAC-to-MAC Connection In MII Mode (Transparent)

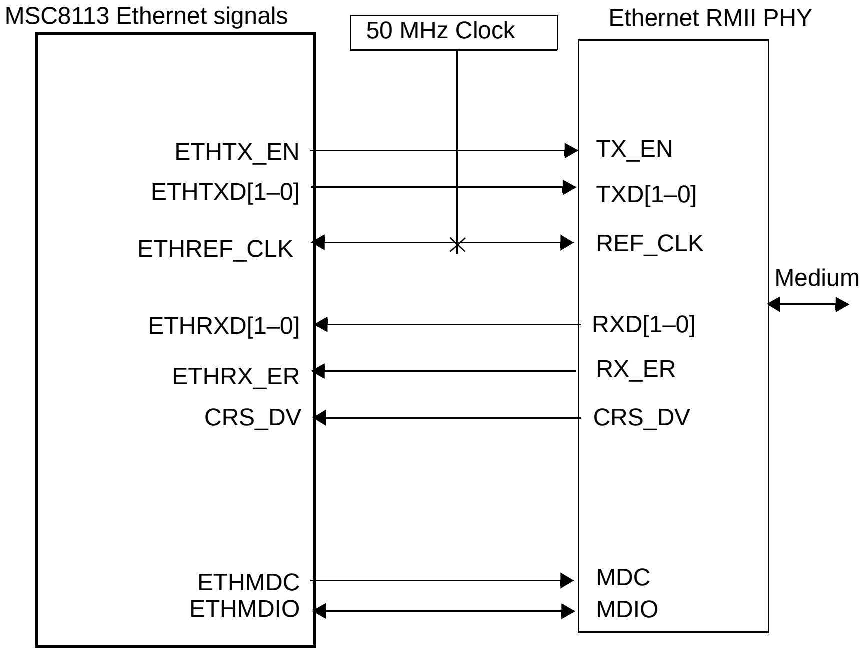


Figure 25-27. Ethernet Controller-to-PHY Connection in RMII Mode

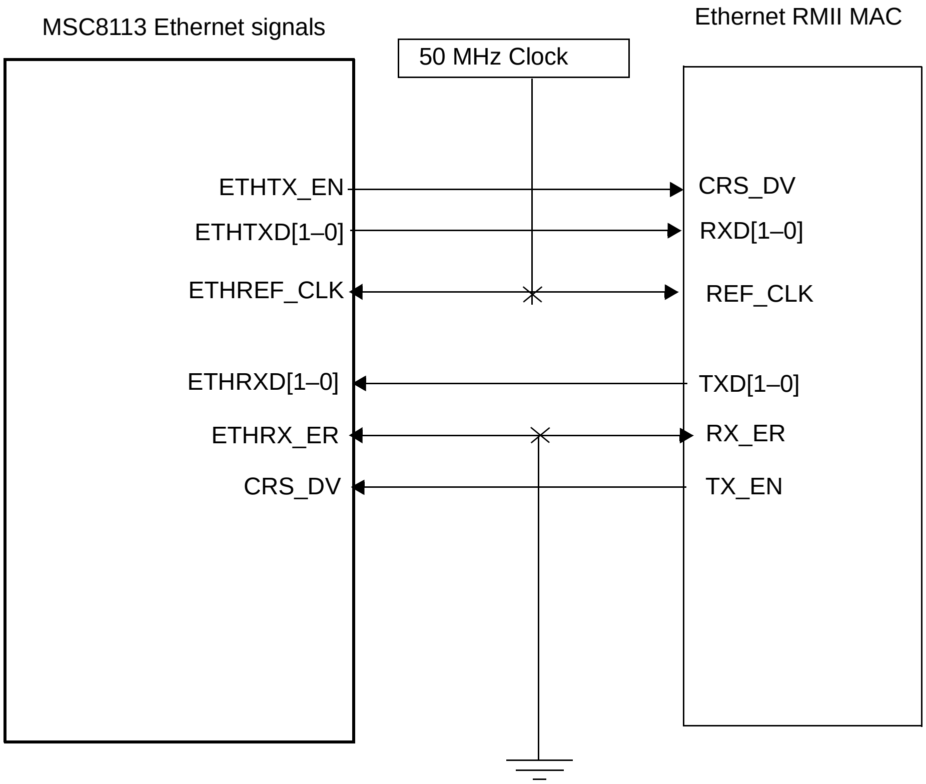


Figure 25-28. Full Duplex MAC-to-MAC Connection In RMII Mode

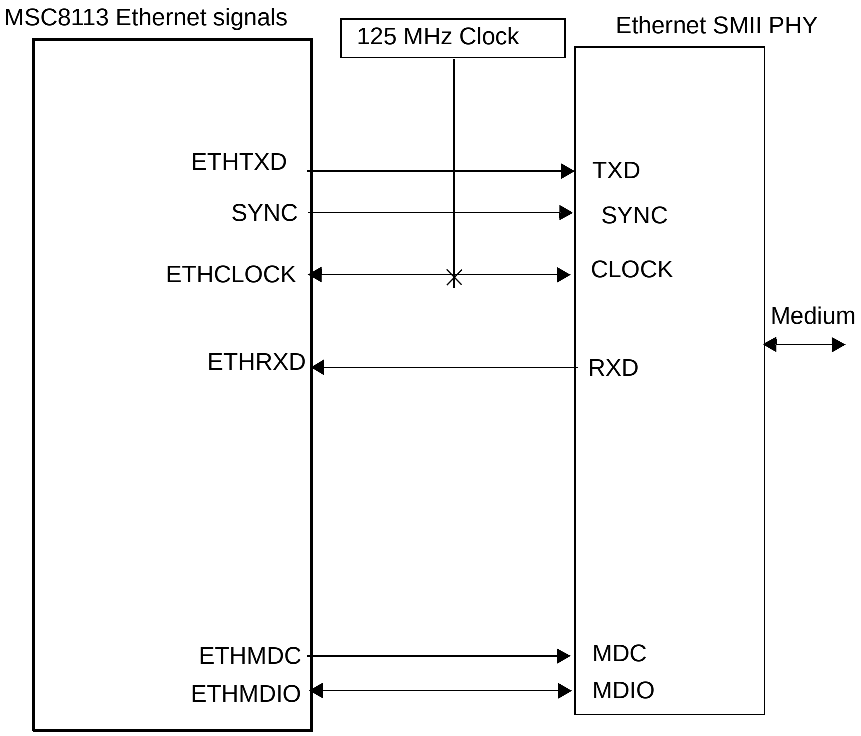


Figure 25-29. Ethernet Controller-to-PHY Connection in SMII Mode

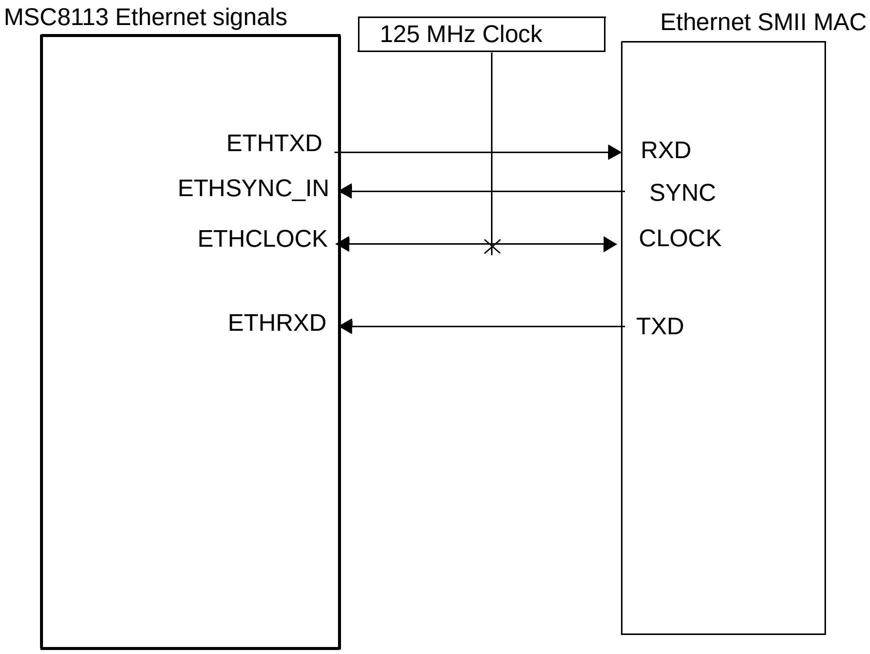


Figure 25-30. Ethernet Controller-to-MAC Connection in SMII Mode

25.16 Initialization and Reset

The Ethernet controller can supports the following types of reset:

- Power-on reset ($\overline{\text{PORESET}}$)
- External hard reset ($\overline{\text{HRESET}}$)
- Ethernet internal reset is initiated by writing a value of 1 to MIIGSK_GPR[IR]. The Ethernet internal reset is released by writing a value of 0 to MIIGSK_GPR[IR].

Note: The Ethernet Internal Reset does not reset the MIIGSK_GPR. The internal reset must be valid for at least 10 core cycles.

- MAC software reset is initiated by writing a 1 to the MACCFG1R[SRESET] bit. The MAC software reset is released by writing a value of 0 to MACCFG1R[SRESET].

Note: The MAC software resets only resets the Ethernet controller MAC logic.

This section describes the registers that are reset by assertion of $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, or the Ethernet Internal Reset and the registers that must be initialized before the Ethernet controller is enabled. All Ethernet controller registers and control logic are reset to their default states.

After the system undergoes reset, software must initialize certain Ethernet controller registers. Other registers can also be initialized, but they are optional and must be determined on the basis of system requirements. **Table 25-17** describes the minimum steps for register initialization.

Table 25-17. Minimum Register Initialization

Initialization Step	Register(s)	Page
1. Set the MIIGSK_ENR[EN] bit.	MIIGSK Enable Register (MIIGSK_ENR).	page 25-98.
2. Wait until the MIIGSK_ENR[R] is set.	MIIGSK Enable Register (MIIGSK_ENR).	page 25-98.
3. Set and clear MACCFG1R[SRESET].	MAC Configuration Register 1 (MACCFG1R).	page 25-84.
4. Initialize the media access control (MAC) configuration register, which adjusts frame length and preamble length, specifies various CRC/pad combinations, and specifies Full- or Half-Duplex operating mode.	MAC Configuration Register 2 (MACCFG2R).	page 25-85.
5. Initialize the MAC station address.	MAC Station Address Part 1/Part 2 registers (MACSTNADDR1, MACSTNADDR1).	page 25-90, page 25-91.
6. Set up the PHY using the MII management interface. **Note:** **1** This step is required only when the Ethernet controller connects to a transceiver (MAC-to-PHY mode). **2.** Before you change the PHY speed via software, you must first clear MACCFG1R[TXEN, RXEN] bits (page 25-84).	MII Management Interface registers are discussed in **Section 25.17.8**.	page 25-105.
7. Select MII, RMII, or SMII mode.	MIIGSK Configuration Register (MIIGSK_CFGR)	page 25-96.

Table 25-17. Minimum Register Initialization (Continued)

Initialization Step	Register(s)	Page
8. Select RMII or SMII speed (10Mbps/100Mbps).	MIIGSK Configuration Register (MIIGSK_CFGR)	page 25-96.
9. Clear interrupts to prepare for interrupt events.	Interrupt Event Register (IEVENT).	page 25-53.
10. Initialize the interrupt mask to prepare for interrupt events.	Interrupt Mask Register (IMASK).	page 25-55.
11. Set the DMACTRL[30] bit.	DMA Control Register (DMACTRL).	page 25-59.
12. Initialize the DMA Maintenance Register by writing a 1 to DMAMR[9].	DMA Maintenance Register (DMAMR).	page 25-61.
13. Initialize the FIFO Receive Control Register by writing a 1 to FRXCTRLR[30].	FIFO Receive Control Register (FRXCTRLR).	page 25-63.
14. Initialize the Pattern Match registers. **Note:** When the Ethernet controller is configured to accept frames based on destination address recognition flow (see page 25-25), you must initialize the individual address hash table entries registers (see page 25-132) and the group address hash table entries (see page 25-132).	PMDn PMASKn PCNTRLn PATTRBn	page 25-133 page 25-133 page 25-134 page 25-135
15. Initialize control of the receive block operating mode.	Receive Control Register (RCTRL).	page 25-78
16. Initialize the DMA controller.	DMA Control Register (DMACTRL).	page 25-59

After the registers are initialized, you must execute the following steps in the order described to bring the Ethernet controller into a functional state out of reset:

1. To transmit Ethernet frames, build the TxBDs in memory, link them together as a ring, and point to the ring. A minimum of two TxBDs per ring is required. Use one of the following two methods to handle the transmit buffers:

   a. Method one:

   - Set the DMACTRL[WOP] bit (see page 25-59).
   - Before setting the TxBD[R] bit, clear the TSTAT[THLT] bit if set by writing to the TSTAT[THLT] bit (see page 25-70 and page 25-137).

   b. Method two:

   - Clear the DMACTRL[WOP] bit (see page 25-59).
   - Before setting the TxBD[R] bit, set the DMACTRL[GTS] bit (see page 25-59 and page 25-137).
   - Set the TxBD[R] bit (see page 25-137).
   - Clear the DMACTRL[GTS] bit (see page 25-59).

   For both methods, ensure that the top (first) TxBD in the transmitter ring is the last to have its ready bit set. For example, if the frame is described by three TxBDs, set TxBD[R] in the last TxBD first, then the middle TxBD, perform either method one or two above. For either method, make sure that the first TxBD is the last for which TxBD[R] is set.

2. To receive Ethernet frames, link the RxBDs together as a ring and point the corresponding registers to them. Both transmit and receive can be gracefully stopped after transmission and reception begins.
3. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. The DMACTRL[GRS] must be cleared if the receiver had been previously stopped. Refer to the *DMACTRL DMA Control Register*, on page 25-59, and **Section 25.10.1** for more information.
4. Write to MACCFG1R and set the appropriate bits, including RXEN and TXEN. To enable flow control, RXFL and TXFL should also be set.

Before issuing any type reset to and/or reconfiguring the MAC with new parameters, you must properly shut down the DMA controller and ensure that it is in an idle state by setting both the DMACTRL[GRS, GTS] bits. Wait for both the IEVENT[GRSC, GTSC] bits to be set, clear them by writing 1 and then set the Ethernet controller internal reset bit (MIIGSK_GPR[IR]). The internal reset must be valid for at least 10 core cycles. Then clear the MIIGSK_GPR[IR] bit and reconfigure the Ethernet controller. During the MAC configuration, the TBASE register and the RBASEn registers must be written with the pointers that points to the TX/RX set of descriptors.

Use the following procedure for resetting and reconfiguring the MAC. The page numbers indicate the location of the appropriate register descriptions.

1. Set the DMACTRL[GRS, GTS] bits (page 25-59).
2. Poll the IEVENT[GRSC, GTSC] bit until both are set (page 25-53).
3. Clear the DMACTRL[GRS, GTS] bits (page 25-59).
4. Set the MIIGSK_GPR[IR] bit (page 25-97)
5. Wait 20 Core Cycles.
6. Clear the MIIGSK_GPR[IR] bit (page 25-97).
7. Set the MIIGSK_ENR[EN] bit (page 25-98).
8. Wait until the MIIGSK_ENR[R} bit sets (page 25-98).
9. Set the MACCFG1R[SRESET] bit (page 25-84).
10. Clear the MACCFG1R[SRESET] bit (page 25-84).
11. Write 01 to the MACCFG2R[22–23] bits (page 25-85).
12. Set DMAMR[9] (page 25-61).
13. Set FRXCTRLR[30] (page 25-63).
14. Configure the MIIGSK_CFGR (page 25-96).

15. Configure the other MAC registers (**Section 25.17.5**, *MAC Registers*, on page 25-84).
16. Set the DMACTRL[30] bit (page 25-590.
17. Clear DMACTRL[GTS] to trigger the transmission of frame data if the transmitter is stopped. DMACTRL[GRS] must be cleared if the receiver is stopped. For details, refer to DMACTRL on page 25-59, and **Section 25.10.1**, *Data Buffer Descriptor*, on page 25-28.
18. Clear the RSTAT[THLT] bit by writing a value of 1 to the bit (page 25-79).
19. Clear the DMACTRL[GRS, GTS] bits (do not change other bits) (page 25-59).
20. Load the corresponding RBASEn with new RxBD pointer (page 25-84).
21. Enable the MACCFG1R[TXEN, RXEN] bits (page 25-84).

If reconfiguration occurs while the Ethernet controller is receiving frames, the first frame after the Ethernet controller is enabled may be detected as an error. In SMII mode, you should also initialize the SMII Receive inter frame bit status interrupt (RIFGSI) using the following steps:

1. Write 0xFF to the MIIGSK_IMASK register to enable all events.
2. Wait for the interrupt.
3. The interrupt handler should disable all events by writing 0x00 to the MIIGSK_IMASK register, and then write 0xFF to the MIIGSK_IEVENT register to clear all events.
4. After initialization, you can configure the MIIGSK_IMASK and MIIGSK_ERIFBR register as required.

25.17 Ethernet Controller Programming Model

Note: The Ethernet controller software model is similar to that of the fast Ethernet functionality in the Freescale MPC8560 device.

The Ethernet controller device is programmed by a combination of control/status registers (CSR) used for mode control and interrupts and BDs (Ethernet controller with IPI magenta interface) that pass data buffers and related buffer status or frame information between the hardware and software. All accesses via IPBus1 to and from the registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.

This section describes the Ethernet controller registers in detail. The discussion is organized around the following register groupings. Refer to **Chapter 8**, *Memory Map* for information on the location of the registers.

Table 25-18. Ethernet Controller Registers Summary

Register			Page
General Control and Status Registers			
	Interrupt Event Register	IEVENT	page 25-53
	Interrupt Mask Register	IMASK	page 25-55
	Ethernet Control Register	ECNTRL	page 25-57
	Minimum Frame Length Register	MINFLR	page 25-58
	Pause Time Value Register	PTV	page 25-58
	DMA Control Register	DMACTRL	page 25-59
	DMA Maintenance Register	DMAMR	page 25-61
FIFO Control and Status Registers			
	FIFO Receive Status Register	FRXSTATR	page 25-62
	FIFO Receive Control Register	FRXCTRLR	page 25-63
	FIFO Receive Alarm Register	FRXALAR	page 25-63
	FIFO Receive Alarm Shutoff Register	FRXSHR	page 25-64
	FIFO Receive Panic Register	FRXPAR	page 25-65
	FIFO Receive Panic Shutoff Register	FRXPSR	page 25-65
	FIFO Transmit Status Register	FTXSTATR	page 25-66
	FIFO Transmit Threshold Register	FTXTHR	page 25-67
	FIFO Transmit Space Available Register	FTXSPR	page 25-67
	FIFO Transmit Starve Register	FTXSR	page 25-68
	FIFO Transmit Starve Shutoff Register	FTXSSR	page 25-68
Transmit Control and Status Registers			
	Transmit Control Register	TCTRL	page 25-69
	Transmit Status Register	TSTAT	page 25-70
	TxBD Data Length Register	TBDLEN	page 25-71
	Current TxBD Pointer	CTBPTR	page 25-71
	TxBD Pointer	TBPTR	page 25-72
	Transmit Descriptor Base Address	TBASE	page 25-72
	Out-of-Sequence TxBD Register	OSTBD	page 25-73
	Out-of-Sequence Tx Data Buffer Pointer Register	OSTBDP	page 25-73
	Out-of-Sequence 32 Bytes Tx Data Buffer Pointer Register	OS32TBDP	page 25-75
	Out-of-Sequence 32 Bytes TxBD Insert Pointer Register	OS32IPTR	page 25-76
	Out-of-Sequence 32 Bytes TxBD Reserved Register	OS32TBDR	page 25-76
	Out-of-Sequence 32 Bytes TxBD Insert Index/Length Register	OS32IIL	page 25-77
Receive Control and Status Registers			
	Receive Control Register	RCTRL	page 25-78
	Receive Status Register	RSTAT	page 25-79
	RxBD Data Length Register	RBDLEN	page 25-80
	Current RxBD Pointer	CRBPTR	page 25-80
	Maximum Receive Buffer Length R0R1 Register	MRBLR0R1	page 25-81
	Maximum Receive Buffer Length R2R3 Register	MRBLR2R3	page 25-82
	RxBD Pointer 0–3	RBPTRn	page 25-82
	Receive Descriptor Base Address	RBASEn	page 25-83

Table 25-18. Ethernet Controller Registers Summary (Continued)

Register			Page
MAC Registers			
	MAC Configuration 1 Register	MACCFG1R	page 25-84
	MAC Configuration 2 Register	MACCFG2R	page 25-85
	Inter-Packet Gap/Inter-Frame Gap Register	IPGIFGR	page 25-86
	Half-Duplex Register	HAFDUPR	page 25-87
	Maximum Frame Length Register	MAXFRMR	page 25-89
	Interface Status Register	IFSTATR	page 25-89
	MAC Station Address Part 1 Register	MACSTADDR1R	page 25-90
	MAC Station Address Part 2 Register	MACSTADDR2R	page 25-91
MII Management Registers			
	MII Management Configuration Register	MIICFGR	page 25-92
	MII Management Command Register	MIIMCOMR	page 25-93
	MII Management Address Register	MIIMADDR	page 25-94
	MII Management Control Register	MIIMCONR	page 25-95
	MII Management Status Register	MIIMSTATR	page 25-95
	MII Management Indicator Register	MIIMINDR	page 25-95
MIIGSK Registers			
	MIIGSK Configuration Register	MIIGSK_CFGR	page 25-96
	MIIGSK General-Purpose Register	MIIGSK_GPR	page 25-97
	MIIGSK Enable Register	MIIGSK_ENR	page 25-98
	MIIGSK SMII SYNC Direction Register	MIIGSK_SYNCDIR	page 25-98
	MIIGSK Transmit Inter-Frame Bits Register	MIIGSK_TIFBR	page 25-99
	MIIGSK Receive Inter-Frame Bits Register	MIIGSK_RIFBR	page 25-101
	MIIGSK Expected Receive Inter-Frame Bits Register	MIIGSK_ERIFBR	page 25-102
	MIIGSK SMII Interrupt Event Register	MIIGSK_IEVENT	page 25-102
	MIIGSK SMII Interrupt Mask Register	MIIGSK_IMASK	page 25-104
RMON Counters (MIB)			
	Transmit and Receive 64-Byte Frame Counter	TR64	page 25-105
	Transmit and Receive 65- to 127-Byte Frame Counter	TR127	page 25-106
	Transmit and Receive 128- to 255-Byte Frame Counter	TR255	page 25-106
	Transmit and Receive 256- to 511-Byte Frame Counter	TR511	page 25-107
	Transmit and Receive 512- to 1023-Byte Frame Counter	TR1K	page 25-107
	Transmit and Receive 1024- to 1518-Byte Frame Counter	TRMAX	page 25-108
	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter	TRMGV	page 25-108
	Receive Byte Counter	RBYT	page 25-109
	Receive Packet Counter	RPKT	page 25-109
	Receive FCS Error Counter	RFCS	page 25-110
	Receive Multicast Packet Counter	RMCA	page 25-110
	Receive Broadcast Packet Counter	RBCA	page 25-111
	Receive Control Frame Packet Counter	RXCF	page 25-111
	Receive Pause Frame Packet Counter	RXPF	page 25-112
	Receive Unknown OPCode Packet Counter	RXUO	page 25-112
	Receive Alignment Error Counter	RALN	page 25-113
	Receive Frame Length Error Counter	RFLR	page 25-113
	Receive Code Error Counter	RCDE	page 25-114
	Receive Carrier Sense Error Counter	RCSE	page 25-114
	Receive Undersize Packet Counter	RUND	page 25-115
	Receive Oversize Packet Counter	ROVR	page 25-115

Table 25-18. Ethernet Controller Registers Summary (Continued)

Register			Page
RMON Registers (MIB) (continued)			
	Receive Fragments Counter	RFRG	page 25-116
	Receive Jabber Counter	RJBR	page 25-116
	Receive Dropped Packet Counter	RDRP	page 25-117
	Transmit Byte Counter	TBYT	page 25-117
	Transmit Packet Counter	TPKT	page 25-118
	Transmit Multicast Packet Counter	TMCA	page 25-118
	Transmit Broadcast Packet Counter	TBCA	page 25-119
	Transmit Pause Control Frame Counter	TXPF	page 25-119
	Transmit Deferral Packet Counter	TDFR	page 25-120
	Transmit Excessive Deferral Packet Counter	TEDF	page 25-120
	Transmit Single Collision Packet Counter	TSCL	page 25-121
	Transmit Multiple Collision Packet Counter	TMCL	page 25-121
	Transmit Late Collision Packet Counter	TLCL	page 25-122
	Transmit Excessive Collision Packet Counter	TXCL	page 25-122
	Transmit Total Collision Counter	TNCL	page 25-123
	Transmit Jabber Frame Counter	TJBR	page 25-123
	Transmit FCS Error Counter	TFCS	page 25-124
	Transmit Control Frame Counter	TXCF	page 25-124
	Transmit Oversize Frame Counter	TOVR	page 25-125
	Transmit Undersize Frame Counter	TUND	page 25-125
	Transmit Fragment Counter	TFRG	page 25-126
	Carry Register One	CAR1	page 25-126
	Carry Register Two	CAR2	page 25-127
	Carry Register One Mask	CAM1	page 25-129
	Carry Register Two Mask	CAM2	page 25-130
Hash Function Registers			
	Individual Address Registers 0–7	IADDRn	page 25-132
	Group Address Registers 0–7	GADDRn	page 25-132
Pattern Matching Registers			
	Pattern Match Data 0–15	PMDn	page 25-133
	Pattern Mask Register 0–15	PMASKn	page 25-133
	Pattern Match Control Register 0–15	PCNTRLn	page 25-134
	Pattern Match Attributes Register 0–15	PATTRBn	page 25-134
	Default Attribute Register	DATTR	page 25-134
Data Structures (Buffer Descriptors)			
	8-Byte Transmit Data Buffer Descriptor	TxBD	page 25-137
	32-Byte Transmit Data Buffer Descriptor	TxBD	page 25-139
	8-Byte Receive Buffer Descriptor	RxBD	page 25-144
	32-Byte Receive Buffer Descriptor	RxBD	page 25-146

25.17.1 General Control and Status Registers

IEVENT Interrupt Event Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	RXC	BSY	EBERR	—	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	IE	LC	CRL	XFUN
Type	R/W				R	R	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
	RXB0	RXB1	RXB2	RXB3	—			GRSC	RXF0	RXF1	RXF2	RXF3	—			
Type	R/W				R			R/W					R			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IEVENT contains bits that generate an interrupt if the corresponding bit in the Interrupt Mask Register (IMASK) is also set. The bit in IEVENT is cleared if a 1 is written to that bit position. A write of 0 has no effect. These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts:

- Interrupts that occur during normal operation are: GTSC, GRSC, TXF, TXB, TXC, RXF, RXB, and RXC.
- Interrupts resulting from errors/problems detected in the network or transceiver are: BABT, LATE_COL and COL_RETRY_LIM.
- Interrupts resulting from internal errors are: EBERR, XFIFO_UN and BSY

Some error interrupts are independently counted in the management information base (MIB) block counters. Software may mask these interrupts because these errors are visible to network management via the MIB counters.

Table 25-19. IEVENT Bit Descriptions

Name	Reset	Description	Settings
0	0	Reserved. Write to zero for future compatibility.	
RXC 1	0	**Receive Control Interrupt** A control frame was received (MACCFG1R[RFCE] must be set). As soon as the transmitter finishes sending the current frame, a pause operation is performed.	0 No control frame. 1 Control frame received.
BSY 2	0	**Busy Condition Interrupt** Set if a frame is received and discarded due to a lack of buffers.	0 No frame discarded. 1 Frame discarded.
EBERR 3	0	**Ethernet Bus Error** A system bus error occurred during a DMA transaction. If EBERR is set while a transmission is in progress, the DMA controller stops sending data to the Tx FIFO, which eventually causes an underrun error (XFUN). If EBERR is set while a frame is being received, the DMA controller discards the frame.	0 No system bus error. 1 System bus error.
— 4	0	Reserved. Write to zero for future compatibility.	
MSRO 5	0	**MSTAT Register Overflow** Generates an interrupt if the count for one of the MSTAT registers exceeds the size of the register.	0 No MSTAT Register overflow. 1 MSTAT Register overflow.

Table 25-19. IEVENT Bit Descriptions (Continued)

Name	Reset	Description	Settings
GTSC 6	0	**Graceful Transmit Stop Complete** Generates an interrupt for one of two reasons. • Completion of a graceful stop initiated by setting DMACTRL[GTS]. • Completion of a graceful stop initiated by setting TCTRL[TFCP]. During a graceful stop, the transmitter is put into a pause state after completion of the frame being transmitted.	0 No interrupt. 1 Graceful stop complete interrupt.
BABT 7	0	**Babbling Transmit Error** The transmitted frame length has exceeded the value in the MACs Maximum Frame Length register (see page 25-89). Frame truncation occurs when this condition occurs.	0 Normal Operation. 1 Babbling transmit error.
TXC 8	0	**Transmit Control Interrupt** A control frame was transmitted.	0 No control frame. 1 Control frame transmitted.
TXE 9	0	**Transmit Error** An error on the transmitted channel that caused the Ethernet controller to set TSTAT[THLT]. This bit is set when any transmit error causes the transmitter to halt (EBERR, LC, CRL, XFUN).	0 No transmit error. 1 Transmit error has halted the transmitter.
TXB 10	0	**Transmit Buffer** A TxBD whose Interrupt (I) bit was set in its status word was updated but was not the last BD of the frame.	0 Normal operation. 1 Transmit buffer.
TXF 11	0	**Transmit Frame Interrupt** A frame was transmitted and the last corresponding TxBD is updated. This occurs only if the Interrupt (I) bit in the status word of the BD is set.	0 No frame transmitted. 1 Frame transmitted and last BD updated.
IE 12	0	**Insertion Error** An insertion error occurred during an attempt to insert data during transmission of a frame.	0 No insertion error. 1 Insertion error.
LC 13	0	**Late Collision** A collision occurred beyond the collision window (slot time) in Half-Duplex mode. The frame is truncated with a bad CRC, and the remainder of the frame is discarded.	0 No collision. 1 Collision occurred beyond collision window.
CRL 14	0	**Collision Retry Limit** The number of successive transmission collisions has exceeded the MAC Half-Duplex Register retransmission maximum count. The frame is discarded without being transmitted, and transmission of the next frame commences. This occurs only in Half-Duplex mode.	0 No excessive transmission collisions. 1 The number of successive transmission collisions has exceeded the maximum count.
XFUN 15	0	**Transmit FIFO Underrun** The transmit FIFO emptied before the complete frame was transmitted.	0 No underrun. 1 Transmit FIFO underrun.
RXB0 16	0	**Receive Buffer 0** An RxBD from queue 0 with the Interrupt (I) bit in its status word was updated but was not the last BD of the frame.	0 Normal operation. 1 Receive buffer.
RXB1 17	0	**Receive Buffer 1** An RxBD from queue 1 with the Interrupt (I) bit in its status word was updated but was not the last BD of the frame.	0 Normal operation. 1 Receive buffer.
RXB2 18	0	**Receive Buffer 2** An RxBD from queue 2 with the Interrupt (I) bit in its status word was updated but was not the last BD of the frame.	0 Normal operation. 1 Receive buffer.
RXB3 19	0	**Receive Buffer 3** An RxBD from queue 3 with the Interrupt (I) bit in its status word was updated but was not the last BD of the frame.	0 Normal operation. 1 Receive buffer.
— 20–22	0	Reserved. Write to zero for future compatibility.	

Table 25-19. IEVENT Bit Descriptions (Continued)

Name	Reset	Description	Settings
GRSC 23	0	**Graceful Receive Stop Complete** Generates an interrupt when a graceful receive stop is completed. It indicates that it is safe to write to the receive registers (status, control or configuration registers) in use by the system during normal operation.	0 No graceful stop completed. 1 Graceful stop completed.
RXF0 24	0	**Receive Frame Interrupt 0** A frame was received in queue 0 and the last RxBD in that frame was updated. This occurs only if the Interrupt (I) bit in the BD status word is set.	0 No receive frame interrupt. 1 Receive frame interrupt.
RXF1 25	0	**Receive Frame Interrupt 1** A frame was received in queue 1 and the last RxBD in that frame was updated. This occurs only if the Interrupt (I) bit in the BD status word is set.	0 No receive frame interrupt. 1 Receive frame interrupt.
RXF2 26	0	**Receive Frame Interrupt 2** A frame was received in queue 2 and the last RxBD in that frame was updated. This occurs only if the Interrupt (I) bit in the BD status word is set.	0 No receive frame interrupt. 1 Receive frame interrupt.
RXF3 27	0	**Receive Frame Interrupt 3** A frame was received in queue 3 and the last RxBD in that frame was updated. This occurs only if the Interrupt (I) bit in the BD status word is set.	0 No receive frame interrupt. 1 Receive frame interrupt.
— 28–31	—	Reserved. Write to zero for future compatibility.	

IMASK Interrupt Mask Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—	RXCEN	BSYEN	EBERREN	—	MSROEN	GTSCEN	BTEN	TXCEN	TXEEN	TXBEN	TXFEN	IEEN	LCEN	CRLEN	XFUNEN
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**
	RXBEN0	RXBEN1	RXBEN2	RXBEN3	—			GRSCEN	RXFEN0	RXFEN1	RXFEN2	RXFEN3	—			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMASK controls which interrupt events can generate an interrupt. All implemented bits in this CSR are R/W. This register is cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, an interrupt is generated. The interrupt signal remains asserted until the IEVENT bit is cleared either by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Table 25-20. IMASK Bit Descriptions

Bit	Reset	Description	Settings
0	0	Reserved. Write to zero for future compatibility.	
RXCEN 1	0	**Receive Control Interrupt Enable**	0 RCI disabled. 1 RCI enabled.

Table 25-20. IMASK Bit Descriptions (Continued)

Bit	Reset	Description	Settings
BSYEN 2	0	**Busy Interrupt Enable**	0 BI disabled. 1 BI enabled.
EBERREN 3	0	**Ethernet Controller Bus Error Enable**	0 EBERR disabled. 1 EBERR enabled.
— 4	0	Reserved. Write to zero for future compatibility.	
MSROEN 5	0	**MSTAT Register Overflow Interrupt Enable**	0 MSROI disabled. 1 MSROI enabled.
GTSCEN 6	0	**Graceful Transmit Stop Complete Interrupt Enable**	0 GTSCI disabled. 1 GTSCI enabled.
BTEN 7	0	**Babbling Transmitter Interrupt Enable**	0 BTI disabled. 1 BTI enabled.
TXCEN 8	0	**Transmit Control Interrupt Enable**	0 TCI disabled. 1 TCI enabled.
TXEEN 9	0	**Transmit Error Interrupt Enable**	0 TEI disabled. 1 TEI enabled.
TXBEN 10	0	**Transmit Buffer Interrupt Enable**	0 TBI disabled. 1 TBI enabled.
TXFEN 11	0	**Transmit Frame Interrupt Enable**	0 TFI disabled. 1 TFI enabled.
IEEN 12	0	**Insertion Error Interrupt Enable**	0 IEI disabled. 1 IEI enabled.
LCEN 13	0	**Late Collision Enable**	0 LC disabled. 1 LC enabled.
CRLEN 14	0	**Collision Retry Limit Enable**	0 CRL disabled. 1 CRL enabled.
XFUNEN 15	0	**Transmit FIFO Underrun Enable**	0 TFU disabled. 1 TFU enabled.
RXBEN0 16	0	**Receive Buffer Queue 0 Interrupt Enable**	0 RBQ0I disabled. 1 RBQ0I enabled.
RXBEN1 17	0	**Receive Buffer Queue 1 Interrupt Enable**	0 RBQ10I disabled. 1 RBQ1I enabled.
RXBEN2 18	0	**Receive Buffer Queue 2 Interrupt Enable**	0 RBQ2I disabled. 1 RBQ2I enabled.
RXBEN3 19	0	**Receive Buffer Queue 3 Interrupt Enable**	0 RBQ3I disabled. 1 RBQ3I enabled.
— 20–22	0	Reserved. Write to zero for future compatibility.	
GRSCEN 23	0	**Graceful Receive Stop Complete Interrupt Enable**	0 GRSCI disabled. 1 GRSCI enabled.
RXFEN0 24	0	**Receive Frame Queue 0 Interrupt Enable**	0 RFQ0I disabled. 1 RFQ0I enabled.
RXFEN1 25	0	**Receive Frame Queue 1 Interrupt Enable**	0 RFQ10I disabled. 1 RFQ1I enabled.
RXFEN2 26	0	**Receive Frame Queue 2 Interrupt Enable**	0 RFQ2I disabled. 1 RFQ2I enabled.

Table 25-20. IMASK Bit Descriptions (Continued)

Bit	Reset	Description	Settings
RXFEN3 27	0	**Receive Frame Queue 3 Interrupt Enable**	0 RFQ3I disabled. 1 RFQ3I enabled.
— 28–31	0	Reserved. Write to zero for future compatibility.	

ECNTRL Ethernet Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—	CLRCNT	AUTOZ	STEN	—				DBDS	—	—	—	—	—	—	—
Type	R	R/W			R				R/W	R						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ECNTRL resets, configures, and initializes the Ethernet controller.

Table 25-21. ECNTRL Bit Descriptions

Bit	Reset	Description	Settings
— 0–16	0	Reserved. Write to zero for future compatibility.	
CLRCNT 17	0	**Clear All Statistics Counters** Specifies whether MSTAT counters continue to increment or are all reset. This bit is self-resetting.	0 MSTAT counters continue to increment. 1 Resets all MSTAT counters.
AUTOZ 18	0	**Automatically Zero-Addressed Statistical Counter Values** Specifies whether a value of zero is automatically written to the addressed counter after a host read or whether you must explicitly write the value of zero. This is a steady state signal and must be set before the Ethernet controller is enabled. It must not be changed without proper care. The addressed counter values are input to the MSTAT module.	0 User must write a value of zero to the addressed counter after a host read. 1 A value of zero is automatically written to the addressed counter after a host read.
STEN 19	0	**Statistics Enabled** Specifies whether statistics are enabled so that internal counters can update. This is a steady state signal and must be set before the Ethernet controller is enabled. It must not be changed without proper care. The values of the internal counters are input to the MSTAT module	0 Statistics are not enabled. 1 Statistics are enabled so that internal counters can update.
— 20–23	0	Reserved. Write to zero for future compatibility.	
DBDS 24	0	**Data Buffer Descriptor Size** Specifies whether the BD size is 8 bytes or 32 bytes. This bit must be set must be set before the Ethernet controller is enabled. It must not be changed without proper care.	0 8-byte BD format. 1 32-byte BD format.
— 25–31	0	Reserved. Write to zero for future compatibility.	

MINFLR Minimum Frame Length Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—									MINFLR						
Type	R															
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

MINFLR specifies the smallest packet to place into a receive buffer indicated by the RxBD.

Table 25-22. MINFLR Field Descriptions

Bit	Reset	Description
— 0–24	0	Reserved. Write to zero for future compatibility.
MINFLR 25–31	1000000	**Minimum Receive Frame Length** (typically 64 decimal) Determines the minimum size of acceptable receive frames. If the Ethernet controller receives an incoming frame shorter than MINFLR, it discards that frame unless RCTRL[RSF] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The largest allowable value for MINFLR is 64. Unlike the MPC8260, in which PADs are added to make the transmit frame equal to MINFLR bytes, if padding is requested, the Ethernet controller always pads transmit frames to 64 bytes, ignoring MINFLR.

PTV Pause Time Value Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	PTE															
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**
	PT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PTV stores the pause duration when the Ethernet controller initiates a pause frame via TCTRL[TFCP]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of pause_quanta, equal to 512 bit times.The pause time can range from 0 to 65535 pause_quanta or 0 to 33553920 bit times. For details, see **Section 25.11**, *Flow Control*, on page 25-38.

Table 25-23. PTV Bit Descriptions

Bit	Reset	Description
PTE 0–15	0	**Extended pause control.** Allows software to add a 16-bit additional control parameter into the pause frame to be sent when TCTRL[TFC_PAUSE]=1. Note that current 802.3 pause frame format requires this parameter to be set to 0.
PT 16–31	0	**Pause Time Value** This pause value is part of the pause frame to be sent when TCTRL[TFC_PAUSE]=1. The pause period is measured in pause_quanta, a speed independent constant of 512 bit-times (unlike slot time). The most-significant octet is transmitted first. with a two pause_quanta resolution. **Note:** Because the pause period has a resolution of two pause_quanta, the value programmed in this field is rounded up to the nearest even number before it is used, as follows: **MAC Parameter Value** — **Pause Period** 0 — None 1 or 2 — 2 × pause_quanta 3 or 4 — 4 × pause_quanta

DMACTRL DMA Control Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—											GRS	GTS	—	—	WOP
Type	R								R/W		R	R/W				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMACTRL configures the DMA controller.

Table 25-24. DMACTRL Bit Descriptions

Bit	Reset	Description	Settings
— 0–26	0	Reserved. Write to zero for future compatibility.	

Table 25-24. DMACTRL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
GRS 27	0	**Graceful Receive Stop** Causes the Ethernet controller to stop receiving frames after receiving the current frame—that is, after a valid end of frame is received. The buffer of the receive frame associated with the EOF is closed and the IEVENT[GRSC] bit is set to generate an interrupt. Because the receive enable bit of the MAC may still be set, the MAC may continue to receive, but the Ethernet controller ignores the receive data until GRS is cleared. When GRS is cleared, the Ethernet controller scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. When GRS is set, you must monitor the graceful receive stop complete IEVENT[GRSC] bit to ensure that the graceful receive stop completed. You can then clear IEVENT[GRSC] and write to receive registers that are accessible to both you and the Ethernet controller hardware without fear of conflict. **Note:** After setting this bit, you must reconfigure the Ethernet controller. See **Section 25.16**.	0 Ethernet controller stops receiving frames after processing the current frame. 1 Ethernet controller resumes receiving frames.
GTS 28	0	**Graceful Transmit Stop** Causes the Ethernet controller to stop transmitting frames after transmitting the current frame, and the IEVENT[GTSC] is set to generate an interrupt. If frame transmission is not currently underway, the GTSC interrupt is generated immediately. Once transmission completes, clearing GTS causes a “restart.” **Note:** After setting this bit, do not clear MACCFG1[TX_EN].	0 Ethernet controller resumes transmitting frames. 1 Ethernet controller stops transmitting frames after processing the current frame.
— 29	0	Reserved. Always write a 0 to this bit after any reset or configuration.	
— 30	0	Reserved. Always write a 1 to this bit after any reset or configuration.	
WOP 31	0	**Wait or Poll** Provides the option for the Ethernet controller to poll a TxBD periodically or to wait for software to tell it to fetch a BD. In the “Wait” mode, the Ethernet controller allows two additional reads of a descriptor that is not ready before it enters a halt state. No interrupt is driven. To resume transmission, software must clear TSTAT[THLT].	0 Poll TxBD based on the setting of DMAMR[PCNT]. 1 Do not poll, but wait for a write to TSTAT[THLT].

DMAMR DMA Maintenance Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—				PCNT		—	DOOS	—				APR		BDPR	
Type	R				R/W		R	R/W	R/W	R/W	R/W	R	R/W		R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
Boot	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1

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

Table 25-25. DMAMR Field Descriptions

Bit	Reset	Description	Settings
— 1–3	0	Reserved. Always write a 0 to this bit after any reset or configuration.	
PCNT 4–5	0	**Polling Count** This field sets the polling frequency of the transmitter. The polling frequency is proportional to the Ethernet MII clock speed (2.5, 25 MHz)	00 512 clocks 01 256 clocks 10 128 clocks 11 64 clocks
— 6	0	Reserved. Always write a 0 to this bit after any reset or configuration.	
DOOS 7	0	**Disable Out-of-Sequence Buffer Descriptor**	0 Out-of-Sequence buffer descriptor polling is enabled 1 Out-of-Sequence buffer descriptor polling is disabled
— 8	0	Reserved. Always write a 0 to this bit after any reset or configuration.	
— 9	0	Reserved. Always write a 1 to this bit after any reset or reconfiguration.	
— 10–11	0	Reserved. Always write a 0 to this bit after any reset or configuration.	
APR 12–13	11	**Alarm Mode Priority** Sets the transmit/receive transaction priority if the Ethernet controller is in alarm mode. In alarm mode (used to help prevent potential underrun/overrun conditions), both reads and writes of TxBDs/RxBDs have a priority set to the APR value.	00 Low priority 01 Mid priority 10 Mid priority 11 High priority
BDPR 14–15	01	**Buffer Descriptor Fetches Priority prior to Alarm Mode** Sets the transmit/receive transaction priority.	00 Low priority 01 Mid priority 10 Mid priority 11 High priority
— 16–31	0	Reserved. Always write a 0 to this bit after any reset or configuration.	

25.17.2 FIFO Control and Status Registers

The registers discussed in this section allow you to change default settings in the FIFO that can be used to optimize operation for performance or for safety. These default settings must be changed carefully to avoid an underrun condition. Underrun is an error condition in which data is not retrieved from external memory quickly enough, leaving the TX FIFO empty before the complete frame is transmitted. Because different combinations of events, several of which you determine, can lead to underrun, the Ethernet controller provides FIFO registers that allow you to select the proper setting to tune the system and obtain the maximum performance with minimal chance of underrun. The principal causes of underrun in the Ethernet controller are:

- Misaligned data buffer addresses
- Small data buffer sizes
- Multiple insertion
- Combinations of the above (that is, multiple insertions in small data payloads)

The minimum data buffer size should be 64 bytes, and data buffers should be 64-byte aligned. Also, one insertion per frame should be used. You can deviate from these recommended values to increase performance or use less memory, but unless the default values of some of the FIFO registers are adjusted, the probability of an underrun may also increase. The FTXTHR (default is 256 entries or 1 KB) indicates the amount of data required to be in the FIFO before starting the transmission of a frame. The FTXSTR (default is 128 entries or 512 bytes) indicates when the amount of data in the FIFO is so low that the risk of underrun is extremely high. The FTXSTSHR (default is 256 entries or 1 KB) contains the watermark level to be used for exiting the starve state. These registers allow you to make the proper trade-off. If triggered, the starve mode, for instance, automatically raises the priority of Ethernet controller fetches from memory.

FRXSTATR FIFO Receive Status Register

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

FRXSTATR contains the status bits of the Rx FIFO controller. This register is read/write by software. This register is cleared at system reset.

Table 25-26. FRXSTATR Field Descriptions

Bit	Reset	Description	Settings
— 0–27	0	Reserved.	
PFS 27	0	**Pause Frame Sent** A pause frame was sent.	0 No pause frame sent. 1 Pause frame sent.
— 28	0	Reserved. Write to zero for future compatibility.	
FULL 29	0	**Rx FIFO Full** Specifies whether the Rx FIFO is full.	0 Not Full 1 Full
EMPTY 30	1	**Rx FIFO Empty** Specifies whether Rx FIFO is empty. It defaults to empty.	0 Not empty. 1 Empty.
31	—	Reserved.	

FRXCTRLR FIFO Receive Control Register

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="14">—</td><td>1</td><td>—</td></tr>
<tr><td>Type</td><td colspan="11">R</td><td colspan="4">R/W</td><td>R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

FRXCTRLR is read/write by software and is cleared at system reset.

Table 25-27. FRXCTRLR Bit Descriptions

Bit	Reset	Description
— 0–29	0	Reserved. Write to zero for future compatibility.
— 30	0	Always write a 1 to this bit after any reset or reconfiguration.
— 31	0	Reserved. Write to zero for future compatibility.

FRXALAR FIFO Receive Alarm Register

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="7">—</td><td colspan="9">FRXALAR</td></tr>
<tr><td>Type</td><td colspan="7">R</td><td colspan="9">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

FRXALAR informs the system of an imminent system overrun condition. It represents the numerical SRAM entry (0–511 for a 2K FIFO) to trigger the alarm function. If the value in the Used Entry Count Register in the FIFO is equal to or greater than that in the Alarm Register, the alarm triggers. This triggered condition is used to change the Rx transaction priority and can be used to send PAUSE frames. This register is read/write by software and is cleared at system reset.

Table 25-28. FRXALAR Field Descriptions

Bit	Reset	Description
0–22	—	Reserved. Write to zero for future compatibility.
FRXALAR 23–31	100000000	**FIFO Receive Alarm** Indicates the value to trigger the receive alarm function. The alarm triggers when the FIFO Receive Used Entry Count is equal to or greater than the FIFO Rx Alarm. The alarm turns off only if the FIFO Receive Used Entry Count falls to less than or equal to the value in the FIFO Rx Shutoff Register.

FRXSHR FIFO Receive Alarm Shutoff Register

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

FRXSHR contains the watermark level to be used for coming out of the alarm state. If the alarm state is in effect and the number of valid entries in the FIFO falls to a value less than or equal to the value in the FRXSHR, the alarm condition ends. This register is read/write by software.

Table 25-29. FRXSHR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FRXSH 23–31	100000000	**FIFO Receive Alarm Shutoff** Indicates the value to turn off the alarm state. The alarm turns off if the FIFO Rx Used Entry Count falls to less than or equal to that of the FIFO Receive Alarm Shutoff Register.

FRXPAR FIFO Receive Panic Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—							FRXPA								
Type	R							R/W								
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

FRXPAR informs the system of an extremely imminent system overrun condition. It represents the numerical SRAM entry (0–511 for a 2K FIFO) to trigger the panic function. If the value in the Used Entry Count Register in the FIFO is equal to or greater than the value in the panic register, a panic alert is triggered.

Table 25-30. FRXPAR Field Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FRXPA 23–31	110000000	**FIFO Receive Panic** The value to trigger the receive panic function, which triggers when the FIFO Receive Used Entry Count is equal to or greater than the value in the. The panic state turns off if the FIFO Rx Used Entry Count falls to less than or equal to the value in the FIFO Receive Panic Shutoff Register (FRXPSR).

FRXPSR FIFO Receive Panic Shutoff Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—							FRXPS								
Type	R							R/W								
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

FRXPSR contains the watermark level to be used for exiting the alarm state. If the alarm state is in effect and the number of valid entries in the FIFO falls to a value less than or equal to the value in the FRXPSR, the panic condition ends. This register is read/write by software.

Table 25-32. FRXPSR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FRXPS 23–31	100000000	**FIFO Receive Panic Shutoff** The value to use to turn off the panic state. The panic state is turned off if the FIFO Rx Used Entry Count falls to less than or equal to the value in FRXPSR.

FTXSTATR FIFO Transmit Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—													FULL	EMPTY	—
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

FTXSTATR contains the status bits of the Tx FIFO controller. This register is read/write by software and is cleared at system reset.

Table 25-33. FTXSTATR Field Descriptions

Bit	Reset	Description	Settings
— 0–28	0	Reserved. Write to zero for future compatibility.	
FULL 29	0	**Tx FIFO Full** Indicates whether the Tx FIFO is full.	0 Not Full 1 Full
EMPTY 30	1	**Tx FIFO Empty** Indicates whether the Tx FIFO is empty.	0 Not empty. 1 Empty.
— 31	0	Reserved. Write to zero for future compatibility.	

FTXTHR FIFO Transmit Threshold Register

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

FTXTHR triggers the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0–511 for a 2K FIFO) to trigger the threshold function. If the number of valid entries in the FIFO Used Entry Count Register is equal to or greater than that in FTXTHR, transmission can begin. This register is read/write by software.

Table 25-34. FTXTHR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FTT 23–31	100000000	**FIFO Transmit Threshold** Specifies the number of entries in the transmit FIFO that trigger the unloading of frame data into the MAC.

FTXSPR FIFO Transmit Space Available Register

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

FTXSPR defines the size of the available transmit space. When the transmit used entry is equal to or greater than the available transmit space, the system sends an indication to the DMA controller. This register is read/write by software.

Table 25-35. FTXSPR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FTXSP 23–31	000010000	**FIFO Transmit Space Available** Indicates the value to indicate when the transmit used entry exceeds the available transmit space, a condition that inhibits DMA writes to the FIFO. **Note:** Before configuring the Ethernet controller to half duplex MII mode, write a value of 0x25 to this field to prevent data loss if multiple collisions occur during data transfer.

FTXSR FIFO Transmit Starve Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—							FTXS								
Type	R							R/W								
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

FTXSR informs the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0–511 for a 2K FIFO) to trigger the starve function. If the Used Entry Count Register shows that the number of valid entries in the FIFO is less than or equal to that in the FTXSR, a starve alert is triggered. This triggered condition is used to throttle back the OCN2CU interface at the Elf Coherency Module (ECM), to change the Tx transaction priority. This register is read/write by software.

Table 25-36. FTXSR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FTXS 23–31	100000000	**FIFO Transmit Starve** Indicate the value to trigger the transmit starve function, which triggers when the number of valid entries in the FIFO is less than or equal to the that in the FTXSR. The starve state turns off when the number of valid entries in the FIFO becomes greater than or equal to that in the FIFO Tx Starve Shutoff Register (FTXSSR).

FTXSSR FIFO Transmit Starve Shutoff Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—							FTXSS								
Type	R							R/W								
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

FTXSSR contains the watermark level to be used for exiting the starve state. If the starve state is in effect and the number of valid entries in the FIFO becomes greater than or equal to the value in the FTSSR, the starve condition ends. This register is read/write by software.

Table 25-37. FTXSSR Bit Descriptions

Bit	Reset	Description
— 0–22	0	Reserved. Write to zero for future compatibility.
FTXSS 23–31	100000000	**FIFO Transmit Starve Shutoff** Indicates the value at which to exit the starve state. The starve state turns off if the number of valid entries in the FIFO becomes greater than or equal to the value in FTXSSR.

25.17.3 Transmit Control and Status Registers

TCTRL Transmit Control Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—				THDF	—						RFCP	TFCP	—		
Type	R				R/W	R							R/W	R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TCTRL configures the transmit block.

Table 25-38. TCTRL Bit Descriptions

Bit	Reset	Description	Settings
— 0–19	0	Reserved. Write to zero for future compatibility.	
THDF 20	0	**Transmit Half-Duplex Flow** Control (written by user) This bit is not self-resetting.	0 Disable back pressure 1 Back pressure is applied to media
— 21–26	0	Reserved. Write to zero for future compatibility.	
RFCP 27	0	**Receive Flow Control Pause Frame** Set if a flow control pause frame is received and the transmitter pauses for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. RFCP is written by Ethernet controller.	0 No flow control pause frame. 1 Flow control pause frame received.

Table 25-38. TCTRL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
TFCP 28	0	**Transmit Flow Control Pause Frame** Set this bit to transmit a pause frame. If this bit is set, the MAC stops transmitting data frames when the current transmission completes. Next, the IEVENT[GTSC] bit generates an interrupt. With transmission of data frames stopped, the MAC transmits a MAC control pause frame with the duration value obtained from the PTV register. The TXC interrupt occurs after the control pause frame is sent. Next, the MAC clears TFCP and resumes transmitting data frames. Note that if the transmitter pauses because of user assertion of GTS or reception of a pause frame, the MAC may still transmit a MAC control pause frame.	0 No pause. 1 Stop transmitting data frames for the specified duration.
— 29–31	0	Reserved. Write to zero for future compatibility.	

TSTAT Transmit Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	THLT	—														
Type	R/W	R														
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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSTAT is a register the Ethernet controller reads/writes to convey DMA status information.

Table 25-39. TSTAT Bit Descriptions

Bit	Reset	Description	Settings
THLT 0	0	**Transmit Halt** The Ethernet controller writes to THLT to inform the user that it is no longer processing transmit frames and that hardware has disabled the transmit DMA function. To restart the transmission function, you must clear this bit by writing a one to it.	0 No hardware initiated transmission halt. 1 Ethernet controller transmission function halted.
— 1–31	0	Reserved. Write to zero for future compatibility.	

TBDLEN TxBD Data Length Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	TBDLEN															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit or insert buffer.

Table 25-40. TBDLEN Bit Descriptions

Bit	Name	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
TBDLEN 16–31	0	**TxBD Data Length** Specifies the length of the transmit or insert buffer. The DMA module writes to TBDLEN internally. The transmit channel remains active until TBDLEN contains a value of 0.

CTBPTR Current TxBD Pointer

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	CTBPTR															
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**
	CTBPTR													—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTBPTR contains the low-order bits of the address of the TxBD that is either currently being processed or processed most recently.

Table 25-41. CTBPTR Field Descriptions

Bit	Reset	Description
CTBPTR 0–28	0	**Current Transmit Buffer Descriptor Pointer** The value of this field increments by eight (bytes) or 32 (bytes), subject to ECNTRL[DBDS], each time a buffer descriptor is read from memory. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned so that bits 27 and 28 are reserved in 32-byte mode. The DMA module writes to CTBPTR internally.
— 29–31	0	Reserved. Write to zero for future compatibility.

TBPTR TxBD Pointer

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	TBPTR															
Type	R/W															
Reset	0	0	0	0	1	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**
	TBPTR													-—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TBPTR contains the low-order 32 bits of the next transmit buffer descriptor address.This register takes on the value of TBASE when the TBASE register is written by software.

Table 25-42. TBPTR Bit Descriptions

Bit	Reset	Description
TBPTR 0-28	0	**Transmit Buffer Descriptor Pointer** The TBPTR register is internally written by the DMA module. The value increments by eight (bytes) or 32 (bytes), subject to ECNTRL[DBDS], each time a descriptor is read from memory. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned. This means that bits 27 and 28 are reserved in 32-byte mode.
— 29-31	0	Reserved. Write to zero for future compatibility.

TBASE Transmit Descriptor Base Address

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	TBASE															
Type	R/W															
Reset	0	0	0	0	1	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**
	TBASE													-—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TBASE is the register to which you write the TxBD base address. The value must be divisible by eight for 8-byte data BDs or by 32 for 32-byte data BDs.

Table 25-43. TBASE Bit Descriptions

Bit	Reset	Description
TBASE 0-28	0	**Transmit Descriptor Base Address** Defines the starting location in the memory map for the Ethernet controller TxBDs. In 8-byte mode (ECNTRL[DBDS] is cleared), this field must be 8-byte aligned. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned so that bits 27 and 28 are reserved in 32-byte mode. In addition to setting the W (wrap) bit in the last BD, you can select how many BDs to allocate for the transmit packets. You must initialize TBASE before enabling the Ethernet controller transmit function.
— 29-31	0	Reserved. Write to zero for future compatibility.

OSTBD Out-of-Sequence TxBD Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	R	PAD	W	I	L	TC	DEF	—	LC	RL	RC				UN	—
Type	R/W															R
Reset	0	0	0	0	1	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**
	OSTBDLEN															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OSTBD is the out-of-sequence TxBD register, which includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. The OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. You must set OSTBD[L] while preparing this BD. If a flow control frame is sent and OSTBD[I] is set, a TXC event is generated after frame transmission. This register must be cleared while not in use. When the Ethernet controller is in pause mode, the out-of-sequence BD cannot be used to send another flow control frame because the MAC regards it as a regular TxBD.

Table 25-44. OSTBD Bit Descriptions

Bit	Reset	Description	Settings
R 0	0	**Ready** Indicates whether the data buffer associated with a BD is ready for transmission. When this bit is cleared, you can manipulate this BD or its associated data buffer. The Ethernet controller clears this bit after the buffer is transmitted or after an error condition is encountered. When this bit is set, you cannot write to any fields of this BD. This bit is written by the Ethernet controller and the user.	0 The data buffer associated with this BD is not ready for transmission. 1 The data buffer that the user has prepared for transmission was not transmitted or is currently being transmitted.
PAD 1	0	**Padding for Short Frames** Enables/disables padding for short frames. This bit is cleared only while one TxBD is used (L is set) and the MACCFG2R[PADCRC, CRCEN] bits are cleared. Otherwise, pads are added to short frames. When PAD is set, padding bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 device, which pads up to the MINFLR value, the Ethernet controller always pads up to the **IEEE** minimum frame length of 64 bytes.	0 Do not add padding to short frames unless TxBD[TC] is set. 1 Add padding to short frames.
W 2	0	**Wrap** Wrap, written by user. This bit is ignored by the Ethernet controller.	0 The next BD is found in the consecutive location 1 The next BD is found at the location defined by the user.
I 3	0	**Interrupt** Causes an interrupt if IEVENT[TXFEN] is enabled. This bit is written by user.	0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXF] is set after this buffer is serviced.
L 4	1	**Last in Frame** The OSTBD is always the last in the frame, so L is always set. This bit is hardwired to a value of 1	

Table 25-44. OSTBD Bit Descriptions (Continued)

Bit	Reset	Description	Settings
TC 5	0	**Tx Cyclic Redundancy Check** Indicates whether a CRC sequence occurs after the TxBD is transmitted. This bit is cleared only while one TxBD is used (L is set), TxBD[PAD] is cleared, and the MACCFG2R[PADCRC, CRCEN] bits are cleared. Otherwise, a CRC is added to all frames. This bit is written by user.	0 End transmission immediately after the last data byte, unless TxBD[PAD] is set. 1 Transmit the CRC sequence after the last data byte.
DEF 6	0	**Defer Indication** Hardware updates this bit after transmitting a frame that is used as a defer indicator. Software or the user updates this bit while building a transmit BD if the BD is to be used as a hardware event indicator.	0 This frame was not deferred. 1 This frame did not have a collision before it was sent, but it was sent late because of deferring.
— 7	0	Reserved. Write to zero for future compatibility.	
LC 8	0	**Late Collision** When this bit is set, the Ethernet controller terminates the transmission and updates this bit value.	0 No late collision. 1 A collision occurred after 64 bytes are sent.
RL 9	0	**Retransmission Limit** When this bit is set, the Ethernet controller terminates the transmission and updates the bit value.	0 Transmission before maximum retry limit is hit 1 The transmitter failed (maximum retry limit + 1) attempts to send a message successfully due to repeated collisions.
RC 10–13	0	**Retry Count** Indicates whether a retry was necessary to send the frame, as well as the number of retries needed. For example, if this field holds a value of 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer. This bit is written by Ethernet controller.	0 The frame is sent correctly the first time. 1 More than zero attempts were needed to send the transmit frame.
UN 14	0	**Underrun** Indicates whether there is a transmitter underrun condition. When this bit is cleared, the data is retrieved from external memory in time to send a complete frame. When this bit is set, the Ethernet controller terminates the transmission and updates UN. This bit is written by Ethernet controller.	0 No underrun encountered. 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer.
— 15	0	Reserved. Write to zero for future compatibility.	
OSTBDLEN 16–31	0	**Out-of-sequence TxBD Data Length** Data length is the number of octets the Ethernet controller should transmit from the BD data buffer. The Ethernet controller never modifies this value. This bit is written by the user.	

OSTBDP Out-of-Sequence Tx Data Buffer Pointer Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	OSTBDP															
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**
	OSTBDP															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OSTBDP is the out-of-sequence Tx data buffer pointer register, which is written by the user and contains the data buffer pointer fields in the same format as a regular TxBD. Together with OSTBD, this register provides the complete 8-byte descriptor. This register must be cleared when it is not in use.

Table 25-45. OSTBDP Bit Descriptions

Bit	Reset	Description
OSTDBP 0–31	0	**Out-of-sequence Tx Data Buffer Pointer** Contains the address of the associated data buffer. There are no alignment requirements for this address.

OS32TBDP Out-of-Sequence 32 Byte Tx Data Buffer Pointer Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	OS32TBDP															
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**
	OS32TBDP															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OS32TBDP is the Tx data buffer pointer for the 32-byte out-of-sequence TxBD. This register contains the data buffer pointer fields in the same format as a regular 32-byte TxBD. Together with OSTBDP, it provides the pointer for the 32-byte format BD. This area must be cleared while not in use.

Table 25-46. OS32TBDP Field Descriptions

Bit	Reset	Description
OS32TBDP 0–31	0	**Out-of-sequence 32-byte Tx Data Buffer Pointer** Contains the address of the associated data buffer. There are no alignment requirements for this address. The buffer must reside in memory external to the Ethernet controller. The Ethernet controller never modifies this value.

OS32IPTR Out-of-Sequence 32 Byte TxBD Insert Pointer Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	OS32IPTR															
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**
	OS32IPTR															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OS32IPTR is the Tx insert buffer pointer for the 32-byte out-of-sequence TxBD. This register contains the buffer pointer fields in the same format as a regular 32-byte TxBD. This register must be cleared when it is not in use.

Table 25-47. OS32IPTR Bit Descriptions

Bit	Reset	Description
OS32IPTR 0–31	0	**Out-of-sequence Tx Insert Buffer Pointer** Contains the address of a buffer to contain the inserted data. There are no alignment requirements. The buffer resides in memory external to the Ethernet controller. The inserted data is placed inside the transmit frame as defined by the insert index and insert length fields. You can choose insertion by replacement or insertion by expansion (see **Figure 25-24** on page 25-37). You are responsible for ensuring that the value in the frame's type/length field is correct. The combination of the data length, insert index, and insert length fields must be valid or else IEVENT[IE] is set. Insertion errors occur for expansion or replacement if the index is greater than the TxBD[DL]. Insertion error occurs for replacement if the index + length is greater than TxBD[DL]. The Ethernet controller never modifies this value.

OS32TBDR Out-of-Sequence 32 Byte TxBD Reserved Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—													IE	IT	
Type	R													R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OS32TBDR is the first reserved field of the 32-byte out-of-sequence TxBD. This register contains the insert error (IE) and insert type (IT) fields but largely contains reserved bits. **Table 25-48** describes the OS32TBDR.

Table 25-48. OS32TBDR Bit Descriptions

Bit	Reset	Description	Setting
— 0–28	0	Reserved. Write to zero for future compatibility.	
IE 29	0	**Insertion Error** Indicates whether there is an insertion error. If this bit is set, the insert length may be too large for the transmit buffer. The Ethernet controller terminates the transmission and updates IE and sets IEVENT[IE] to generate an interrupt. An insertion error occurs for expansion or replacement if the index is greater than the TxBD[DL]. An insertion error occurs for replacement if the index + length is greater than the TxBD[DL]. This bit is written by Ethernet controller.	0 If TxBD[IT] = 01 or 10, no insertion error. 1 An error occurred during an attempt to insert data.
IT 30–31	0	**Insert Type** Defines the type of insertion to be perform.	00 No insertion 01 Replacement 10 Expansion 11 Reserved

OS32IIL Out-of-Sequence 32 Byte TxBD Insert Index/Length Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	OS32INX															
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**
	OS32ILEN															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OS32IIL contains the insert index and insert length fields for the 32-byte out-of-sequence TxBD. This register must be cleared while not in use.

Table 25-49. OS32IIL Bit Descriptions

Bit	Reset	Description
OS32INX 0–15	0	**Out-of-sequence 32-byte TxBD Insert Index** Contains the number of bytes to jump within the transmit buffer before inserting data into the buffer to which the Tx insert buffer pointer is pointing. The value in this field must be less than or equal to TxBD[DL]. The Ethernet controller never modifies this value. If insert index = 0, insertion starts at the beginning of the buffer.
OS32ILEN 16–31	0	**Out-of-sequence 32-byte TxBD Insert Length** Contains the number of bytes of data that is inserted into the buffer. If this field holds a value of zero, no insertion occurs, and the values in the Tx insert buffer pointer and the insert index field are considered invalid. The Ethernet controller never modifies this value.

25.17.4 Receive Control and Status Registers

RCTRL Receive Control Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
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**
	—				CRCLSEL	—				PMEN	—	BCREJ	PROM	RSF	RA	—
Type	R				R/W	R				R/W	R	R/W				R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RCTRL controls the operational mode of the receive block. It must be written only after a system reset (at initialization) or if DMACTRL[GRS] is cleared.

Table 25-50. RCTRL Bit Descriptions

Bit	Reset	Description	Settings
— 0–19	0	Reserved. Write to zero for future compatibility.	
CRCLSEL 20	0	**CRC LSB Select** Specifies whether the eight most significant bits of the CRC remainder are to be used to map the destination address to the hash table entry.	0 Use the eight most significant bits of the CRC remainder to map the DA to the hash table entry. 1 Ignore the most-significant bit and use the next eight most-significant bits of the CRC remainder to map the DA to the hash table entry.
— 21–24	0	Reserved. Write to zero for future compatibility.	
PMEN 25	0	**Pattern Match Enabled** Enables/disables pattern matching.	0 Pattern match is disabled 1 Pattern match is enabled
— 26	0	Reserved. Write to zero for future compatibility.	
BCREJ 27	0	**Broadcast Frame Reject** Rejects frames with a destination address (DA) = FFFF_FFFF_FFFF unless RCTRL[PROM] is set. If both BCREJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the MISS (M) bit is set in the receive BD. A pattern match reject hit or the setting of RA prevents a frame from being accepted even if RCTRL[PROM] is set.	0 No effect. 1 Reject frames with the destination address = FFFF_FFFF_FFFF.
PROM 28	0	**Promiscuous Mode** All frames, regardless of the addresses, are accepted unless there is a pattern match reject hit or RA is set.	0 No effect. 1 Accept all frames.
RSF 29	0	**Receive Short Frame Mode** Enables the reception of frames shorter than MINFLR bytes. **Note:** For short frames to be received when RSF=1, a DA hit or a pattern match hit accept needs to occur. When RSF=0, all frames shorter than MINFLR are automatically rejected.	0 No effect. 1 Receive frames shorter than MINFLR bytes.

Table 25-50. RCTRL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
RA 30	0	**Reject All Mode** Do not accept any frames on the basis of a DA hit. Only frames with pattern match accept hits are received. This bit is ignored if PMEN is cleared.	0 No effect. 1 No frames accepted on the basis of a DA hit.
— 31	0	Reserved. Write to zero for future compatibility.	

RSTAT Receive Status Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	RHLT	—							Q0HLT	Q1HLT	Q2HLT	Q3HLT	—			
Type	R								R/W				R			
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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSTAT is written by the Ethernet controller when the receiver runs out of descriptors or the receive halts because an error condition occurred while a frame was being received. Software should clear the appropriate Q*n*HLT bits to bring the Ethernet controller receiver function out of a halt state.

Table 25-51. RSTAT Bit Descriptions

Bit	Reset	Description	Settings
RHLT 0	0	**Receive Halted** The logical OR of the Q*n*HLT bits.	0 None of the Ethernet RXDB queues are halted. 1 Ethernet controller receive activity in at least one RXBD queue is halted.
— 1–7	0	Reserved. Write to zero for future compatibility.	
Q0HLT 8	0	**RxBD Queue 0 Halted** Halts all receive activity in RxBD queue 0. The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.	0 RxBD queue 0 is enabled for Ethernet reception. 1 All Ethernet controller receive activity to RxBD queue 0 is halted.
Q1HLT 9	0	**RxBD Queue 1 Halted** Halts all receive activity in RxBD queue 1. The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.	0 RxBD queue 1 is enabled for Ethernet reception. 1 All Ethernet controller receive activity to RxBD queue 1 is halted.
Q2HLT 10	0	**RxBD Queue 2 Halted** Halts all receive activity in RxBD queue 2. The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.	0 RxBD queue 2 is enabled for Ethernet reception. 1 All Ethernet controller receive activity to RxBD queue 2 is halted.

Table 25-51. RSTAT Bit Descriptions (Continued)

Bit	Reset	Description	Settings
Q3HLT 11	0	**RxBD Queue 3 Halted** Halts all receive activity in RxBD queue 3. The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.	0 RxBD queue 3 is enabled for Ethernet reception. 1 All Ethernet controller receive activity to RxBD queue 3 is halted.
— 12–31	0	Reserved. Write to zero for future compatibility.	

RBDLEN RxBD Data Length Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	RBDLEN															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer.

Table 25-52. RBDLEN Bit Descriptions

Bit	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RBDLEN 16–31	0	**RxBD Data Length** If this bit is cleared, all activity in the receive channel stops. The DMA module writes to RBDLEN internally, and you should avoid writing to this field.User writes can cause unpredictable Ethernet controller behavior

CRBPTR Current RxBD Pointer

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	CRBPTR															
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**
	CRBPTR													—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRBPTR contains the address of the RxBD that is either being processed or was most recently processed. The DMA module writes to this register internally, and you should never write to it.

Table 25-53. CRBPTR Bit Descriptions

Bit	Reset	Description
CRBPTR 0–28	0	**Current Receive Buffer Descriptor Pointer** The value in this field increments by eight (bytes) or 32 (bytes), subject to ECNTRL[DBDS], each time a descriptor is read from memory. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned so that bits 27 and 28 are reserved in 32-byte mode.
— 29–31	0	Reserved. Write to zero for future compatibility.

MRBLR0R1 Maximum Receive Buffer Length R0R1 Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	MRBLR1										—					
Type	R/W										R					
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**
	MRBLR0										—					
Type	R/W										R					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MRBLR0R1 specifies for the Ethernet controller how much space is available in each receive buffer to which the RxBD is pointing.

Table 25-54. MRBLR0R1 Field Descriptions

Bit	Reset	Description
MRBLR1 10–9	0	**Maximum Receive Buffer Length for Ring 1** Specifies the number of bytes that the Ethernet controller receiver writes to receive buffer ring 1 before moving to the next buffer. You write to MRBLR1 with a multiple of 64 for all modes. The Ethernet controller can write fewer bytes to the buffer than the value set in MRBLR1 if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR1 value. Therefore, user-supplied buffers must be at least as large as MRBLR1. Note that you can assign transmit buffers varying lengths by programming TxBD[DL] as needed, and they are not affected by the value in MRBLR*n*. MRBLR*n* is not to be changed dynamically while the Ethernet controller is operating. Change MRBLR*n* only when the Ethernet controller receive function is disabled.
— 10–15	0	To ensure that MRBL1 and MRBLR0 are multiples of 64, these bits are reserved and must be cleared.
MRBLR0 16–25	0	**Maximum Receive Buffer Length for Ring 0** Specifies the number of bytes that the Ethernet controller receiver writes to receive buffer ring 0 before moving to the next buffer. You write to MRBLR0 register with a multiple of 64 for all modes. The Ethernet controller can write fewer bytes to the buffer than the value set in MRBLR0 if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR0 value. Therefore, user-supplied buffers must be at least as large as the MRBLR0.
— 26–31	0	To ensure that MRBLR1 and MRBLR0 are multiples of 64, these bits are reserved and should be cleared.

MRBLR2R3 Maximum Receive Buffer Length R2R3 Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	MRBLR3										—					
Type	R/W										R					
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**
	MRBLR2										—					
Type	R/W										R					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MRBLR2R3 specifies for the Ethernet controller how much space is available in each receive buffer to which the RxBD is pointing.

Table 25-55. MRBLR2R3 Field Descriptions

Bit	Reset	Description
MRBLR3 0–9	0	**Maximum Receive Buffer Length for Ring 3** Specifies the number of bytes that the Ethernet controller receiver writes to receive buffer ring 3 before moving to the next buffer. You write to MRBLR3 register with a multiple of 64 for all modes. The Ethernet controller can write fewer bytes to the buffer than the value set in MRBLR3 if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR3 value. Therefore, user-supplied buffers must be at least as large as MRBLR3. Note that you can assign transmit buffers varying lengths by programming TxBD[DL], as needed. They are not affected by the value in a MRBLR*n*. MRBLR*n* is not to be changed dynamically while the Ethernet controller is operating. Change MRBLR*n* only when the Ethernet controller receive function is disabled.
— 10–15	0	To ensure that MRBLR2 and MRBLR3 are multiples of 64, these bits are reserved and should be cleared.
MRBLR2 16–25	0	**Maximum Receive Buffer Length for Ring 2** Specifies the number of bytes that the Ethernet controller receiver writes to receive buffer ring 2 before moving to the next buffer. You write to MRBLR2 with a multiple of 64 for all modes. The Ethernet controller can write fewer bytes to the buffer than the value set in MRBLR2 if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR2 value. Therefore, user-supplied buffers must be at least as large as MRBLR2.
— 26–31	0	To ensure that MRBLR2 and MRBLR3 are multiples of 64, these bits are reserved and should be cleared.

RBPTRn RxBD Pointer 0-3

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	RBPTRn															
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**
	RBPTRn													—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBPTRn contains the receive buffer descriptor address for the respective ring.This register takes on the value of RBASEn when the RBASEn register is written by software.

Table 25-56. RBPTRn Field Descriptions

Bit	Reset	Description
RBPRTn 0–28	0	**Receive Buffer Descriptor Pointer 0 - 3** The RBPTRn register is internally written by the DMA module. The value increments by eight (bytes) or 32 (bytes), depending on ECNTRL[DBDS], each time a descriptor is read from memory. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned. This means that bits 27 and 28 are reserved in 32-byte mode.
— 29–31	0	Reserved. Write to zero for future compatibility.

RBASEn Receive Descriptor Base Address 0-3

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	RBASEn															
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**
	RBASEn													—		
Type	R/W													R		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBASEn are written by the user with the RxBD base address and must be divisible by eight for 8-byte BDs or by 32 for the 32-byte extended BDs. There are four RBASELn registers for four RxBD queues to support filing of frames based on address recognition or pattern match. Refer to the pattern match attributes register PATTRBn[QC] for information on tying a RBASEn to a RxBD ring.

Table 25-57. RBASEn Field Descriptions

Bit	Reset	Description
RBASEn 0–28	0	**Receive Descriptor Base Address n** Defines the starting location in the memory map for the Ethernet controller RxBDs. In 8-byte mode (ECNTRL[DBDS] is cleared), this field must be 8-byte aligned. In 32-byte mode (ECNTRL[DBDS] is set), this field must be 32-byte aligned so that bits 27 and 28 are reserved in 32-byte mode. Together with setting the wrap (W) bit in the last BD, you can select how many BDs to allocate for the receive packets.
— 29–31	0	Reserved. Write to zero for future compatibility.

25.17.5 MAC Registers

MAC configuration registers 1 and 2 configure the MAC in multiple ways:

- *Adjusting the preamble length*. The adjustment is made from the nominal seven bytes to some other (non-zero and not greater than 7) value.
- *Varying pad/CRC combinations*. Three different pad/CRC combinations are provided to handle a variety of system requirements. Frames that already have a valid FCS field are the most simple. The CRC is checked and reported via the transmit statistics vector (TSV[51–0]). The other two options include appending a valid CRC or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

MACCFG1R MAC Configuration 1 Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	SRESET	—											RRXM	RTXM	RRXF	RTXF
Type	R/W		R										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**
	—							MIILB	—		RXFL	TXFL	SYRXEN	RXEN	SYTXEN	TXEN
Type	R							R/W	R		R/W		R	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MACCFG1R is a user-programmable register that configures several MAC features.

Table 25-58. MACCFG1R Field Descriptions

Bit	Reset	Description	Settings
SRESET 0	0	**Soft Reset** Puts all MAC modules into reset. For details on setting this bit, see **Section 25.16**, *Initialization and Reset*.	0 Normal operation. 1 Soft reset.
— 2–11	0	Reserved. Write to zero for future compatibility.	
RRXM 12	0	**Reset Rx MAC** Puts the receive MAC control block into reset. This block detects control frames and contains the pause timers.	0 Normal operation. 1 Resets the MAC receive block.
RTXM 13	0	**Reset Tx MAC** Puts the PETMC transmit MAC control block into reset. This block multiplexes data and control frame transfers. It also responds to XOFF pause control frames.	0 Normal operation. 1 Resets the PETMC transmit control block.
RRXF 14	0	**Reset Rx Function** Puts the receive function block into reset. This block performs the receive frame protocol.	0 Normal operation. 1 Resets the receive function block.
RTXF 15	0	**Reset Tx Function** Puts the transmit function block into reset. This block performs the frame transmission protocol.	0 Normal operation. 1 Resets the transmit function block.
— 16–22	0	Reserved. Write to zero for future compatibility.	

Table 25-58. MACCFG1R Field Descriptions (Continued)

Bit	Reset	Description	Settings
MIILB 23	0	**MII Loopback** Causes the MII MAC transmit outputs to be looped back to the MAC receive inputs.	0 Normal operation. 1 MII Loopback mode.
— 24–25	0	Reserved. Write to zero for future compatibility.	
RXFL 26	0	**Rx Flow** Causes the receive MAC control to detect and act on pause flow control frames.	0 Ignore receive pause flow control frames. 1 Detect and act on receive pause flow control frames.
TXFL 27	0	**Tx Flow** Allows the transmit MAC control to send pause flow control frames if the system requests them. Clearing this bit prevents the transmit MAC control from sending flow control frames.	0 No transmit pause flow control frames. 1 Detect and act on transmit pause flow control frames.
SYRXEN 28	0	**Synchronized Rx Enable** Receive enable synchronized to the receive stream.	0 Frame reception is not enabled. 1 Frame reception is enabled.
RXEN 29	0	**Receive Enable** Allows the MAC to receive frames from the PHY. Clearing this bit prevents the reception of frames.	0 MAC cannot receive frames. 1 MAC can receive frames.
SYTXEN 30	0	**Synchronized TX Enable** Transmit Enable synchronized to the transmit stream.	0 Frame transmission is not enabled. 1 Frame transmission is enabled.
TXEN 31	0	**Transmit Enable** Allows the MAC to transmit frames from the system. Clearing this bit prevents the transmission of frames. This bit is cleared by default.	0 MAC cannot transmit frames. 1 MAC can transmit frames.

MACCFG2R MAC Configuration 2 Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	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**
	PREAL				—						—	LENC	—	PADCRC	CRCEN	FDUP
Type	R/W				R		R/W		R		R	R/W	R	R/W		
Reset	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Boot	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0

MACCFG2R is a user-programmable register that configures several MAC features.

Table 25-59. MACCFG2R Field Descriptions

Bit	Reset	Description	Settings
— 0–15	0	Reserved. Write to zero for future compatibility.	
PREAL 16–19	0111	**Preamble Length** Determines the length in bytes of the preamble field in the packet. The maximum value is 0x7, which is the default value. A preamble length of 0 is not supported.	
— 20–25	000100	Reserved. Write to 000100 for future compatibility.	
26	0	Reserved. Write to zero for future compatibility.	
LENC 27	0	**Length Check** Causes the MAC to check the frame length field to ensure that it matches the actual data field length.	0 No length field checking. 1 The MAC checks the frame length field.
— 28	0	Reserved. Write to zero for future compatibility.	
PADCRC 29	0	**PAD/CRC** Indicates padding and CRC status.	0 No padding and no CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame.
CRCEN 30	0	**CRC Enable** Enables MAC CRC checking. **Note:** If the configuration bit PAD/CRC ENABLE or the per-packet PAD/CRC ENABLE is set, CRC ENABLE is ignored.	0 Frame valid with valid CRC. 1 Frame or CRC not valid.
FDUP 31	O	**Full Duplex** Selects half-duplex or full-duplex mode.	0 Half-duplex mode. 1 Full-duplex mode.

IPGIFGR Inter-Packet Gap/Inter-Frame Gap Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—	NBBIPG1							—	NBBIPG2						
Type	R	R/W							R	R/W						
Reset	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	MIFGE								—	BBIPG						
Type	R/W								R	R/W						
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0

IPGIFGR specifies the amount of time a transmitting station must wait between packets.

Table 25-60. IPGIFGR Bit Descriptions

Bit	Reset	Description
— 0	0	Reserved. Write to zero for future compatibility.
NBBIPG1 1–7	1000000	**Non-Back-to-Back Inter-Packet Gap, Part 1** This programmable field represents the optional carrier sense window referenced in **IEEE** Std. 802.3™ 4.2.3.2.1, Carrier Deference. If a carrier is detected during the timing of IPGR1, the MAC defers to the carrier. However, if the carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits the data, causing a collision and thus ensuring fair access to the medium. The range of values is 0x00 to IPGR2. The default is 0x40 (64d), which is in accordance with the two-thirds/one-third guideline.
— 8	0	Reserved. Write to zero for future compatibility.
NBBIPG2 9–15	1100000	**Non-Back-to-Back Inter-Packet Gap, Part 2** This programmable field represents the non-back-to-back inter-packet gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
MIFGE 16–23	01010000	**Minimum IFG Enforcement** This programmable field represents the minimum number of bits of the IFG to enforce between frames. A frame with an IFG is less than that programmed is dropped. The default setting of 0x50 (80d) represents half the nominal minimum IFG, which is 160 bits.
— 24	0	Reserved. Write to zero for future compatibility.
BBIPG 25–31	1100000	**Back-to-Back Inter-Packet Gap** This programmable field represents the IPG between back-to-back packets. This is the IPG parameter used exclusively in Full-Duplex mode and Half-Duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

HAFDUPR Half-Duplex Register

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

HAFDUPR controls the carrier-sense multiple access/collision detection (CSMA/CD) logic. Half-duplex is supported for both 10 Mbps and 100 Mbps operation. This register is user-programmable.

Table 25-61. HAFDUPR Bit Descriptions

Bit	Reset	Description	Settings
— 0–7	0	Reserved. Write to zero for future compatibility.	
ABEBT 8–11	0101	**Alternate Binary Exponential Back-off Truncation** In use while ABEB is set. The value programmed is substituted for the Ethernet standard value of ten. The default is 0xA.	
ABEB 12	0	**Alternate Binary Exponential Back-off Enable** Configures the Tx MAC to use the alternate binary exponential back-off truncation (ABEBT) setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum back-off time.	0 Tx MAC follows the binary exponential back-off rule. 1 Tx MAC uses alternate binary exponential back-off rule.
BPNB 13	0	**Back Pressure No Back-off** Configures the Tx MAC to retransmit the data immediately after a collision, during a back pressure operation.	0 Tx MAC follows the binary exponential back-off rule. 1 No back-off during a back pressure operation.
NB 14	0	**No Back-off** Configures the Tx MAC to immediately re-transmit following a collision.	0 Tx MAC follows the binary exponential back-off rule. 1 No back-off.
ED 15	1	**Excess Defer** Configures the Tx MAC to allow the transmission of a packet that is excessively deferred.	0 Abort an excessively deferred packet. 1 Transmit an excessively deferred packet.
RTXM 16–19	1111	**Retransmission Maximum** This programmable field specifies the number of retransmission attempts following a collision before the packet is aborted due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d).	
— 20–25	0	Reserved. Write to zero for future compatibility.	
CW 26–31	110111	**Collision Window** This programmable field represents the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.	

MAXFRMR Maximum Frame Length Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	MF															
Type	R/W															
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

MAXFRMR is a user-programmable register that specifies the maximum frame size in both the transmit and receive directions.

Table 25-62. MAXFRM Descriptions

Bit	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
MF 16–31	0000011000000000	**Maximum Frame** By default this field is set to 0x0600. It sets the maximum frame size in both the transmit and receive directions.

IFSTATR Interface Status Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—						EXD	—								
Type	R						R/W	R								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IFSTATR is a user-programmable register that ensures that the MAC does not excessively defer a transmission.

Table 25-63. IFSTAT Bit Descriptions

Bit	Reset	Description
— 0–21	0	Reserved. Write to zero for future compatibility.
EXD 22	0	**Excess Deferral** Set if the MAC excessively defers a transmission. It clears if read. This bit latches high.
— 23–31	000000001	Reserved.

MACSTADDR1R MAC Station Address Part 1 Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SA1								SA2							
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**
	SA3								SA4							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MACSTNADDR1 is one of two user-programmable registers holding the physical address that the Ethernet controller compares with the destination address field of the received frame. If the destination does not match the station address, the Ethernet controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. The value of the station address written into MACSTNADDR1 and MACSTNADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678abcd, MACSTNADDR1 is set to 0xcdab7856 and MACSTNADDR2 is set to 0x34120000. When the user reads MACSTNADDR1, 0xcdab7856 will be returned. A read of MACSTNADDR2 will return a value of 0x34120000.Note, the I/G and U/L bits of the frame’s DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

Table 25-64. MACSTNADR1 Bit Descriptions

Bit	Reset	Description
SA1 0–7	0	**Station Address, Octet 1** Holds the first octet of the station address, which defaults to a value of 0x00.
SA2 8–15	0	**Station Address, Octet 2** Holds the second octet of the station address, which defaults to a value of 0x00.
SA3 16–23	0	**Station Address, Octet 3** Holds the third octet of the station address, which defaults to a value of 0x00.
SA4 24–31	0	**Station Address, Octet 4** Holds the fourth octet of the station address, which defaults to a value of 0x00.

MACSTADDR2R MAC Station Address Part 2 Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SA5								SA6							
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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MACSTNADDR2 is one of two user-programmable registers holding the physical address that the Ethernet controller compares with the destination address field of the received frame. If the destination does not match the station address, the Ethernet controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. The value of the station address written into MACSTNADDR1 and MACSTNADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678abcd, MACSTNADDR1 is set to 0xcdab7856 and MACSTNADDR2 is set to 0x34120000.

Note: The I/G and U/L bits of the frame DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

Table 25-65. MACSTNADDR2 Bit Descriptions

Bit	Reset	Description
SA5 0–7	0	**Station Address, Octet 5** Holds the fifth octet of the station address, which defaults to a value of 0x00.
SA6 8–15	0	**Station Address, Octet 6** Holds the sixth octet of the station address, which defaults to a value of 0x00.
— 16–31	0	Reserved. Write to zero for future compatibility.

25.17.6 MII Management Registers

MIIMCFGR MII Management Configuration Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	RMGT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	R/W	R														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	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
	—	—	—	—	—	—	—	—	—	—	—	NOPRE	—	MGTCS		
Type	R											R/W	R	R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

MIIMCFGR is a user-programmable register that contains configuration bits for various features and modes of the Ethernet controller.

Table 25-66. MIIMCFGR Field Descriptions

Bit	Reset	Description	
RMGT 0	0	**Reset Management** Resets the MII management. Clearing this bit allows the MII management to perform management read/write cycles if requested.	0 MII management enabled. 1 MII management reset.
— 1–26	0	Reserved. Write to zero for future compatibility.	
NOPRE 27	0	**No Preamble** Setting this bit causes the MII management to suppress preamble generation and reduce the management cycle from 64 clocks to 32 clocks, in accordance with **IEEE** Std. 802.3 22.2.4.4.2. Clearing this bit causes the MII management to perform management read/write cycles with the 32 clocks of preamble. It is cleared by default.	0 Preamble generated. 1 Preamble suppressed.
— 28	0	Reserved. Write to zero for future compatibility.	
MGTCS 29–31	0	**Management Clock Select** Determines the clock frequency of the management clock (EC_MDC). Its default value is 000.	000 BUSES_CLOCK/8 divided by 4. 001 BUSES_CLOCK/8 divided by 4. 010 BUSES_CLOCK/8 divided by 6. 011 BUSES_CLOCK/8 divided by 8. 100 BUSES_CLOCK/8 divided by 10. 101 BUSES_CLOCK/8 divided by 14. 110 BUSES_CLOCK/8 divided by 20. 111 BUSES_CLOCK/8 divided by 28.

MIIMCOMR MII Management Command Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—														SCYC	RCYC
Type	R														R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIMCOMR provides the ability to perform continuous read cycles (called a scan cycle), although scan cycles are not explicitly defined in the standard. If a scan cycle is requested, the device performs repetitive read cycles of the PHY status register, for example. PHY link characteristics can therefore be monitored more efficiently.

Table 25-67. MIIMCOMR Bit Descriptions

Bit	Reset	Description
— 0–29	0	Reserved. Write to zero for future compatibility.
SCYC 30	0	**Scan Cycle** Causes the MII management to perform continuous read cycles, which is useful for monitoring link fail, for example.
RCYC 31	0	**Read Cycle** Causes the MII management to perform a single read cycle. If RCYC is set, an MII management read cycle is performed using the PHY address at MIIMADD[PHYADDR] and the register address (at MIIMADD[RADDR]. The read data is returned in the MIIMSTATR[PHYS] bit.

MIIMADDR MII Management Address Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—			PHYADDR					—			RADDR				
Type	R			R/W					R			R/W				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIMADDR provides access to the PHY address and register address fields of management cycles. Note that the offset varies depending on the Ethernet controller used.

Table 25-68. MIIMADDR Bit Descriptions

Bit	Reset	Description
— 0–18	0	Reserved. Write to zero for future compatibility.
PHYADDR 19–23	0	**PHY Address** The 5-bit PHY address field of management cycles. Up to 31 PHYs can be addressed (0 is reserved). The default value is 0x00.
— 24–26	0	Reserved. Write to zero for future compatibility.
RADDR 27–31	0	**Register Address** The 5-bit register address field of management cycles. Up to 32 registers can be accessed. The default value is 0x00.

MIIMCONR MII Management Control Register

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

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

MIIMCONR is written by the user. Note that the offset varies depending on the Ethernet controller used.

Table 25-69. MIIMCONR Field Descriptions

Bit	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
PHYC 16–31	0	**PHY Control** Causes an MII management write cycle to be performed using this 16-bit data field, the pre-configured PHY address at MIIMADD[PHYADDR]), and the register address (at MIIMADD[RADDR]).

MIIMSTATR MII Management Status Register

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

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

MIIMSTATR is used by the host to read the fields in the MII Management Indicator Register (MIIMIND) (scan, not valid, and busy) indicate availability of each read of the scan cycle.

Table 25-70. MIIMSTATR Bit Descriptions

Bit	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
PHYS 16–31	0	**PHY Status** Following an MII management read cycle, you can read the 16-bit data from this location. The default value is 0x0000.

MIIMINDR MII Management Indicator Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	R															
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
	—	—	—	—	—	—	—	—	—	—	—	—	—	NV	SCAN	BUSY
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIMIND fields (scan, not valid, and busy) indicate the availability of each read of a scan cycle to the host from MIIMSTATR[PHYS].

Table 25-71. MIIMIND Bit Descriptions

Bit	Reset	Description	Settings
— 0–28	0	Reserved.	
NV 29	0	**Not Valid** Indicates that the MII management read cycle has not completed and the read data is not valid.	0 No read cycle or read cycle complete. 1 Read cycle not complete.
SCAN 30	0	**Scan** Indicates that a scan operation (continuous MII management read cycles) is in progress.	0 No scan cycle. 1 Scan cycle is in progress.
BUSY 31	0	**Busy** Indicates that an MII management block is performing an MII management read or write cycle.	0 No read or write cycle. 1 Read or write cycle is in progress.

25.17.7 MIIGSK Registers

The MIIGSK contains nine memory-mapped, read/write, 32-bit registers. All MIIGSK registers are accessible via the IPI line.

MIIGSK_CFGR MIIGSK Configuration Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—									FRCONT	—	LBMODE	EMODE	—	IFMODE	
Type	R								R/W		R	R/W		R	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_CFGR contains configuration bits for various Ethernet controller features and modes.

Table 25-72. MIIGSK_CFGR Bit Descriptions

Bit	Reset	Description	Settings
— 0–24	0	Reserved.	
FRCONT 25	0	**Frequency Control.** Determines the frequency of the clock source to the Ethernet controller to support 10/100 Mbps operations in RMII/SMII modes of operation. (In SMII mode, ETHCLOCK, and in RMII mode, ETHREF_CLK. **Note:** This field has no effect in MII mode.	0 In RMII mode, the clock source (ETHREF_CLK) is 50 MHz to support 100 Mbps operation. In SMII mode, the clock source (ETHCLOCK) is 125 MHz to support 100 Mbps operation. 1 In RMII mode, the clock source (ETHREF_CLK) is divided by 10 (5 MHz) to support 10 Mbps Operation. In SMII mode, the clock source (ETHCLOCK) is divided by 10 (12.5 MHz) to support 10Mbps Operation.
— 26	0	Reserved.	
LBMODE 27	0	**RMII/SMII Sync Out - Internal Loopback Mode** Causes the Ethernet controller RMII/SMII transmit outputs to be looped back to the Ethernet controller RMII/SMII receive inputs.Proper operation is guaranteed only when: • MIIGSK_CFGR[IFMODE] = 01 or 10 • MACCFG[MIILB]=0 • MIIGSK_CFGR[EMODE] = 0	0 Normal operation. 1 RMII/SMII transmit outputs are looped back to the RMII/SMII receive inputs.
EMODE 28	0	**Echo Mode** Causes the Ethernet controller MII receive inputs from the MII PHY to be looped back to the Ethernet controller transmit outputs to the MII PHY. Proper operation is guaranteed only when: • MIIGSK_CFGR[IFMODE] = 00 • MIIGSK_CFGR[LBMODE] = 0	0 Normal operation (the default). 1 MII receive inputs are looped back to the Ethernet controller transmit outputs.

Table 25-72. MIIGSK_CFGR Bit Descriptions (Continued)

Bit	Reset	Description	Settings
— 29	0	Reserved.	
IFMODE 30–31	0	**Interface Mode** Specifies the type of interface to which the Ethernet controller is connected.	00 MII mode. 01 RMII mode. 10 SMII mode. 11 Reserved.

MIIGSK_GPR MIIGSK General-Purpose Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—											IR	—			DS
Type	R												R/W			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-73. MIGSK_GPR Bit Descriptions

Bit	Reset	Description	Settings
— 0–27	0	Reserved.	
IR 27	—	**Ethernet Controller Internal Reset** Puts all Ethernet controller modules into reset. **Note:** MIIGSK_GPR is not reset by the Ethernet controller Internal Reset.	0 Normal operation 1 Internal Reset
— 30-28		Reserved	
DS 31	0	**Drive Strength** Select standard pad drive strength.	0 Select slow drive 1 Select high drive

MIIGSK_ENR MIIGSK Enable Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—														READY	EN
Type	R															R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_ENR contains a bit that allows you to enable/disable Ethernet controller operation and a bit that indicates whenever the Ethernet controller is ready for use. The ready bit ensures proper configuration of the Ethernet controller. Z

Table 25-74. MIGSK_ENR Bit Descriptions

Bit	Reset	Description	Settings
— 0–29	0	Reserved.	
READY 30	0	**Ready** This bit is set when the Ethernet controller is ready for use. **Note:** This bit is read-only.	0 Ethernet controller not ready for use. 1 Ethernet controller ready for use.
EN 31	0	**Enable** Enables/disables Ethernet controller operation.	0 No transmission/reception of frames to/from the Ethernet controller. 1 The Ethernet controller can transmit/receive frames.

MIIGSK_SMII_SYNCDIR MIIGSK SMII SYNC Direction Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—														SYNC_IN	SYNC
Type	R														R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_SMII_SYNCDIR contains two bits to allow you to determine the direction of the SMII SYNC signal; that is, whether the Ethernet controller is connected to another SMII MAC or to a SMII PHY.

- For the SMII MAC-to-MAC connection, the Ethernet controller is synchronized on the receive SYNC signal, and the receive and transmit operation proceeds according to the input sync signal, ETHSYNC_IN. In this mode, the generation of the transmit output signal is disabled.

- For the SMII MAC-to-PHY connection, both transmit and receive operation are synchronized to the Ethernet controller output sync signal ETHSYNC. In this mode, receiving an external SYNC signal is disabled.

If the Ethernet controller is connected to another MAC with the same capability to transmit and receive frames synchronously to an incoming sync signal, the Ethernet controller drives the output sync signal, and the ETHSYNC_IN signal is disabled.

Table 25-75. MIGSK_SMII_SYNCDIR Bit Descriptions

Bit	Reset	Description	Settings
— 0–29	0	Reserved.	
SYNC_IN 30	0	**SYNC_IN Enable** Enables/disables the ETHSYNC_IN input control signal.	0 ETHSYNC_IN input control signal is disabled. 1 ETH SYNC_IN input control signal is enabled.
SYNC 31	0	**SYNC Enable** Enables/disables the ETHSYNC output control signal.	0 ETHSYNC output control signal is disabled. 1 ETHSYNC output control signal is enabled.

MIIGSK_TIFBR MIIGSK SMII Transmit Inter-Frame Bits Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—								TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_TIFBR allows you to determine the value of the TXD[7–0] signals that transfer data between frames in the programmable inter-frame gap (IFG) period for both MAC-to-PHY and MAC-to-MAC connections for SMII. On the PHY side of a MAC-to-PHY connection, TXD[7–0] convey only packet data. For a MAC-to-MAC connection, TXD[7–0] transfer signal status values. You can write any value to these bits, but the SMII specification provides recommended status bit definitions.

Note: MIIGSK_TIFBR can be programmed only when MIIGSK_ENR[EN] = 0, that is, the Ethernet controller is disabled.

Table 25-76. MIIGSK_TIFBR Bit Descriptions

Bit	Reset	Description	Cisco SMII Specification Status Bit Values
— 0–23	0	Reserved.	
TXD7 24	0	**IFG Transmit Segment Data Bit 7** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** if the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	1
TXD6 25	0	**IFG Transmit Segment Data Bit 6** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	1
TXD5 26	0	**IFG Transmit Segment Data Bit 5** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	1
TXD4 27	0	**IFG Transmit Segment Data Bit 4** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	0 No jabber. 1 Jabber.
TXD3 28	0	**IFG Transmit Segment Data Bit 3** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	0 Link down. 1 Link up.
TXD2 29	0	**IFG Transmit Segment Data Bit 2** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	0 Half-duplex. 1 Full-duplex.
TXD1 30	0	**IFG Transmit Segment Data Bit 1** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	0 10 Mbps. 1 100 Mbps.
TXD0 31	0	**IFG Transmit Segment Data Bit 0** Part of the 10-bit data segments transferred in the inter-packet gap between frames. **Note:** If the TX_EN bit = 1, this is data. If TX_EN = 0, this is a status bit.	0 No forced error. 1 Forced error.

MIIGSK_RIFBR MIIGSK SMII Receive Inter-Frame Bits Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	R															
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**
	—	—	—	—	—	—	—	—	RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_RIFBR contains 8 bits that allow you to read the value of the received inter frame segment bits. See the description of the ETHRXD signal in **Section 25.5**, *External Signals*, on page 25-10, as well as **Figure 25-10**.

Table 25-77. MIIGSK_RIFBR Bit Descriptions

Bit	Reset	Description	Cisco SMII Specification Status Bit Values
— 0–23	0	Reserved.	
RXD7 24	0	**IFG Receive Segment Data Bit 7** Allows you to read the value of the received inter-frame segment bits.	1
RXD6 25	0	**IFG Receive Segment Data Bit 6** Allows you to read the value of the received inter-frame segment bits.	0 No false carrier detected. 1 False carrier detected.
RXD5 26	0	**IFG Receive Segment Data Bit 5** Allows you to read the value of the received inter-frame segment bits.	0 Upper nibble invalid. 1 Upper nibble valid.
RXD4 27	0	**IFG Receive Segment Data Bit 4** Allows you to read the value of the received inter-frame segment bits.	0 No Error. 1 Jabber error.
RXD3 28	0	**IFG Receive Segment Data Bit 3** Allows you to read the value of the received inter-frame segment bits.	0 Link down. 1 Link up.
RXD2 29	0	**IFG Receive Segment Data Bit 2** Allows you to read the value of the received inter-frame segment bits.	0 Half-duplex. 1 Full-duplex.
RXD1 30	0	**IFG Receive Segment Data Bit 1** Allows you to read the value of the received inter-frame segment bits.	0 10 Mbps. 1 100 Mbps.
RXD0 31	0	**IFG Receive Segment Data Bit 0** Allows you to read the value of the received inter-frame segment bits.	0 No previous RX error. 1 RX error previous frame.

MIIGSK_ERIFBR MIIGSK SMII Expected Receive Inter-Frame Bits Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—								ERXD7	ERXD6	ERXD5	ERXD4	ERXD3	ERXD2	ERXD1	ERXD0
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_ERIFBR contains 8 bits that allow you to determine the expected values of the receive bits while receiving inter-frame gap frames. Bits 0 to 7 of the receive inter-frame gap segment are compared with the MIIGSK_ERIFBR bits. A difference between a received inter-frame gap bit in the MIIGSK_RIFBR register and its corresponding bit in the MIIGSK_ERIFBR register causes a corresponding bit in the MIIGSK_IEVENT register to be set.

Table 25-78. MIIGSK_ERIFBR Bit Descriptions

Bit	Reset	Description
— 0–23	0	Reserved.
ERXD[7–1] 24–30	0	**Expected IFG Receive Segment Bits** Allow you to determine the expected values of the received inter-frame segment bits.
ERXD0 31	0	**Bit 0 of the Expected IFG Receive Segment Bits** In SMII mode, this is IRX_ER and should be detected by interrupt event 0.

MIIGSK_IEVENT MIIGSK SMII Interrupt Event Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—								IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_IEVENT generates an interrupt if the corresponding bit in the MIIGSK_IMASK register is also set. The bit in MIIGSK_IEVENT is cleared if a 1 is written to that bit position. Writing a 0 has no effect.

Table 25-79. MIGSK_IEVENT Bit Descriptions

Bit	Reset	Description	Settings
— 0–23	0	Reserved.	
IE7 24	0	**Interrupt Event 7** A difference was discovered between the MIIGSK_RIFBR[RXD7] bit and the MIIGSK_ERIFBR[ERXD7] bit.	0 No effect. 1 Difference in bit 7.
IE6 25	0	**Interrupt Event 6** A difference was discovered between the MIIGSK_RIFBR[RXD6] bit and the MIIGSK_ERIFBR[ERXD6] bit.	0 No effect. 1 Difference in bit 6.
IE5 25	0	**Interrupt Event 5** A difference was discovered between the MIIGSK_RIFBR[RXD5] bit and the MIIGSK_ERIFBR[ERXD5] bit.	0 No effect. 1 Difference in bit 5.
IE4 25	0	**Interrupt Event 4** A difference was discovered between the MIIGSK_RIFBR[RXD4] bit and the MIIGSK_ERIFBR[ERXD4] bit.	0 No effect. 1 Difference in bit 4.
IE3 25	0	**Interrupt Event 3** A difference was discovered between the MIIGSK_RIFBR[RXD3] bit and the MIIGSK_ERIFBR[ERXD3] bit.	0 No effect. 1 Difference in bit 3.
IE2 25	0	**Interrupt Event 2** A difference was discovered between the MIIGSK_RIFBR[RXD2] bit and the MIIGSK_ERIFBR[ERXD2] bit.	0 No effect. 1 Difference in bit 2.
IE1 25	0	**Interrupt Event 1** A difference was discovered between the MIIGSK_RIFBR[RXD1] bit and the MIIGSK_ERIFBR[ERXD1] bit.	0 No effect. 1 Difference in bit 1.
IE0 25	0	**Interrupt Event 0** A difference was discovered between the MIIGSK_RIFBR[RXD0] bit and the MIIGSK_ERIFBR[ERXD0] bit.	0 No effect. 1 Difference in bit 0.

The following steps are recommended to initialize the SMII inter frame status interrupt:

1. Before enabling the Ethernet controller to transmit and receive frames in SMII mode:
   a. Write 0xFF to the MIIGSK_IMASK register.
   b. Clear the MIIGSK_ERIFBR.
2. Wait for the interrupt.
3. In the interrupt handler, use the following steps:
   a. Disable all interrupts by clearing the MIIGSK_IEVENT register.
   b. Clear the IEVENT register.
   c. Update the MIIGSK_ERIFBR according to the expected value.
   d. Enable the desired interrupt in the MIIGSK_IMASK register.

MIIGSK_IMASK MIIGSK SMII Interrupt Mask Register

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—								IE7EN	IE6EN	IE5EN	IE4EN	IE3EN	IE2EN	IE1EN	IE0EN
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MIIGSK_IMASK controls which interrupt events are allowed to generate an actual interrupt. If the corresponding bits in both the MIIGSK_IEVENT and MIIGSK_IMASK registers are set, an interrupt is generated. The interrupt signal remains asserted until the IEVENT bit is cleared either by writing a value of 1 to it or by writing a value of 0 to the corresponding IMASK bit.

Table 25-80. MIGSK_IEVENT Bit Descriptions

Bit	Reset	Description	Settings
— 0–23	0	Reserved.	
IE7EN 24	0	**Interrupt Event 7 Enable** Enabled/disables interrupt event 7.	0 Interrupt 7 disabled. 1 Interrupt 7 enabled.
IE6EN 25	0	**Interrupt Event 6 Enable** Enabled/disables interrupt event 6.	0 Interrupt 6 disabled. 1 Interrupt 6 enabled.
IE5EN 25	0	**Interrupt Event 5 Enable** Enabled/disables interrupt event 5.	0 Interrupt 5 disabled. 1 Interrupt 5 enabled.
IE4EN 25	0	**Interrupt Event 4 Enable** Enabled/disables interrupt event 4.	0 Interrupt 4 disabled. 1 Interrupt 4 enabled.
IE3EN 25	0	**Interrupt Event 3 Enable** Enabled/disables interrupt event 3.	0 Interrupt 3 disabled. 1 Interrupt 3 enabled.
IE2EN 25	0	**Interrupt Event 2 Enable** Enabled/disables interrupt event 2.	0 Interrupt 2 disabled. 1 Interrupt 2 enabled.
IE1EN 25	0	**Interrupt Event 1 Enable** Enabled/disables interrupt event 1.	0 Interrupt 1 disabled. 1 Interrupt 1 enabled.
IE0EN 25	0	**Interrupt Event 0 Enable** Enabled/disables interrupt event 0.	0 Interrupt 0 disabled. 1 Interrupt 0 enabled.

25.17.8 RMON Management Information Base (MIB)

This section describes the management information base (MIB) counters. The Ethernet controller PE-MSTAT module has 37 separate statistics counters that count or accumulate statistical events as packets are transmitted and received. These counters support RMON MIB groups 1–3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB. The detection of one or more of these statistical events triggers the PE-MSTAT module to update its statistics counters. These counters are stored in internal data registers. You can access these internal data registers at any time.

An interrupt can be generated upon any counter rollover condition via a carry interrupt output from the PE-MSTAT. Internal masking registers allow you to mask each counter rollover condition from causing an interrupt. In addition, each individual counter value can be reset on read access, or all counters can be simultaneously reset by asserting an external module input pin.

TR64 Transmit and Receive 64-Byte Frame Counter

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—										TR64					
Type	R										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**
	TR64															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TR64 is one of the RMON MIB counters; it counts the 64-byte frames.

Table 25-81. TR64 Field Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TR64 10–31	0	**Transmit and Receive 64-byte Frame Counter** Increments for each good or bad frame transmitted and received that is 64 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TR127 Transmit and Receive 65- to 127-Byte Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TR127					
Type	R										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**
	TR127															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TR127 is one of the RMON MIB counters; it counts the 65- to 127-byte frames.

Table 25-82. TR127 Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TR127 10–31	0	**Transmit and Receive 65- to 127-Byte Frame Counter** Increments for each good or bad frame transmitted and received that is 65 to 127 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TR255 Transmit and Receive 128- to 255-Byte Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TR255					
Type	R										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**
	TR255															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TR255 is one of the RMON MIB counters; it counts the 128- to 255-byte frames.

Table 25-83. TR255 Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TR255 10–31	0	**Transmit and Receive 128- to 255-Byte Frame Counter** Increments for each good or bad frame transmitted and received that is 128 to 255 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TR511 Transmit and Receive 256- to 511-Byte Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TR511					
Type	R										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**
	TR511															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TR511 is one of the RMON MIB counters; it counts the 256- to 511-byte frames.

Table 25-84. TR511 Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TR511 10–31	0	**Transmit and Receive 256- to 511-Byte Frame Counter** Increments for each good or bad frame transmitted and received that is 256 to 511 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TR1K Transmit and Receive 512- to 1023-Byte Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TR1K					
Type	R										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**
	TR1K															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TR1K is one of the RMON MIB counters; it counts the 512- to 1023-byte frames.

Table 25-85. TR1K Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TR1K 10–31	0	**Transmit and Receive 512- to 1023-Byte Frame Counter** Increments for each good or bad frame transmitted and received that is 512 to 1023 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TRMAX Transmit and Receive 1024- to 1518-Byte Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TRMAX					
Type	R										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**
	TRMAX															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TRMAX is one of the RMON MIB counters; it counts the 1024- to 1518-byte frames.

Table 25-86. TRMAX Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TRMAX 10–31	0	**Transmit and Receive** 1024- to 1518-B**yte Frame Counter** Increments for each good or bad frame transmitted and received that is 1024 to 1518 bytes long, inclusive (excluding preamble and SFD but including FCS bytes).

TRMGV Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TRMGV					
Type	R										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**
	TRMGV															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TRMGV is one of the RMON MIB counters; it counts the 1519- to 1522-byte frames.

Table 25-87. TRMGV Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TRMGV 10–31	0	**Transmit and Receive** 1519- to 1522-B**yte Frame Counter** Increments for each good or bad frame transmitted and received that is 1519 to 1522 bytes long, inclusive (excluding Preamble and SFD but including FCS bytes).

RBYT Receive Byte Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	RBYT														
Type	R	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**
	RBYT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBYT holds the statistics byte counter for receive frames.

Table 25-88. RBYT Bit Descriptions

Bits	Reset	Description
— 0	0	Reserved. Write to zero for future compatibility.
RBYT 1–31	0	**Receive Byte Counter** The statistics counter increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes.

RPKT Receive Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										RPKT					
Type	R										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**
	RPKT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RPKT holds the counter for frame received packets.

Table 25-89. RPKT Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
RPKT 10–31	0	**Receive Packet Counter** Increments for each frame received packet (including bad packets, all unicast, broadcast, and multicast packets).

RFCS Receive FCS Error Counter

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">RFCS</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

RFCS holds the counter for frame receive packets containing check sequence errors.

Table 25-90. RFCS Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RFCS 16–31	0	**Receive FCS Error Counter** Increments for each frame received that has an integral 64 to 1518 length and contains a frame check sequence error.

RMCA Receive Multicast Packet Counter

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="10">—</td><td colspan="6">RMCA</td></tr>
<tr><td>Type</td><td colspan="10">R</td><td colspan="6">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">RMCA</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

RMCA holds the counter for valid non-VLAN multicast packets.

Table 25-91. RMCA Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
RMCA 10–31	0	**Receive Multicast Packet Counter** Increments for each multicast good frame of lengths 64 to 1518 (non-VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

RBCA Receive Broadcast Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										RBCA					
Type	R										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**
	RBCA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RBCA holds the counter for valid non-VLAN broadcast frames.

Table 25-92. RBCA Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
RBCA 10–31	0	**Receive Broadcast Packet Counter** Increments for each valid broadcast frame of lengths 64 to 1518 (non-VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

RXCF Receive Control Frame Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	Rxcf															
Type	R/w															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RXCF holds the counter for MAC control frames.

Table 25-93. RXCF Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RXCF 16–31	0	**Receive Control Frame Packet Counter** Increments for each MAC control frame received (PAUSE and unsupported).

RXPF Receive Pause Frame Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RXPF															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RXPF holds the counter for pause MAC control frames.

Table 25-94. RXPF Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RXPF 16–31	0	**Receive Pause Frame Packet Counter** Increments each time a valid pause MAC control frame is received.

RXUO Receive Unknown OPCode Packet Counter

Bit	0	R1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RXUO															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RXUO holds the counter for MAC control frames with no pause opcode.

Table 25-95. RXUO Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RXUO 16–31	0	**Receive Unknown Opcode Counter** Increments each time a MAC control frame is received that contains an opcode other than a pause.

RALN Receive Alignment Error Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RALN															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RALN holds the counter for receive frames containing an invalid FCS is not an integral number of bytes.

Table 25-96. RALN Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RALN 16–31	0	**Receive Alignment Error Counter** Increments for each received frame from 64 to 1518 (non-VLAN) or 1522 (VLAN) that contains an invalid FCS and is not an integral number of bytes.

RFLR Receive Frame Length Error Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RFLR															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RFLR holds the counter for receive frames with a length field that does not match the number of data bytes actually received.

Table 25-97. RFLR Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RFLR 16–31	0	**Receive Frame Length Error Counter** Increments for each frame received in which the 802.3 length field does not match the number of data bytes actually received (46 –1500 bytes). The counter does not increment if the length field is not a valid 802.3 length, such as an ethertype value.

RCDE Receive Code Error Counter

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">RCDE</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

RCDE holds the counter for receive frames containing an invalid data symbol.

Table 25-98. RCDE Field Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RCDE 16–31	0	**Receive Code Error Counter** Increments each time a valid carrier is present and at least one invalid data symbol is detected.

RCSE Receive Carrier Sense Error Counter

<table>
<tr><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td></td><td colspan="16">—</td></tr>
<tr><td>Type</td><td colspan="16">R</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><th>Bit</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td></td><td colspan="16">RCSE</td></tr>
<tr><td>Type</td><td colspan="16">R/W</td></tr>
<tr><td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

RCSE holds the counter receive packets containing a false carrier.

Table 25-99. RCSE Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RCSE 16–31	0	**Receive False Carrier Counter** Increments each time a false carrier is detected during idle, as defined by a 1 on Ethernet Controller*n*_RXER and an 0xE on Ethernet Controller*n*_RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.

RUND Receive Undersize Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RUND															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RUND holds the counter for well-formed receive frames that are less than 64 bytes long.

Table 25-100. RUND Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RUND 16–31	0	**Receive Undersize Packet Counter** Increments each time a frame is received that is less than 64 bytes long and contains a valid FCS and is otherwise well-formed. This count does not include range length errors.

ROVR Receive Oversize Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	ROVR															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ROVR holds the counter for receive frames that exceed the maximum length but are otherwise well-formed.

Table 25-101. ROVR Field Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
ROVR 16–31	0	**Receive Oversize Packet Counter** Increments each time a frame is received that exceeds 1518 (non-VLAN) or 1522 (VLAN) and contains a valid FCS and is otherwise well formed. This count does not include range length errors.

RFRG Receive Fragments Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RFRG															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RFRG holds the counter for receive frames that are less than 64 bytes long and contain an invalid FCS.

Table 25-102. RFRG Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RFRG 16–31	0	**Receive Fragments Counter** Increments for each frame received that is less than 64 bytes long and contains an invalid FCS. This includes integral and non-integral lengths.

RJBR Receive Jabber Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RJBR															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RJBR holds the counter for receive frames that exceed the maximum data length and contain an invalid FCS.

Table 25-103. RJBR Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RJBR 16-31	0	**Receive Jabber Counter** Increments for frames received that exceed 1518 (non-VLAN) or 1522 (VLAN) bytes and contain an invalid FCS. This includes alignment errors.

RDRP Receive Dropped Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	RDRP															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RDRP holds the counter for receive frames that are streamed to the system but later dropped.

Table 25-104. RDRP Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
RDRP 16–31	0	**Receive Dropped Packets Counter** Increments for frames received that are streamed to the system but are later dropped due to lack of system resources.

TBYT Transmit Byte Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—	TBYT														
Type	R	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**
	TBYT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TBYT holds the counter for transmit bytes.

Table 25-105. TBYT Bit Descriptions

Bits	Reset	Description
— 0	0	Reserved. Write to zero for future compatibility.
TBYT 1–31	0	**Transmit Byte Counter** Increments by the number of bytes that are put on the wire, including fragments of frames involved with collisions. This count does not include preamble/SFD or jam bytes.This counter does not count if the frame is a truncated frame

TPKT Transmit Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TPKT					
Type	R										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**
	TPKT															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TPKT holds the counter for transmitted packets.

Table 25-106. TPKT Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TPKT 10–31	0	**Transmit Packet Counter** Increments for each transmitted packet, including bad packets, excessively deferred packets, excessive collision packets, late collision packets, and all unicast, broadcast, and multicast packets.

TMCA Transmit Multicast Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TMCA					
Type	R										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**
	TMCA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMCA holds the counter for valid multicast transmit frames.

Table 25-107. TMCA Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TMCA 10–31	0	**Transmit Multicast Packet Counter** Increments for each valid multicast frame transmitted, excluding broadcast frames.

TBCA Transmit Broadcast Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—										TBCA					
Type	R										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**
	TBCA															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TBCA holds the counter for transmitted broadcast frames.

Table 25-108. TBCA Bit Descriptions

Bits	Reset	Description
— 0–9	0	Reserved. Write to zero for future compatibility.
TBCA 10–31	0	**Transmit Broadcast Packet Counter** Increments for each broadcast frame transmitted, excluding multicast frames.

TXPF Transmit Pause Control Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	TXPF															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TXPF holds the counter for valid transmit pause MAC control frames.

Table 25-109. TXPF Bit Descriptions

Bits	Reset	Description
— 0–15	0	Reserved. Write to zero for future compatibility.
TXPF 16–31	0	**Transmit Pause Frame Packet Counter** Increments each time a valid pause MAC control frame is transmitted.

TDFR Transmit Deferral Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TDFR											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDFR holds the counter for frames that are deferred on their first transmission attempt.

Table 25-110. TDFR Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TDFR 20–31	0	**Transmit Deferral Packet Counter** Increments for each frame that is deferred on its first transmission attempt. This count does not include frames involved in collisions.

TEDF Transmit Excessive Deferral Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TEDF											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TEDF holds the counter for transmit frames that are aborted because they were deferred for an excessive period of time.

Table 25-111. TEDF Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TEDF 20–31	0	**Transmit Excessive Deferral Packet Counter** Increments for aborted frames that were deferred for an excessive period of time (3036 byte times).

TSCL Transmit Single Collision Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TSCL											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSCL holds the counter for transmit frames that experience exactly one collision.

Table 25-112. TSCL Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TSCL 20–31	0	**Transmit Single Collision Packet Counter** Increments for each frame that experienced exactly one collision during transmission.

TMCL Transmit Multiple Collision Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TMCL											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMCL holds the counter for transmit frames that experience two to fifteen collisions.

Table 25-113. TMCL Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TMCL 20–31	0	**Transmit Multiple Collision Packet Counter** Increments for each frame that experienced 2–15 collisions, including any late collisions, during transmission as defined by the HAFDUPR[RTXM] field.

TLCL Transmit Late Collision Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TLCL											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TLCL holds the counter for transmit frames that experience a late collision.

Table 25-114. TLCL Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TLCL 20–31	0	**Transmit Late Collision Packet Counter** Increments for each transmit frame that experienced a late collision during a transmission attempt.

TXCL Transmit Excessive Collision Packet Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TXCL											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TXCL holds the counter for transmit frames that experience 16 collisions during transmission and are aborted.

Table 25-115. TXCL Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TXCL 20–31	0	**Transmit Excessive Collision Packet Counter** Increments for each frame that experiences 16 collisions during transmission and is aborted.

TNCL Transmit Total Collision Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TNCL											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TNCL holds the counter for the total number of transmit frame collisions.

Table 25-116. TNCL Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TNCL 20–31	0	**Transmit Total Collision Counter** Increments by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits—that is, transmitting and receiving at the same time. **Note:** This count does not include collisions that result in an excessive collision condition.

TJBR Transmit Jabber Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TJBR											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TJBR holds the counter for oversized transmit frames with an incorrect FCS value.

Table 25-117. TJBR Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TJBR 20–31	0	**Transmit Jabber Frame Counter** Increments for each oversized transmitted frame with an incorrect FCS value.

TFCS Transmit FCS Error Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TFCS											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TFCS holds the counter for transmit packets with a valid size but an incorrect FCS value.

Table 25-118. TFCS Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TFCS 20–31	0	**Transmit FCS Error Counter** Increments for every transmit packet with a valid size but an incorrect FCS value.

TXCF Transmit Control Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TXCF											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TXCF holds the counter for transmit frames with a valid size and a type field signifying a control frame.

Table 25-119. TXCF Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TXCF2 0–31	0	**Transmit Control Frame Counter** Increments for every transmit frame with a valid size and a type field signifying a control frame.

TOVR Transmit Oversize Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TOVR											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TOVR holds the counter for each oversized transmit frame with a correct FCS value.

Table 25-120. TOVR Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TOVR 20–31	0	**Transmit Oversize Frame Counter** Increments for each oversized transmitted frame with a correct FCS value.

TUND Transmit Undersize Frame Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TDFR											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TUND holds the counter for transmit frames less than 64 bytes long but with a correct FCS value.

Table 25-121. TUND Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TDFR 20–31	0	**Transmit Undersize Frame Counter** Increments for every frame less than 64 bytes long with a correct FCS value.

TFRG Transmit Fragment Counter

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—				TFRG											
Type	R				R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TFRG holds the counter for transmit frames less than 64 bytes long and with an incorrect FCS value.

Table 25-122. TFRG Bit Descriptions

Bits	Reset	Description
— 0–19	0	Reserved. Write to zero for future compatibility.
TFRG 20–31	0	**Transmit Fragment Counter** Increments for every frame less than 64 bytes long and with an incorrect FCS value.

CAR1 Carry Register One

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	C164	C1127	C1255	C1511	C11K	C1MAX	C1MGV	—								C1RBY
Type	R/W							R								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**
	C1RPK	C1RFC	C1RMC	C1RBC	C1RXC	C1RXP	C1RXU	C1RAL	C1RFL	C1RCD	C1RCS	C1RUN	C1ROV	C1RFR	C1RJB	C1RDR
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAR1 bits are cleared on carry register writes when the respective bits are set.

Table 25-123. CAR1 Bit Descriptions

Bits	Reset	Description
C164 0	0	Carry Register 1 TR64 Counter Carry
C1127 1	0	Carry Register 1 TR127 Counter Carry
C1255 2	0	Carry Register 1 TR255 Counter Carry
C1511 3	0	Carry Register 1 TR511 Counter Carry
C11K 4	0	Carry Register 1 TR1K Counter Carry
C1MAX 5	0	Carry Register 1 TRMAX Counter Carry

Table 25-123. CAR1 Bit Descriptions (Continued)

Bits	Reset	Description
C1MGV 6	0	Carry Register 1 TRMGV Counter Carry
— 7–14	0	Reserved
C1RBY 15	0	Carry Register 1 RBYT Counter Carry
C1RPK 16	0	Carry Register 1 RPKT Counter Carry
C1RFC 17	0	Carry Register 1 RFCS Counter Carry
C1RMC 18	0	Carry Register 1 RMCA Counter Carry
C1RBC 19	0	Carry Register 1 RBCA Counter Carry
C1RXC 20	0	Carry Register 1 RXCF Counter Carry
C1RXP 21	0	Carry Register 1 RXPF Counter Carry
C1RXU 22	0	Carry Register 1 RXUO Counter Carry
C1RAL 23	0	Carry Register 1 RALN Counter Carry
C1RFL 24	0	Carry Register 1 RFLR Counter Carry
C1RCD 25	0	Carry Register 1 RCDE Counter Carry
C1RCS 26	0	Carry Register 1 RCSE Counter Carry
C1RUN 27	0	Carry Register 1 RUND Counter Carry
C1ROV 28	0	Carry Register 1 ROVR Counter Carry
C1RFR 29	0	Carry Register 1 RFRG Counter Carry
C1RJB 30	0	Carry Register 1 RJBR Counter Carry
C1RDR 31	0	Carry Register 1 RDRP Counter Carry

CAR2 Carry Register Two

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—												C2TJB	C2TFC	C2TCF	C2TOV
Type	R												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**
	C2TUN	C2TFG	C2TBY	C2TPK	C2TMC	C2TBC	C2TPF	C2TDF	C2TED	C2TSC	C2TMA	C2TLC	C2TXC	C2TNC	—	C2TDP
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAR2 bits are cleared on a carry register write when the respective bits are set.

Table 25-124. CAR2 Bit Descriptions

Bits	Reset	Description
— 0–11	0	Reserved. Write to zero for future compatibility.
C2TJB 12	0	Carry Register 2 TJBR Counter Carry
C2TFC 13	0	Carry Register 2 TFCS Counter Carry
C2TCF 14	0	Carry Register 2 TXCF Counter Carry
C2TOV 15	0	Carry Register 2 TOVR Counter Carry
C2TUN 16	0	Carry Register 2 TUND Counter Carry
C2TFG 17	0	Carry Register 2 TFRG Counter Carry
C2TBY 18	0	Carry Register 2 TBYT Counter Carry
C2TPK 19	0	Carry Register 2 TPKT Counter Carry
C2TMC 20	0	Carry Register 2 TMCA Counter Carry
C2TBC 21	0	Carry Register 2 TBCA Counter Carry
C2TPF 22	0	Carry Register 2 TXPF Counter Carry
C2TDF 23	0	Carry Register 2 TDFR Counter Carry
C2TED 24	0	Carry Register 2 TEDF Counter Carry
C2TSC 25	0	Carry Register 2 TSCL Counter Carry
C2TMA 26	0	Carry Register 2 TMCL Counter Carry
C2TLC 27	0	Carry Register 2 TLCL Counter Carry
C2TXC 28	0	Carry Register 2 TXCL Counter Carry
C2TNC 29	0	Carry Register 2 TNCL Counter Carry
— 30	0	Reserved. Write to zero for future compatibility.
C2TDP 31	0	Carry Register 2 TDRP Counter Carry

CAM1 Carry Register One Mask

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M164	M1127	M1255	M1511	M11K	M1MAX	M1MGV	—								M1RBY
Type	R/W							R								R/W
Reset	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	M1RPK	M1RFC	M1RMC	M1RBC	M1RXC	M1RXP	M1RXU	M1RAL	M1RFL	M1RCD	M1RCS	M1RUN	M1ROV	M1RFR	M1RJB	M1RDR
TYPE	R/W															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

While one of the mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits all default to a set state.

Table 25-125. CAM1 Bit Descriptions

Bits	Reset	Description
M164 0	1	Mask Register 1 TR64 Counter Carry Mask
M1127 1	1	Mask Register 1 TR127 Counter Carry Mask
M1255 2	1	Mask Register 1 TR255 Counter Carry Mask
M1511 3	1	Mask Register 1 TR511 Counter Carry Mask
M11k 4	1	Mask Register 1 TR1K Counter Carry Mask
M1MAX 5	1	Mask Register 1 TRMAX Counter Carry Mask
M1MGV 6	1	Mask Register 1 TRMGV Counter Carry Mask
— 7–14	0	Reserved. Write to zero for future compatibility.
M1RBY 15	1	Mask Register 1 RBYT Counter Carry Mask
M1RPK 16	1	Mask Register 1 RPKT Counter Carry Mask
M1RFC 17	1	Mask Register 1 RFCS Counter Carry Mask
M1RMC 18	1	Mask Register 1 RMCA Counter Carry Mask
M1RBC 19	1	Mask Register 1 RBCA Counter Carry Mask
M1RXC 20	1	Mask Register 1 RXCF Counter Carry Mask
M1RXP 21	1	Mask Register 1 RXPF Counter Carry Mask
M1RXU 22	1	Mask Register 1 RXUO Counter Carry Mask

Table 25-125. CAM1 Bit Descriptions (Continued)

Bits	Reset	Description
M1RAL 23	1	Mask Register 1 RALN Counter Carry Mask
M1RFL 24	1	Mask Register 1 RFLR Counter Carry Mask
M1RCD 25	1	Mask Register 1 RCDE Counter Carry Mask
M1RCS 26	1	Mask Register 1 RCSE Counter Carry Mask
M1RUN 27	1	Mask Register 1 RUND Counter Carry Mask
M1ROV 28	1	Mask Register 1 ROVR Counter Carry Mask
M1RFR 29	1	Mask Register 1 RFRG Counter Carry Mask
M1RJB 30	1	Mask Register 1 RJBR Counter Carry Mask
M1RDR 31	1	Mask Register 1 RDRP Counter Carry Mask

CAM2 Carry Register Two Mask

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—												M2TJB	M2TFC	M2TXC	M2TOV
Type	R												R/W			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Bit	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
	M2TUN	M2TFG	M2TBY	M2TPK	M2TMC	M2TBC	M2TPF	M2TDF	M2TED	M2TSC	M2TMA	M2TLC	M2TXC	M2TNC	M2TPH	M2TDP
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

While one of the mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on an output carry. These bits default to a set state.

Table 25-126. CAM2 Field Descriptions

Bits	Name	Description
— 0–11	0	Reserved. Write to zero for future compatibility.
M2TJB 12	1	Mask Register 2 TJBR Counter Carry Mask
M2TFC 13	1	Mask Register 2 TFCS Counter Carry Mask
M2TXC 14	1	Mask Register 2 TXCF Counter Carry Mask
M2TOV 15	1	Mask Register 2 TOVR Counter Carry Mask

Table 25-126. CAM2 Field Descriptions (Continued)

Bits	Name	Description
M2TUN 16	1	Mask Register 2 TUND Counter Carry Mask
M2TFG 17	1	Mask Register 2 TFRG Counter Carry Mask
M2TBY 18	1	Mask Register 2 TBYT Counter Carry Mask
M2TPK 19	1	Mask Register 2 TPKT Counter Carry Mask
M2TMC 20	1	Mask Register 2 TMCA Counter Carry Mask
M2TBC 21	1	Mask Register 2 TBCA Counter Carry Mask
M2TPF 22	1	Mask Register 2 TXPF Counter Carry Mask
M2TDF 23	1	Mask Register 2 TDFR Counter Carry Mask
M2TED 24	1	Mask Register 2 TEDF Counter Carry Mask
M2TSC 25	1	Mask Register 2 TSCL Counter Carry Mask
M2TMA 26	1	Mask Register 2 TMCL Counter Carry Mask
M2TLC 27	1	Mask Register 2 TLCL Counter Carry Mask
M2TXC 28	1	Mask Register 2 TXCL Counter Carry Mask
M2TNC 29	1	Mask Register 2 TNCL Counter Carry Mask
— 30	1	Reserved. Write to zero for future compatibility.
M2TDP 31	1	Mask Register 2 TDRP Counter Carry Mask

25.17.9 Hash Function Registers

If the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to a hash table entry. You can enable a hash entry by setting the appropriate bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. You should further filter the address. For details, see **Section 25.9.3**, *Hash Table Algorithm*, on page 25-27.

IADDR[0–7] Individual Address Registers 0–7

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	IADDR*n*															
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**
	IADDR*n*															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IADDR*n* is a set of user-programmable registers that represent 256 entries of the individual (unicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. You can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

Table 25-127. IADDR*n* Bit Descriptions

Bits	Name	Description
IADDR*n* 0–31	0	**Individual Address** Represents the 32-bit value associated with the corresponding register. IADDR0 contains the high-order 32 bits of the 256-entry hash table and IADDR7 represents the low-order 32 bits.

GADDR[0–7] Group Address Registers 0–7

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	GADDR*n*															
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**
	GADDR*n*															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GADDR*n* is a set of user-programmable registers that represent 256 entries of the group (multicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. You can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

25.17.10 Pattern Matching Registers

PMD[0–15] Pattern Match Data 0–15

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	PMD*n*															
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**
	PMD*n*															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMD*n* is a user-programmable set of registers that contain 32-bit (bytes 0, 1, 2, 3) data to compare with the data of receive frames. Data is matched on a bit-by-bit basis. These registers allow up to 16 different 4-byte patterns to be recognized. Each PMD*n* register has a corresponding pattern mask register (PMASK*n*), pattern control register (PCNTRL*n*), and pattern attribute register (PATTRB*n*). Pattern matching occurs at the matching index defined in the corresponding PCNTRLn] field. The pattern registers can be configured to support patters longer than 4 bytes by setting the PCNTRL*n*[CP] field. The pattern match status of each accepted frame is written into the status field of the last RxBD. The pattern match feature is supported in both 8-byte and 32-byte RxBD mode. In 8-byte mode, however, the limited size of the BD limits only the reporting of the results of the matching process, not the pattern matching itself.

Table 25-128. PMD*n* Field Description

Bits	Reset	Description
PMD*n* 0–31	0	Specifies 32 bits of data to compare against the frame data.

PMASK[0–15] Pattern Mask Data 0–15

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	PMASK*n*															
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**
	PMASK*n*															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMASK*n* is a set of user-programmable registers (bytes 0, 1, 2, 3) to mask a pattern match associated with the PMD*n* registers. Data is masked on a bit-by-bit basis, thus allowing contiguous and noncontiguous patterns to be recognized. For each PMD*n* register, there is a corresponding PMASK*n* register.

Table 25-129. PMASK*n* Field Description

Bits	Reset	Description	Setting
PMASK*n* 0–31	0	**Pattern Mask** Specifies the 32-bit mask used for pattern matching. If a bit is masked, it is considered a match. When the data bit is unmasked, the corresponding PMD*n* bit is enabled for pattern match compares (with frame data).	0 Mask data bit. Corresponding PMD*n* bit is not enabled for pattern match compares. 1 Unmask data bit.

PCNTRL[0–15] Pattern Match Control Register 0–15

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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
	—		MI						CSE	CP	—				PMC	
Type	R		R/W								R					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCNTRL*n* is a set of user-programmable registers that specify the control variables for pattern matching.

Table 25-130. PCNTRL*n* Bit Descriptions

Bits	Reset	Description	Settings
0–17	—	Reserved. Write to zero for future compatibility.	
MI 18–23	0	**Matching Index** Specifies the index, in multiples of 4 bytes, from the start of the receive frame (from the DA field to FCS inclusive) to perform the pattern matching. For example, MI as cleared corresponds to the first 4-bytes of the destination address. The maximum programmed value for MI is 63 (252-byte offset). The MI value for each 4-byte pattern is always honored (regardless of PCNTRL*n*[CP]), allowing for contiguous or non-contiguous patterns.	
CSE 24	0	**Continue Search Enable** Indicates that if a match occurs on an entry in which the CSE bit is set, the pattern matching should continue. If no other matches are encountered, the attributes corresponding to the last matched entry are used. If a pattern match reject occurs, CSE is ignored (the frame is rejected and searching is discontinued).	0 If the pattern matched, discontinue the search for all other patterns. 1 If the pattern matched, continue searching for other patterns up unto the 256-byte maximum.

Table 25-130. PCNTRL*n* Bit Descriptions (Continued)

Bits	Reset	Description	Settings
CP 25	0	**Concatenated Pattern** The immediate pattern registers that follow are regarded as a continuation of this pattern. For example, PCNTRL0[CP] as set means pattern 0 and pattern 1 are joined together as one pattern. CP as set is always honored and PCNTRL15[CP] is always regarded as cleared, regardless. The lowest numerical PCNTRL*n* register in which CP is set contains the pattern matching control and attribute information (except MI) that is used for concatenated patterns. For each concatenated pattern, the MI field must be set to the appropriate 4-byte multiple. Otherwise, all the patterns attempt to match to the first 4-bytes of the DA (if MI is left cleared).	0 No pattern concatenation with the following pattern is performed. 1 The immediate pattern that follows is concatenated to the current one.
— 26–29	0	Reserved. Write to zero for future compatibility.	
PMC 30–31	0	**Pattern Match Control** Controls the filtering of frames based on pattern matching.	00 Entry disabled. 01 Pattern Match. The pattern is not the criterion used for accepting or rejecting a frame if a match occurs. It is used only for filing data on a frame that is accepted based on a previous pattern (one with a PMC = 10) or on DA recognition. 10 Pattern Match Accept. The frame is accepted (subject to the CSE bit). 11 Pattern Match Reject. The frame is rejected if a match occurs.

PATTRB[0–15] Pattern Match Attributes Register 0–15

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	—															
Type	R															
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**
	—						PMF	—	RDSEN	RBDSEN	—				QC	
Type	R	R/W		R	R/W		R/W				R				R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PATTRB*n* registers are user-programmable registers that specify the action to take when a match occurs, where to file, and how to store receive frames and their associated BDs.

Table 25-131. PATTRB*n* Field Descriptions

Bits	Reset	Description	Settings
— 0–21	0	Reserved. Write to zero for future compatibility.	
PMF 22	0	**Pattern Match File** Specifies which field is used to determine where the frame is filled.	0 If a match occurs, the DATTR[QC] field is used to determine where the frame is filed. 1 If a match occurs, the PATTRB[QC] field is used to determine where the frame is filed.

Table 25-131. PATTRB*n* Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 23	0	Reserved. Write to zero for future compatibility.	
RDSEN 24	0	**Rx Data Snoop Enable** Enables/disables snooping of all receive frame data to memory.	0 Disables snooping of all receive frame data to memory. 1 Enables snooping of all receive frame data to memory.
RBDSEN 25	0	**RxBD Snoop Enable** Enables/disables snooping of all RxBD memory accesses.	0 Disables snooping of all receive BD memory accesses. 1 Enables snooping of all receive BD memory accesses.
— 26–29	0	Reserved. Write to zero for future compatibility.	
QC 30–31	00	**Queue Classification** Specifies the receive queue classification in which to file an incoming frame if the PATTRBn[PMF] field is set and a corresponding pattern match occurs. In the case of concatenated pattern configurations the QC used is from the first 4-byte pattern.	00 0 queue. 01 1 queue. 10 2 queue. 11 3 queue.

DATTR Default Attribute Register

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	—															
Type	R															
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**
	—								RDSEN	RBDSEN	—				QC	
Type	R	R/W		R	R/W		R		R/W		R				R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A frame is processed using DATTR instead of PATTRB*n* in the following cases:

- Pattern matching is disabled (destination address recognition mode).
- Pattern matching enabled but no pattern hit detected.
- Pattern matching enabled but PATTRB*n*[PMF] is cleared (only the DATTR[QC] is used).

Write to DATTR to specify where to file the received frames and their associated BDs. The reset condition of this register is to file in queue 0. If PATTRB*n*[PMF] is cleared and there is a hit on a pattern while the DAATR[QC] is used, PATTRB*n*[RBDSEN] is used instead of DATTR[RBDSEN] to process the received frames, allowing you to override these fields in the DATTR. You must ensure that this override does not happen inadvertently. For a snoop enable bit, for example, the PATTRB*n*[RBDSEN] and PATTRB*n*[RDSEN] should match the corresponding DATTR fields unless you want to file to the default queue with a different snoop mode than the default snoop mode. To avoid inadvertent overrides, DATTR must be programmed first and then each PATTRB*n* register can be programmed accordingly.

Table 25-132. DATTR Bit Descriptions

Bits	Reset	Description	Settings
— 0–23	0	Reserved. Write to zero for future compatibility.	
RDSEN 24	0	**Rx Data Snoop Enable** Enables/disables snooping of all receive frame data to memory.	0 Disables receive frame data to memory snooping. 1 Enables receive frame data to memory snooping.
RBDSEN 25	0	**RxBD Snoop Enable** Enables/disables snooping of all receive BD memory accesses.	0 Disables receive BD memory access snooping. 1 Enables receive BD memory access snooping.
— 26–29	0	Reserved. Write to zero for future compatibility.	
QC 30–31	0	**Queue Classification** Specifies the receive default queue classification in which to file an incoming frame if there is no pattern hit or if a pattern hit occurs with the PATTRBn[PMF] field equal to zero.	00 Queue 0. 01 Queue 1. 10 Queue 2. 11 Queue 3.

25.17.11 Data Structures (Buffer Descriptors)

Data is presented to the Ethernet controller for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, L, HE and TC bits and the length (in bytes) in the first word and the buffer pointer in the second word. The Ethernet controller clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block.

TxBD 8-Byte Transmit Data Buffer Descriptor

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	R	PADCRC	W	I	L	TC	DEF	0	HFELC	RL	RC				UN	0
Offset + 0x02	Data Length (DL)															
Offset + 0x04	Tx Data Buffer Pointer															
Offset + 0x06																

Table 25-133. 8-Byte Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Name	Description	Settings
Offset + 0	**R** 0	**Ready** Specifies whether this BD or its buffer is ready to be sent. If it is not ready, this BD or its buffer can be modified. The Ethernet controller clears this bit after the buffer is transmitted or after an error condition is encountered. The BD cannot be modified once R is set. This bit is written by the Ethernet controller and the user.	0 Not ready to be sent. 1 Ready for transmission or is being sent.
Offset + 0	**PADCRC** 1	**Padding and CRC Attachment for Frames** Indicates whether to add padding and CRCs to frames. This bit is valid only while it is set in the first BD and MACCFG2R[PADCRC] is cleared). If MACCFG2R[PADCRC] is set, this bit is ignored. When PADCRC is set, padding bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 device, which PADs up to the MINFLR value, the Ethernet controller always inserts padding until the frame attains the **IEEE** minimum frame length of 64 bytes. CRC is always appended.	0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames.
Offset + 0	**W** 2	**Wrap** Indicates whether this BD is the last BD in the TxBD table. This bit is written by the user.	0 The next BD is in the consecutive location. 1 The next BD is in the location defined in TBASE.
Offset + 0	**I** 3	**Interrupt** Specifies whether an interrupt is generated after this buffer is processed. This bit is written by the user.	0 No interrupt is generated after this buffer is serviced. 1 EVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (that is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
+ 0x00	**L** 4	**Last in Frame** Specifies whether this buffer is the last one in the transmit frame. This bit is written by the user.	0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
+ 0x00	**TC** 5	**Tx CRC** Specifies whether to append a hardware-generated CRC after the last data byte is transmitted. This bit is written by the user. It is valid only when it is set in the first BD, TxBD[PADCRC] is cleared, MACCFG2R[PADCRC] is cleared, and MACCFG2R[CRC] is cleared.) If MACCFG2R[PADCRC] is set or MACCFG2R[CRC] is set, this bit is ignored.	0 End the transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PADCRC] is set. 1 Transmit the CRC sequence after the last data byte.
+ 0x00	**DEF** 6	**Defer Indication** Indicates whether this frame was deferred. Hardware updates this bit after transmitting a frame (TxBD[L] is set.	0 This frame was not deferred. 1 If HAFDUP[EXCESS_DEFER]=1, this frame did not have a collision before it was sent but it was sent late because of deferring. If HAFDUP[EXCESS_DEFER]=0, this frame was aborted and not sent.

Table 25-133. 8-Byte Transmit Data Buffer Descriptor (TxBD) Field Descriptions (Continued)

Offset	Name	Description	Settings
+ 0x00	**HFE/LC** 8	**Huge Frame Enable (written by user)/Late collision (written by Ethernet controller)**. When a collision occurs, the Ethernet controller terminates the transmission and updates LC.	0 Truncate transmit frame if its length is greater than the value in the MAC’s Maximum Frame Length register. 1 Do not truncate the transmit frame. 0 No late collision. 1 A collision occurred after 64 bytes were sent.
+ 0x00	**RL** 9	**Retransmission Limit** Indicates when a transmission attempt exceeds the maximum retry limit for sending the message. When RL is set, the Ethernet controller terminates the transmission and updates RL.	0 Transmission before maximum retry limit is hit. 1 The transmitter failed (maximum retry limit + 1) attempts to successfully send a message due to repeated collisions.
+ 0x00	**RC** 10–13	**Retry Count** Indicates the number attempts required to transmit a frame. If the value in this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.	0 The frame is sent correctly the first time. x More than zero attempts were needed to send the transmit frame.
+ 0x00	**UN** 14	**Underrun** Indicates when a transmitter underrun condition is encountered. When this bit is set, the Ethernet controller terminates the transmission and updates UN.	0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer.
+ 0x00	— 15	Reserved. Write to zero for future compatibility.	
+0x02	**DL** 0–15	**Data Length** The number of octets the Ethernet controller should transmit from this BD’s data buffer. The Ethernet controller never modifies this value.This field must be greater than zero.	
+0x0 4	**TXDBPT** 0–31	**Transmit Data Buffer Pointer** Contains the address of the associated data buffer. There are no alignment requirements for this address.	

TxBD 32-Byte Transmit Data Buffer Descriptor

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Offset + 0x00	R	PADCRC	W	I	L	TC	DEF	0	HFE/LC	RL	RC				UN	0
Offset + 0x02	Data Length (DL)															
Offset + 0x04	Reserved															
Offset + 0x06																

TxBD 32-Byte Transmit Data Buffer Descriptor (Continued)

Offset + 0x08	TX Data Buffer Pointer		
Offset + 0x10			
Offset + 0x12	Reserved		
Offset + 0x14			
Offset +0x 16	TX Insert Buffer Pointer		
Offset + 0x18			
Offset + 0x20	Reserved		
Offset + 0x22	Reserved	IE	IT
Offset + 0x24	Insert Index		
Offset + 0x26	Insert Length		
Offset + 0x28	Reserved		
Offset + 0x30	Reserved		

The 32-byte TxBD is an extended version of the 8-byte TxBD. It supports insertion of data on a per-buffer descriptor basis, either insertion with replacement or insertion with expansion. Software programs the right TYPE/LENGTH field in the BD to take into account any inserted data.

Table 25-134. 32-Byte TxBD Bit Descriptions

Offset	Bits	Description	Settings
+ 0x00	**R** 0	**Ready** Specifies whether this BD or its buffer is ready to be sent. If it is not ready, this BD or its buffer can be modified. The Ethernet controller clears this bit after the buffer is transmitted or after an error condition is encountered. The BD cannot be modified once R is set. This bit is written by the Ethernet controller and the user.	0 Not ready to be sent. 1 Ready for transmission or is being sent.
+ 0x00	**PADCRC** 1	**Padding and CRC Attachment for Frames** Indicates whether to add padding and CRCs to frames. This bit is valid only while it is set in the first BD and MACCFG2R[PADCRC] is cleared). If MACCFG2R[PADCRC] is set, this bit is ignored. When PADCRC is set, padding bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 device, which PADs up to the MINFLR value, the Ethernet controller always inserts padding until the frame attains the **IEEE** minimum frame length of 64 bytes. CRC is always appended.	0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames.
+ 0x00	**W** 2	**Wrap** Indicates whether this BD is the last BD in the TxBD table. This bit is written by the user.	0 The next BD is in the consecutive location. 1 The next BD is in the location defined in TBASE.
+ 0x00	**I** 3	**Interrupt** Specifies whether an interrupt is generated after this buffer is processed. This bit is written by the user.	0 No interrupt is generated after this buffer is serviced. 1 EVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (that is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
+ 0x00	**L** 4	**Last in Frame** Specifies whether this buffer is the last one in the transmit frame. This bit is written by the user.	0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
+ 0x00	**TC** 5	**Tx CRC** Specifies whether to append a hardware-generated CRC after the last data byte is transmitted. This bit is written by the user. It is valid only when it is set in the first BD, TxBD[PADCRC] is cleared, MACCFG2R[PADCRC] is cleared, and MACCFG2R[CRC] is cleared.) If MACCFG2R[PADCRC] is set or MACCFG2R[CRC] is set, this bit is ignored.	0 End the transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PADCRC] is set. 1 Transmit the CRC sequence after the last data byte.
+ 0x00	**DEF** 6	**Defer Indication** Indicates whether this frame was deferred. Hardware updates this bit after transmitting a frame (TxBD[L] is set.	0 This frame was not deferred. 1 If HAFDUP[EXCESS_DEFER]=1, this frame did not have a collision before it was sent but it was sent late because of deferring. If HAFDUP[EXCESS_DEFER]=0, this frame was aborted and not sent.

Table 25-134. 32-Byte TxBD Bit Descriptions (Continued)

Offset	Bits	Description	Settings
+ 0x00	**HFE/LC** 8	**Huge Frame Enable (written by user)/Late collision (written by Ethernet controller)** HFE is valid only when it is set in the first BD. When a collision occurs, the Ethernet controller terminates the transmission and updates LC.	0 Truncate transmit frame if its length is greater than the value in the MAC’s Maximum Frame Length register. 1 Do not truncate the transmit frame. 0 No late collision. 1 A collision occurred after 64 bytes were sent.
+ 0x00	**RL** 9	**Retransmission Limit** Indicates when a transmission attempt exceeds the maximum retry limit for sending the message. When RL is set, the Ethernet controller terminates the transmission and updates RL.	0 Transmission before maximum retry limit is hit. 1 The transmitter failed (maximum retry limit + 1) attempts to successfully send a message due to repeated collisions.
+ 0x00	**RC** 10–13	**Retry Count** Indicates the number attempts required to transmit a frame. If the value in this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.	0 The frame is sent correctly the first time. x More than zero attempts were needed to send the transmit frame.
+ 0x00	**UN** 14	**Underrun** Indicates when a transmitter underrun condition is encountered. When this bit is set, the Ethernet controller terminates the transmission and updates UN.	0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer.
+ 0x00	— 15	Reserved. Write to zero for future compatibility.	
+ 0x02	**DL** 0–15	**Data Length** The number of octets the Ethernet controller should transmit from this BD’s data buffer. The Ethernet controller never modifies this value.This field must be greater than zero	
+0x04	— 0–31	Reserved. Write to zero for future compatibility.	
+ 0x08	**TXDBPT** 0–31	**Transmit Data Buffer Pointer** Contains the address of the associated data buffer. There are no alignment requirements for this address.	
+0x12	— 0–31	Reserved. Write to zero for future compatibility.	

Table 25-134. 32-Byte TxBD Bit Descriptions (Continued)

Offset	Bits	Description	Settings
+0x16	**TXIBPL** 0–31	**Tx Insert Buffer Pointer** Contains the user-specified address of a buffer to contain the insert data. There are no alignment requirements. The buffer resides in memory external to the Ethernet controller. The insert data is placed into the transmit frame as defined by the insert index and insert length fields, as shown in **Figure 25-24**, *Insertion by Replacement and Insertion by Expansion,* on page 25-37. The combination of the data length, insert index, and insert length fields must be valid, or else IEVENT[IE] is set.	
+ 0x20	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x22	— 0–12	Reserved. Write to zero for future compatibility.	
+0x 22	**IE** 13	**Insertion Error** Indicates an error during an attempt to insert data. The Ethernet controller terminates the transmission, updates IE, and sets IEVENT[IE]. Insertion errors occur if the index is greater than the TxBD[DL]. Insertion errors occur for replacement if the index + length is greater than the TxBD[DL]. Transmission of frames continues, so a partial insertion can occur within the frame.	0 If TxBD[IT] = 01 or 10, no insertion error. 1 An error occurred during an attempt to insert data
+0x22	**IT** 14–15	**Insertion Type** Defines the type of insertion to perform. This field is written by the user.	00 No insertion. 01 Replacement. 10 Expansion. 11 Reserved.
+ 0x24	**II**	**Insert Index** Contains the number of bytes to jump within the transmit buffer before the Ethernet controller begins to insert data into the buffer pointed to which the Tx insert buffer pointer is pointing. This field must be less than or equal to the TxBD[DL]. The Ethernet controller never modifies this value. If this field is cleared, insertion starts at the beginning of the buffer.	
+0x 26	**IL**	**Insert Length** Contains the number of bytes of data that are inserted into the buffer. If this field is zero, no insertion takes place, and the Tx insert buffer pointer and the Insert Index field are considered invalid. The Ethernet controller never modifies this value.	
+0x28	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x30	— 0–15	Reserved. Write to zero for future compatibility.	

RxBD 8-Byte Receive Buffer Descriptor

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	E	RO1	W	I	L	F	0	M	BC	MC	LG	NO	SH	CR	OV	TR
Offset + 0x02	Data Length															
Offset + 0x04	RX Data Buffer Pointer															
Offset + 0x06																

In the RxBD, the user initializes the E, HE, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the Ethernet controller modifies the E, L, F, M, BC, MC, LG, NO, SH, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The Ethernet controller modifies the M, BC, MC, LG, NO, SH, CR, OV, and TR bits in the first word of the BD only if the L bit is set. The first word of the RxBD contains control and status bits.

Table 25-135. 8-Byte Receive Buffer Descriptor Field Descriptions

Offset	Bits	Description	Settings
+ 0x00	**E** 0	**Empty** Indicates whether the data buffer associated with this BD empty or full. When this bit is cleared, the status and length fields have been updated as required. This bit is written by the Ethernet controller when cleared and by the user when set.	0 The data buffer of this BD is filled with received data, or data reception is aborted due to an error condition. 1 The data buffer of this BD is empty, or reception is currently in progress.
+ 0x00	**RO1** 1	**Receive Software Ownership** This field is reserved for use by software. Hardware does not modify this read/write bit, nor does its value affect hardware.	
+ 0x00	**W** 2	**Wrap** Indicates whether the next BD is the last one. This bit is written by user.	0 The next BD is found in the consecutive location. 1 The next BD is found at the location defined in RBASE.
+ 0x00	**I** 3	**Interrupt** Specifies whether an interrupt is generated after this buffer is processed. This bit is written by the user. If you want the interrupt to occur only if RXF0 occurs, disable RXB0 (IMASK[RXBEN0] is cleared) and enable RXF0 (IMASK[RXFEN0] is set).	0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB*n*] or IEVENT[RXF*n*] are set after this buffer is serviced. These bits can cause an interrupt if enabled (IMASK[RXBEN*n*] or IMASK[RXFEN*n*]).
+ 0x00	**L** 4	**Last in Frame** Specifies whether this buffer is the last one in the receive frame. This bit is written by the Ethernet controller.	0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.

Table 25-135. 8-Byte Receive Buffer Descriptor Field Descriptions (Continued)

Offset	Bits	Description	Settings
+ 0x00	**F** 5	**First in Frame** Specifies whether this buffer is the first one in the receive frame. This bit is written by the Ethernet controller.	0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
+ 0x00	— 6	Reserved. Write to zero for future compatibility.	
+ 0x00	**M** 7	**Miss** The Ethernet controller sets this bit for frames that are accepted in promiscuous mode but are flagged as a “miss” by the internal address recognition. Thus, when the Ethernet controller is in Promiscuous mode, you can use the M bit to determine quickly whether the frame is destined to this station. This bit is valid only if the L-bit is set and Ethernet controller is in Promiscuous mode.	0 The frame was received because of an address recognition hit. 1 The frame was received because of Promiscuous mode.
+ 0x00	**BC** 8	**Broadcast** Broadcast mode. This bit is written by Ethernet controller and is valid only if L is set. L is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).	0 Normal operation. 1 Broadcast mode.
+ 0x00	**MC** 9	**Multicast** Multicast mode. This bit is written by Ethernet controller and is valid only if L is set. L is set if the DA is multicast and not BC.	0 Normal operation. 1 Multicast mode.
+ 0x00	**LG** 10	**Rx Frame Length Violation** The length of a frame is greater than maximum frame length. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame length exceeds the maximum frame length.
+ 0x00	**NO** 11	**Rx Non-octet Aligned Frame** A frame that contained a number of bits not divisible by eight was received. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame contains a number of bits not divisible by eight.
+ 0x00	**SH** 12	**Short Frame** The length of a frame is less than the minimum length defined for this channel (MINFLR), provided RCTRL[RSF] is set. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame length is less than the minimum length.
+ 0x00	**CR** 13	**Rx CRC Error** The frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 CRC error or receive code group error.
+ 0x00	**OV** 14	**Overrun** A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and are zero. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Receive FIFO overrun.
+ 0x00	**TR** 15	**Truncation** Receive frame is truncated. If this bit is set, the frame must be discarded and the other error bits must be ignored because they may be incorrect. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Receive frame truncated.

Table 25-135. 8-Byte Receive Buffer Descriptor Field Descriptions (Continued)

Offset	Bits	Description	Settings
+ 0x02	**DL** 0–15	**Data Length** The number of octets written by the Ethernet controller into this BD’s data buffer if L is cleared (the value is equal to the corresponding MRBLR[0–3]), or the length of the frame including CRC if L is set. This field is written by the Ethernet controller. written by the Ethernet controller.	
+ 0x04	**RXDBPT** 0–31	**Rx Data Buffer Pointer** Points to the first location of the associated data buffer and must be 64-byte aligned. The buffer must reside in memory external to the Ethernet controller. This field is written by the user.	

RxBD 32-Byte Receive Buffer Descriptor

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	E	RO1	W	I	L	F	0	M	BC	MC	LG	NO	SH	CR	OV	TR
Offset + 0x02	Data Length															
Offset + 0x04	Reserved															
Offset + 0x06																
Offset + 0x08	RX Data Buffer Pointer															
Offset + 0x10																
Offset + 0x12	Reserved															
Offset + 0x14																
Offset + 0x16	Reserved															
Offset + 0x18																
Offset + 0x20	0	0	0	0	MP				PM	MM	ABPM	0	0			0
Offset + 0x22	Reserved															
Offset + 0x24	0															
Offset + x026	0															
Offset + x028	Reserved															
Offset + 0x30	Byte Count															

The 32-byte RxBD is an extended version of the 8-byte RxBD. The extended feature consists of the filing feature See **Section 25.10.4**, *Filing*, on page 25-35. Software can reuse an RxBD only if the E bit is cleared.

Table 25-136. 32-Byte Receive Buffer Descriptor Field Descriptions

Offset	Bits	Description	Settings
+ 0x00	**E** 0	**Empty** Indicates whether the data buffer associated with this BD empty or full. When this bit is cleared, the status and length fields have been updated as required. This bit is written by the Ethernet controller when cleared and by the user when set.	0 The data buffer of this BD is filled with received data, or data reception is aborted due to an error condition. 1 The data buffer of this BD is empty, or reception is currently in progress.
+ 0x00	**RO1** 1	**Receive Software Ownership** This field is reserved for use by software. Hardware does not modify this read/write bit, nor does its value affect hardware.	
+ 0x00	**W** 2	**Wrap** Indicates whether the next BD is the last one. This bit is written by user.	0 The next BD is found in the consecutive location. 1 The next BD is found at the location defined in RBASE.
+ 0x00	**I** 3	**Interrupt** Specifies whether an interrupt is generated after this buffer is processed. This bit is written by the user. If you want the interrupt to occur only if RXF0 occurs, disable RXB0 (IMASK[RXBEN0] is cleared) and enable RXF0 (IMASK[RXFEN0] is set).	0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB*n*] or IEVENT[RXF*n*] are set after this buffer is serviced. These bits can cause an interrupt if enabled (IMASK[RXBEN*n*] or IMASK[RXFEN*n*]).
+ 0x00	**L** 4	**Last in Frame** Specifies whether this buffer is the last one in the receive frame. This bit is written by the Ethernet controller.	0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
+ 0x00	**F** 5	**First in Frame** Specifies whether this buffer is the first one in the receive frame. This bit is written by the Ethernet controller.	0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
+ 0x00	**M** 7	**Miss** The Ethernet controller sets this bit for frames that are accepted in promiscuous mode but are flagged as a “miss” by the internal address recognition. Thus, when the Ethernet controller is in Promiscuous mode, you can use the M bit to determine quickly whether the frame is destined to this station. This bit is valid only if the L-bit is set and Ethernet controller is in Promiscuous mode.	0 The frame was received because of an address recognition hit. 1 The frame was received because of Promiscuous mode.
+ 0x00	**BC** 8	**Broadcast** Broadcast mode. This bit is written by Ethernet controller and is valid only if L is set. L is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).	0 Normal operation. 1 Broadcast mode.

Table 25-136. 32-Byte Receive Buffer Descriptor Field Descriptions (Continued)

Offset	Bits	Description	Settings
+ 0x00	**MC** 9	**Multicast** Multicast mode. This bit is written by Ethernet controller and is valid only if L is set. L is set if the DA is multicast and not BC.	0 Normal operation. 1 Multicast mode.
+ 0x00	**LG** 10	**Rx Frame Length Violation** The length of a frame is greater than maximum frame length. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame length exceeds the maximum frame length.
+ 0x00	**NO** 11	**Rx Non-octet Aligned Frame** A frame that contained a number of bits not divisible by eight was received. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame contains a number of bits not divisible by eight.
+ 0x00	**SH** 12	**Short Frame** The length of a frame is less than the minimum length defined for this channel (MINFLR), provided RCTRL[RSF] is set. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Frame length is less than the minimum length.
+ 0x00	**CR** 13	**Rx CRC Error** The frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 CRC error or receive code group error.
+ 0x00	**OV** 14	**Overrun** A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and are zero. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Receive FIFO overrun.
+ 0x00	**TR** 15	**Truncation** Receive frame is truncated. If this bit is set, the frame must be discarded and the other error bits must be ignored because they may be incorrect. This bit is written by the Ethernet controller and is valid only if L is set.	0 Normal operation. 1 Receive frame truncated.
+ 0x0 2	**DL** 0–15	**Data Length** The number of octets written by the Ethernet controller into this BD's data buffer if L is cleared (the value is equal to the corresponding MRBLR[0–3]), or the length of the frame including CRC if L is set. This field is written by the Ethernet controller. written by the Ethernet controller.	
+ 0x04	— 0–31	Reserved. Write to zero for future compatibility.	
+ 0x06	— 0–31	Reserved. Write to zero for future compatibility.	
+ 0x08	**RXDBPTL** 0–31	**Receive Buffer Pointer** Points to the first location of the associated data buffer and must be 64-byte aligned. The buffer must reside in memory external to the Ethernet controller. This field is written by the user.	
+ 0x10	0–31		

Table 25-136. 32-Byte Receive Buffer Descriptor Field Descriptions (Continued)

Offset	Bits	Description	Settings
+ 0x12	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x14	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x16	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x18	— 0–15	Reserved. Write to zero for future compatibility.	
+ 0x20	— 0–3	Reserved. Write to zero for future compatibility.	
+ 0x20	**MP** 4–7	**Matched Pattern** Indicates which pattern matched and is valid only while PM is set. In general, the last matched entry is the one reported. For multiple matches in the same 4-byte compare window, the numerically lowest match pattern is indicated if no further match is found. The Ethernet controller uses the attributes associated with the pattern indicated by the MP field to process the frame (that is, filing). This field is written by the Ethernet controller.	0000pattern 0 0001pattern 1 .... 1111pattern 15
Offset + 20	**PM** 8	**Pattern Match** written by Ethernet controller.	0 No pattern match. 1 There is a pattern match.
Offset + 20	**MM** 9	**Multiple Match** Indicates that two or more patterns matched. This bit is written by Ethernet controller. It contains valid information only while PM is set.	0 No more than one pattern matched. 1 Two or more patterns matched.
Offset + 20	**ABPM** 10	**Accepted Base on Pattern Match** The frame was accepted based on a pattern match. This bit is written by the Ethernet controller.	0 Frame was accepted based on DA recognition. 1 Frame was accepted because of a pattern match
Offset + 20	— 11–15	Reserved. Write to zero for future compatibility.	
Offset + 22	— 0–15	Reserved. Write to zero for future compatibility.	
Offset + 24	— 0–15	Reserved. Write to zero for future compatibility.	
Offset + 26	— 0–15	Reserved. Write to zero for future compatibility.	
Offset + 28	— 0–15	Reserved. Write to zero for future compatibility.	
Offset + 30	**BCT** 0–15	**Byte Count** Indicates the actual size, in bytes, of the data to which the Rx buffer pointer points, regardless of the L bit value. With the exception of the last BD (L is set), the data length and the byte count are always the same.This field is written by the Ethernet controller.	

Programming Reference A

This reference for programmers includes tables summarizing the interrupt sources for Global Interrupt Controller (GIC) interrupts and Local Interrupt Controller (LIC) interrupts, routing through the Programmable Interrupt Controller (PIC), and programming sheets for key programmable MSC8113 registers.

A.1 Register Addressing

Register addressing is complex because of the flexible addressing model offered in the MSC8113 device. SC140 DSP core and extended core registers are shown on programming sheets with fixed addresses based on the default QBus bank configurations after reset and booting. While the QBus base addresses can be reconfigured, this is not recommended.

Internal peripherals can be addressed through the local bus, QBus, or the Direct Slave Interface (DSI). QBus addressing is through bank 3 and is fixed. The addresses listed on the programming sheets are based on the default QBus bank settings. The DSI addressing is also fixed. Local bus addressing depends on the base addresses selected by the ISBSEL bits in the Hard Reset Configuration Word (HRCW) which is loaded during a power-on reset. The memory map in **Chapter 8** provides a detailed listing of the various possibilities, and the programming sheets reference the specific page where the addresses are listed.

System registers can be addressed through the system bus or the DSI. The DSI addresses are fixed, but the system bus addresses vary depending of the values loaded in the ISBSEL bits in the HRCW. As with the internal peripherals on the IPBus, **Chapter 8** provides a detailed list of the possible system register addresses on the system bus selected by the possible ISB bit combinations, and the programming sheets reference the page that lists the system register addresses for a specific register.

For sequential groups of registers, such as those in the TDM or timer modules, the programming sheets include equations based on register numbers and base addresses with a space to write in the computed address.

A.2 Interrupts

For information on the MSC8113 interrupt controllers, refer to **Chapter 17**, *Interrupt Processing*. The following tables are a summary of interrupt sources for each of the controllers.

Table A-1. GIC $\overline{\text{INT\_OUT}}$ Sources

No.	Source	Description
0	—	Reserved
1	—	Reserved
2	—	Reserved
3	—	Reserved
4	VS24	Virtual System interrupt 24
5	VS16	Virtual System interrupt 16
6	VS8	Virtual System interrupt 8
7	VS0	Virtual System interrupt 0
8	—	Reserved
9	—	Reserved
10	—	Reserved
11	—	Reserved
12	UART	UART Interrupt
13	TMCNT	Time Counter
14	PIT	Periodic Interrupt Timer
15	DMA	DMA global interrupt
16	IRQ15	IRQ15 Signal
17	IRQ14	IRQ14 Signal
18	IRQ13	IRQ13 Signal
19	IRQ12	IRQ12 Signal
20	IRQ11	IRQ11 Signal
21	IRQ10	IRQ10 Signal
22	IRQ9	IRQ9 Signal
23	IRQ8	IRQ8 Signal
24	IRQ7	IRQ7 Signal
25	IRQ6	IRQ6 Signal
26	IRQ5	IRQ5 Signal
27	IRQ4	IRQ4 Signal
28	IRQ3	IRQ3 Signal
29	IRQ2	IRQ2 Signal
30	IRQ1	IRQ1 Signal
31	—	Reserved

Table A-2. LIC Interrupt Group A Sources (Same for all SC140 Cores)

No.	Source	Description
0	TDM0RXER	TDM0 Receive Error (sum of TDM receive error detections).
1	TDM0RSTE	TDM0 Receive Second Threshold Event.
2	TDM0RFTE	TDM0 Receive First Threshold Event.
3	TDM0TXER	TDM0 Transmit Error (sum of TDM transmit error detections).
4	TDM0TSTE	TDM0 Transmit Second Threshold Event.
5	TDM0TFTE	TDM0 Transmit First Threshold Event.
6	TDM1RXER	TDM1 Receive Error (sum of TDM receive error detections).
7	TDM1RSTE	TDM1 Receive Second Threshold Event
8	TDM1RFTE	TDM1 Receive First Threshold Event
9	TDM1TXER	TDM1 Transmit Error (sum of TDM transmit error detections).
10	TDM1TSTE	TDM1 Transmit Second Threshold Event
11	TDM1TFTE	TDM1 Transmit First Threshold Event
12	TDM2RXER	TDM2 Receive Error (sum of TDM receive error detections).
13	TDM2RSTE	TDM2 Receive Second Threshold Event
14	TDM2RFTE	TDM2 Receive First Threshold Event
15	TDM2TXER	TDM2 Transmit Error (sum of TDM transmit error detections).
16	TDM2TSTE	TDM2 Transmit Second Threshold Event
17	TDM2TFTE	TDM2 Transmit First Threshold Event
18	TDM3RXER	TDM3 Receive Error (sum of TDM receive error detections).
19	TDM3RSTE	TDM3 Receive Second Threshold Event
20	TDM3RFTE	TDM3 Receive First Threshold Event
21	TDM3TXER	TDM3 Transmit Error (sum of TDM transmit error detections).
22	TDM3TSTE	TDM3 Transmit Second Threshold Event
23	TDM3TFTE	TDM3 Transmit First Threshold Event
24	DMA	DMA global interrupt (sum of all channel interrupts)
25	DMA_ERROR	DMA error
26	—	Reserved
27	—	Reserved
28	—	Reserved
29	—	Reserved
30	—	Reserved
31	—	Reserved

Table A-3. LIC Interrupt Group B Source for Core 0

No.	Source	Description
0	DMA0	DMA channel 0 interrupt
1	DMA1	DMA channel 1 interrupt
2	DMA2	DMA channel 2 interrupt
3	DMA3	DMA channel 3 interrupt
4	DMA4	DMA channel 4 interrupt
5	DMA5	DMA channel 5 interrupt
6	DMA6	DMA channel 6 interrupt
7	DMA7	DMA channel 7 interrupt
8	TIMER0A	Timer Block A Timer 0 Compare Flag
9	TIMER1A	Timer Block A Timer 1 Compare Flag
10	TIMER2A	Timer Block A Timer 2 Compare Flag
11	TIMER3A	Timer Block A Timer 3 Compare Flag
12	TIMER8A	Timer Block A Timer 8 Compare Flag
13	TIMER9A	Timer Block A Timer 9 Compare Flag
14	TIMER10A	Timer Block A Timer 10 Compare Flag
15	TIMER11A	Timer Block A Timer 11 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	IRQ1	IRQ1 signal (global)
20	VIRQ0	Virtual Interrupt Number 0
21	VIRQ1	Virtual Interrupt Number 1
22	VIRQ2	Virtual Interrupt Number 2
23	VIRQ3	Virtual Interrupt Number 3
24	VIRQ4	Virtual Interrupt Number 4
25	VIRQ5	Virtual Interrupt Number 5
26	VIRQ6	Virtual Interrupt Number 6
27	VIRQ7	Virtual Interrupt Number 7
28	IRQ4	IRQ4 signal (global)
29	IRQ5	IRQ5 signal (global)
30	IRQ6	IRQ6 signal (global)
31	IRQ7	IRQ7 signal (global)

Table A-4. LIC Interrupt Group B Source for Core 1

No.	Source	Description
0	DMA0	DMA channel 0 interrupt
1	DMA1	DMA channel 1 interrupt
2	DMA2	DMA channel 2 interrupt
3	DMA3	DMA channel 3 interrupt
4	DMA4	DMA channel 4 interrupt
5	DMA5	DMA channel 5 interrupt
6	DMA6	DMA channel 6 interrupt
7	DMA7	DMA channel 7 interrupt
8	TIMER4A	Timer Block A Timer 4 Compare Flag
9	TIMER5A	Timer Block A Timer 5 Compare Flag
10	TIMER6A	Timer Block A Timer 6 Compare Flag
11	TIMER7A	Timer Block A Timer 7 Compare Flag
12	TIMER12A	Timer Block A Timer 12 Compare Flag
13	TIMER13A	Timer Block A Timer 13 Compare Flag
14	TIMER14A	Timer Block A Timer 14 Compare Flag
15	TIMER15A	Timer Block A Timer 15 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	IRQ1	IRQ1 signal (global)
20	VIRQ8	Virtual Interrupt Number 8
21	VIRQ9	Virtual Interrupt Number 9
22	VIRQ10	Virtual Interrupt Number 10
23	VIRQ11	Virtual Interrupt Number 11
24	VIRQ12	Virtual Interrupt Number 12
25	VIRQ13	Virtual Interrupt Number 13
26	VIRQ14	Virtual Interrupt Number 14
27	VIRQ15	Virtual Interrupt Number 15
28	IRQ4	IRQ4 signal (global)
29	IRQ5	IRQ5 signal (global)
30	IRQ6	IRQ6 signal (global)
31	IRQ7	IRQ7 signal (global)

Table A-5. LIC Interrupt Group B Source for Core 2

No.	Source	Description
0	DMA8	DMA channel 8 interrupt
1	DMA9	DMA channel 9 interrupt
2	DMA10	DMA channel 10 interrupt
3	DMA11	DMA channel 11 interrupt
4	DMA12	DMA channel 12 interrupt
5	DMA13	DMA channel 13 interrupt
6	DMA14	DMA channel 14 interrupt
7	DMA15	DMA channel 15 interrupt
8	TIMER0B	Timer Block B Timer 0 Compare Flag
9	TIMER1B	Timer Block B Timer 1 Compare Flag
10	TIMER2B	Timer Block B Timer 2 Compare Flag
11	TIMER3B	Timer Block B Timer 3 Compare Flag
12	TIMER8B	Timer Block B Timer 8 Compare Flag
13	TIMER9B	Timer Block B Timer 9 Compare Flag
14	TIMER10B	Timer Block B Timer 10 Compare Flag
15	TIMER11B	Timer Block B Timer 11 Compare Flag
16	UART	UART Tx & Rx Interrupt (global)
17	PIT	Periodic Interrupt Timer (global)
18	TMCNT	Timer Counter (global)
19	IRQ1	IRQ1 signal (global)
20	VIRQ16	Virtual Interrupt Number 16
21	VIRQ17	Virtual Interrupt Number 17
22	VIRQ18	Virtual Interrupt Number 18
23	VIRQ19	Virtual Interrupt Number 19
24	VIRQ20	Virtual Interrupt Number 20
25	VIRQ21	Virtual Interrupt Number 21
26	VIRQ22	Virtual Interrupt Number 22
27	VIRQ23	Virtual Interrupt Number 23
28	IRQ4	IRQ4 signal (global)
29	IRQ5	IRQ5 signal (global)
30	IRQ6	IRQ6 signal (global)
31	IRQ7	IRQ7 signal (global)

Table A-6. MSC8113 Interrupt Routing

VAB[0–5]	Signal	Description	Service Routine Address (Offset from VBA)
0x0	TRAP	Internal exception (generated by trap instruction)	0x0
0x1	—	Reserved	0x40
0x2	ILLEGAL	Illegal instruction or set	0x80
0x3	DEBUG	Debug exception (EOnCE)	0xC0
0x4	—	Reserved	0x100
0x5	OVERFLOW	Overflow exception (DALU)	0x140
0x6	DEFAULT $\overline{\text{NMI}}$	In VAB disabled mode only	0x180
0x7	DEFAULT IRQ	In VAB disabled mode only	0x1C0
0x8–0x1F	—	Reserved	0x200–0x7FF
0x20	IRQ0	Reserved	0x800
0x21	IRQ1	Reserved	0x840
0x22	IRQ2	Reserved	0x880
0x23	IRQ3	Reserved	0x8C0
0x24	IRQ4	Reserved	0x900
0x25	IRQ5	Reserved	0x940
0x26	IRQ6	LIC IRQOUTA0 - Group A	0x980
0x27	IRQ7	LIC IRQOUTA1 - Group A	0x9C0
0x28	IRQ8	LIC IRQOUTA2 - Group A	0xA00
0x29	IRQ9	LIC IRQOUTA3 - Group A	0xA40
0x2A	IRQ10	Bus controller (x-y contention)	0xA80
0x2B	IRQ11	Bus controller (level1 contention)	0xAC0
0x2C	IRQ12	Bus controller (p-x contention)	0xB00
0x2D	IRQ13	Bus controller (nonaligned data error)	0xB40
0x2E	IRQ14	LIC IRQOUTB0- Group B	0xB80
0x2F	IRQ15	External IRQ2 (edge/level configurable)	0xBC0
0x30	IRQ16	GIC - global interrupt	0xC00
0x31	IRQ17	External IRQ3 (edge/level configurable)	0xC40
0x32	IRQ18	LIC IRQOUTB1 - Group B	0xC80
0x33	IRQ19	Reserved	0xCC0
0x34	IRQ20	EOnCE interrupt (edge-triggered)	0xD00
0x35	IRQ21	LIC IRQOUTB2 - Group B	0xD40
0x36	IRQ22	LIC IRQOUTB3 - Group B	0xD80
0x37	IRQ23	LICSEIRQ - LIC Second Edge IRQ (Groups A and B)	0xDC0
0x38	NMI0	GIC Virtual $\overline{\text{NMI}}$ of this core	0xE00
0x39	NMI1	Reserved	0xE40

Table A-6. MSC8113 Interrupt Routing (Continued)

VAB[0–5]	Signal	Description	Service Routine Address (Offset from VBA)
0x3A	NMI2	Bus controller (memory write error)	0xE80
0x3B	NMI3	Bus controller (nonaligned error)	0xEC0
0x3C	NMI4	Bus Controller (bus error - access to unmapped memory space)	0xF00
0x3D	NMI5	System I/F block TEA on System PPC bus	0xF40
0x3E	NMI6	Reserved	0xF80
0x3F	NMI7	SIU $\overline{\text{NMI}}$ (from GIC), for example, S/W watchdog, external NMI, parity error, bus monitor	0xFC0

A.3 Programming Sheets

The programming sheets are grouped in the order shown in **Table A-7**. Each sheet has space to write in the value of each bit and the hexadecimal value for each register. You can photocopy these sheets and reuse them for each application development project. For details on the instruction set of the SC140 core, see the *SC140 DSP Core Reference Manual*, which is available at the web site listed on the back cover of this manual.

Table A-7. Guide to MSC8113 Programming Sheets

Module	Programming Sheet	Type	Page
SC140 Core	Status Register (SR)	R/W	**page A-16**
	Exception and Mode Register (EMR)	R/W	**page A-17**
EQBS Configuration	QBus Mask for Bank 0 (QBUSMR0)	R/W	**page A-18**
	QBus Base for Bank 0 (QBUSBR0)	R/W	**page A-18**
	QBus Mask for Bank 1 (QBUSMR1)	R/W	**page A-18**
	QBus Base for Bank 1 (QBUSBR1)	R/W	**page A-18**
	QBus Mask for Bank 2 (QBUSMR2)	R/W	**page A-18**
	QBus Base for Bank 2 (QBUSBR2)	R/W	**page A-18**
	Instruction Cacheable Area Control Register (ICACR)	R/W	**page A-19**
	Instruction Cacheable Area Base Register (ICABR)	R/W	**page A-19**
	Instruction Fetch Unit Configuration Register (IFUR)	R/W	**page A-20**
	Write Buffer Control Register (WBCR)	R/W	**page A-20**
	Data Area Registers (DBR[0–3])	R/W	**page A-21**
	FlyBy Address Control Register (FLBACR0)	R/W	**page A-21**
Reset Configuration	Hard Reset Configuration Word (HRCW)	R/W	**page A-22**
	Reset Status Register (RSR)	R/W	**page A-23**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
System Interface Unit (SIU)	Bus Configuration Register (BCR)	R/W	**page A-24**
	System Bus Arbiter Configuration Register (PPC_ACR)	R/W	**page A-25**
	System Bus Arbitration-Level Registers (PPC_ALRH/PPC_ALRL)	R/W	**page A-26**
	Local Bus Arbiter Configuration Register (LCL_ACR)	R/W	**page A-27**
	Local Bus Arbitration Level Registers (LCL_ALRH/LCL_ALRL	R/W	**page A-28**
	SIU Module Configuration Register (SIUMCR), Sheet 1	R/W	**page A-29**
	SIU Module Configuration Register (SIUMCR), Sheet 2	R/W	**page A-30**
	Internal Memory Map Register (IMMR)	R/W	**page A-31**
	System Protection Control Register (SYPCR)	R/W	**page A-32**
	Software Service Register (SWSR)	R/W	**page A-33**
	System Bus Error Status and Control Register 1 (TESCR1)	R/W	**page A-34**
	System Bus Error Status and Control Register 2 (TESCR2)	R/W	**page A-35**
	Local Bus Error Status and Control Register (L_TESCR1)	R/W	**page A-36**
	Timer Counter Status and Control Register (TMCNTSC)	R/W	**page A-37**
	Time Counter Register (TMCNT)	R/W	**page A-38**
	Time Counter Alarm Register (TMCNTAL)	R/W	**page A-38**
	Periodic Interrupt Status and Control Register (PISCR)	R/W	**page A-39**
	Periodic Interrupt Timer Count Register (PITC)	R/W	**page A-39**
Memory Controller	Base Registers (BR[0–7, 9–11])	R/W	**page A-40**
	Option Registers (OR[0–7, 9–11]), GPCM mode	R/W	**page A-41**
	Option Registers (OR[0–7, 9–11]), SDRAM mode	R/W	**page A-42**
	Option Registers (OR[0–7, 9–11]), UPM mode	R/W	**page A-43**
	UPM A/B/C Mode Registers (MAMR, MBMR, MCMR)	R/W	**page A-44**
	Bus SDRAM Mode Register (PSDMR), Sheet 1	R/W	**page A-45**
	Bus SDRAM Mode Register (PSDMR), Sheet 2	R/W	**page A-46**
	Memory Data Register (MDR)	R/W	**page A-47**
	Memory Address Register (MAR)	R/W	**page A-47**
	Bus Assigned UPM Refresh Timer (PURT)	R/W	**page A-48**
	Bus Assigned SDRAM Refresh Timer (PSRT)	R/W	**page A-48**
	Memory Refresh Timer Prescaler Register (MPTPR)	R/W	**page A-48**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
Interrupt Controllers	Virtual Interrupt Generation Register (VIGR)	W	**page A-49**
	Virtual Interrupt Status Register (VISR)	R/W	**page A-49**
	Virtual NMI Generation Register (VNMIGR)	W	**page A-50**
	GIC Interrupt Configuration Register (GICR)	R/W	**page A-50**
	GIC External Interrupt Enable Register (GEIER)	R/W	**page A-51**
	GIC Core Interrupt Enable Register (GCIER)	R/W	**page A-52**
	GIC Interrupt Status Register (GISR)	R/W	**page A-52**
	LIC Group A Interrupt Configuration Register (LICAICR[0–3])	R/W	**page A-53**
	LIC Group B Interrupt Configuration Register (LICBICR[0–3])	R/W	**page A-54**
	LIC Group A Interrupt Enable Register (LICAIER)	R/W	**page A-55**
	LIC Group A Interrupt Status Register (LICAISR)	R/W	**page A-55**
	LIC Group A Interrupt Error Status Register (LICAIESR)	R/W	**page A-55**
	LIC Group B Interrupt Enable Register (LICBIER)	R/W	**page A-55**
	LIC Group B Interrupt Status Register (LICBISR)	R/W	**page A-55**
	LIC Group B Interrupt Error Status Register (LICBIESR)	R/W	**page A-55**
	PIC Edge/Level-Triggered Interrupt Priority Register A (ELIRA) PIC Edge/Level-Triggered Interrupt Priority Register B (ELIRB)	R/W	**page A-56**
	PIC Edge/Level-Triggered Interrupt Priority Register C (ELIRC) PIC Edge/Level-Triggered Interrupt Priority Register D (ELIRD)	R/W	**page A-57**
	PIC Edge/Level-Triggered Interrupt Priority Register E (ELIRE) PIC Edge/Level-Triggered Interrupt Priority Register F (ELIRF)	R/W	**page A-58**
	Interrupt Pending Register A Interrupt Pending Register B	R/W	**page A-59**
Instruction Cache	ICache Control Register (ICCR)	R/W	**page A-60**
	ICache Command Register (ICCMR)	W	**page A-60**
Direct Memory Access (DMA) Controller	DMA Channel Configuration Registers (DCHCR[0–15]), Sheet 1	R/W	**page A-61**
	DMA Channel Configuration Registers (DCHCR[0–15]), Sheet 2	R/W	**page A-62**
	DMA Pin Configuration Register (DPCR)	R/W	**page A-63**
	DMA Channel Parameters RAM (DCPRAM) • Buffer Address (BD_ADDR) • Buffer Size (BD_SIZE) • Buffer Attributes Parameter (BD_ATTR) • Buffer Base Size (BD_BSIZE)	R/W	**page A-64** **page A-64** **page A-65** **page A-66** **page A-64**
	DMA Status Register (DSTR)	R/W	**page A-63**
	DMA Internal Mask Register (DIMR)	R/W	**page A-67**
	DMA External Mask Register (DEMR)	R/W	**page A-67**
	DMA Transfer Error Address Status Register (DTEAR)	R/W	**page A-68**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
Direct Slave Interface (DSI)	DSI Control Register (DCR)	R/W	**page A-69**
	DSI Sliding Window Base Address Register (DSWBAR)	R/W	**page A-70**
	DSI Internal Base Address Registers (DIBAR[9, 11])	R/W	**page A-70**
	DSI Internal Address Mask Registers (DIAMR[9, 11])	R/W	**page A-70**
	DSI Chip ID Register (DCIR)	R/W	**page A-71**
	DSI Disable Register (DDR)	R/W R	**page A-71**
	DSI Error Register (DER)	R/W	**page A-71**
IP Bus	Stop Control Register (SCR)	R/W	**page A-72**
Hardware Semaphores	Hardware Semaphore Registers (HSMPR[0–7])	R/W	**page A-73**
TDM Interfaces	TDM[0–3] General Interface Registers (TDM[0–3]GIR)	R/W	**page A-74**
	TDM[0–3] Receive Interface Registers (TDM[0–3]RIR)	R/W	**page A-75**
	TDM[0–3] Transmit Interface Registers (TDM[0–3]TIR)	R/W	**page A-76**
	TDM[0–3] Receive Frame Parameters (TDM[0–3]RFP)	R/W	**page A-77**
	TDM[0–3] Transmit Frame Parameters (TDM[0–3]TFP)	R/W	**page A-78**
	TDM[0–3] Receive Data Buffer Size (TDM[0–3]RDBS)	R/W	**page A-79**
	TDM[0–3] Transmit Data Buffer Size (TDM[0–3]TDBS)	R/W	**page A-79**
	TDM[0–3] Receive Global Base Address (TDM[0–3]RGBA)	R/W	**page A-80**
	TDM[0–3] Transmit Global Base Address (TDM[0–3]TGBA)	R/W	**page A-80**
	TDM[0–3] Adaptation Control Registers (TDM[0–3]ACR)	R/W	**page A-81**
	TDM[0–3] Receive Control Registers (TDM[0–3]RCR)	R/W	**page A-82**
	TDM[0–3] Transmit Control Registers (TDM[0–3]TCR)	R/W	**page A-82**
	TDM[0–3] Receive Data Buffers First Threshold (TDM[0–3]RDBFT)	R/W	**page A-83**
	TDM[0–3] Transmit Data Buffers First Threshold (TDM[0–3]TDBFT)	R/W	**page A-83**
	TDM[0–3] Receive Data Buffers Second Threshold (TDM[0–3]RDBST)	R/W	**page A-84**
	TDM[0–3] Transmit Data Buffers Second Threshold (TDM[0–3]TDBST)	R/W	**page A-84**
	TDM[0–3] Receive Channel Parameter Registers (TDMRCPR[0–255])	R/W	**page A-85**
	TDM[0–3] Transmit Channel Parameter Registers (TDMTCPR[0–255])	R/W	**page A-85**
	TDM[0–3] Receive Interrupt Enable Registers (TDM[0–3]RIER)	R/W	**page A-86**
	TDM[0–3] Transmit Interrupt Enable Registers (TDM[0–3]TIER)	R/W	**page A-86**
	TDM[0–3] Receive Event Registers (TDM[0–3]RER)	R/W	**page A-87**
	TDM[0–3] Transmit Event Registers (TDM[0–3]TER)	R/W	**page A-87**
	TDM[0–3] Adaptation Status Registers (TDM[0–3]ASR)	R/W	**page A-88**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
UART	SCI Baud-Rate Register (SCIBR)	R/W	**page A-89**
	SCI Data Register (SCIDR)	R/W	**page A-89**
	SCI Data Direction Register (SCIDDR)	R/W	**page A-89**
	SCI Control Register (SCICR)	R/W	**page A-90**
Timers	Timer General Configuration Register A (TGCRA)	R/W	**page A-91**
	Timer General Configuration Register B (TGCRB)	R/W	**page A-92**
	Timer Interrupt Enable Register A (TIERA)	R/W	**page A-93**
	Timer Interrupt Enable Register B (TIERB)	R/W	‘**page A-93**
	Timer Configuration Registers A (TCRFA[0–15])	R/W	**page A-94**
	Timer Configuration Registers B (TCRFB[0–15])	R/W	**page A-95**
	Timer Compare Registers A (TCMPA[0–15])	R/W	**page A-96**
	Timer Compare Registers B (TCMPB[0–15])	R/W	**page A-96**
	Timer Control Registers A (TCRA[0–15])	R/W	**page A-97**
	Timer Control Registers B (TCRB[0–15])	R/W	**page A-97**
	Timer Event Register A (TERA)	R/W	**page A-98**
	Timer Event Register B (TERB)	R/W	**page A-98**
GPIOs	Pin Open-Drain Register (PODR)	R/W	**page A-99**
	Pin Data Register (PDAT)	R/W	**page A-99**
	Pin Data Direction Register (PDIR)	R/W	**page A-99**
	Pin Assignment Register (PAR)	R/W	**page A-100**
	Pin Special Options Register (PSOR)	R/W	**page A-100**
Ethernet Controller	Interrupt Event Register (IEVENT)	R/W	**page A-101**
	Interrupt Mask Register (IMASK)	R/W	**page A-102**
	Ethernet Control Register (ECNTRL)	R/W	**page A-103**
	Minimum Frame Length Register (MINFLR)	R/W	**page A-103**
	Pause Time Value Register (PTV)	R/W	**page A-104**
	DMA Control Register (DMACTRL)	R/W	**page A-104**
	DMA Maintenance Register (DMAMR)	R/W	**page A-105**
	FIFO Receive Status Register (FRXSTATR)	R/W	**page A-106**
	FIFO Receive Control Register (FRXCTRLR)	R/W	**page A-106**
	FIFO Receive Alarm Register (FRXALAR)	R/W	**page A-107**
	FIFO Receive Alarm Shutoff Register (FRXSHR)	R/W	**page A-107**
	FIFO Receive Panic Register (FRXPAR)	R/W	**page A-108**
	FIFO Receive Panic Shutoff Register (FRXPSR)	R/W	**page A-108**
	FIFO Transmit Status Register (FTXSTATR)	R/W	**page A-109**
	FIFO Transmit Threshold Register (FTXTHR)	R/W	**page A-109**
	FIFO Transmit Space Available Register (FTXSPR)	R/W	**page A-110**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
Ethernet Controller (cont.)	FIFO Transmit Starve Register (FTXSR)	R/W	**page A-110**
	FIFO Transmit Starve Shutoff Register (FTXSSR)	R/W	**page A-110**
	Transmit Control Register (TCTRL)	R/W	**page A-111**
	Transmit Status Register (TSTAT)	R/W	**page A-111**
	TxBD Data Length Register (TBDLEN)	R/W	**page A-112**
	Current TxBD Pointer (CTBPTR)	R/W	**page A-112**
	TxBD Pointer (TBPTR)	R/W	**page A-113**
	Transmit Descriptor Base Address (TBASE)	R/W	**page A-113**
	Out-of-Sequence TxBD Register (OSTBD)	R/W	**page A-114**
	Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)	R/W	**page A-115**
	Out-of-Sequence 32 Bytes Tx Data Buffer Pointer Register (OS32TBDP)	R/W	**page A-115**
	Out-of-Sequence 32 Bytes TxBD Insert Pointer Register (OS32IPTR)	R/W	**page A-116**
	Out-of-Sequence 32 Bytes TxBD Reserved Register (OS32TBDR)	R/W	**page A-116**
	Out-of-Sequence 32 Bytes TxBD Insert Index/Length Register (OS32IIL)	R/W	**page A-117**
	Receive Control Register (RCTRL)	R/W	**page A-118**
	Receive Status Register (RSTAT)	R/W	**page A-118**
	RxBD Data Length Register (RBDLEN)	R/W	**page A-119**
	Current RxBD Pointer (CRBPTR)	R/W	**page A-119**
	Maximum Receive Buffer Length R0R1 Register (MRBLR0R1)	R/W	**page A-120**
	Maximum Receive Buffer Length R2R3 Register (MRBLR2R3)	R/W	**page A-120**
	RxBD Pointer 0–3 (RBPTR[0–3])	R/W	**page A-121**
	Receive Descriptor Base Address 0–3 (RBASE[0–3])	R/W	**page A-121**
	MAC Configuration 1 Register (MACCFG1R)	R/W	**page A-122**
	MAC Configuration 2 Register (MACCFG2R)	R/W	**page A-123**
	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFGR)	R/W	**page A-124**
	Half-Duplex Register (HAFDUPR)	R/W	**page A-125**
	Maximum Frame Length Register (MAXFRMR)	R/W	**page A-126**
	Interface Status Register (IFSTATR)	R/W	**page A-126**
	MAC Station Address Part 1 Register (MACSTADDR1R)	R/W	**page A-127**
	MAC Station Address Part 2 Register (MACSTADDR2R)	R/W	**page A-128**
	MII Management Configuration Register (MIICFGR)	R/W	**page A-129**
	MII Management Command Register (MIICOMR)	R/W	**page A-129**
	MII Management Address Register (MIIMADDR)	R/W	**page A-130**
	MII Management Control Register (MIIMCONR)	R/W	**page A-130**
	MIIGSK Configuration Register (MIIGSK_CFGR)	R/W	**page A-131**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
Ethernet Controller (cont.)	MIIGSK General-Purpose Register (MIIGSK_GPR)	R/W	**page A-131**
	MIIGSK Enable Register (MIIGSK_ENR)	R/W	**page A-132**
	MIIGSK SMII SYNC Direction Register (MIIGSK_SMII_SYNCDIR)	R/W	**page A-132**
	MIIGSK Transmit Inter-Frame Bits Register (MIIGSK_TIFBR)	R/W	**page A-133**
	MIIGSK Expected Receive Inter-Frame Bits Register (MIIGSK_ERIFBR)	R/W	**page A-133**
	MIIGSK SMII Interrupt Event Register (MIIGSK_IEVENT)	R/W	**page A-134**
	MIIGSK SMII Interrupt Mask Register (MIIGSK_IMASK)	R/W	**page A-134**
	Transmit and Receive 64-Byte Frame Counter (TR64)	R/W	**page A-135**
	Transmit and Receive 65- to 127-Byte Frame Counter (TR127)	R/W	**page A-135**
	Transmit and Receive 128- to 255-Byte Frame Counter (TR255)	R/W	**page A-136**
	Transmit and Receive 256- to 511-Byte Frame Counter (TR511)	R/W	**page A-136**
	Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)	R/W	**page A-137**
	Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)	R/W	**page A-137**
	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)	R/W	**page A-138**
	Receive Byte Counter (RBYT)	R/W	**page A-138**
	Receive Packet Counter (RPKT)	R/W	**page A-139**
	Receive FCS Error Counter (RFCS)	R/W	**page A-139**
	Receive Multicast Packet Counter (RMCA)	R/W	**page A-140**
	Receive Broadcast Packet Counter (RBCA)	R/W	**page A-140**
	Receive Control Frame Packet Counter (RXCF)	R/W	**page A-141**
	Receive Pause Frame Packet Counter (RXPF)	R/W	**page A-141**
	Receive Unknown OPCode Packet Counter (RXUO)	R/W	**page A-142**
	Receive Alignment Error Counter (RALN)	R/W	**page A-142**
	Receive Frame Length Error Counter (RFLR)	R/W	**page A-143**
	Receive Code Error Counter (RCDE)	R/W	**page A-143**
	Receive Carrier Sense Error Counter (RCSE)	R/W	**page A-144**
	Receive Undersize Packet Counter (RUND)	R/W	**page A-144**
	Receive Oversize Packet Counter (ROVR)	R/W	**page A-145**
	Receive Fragments Counter (RFRG)	R/W	**page A-145**
	Receive Jabber Counter (RJBR)	R/W	**page A-146**
	Receive Dropped Packet Counter (RDRP)	R/W	**page A-146**
	Transmit Byte Counter (TBYT)	R/W	**page A-147**
	Transmit Packet Counter (TPKT)	R/W	**page A-147**
	Transmit Multicast Packet Counter (TMCA)	R/W	**page A-148**
	Transmit Broadcast Packet Counter (TBCA)	R/W	**page A-148**
	Transmit Pause Control Frame Counter (TXPF)	R/W	**page A-149**

Table A-7. Guide to MSC8113 Programming Sheets (Continued)

Module	Programming Sheet	Type	Page
Ethernet Controller (cont.)	Transmit Deferral Packet Counter (TDFR)	R/W	**page A-149**
	Transmit Excessive Deferral Packet Counter (TEDF)	R/W	**page A-150**
	Transmit Single Collision Packet Counter (TSCL)	R/W	**page A-150**
	Transmit Multiple Collision Packet Counter (TMCL)	R/W	**page A-151**
	Transmit Late Collision Packet Counter (TLCL)	R/W	**page A-151**
	Transmit Excessive Collision Packet Counter (TXCL)	R/W	**page A-152**
	Transmit Total Collision Counter (TNCL)	R/W	**page A-152**
	Transmit Jabber Frame Counter (TJBR)	R/W	**page A-153**
	Transmit FCS Error Counter (TFCS)	R/W	**page A-154**
	Transmit Control Frame Counter (TXCF)	R/W	**page A-154**
	Transmit Oversize Frame Counter (TOVR)	R/W	**page A-155**
	Transmit Undersize Frame Counter (TUND)	R/W	**page A-155**
	Transmit Fragment Counter (TFRG)	R/W	**page A-156**
	Carry Register One (CAR1)	R/W	**page A-157**
	Carry Register Two (CAR2)	R/W	**page A-158**
	Carry Register One Mask (CAM1)	R/W	**page A-159**
	Carry Register Two Mask (CAM2)	R/W	**page A-160**
	Individual Address Registers 0–7 (IADDR[0–7)	R/W	**page A-161**
	Group Address Registers 0–7 (GADDR[0–7])	R/W	**page A-161**
	Pattern Match Data 0–15 (PMD[0–15])	R/W	**page A-162**
	Pattern Mask Register 0–15 (PMASK[0–15])	R/W	**page A-162**
	Pattern Match Control Register 0–15 (PCNTRL[0–15])	R/W	**page A-163**
	Pattern Match Attributes Register 0–15 (PATTRB[0–15])	R/W	**page A-164**
	Default Attribute Register (DATTR)	R/W	**page A-164**

SC140 Core

SR

Status Register
Reset: 0x00E40000
Read/Write

EXP – Exception Mode, Bit 18	
0	Normal, stack pointer NSP
1	Exception, stack pointer ESP

DI – Disable Interrupts, Bit 19	
0	Interrupts enabled
1	Interrupts disabled

OVE – Disable Reflection of System Bus, Bit 20	
0	Overflow exception generation disabled
1	Overflow exception generation enabled unless EMR[DOVF] = 1

I[2–0] – Interrupt Mask, Bits 23–21

I[2–0]	Exceptions Permitted	Exceptions Masked
000	IPL 1–7	IPL 0
001	IPL 2–7	IPL 0–1
010	IPL 3–7	IPL 0–2
011	IPL 4–7	IPL 0–3
100	IPL 5–7	IPL 0–4
101	IPL 6–7	IPL 0–5
110	IPL 7	IPL 0–6
111	NMI	IPL 0–7

LF[3–0] – Loop Flags 3–0, Bits 30–27	
0	Hardware loop not enabled
1	Hardware loop enabled

Note: LF3 = Loop #4; LF2 = Loop #3; LF1 = Loop #2; LF0 = Loop #1

SLF – Short Loop Flag, Bit 31	
0	Active loop length is 3 or more execution sets
1	Active loop length is 1 or 2 execution sets

VF[3–0] – Viterbi Flags 3–0, Bits 11–8	
0	Appropriate 16-bit portion transferred
1	Appropriate 16-bit portion not transferred

S – Scaling, Bit 6	
0	Reset or Abs. value of data moved <0.25 or ≥0.75
1	Abs. value of data moved ≥0.25 and <0.75

S[1–0] – Scaling Mode, Bits 5–4	
00	Rounding bit 15, no scaling
01	Rounding bit 16, scale down (1-bit arithmetic right shift)
10	Rounding bit 14, scale up (1-bit arithmetic left shift)
11	Reserved

RM – Rounding Mode, Bit 3	
0	Convergent rounding selected
1	Two's complement rounding selected

SM – Saturation Mode, Bit 2	
0	Arithmetic saturation mode not selected
1	Arithmetic saturation mode selected

T – True, Bit 1	
0	Compare or test instruction condition is false
1	Compare or test instruction condition is true

C – Carry, Bit 0	
0	No carry/borrow generated
1	Carry from last addition or borrow from last subtraction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLF	LF3	LF2	LF1	LF0	* 0	* 0	* 0	I2	I1	I0	OVE	DI	EXP	* 0	* 0	* 0	* 0	* 0	* 0	VF3	VF2	VF1	VF0	* 0	S	S1	S0	RM	SM	T	C

* = Reserved. Write to 0 for future compatibility

SC140 Core

EMR

Exception and Mode Register

Reset: 0x00xx0000
Lowest 4 Bits Read/Write
All other bits Read Only

BEM – Big-Endian Memory, Bit 16 (read-only, set by default at reset to 1)	
0	Little-endian configuration
1	Big-endian configuration

GP[6–0] – General Purpose Flags, Bits 23–17 (read-only, set by default or external pin at reset)	
GP6	EE1 value
GP5	0
GP4	ISBSEL2 (Hard Reset Configuration Word, bit 15)
GP3	ISBSEL1 (Hard Reset Configuration Word, bit 14)
GP2	Inverse of ISBSEL0 (Hard Reset Configuration Word, bit 13)
GP1	0
GP0	0

NMID – Non-maskable Interrupt Disable, Bit 3	
0	NMI enabled, no NMI service is executing
1	NMI disabled, NMI service executing

DOVF – Data ALU Overflow, Bit 2	
0	No overflow or arithmetic saturation occurred
1	Overflow or arithmetic saturation occurred

ILST – Illegal Execution Set, Bit 1	
0	No execution set rule violated
1	Execution set rule violated

ILIN – Illegal Instruction, Bit 0	
0	No instruction set violation
1	One or more opcodes not SC140 instruction set

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	GP6	GP5	GP4	GP3	GP2	GP1	GP0	BEM	*	*	*	*	*	*	*	*	*	*	*	*	NMID	DOVF	ILST	ILIN
0	0	0	0	0	0	0	0		0				0	0	1	0	0	0	0	0	0	0	0	0	0	0	0				

* = Reserved. Write to 0 for future compatibility

VBA

Vector Base Address Register

Initial Address after Boot: 0x01077000
Highest 20 Bits Read/Write, Lowest 12 Bits Always 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																				*	*	*	*	*	*	*	*	*	*	*	*
																				0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Extended QBus System (EQBS)

QBUSBR[0–2]

QBus Base Registers 0–2

Addresses: 0x00F0FF02; 0x00F0FF06; 0x00F0FF0A
Reset: QBUSBR0 = 0x00F0; QBUSBR1 = 0x0100; QBUSR2 = 0x0000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bank Base[0–15]															

QBUSMR[1–2]

QBus Mask Registers 0–2

Addresses: 0x00F0FF04; 0x00F0FF08
Reset: QBUSMR1 = 0xFF00 (Boot rewrites to 0XFF80); QBUSMR2 = 0xFFFF
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bank Mask [0–15]															

Note: QBUSMR0 is a read-only register with a reset mask value of 0xFFFF.

Extended QBus System (EQBS)

ICACR

Instruction Cacheable Area Control Register

Address: 0x00F0FF30
Reset: 0x0600
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* **0**	* **0**	* **0**	* **0**	* **0**	EN	REV	SIZE	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**

EN – Enable Area Operation, Bit 5	
0	Area disabled
1	Area enabled

REV – Reverse Cacheable Area, Bit 6	
0	Cacheable area inside the area definition
1	Cacheable area outside the area definition

SIZE – Size Indication, Bit 7	
0	Size is not 64 KB
1	Size is 64 KB

* = Reserved. Write to 0 for future compatibility

ICABR

Instruction Cacheable Area Base Register

Address: 0x00F0FF32
Reset: 0x0080
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Area Base[0–15]															

Extended QBus System (EQBS)

IFUR

Instruction Fetch Unit Configuration Register
Address: 0x00F0FF60
Reset: 0x0010
Read/Write

SIZE – Block Size, Bits 13–15	
SIZE[2–0]	**Block Size**
000	1
001	2
010	4
011–111	reserved

PFOFF – Pre-fetch Off, Bit 11	
0	Pre-fetch enabled
1	Pre-fetch disabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	PFOFF	* 0	SIZE		

* = Reserved. Write to 0 for future compatibility

WBCR

Write Buffer Control Register
Address: 0x00F0FF82
Reset: 0x13FF
Read/Write

WBOFF – Write Buffer Disable, Bit 3	
0	Write Buffer enabled
1	Write Buffer disabled

WD – Watchdog Count, Bits 6–15
Contains the watchdog count value.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	* 0	* 0	WBOFF	* 0	* 0	WD[9–0]									

* = Reserved. Write to 0 for future compatibility

Extended QBus System (EQBS)

DBR[0–3]

Data Area Registers 0–3
Address: DBR0 = 0x00F0FFA0
DBR1 = 0x00F0FFA4
DBR2 = 0x00F0FFA8
DBR3 = 0x00F0FFAC
Reset: 0x00008000
Read/Write

Area Base[31–16, 15–8]– Area Base Address, Bits 0–15, 24–31
Contains the area base address for the data area.

IMM – Immediate, Bits 17–18	
00	Regular write through Write Buffer
01	Write immediate
10	Write immediate with no freeze
11	Reserved, do not use

GBL – Global, Bit 16	
0	Non-global
1	Global

EN – Enable Operation, Bit 21	
0	Disable operation
1	Enable operation

RV – Reverse, Bit 22	
0	Normal area definition
1	Reverse the area definition

SIZE – Size Indication, Bit 23	
0	Size is not 256 bytes
1	Size is 256 bytes

0–15	16	17–18	19	20	21	22	23	24–31
Area Base[0–15]	GBL	IMM	*	*	EN	RV	SIZE	Area Base[24–31]
			0	0				

* = Reserved. Write to 0 for future compatibility

FLBACR0

FlyBy Address Control Register
Address: 0x00F0FFF4
Reset: 0x00000000
Read/Write

FLBSA[20–0]– FlyBy Start Address, Bits 0–20
Contains bits 23–3 of the start address used during a flyby transfer.

0–20	21	22	23	24	25	26	27	28	29	30	31
FLBSA[0–20]	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Reset HRCW

Hard Reset Configuration Word

DSI Address: 0x1BE050
Reset: 0x00E40000
Read/Write

BPS – Boot Port Size, Bits 4–5	
00	64-bit port
01	8-bit port
10	16-bit port
11	32-bit port

EBM – External 60x-Compatible Bus Mode, Bit 3	
0	Single MSC8113 bus mode
1	60x-compatible bus mode

INTOUT – INT_OUT/IRQ7 Selection, Bit 2	
0	IRQ7/INT_OUT signal is IRQ7
1	IRQ7/INT_OUT signal is INT_OUT

EXMC – External Memory Controller, Bit 1	
0	No external memory controller
1	External memory controller

EARB – External Arbitration, Bit 0	
0	Internal arbitration
1	External arbitration

IRPC – Interrupt Pin Configuration, Bit 8	
0	BADDR[29–31]/IRQ[5, 3–2] is IRQ[5, 3–2]
1	BADDR[29–31]/IRQ[5, 3–2] is BADDR[29–31]

ISPS – Internal Space Port Size, Bit 7	
0	MSC8113 is 64-bit slave
1	MSC8113 is 32-bit slave

SCDIS – SC140 Disable, Bit 6	
0	SC140 cores enabled
1	SC140 cores disabled

BBD – Bus Busy Disable, Bit 17	
0	ABB/IRQ4 is ABB DBB/IRQ5 is DBB
1	ABB/IRQ4 is IRQ4 DBB/IRQ5 is IRQ5

DPPC – Data Parity Pin Configuration, Bits 10–11	
00	Pins are Reserved and IRQ[1–7]
01	Pins are DP[0–7]
10	Pins are DREQ[1–4], DACK[1–4]
11	Pins are EXT_BR[2–3], EXT_BG[2–3], EXT_DBG[2–3], and IRQ[6–7]

NMIOUT – NMI Out, Bit 12	
0	SC140 cores service NMI
1	External host services NMI

ISBSEL – Initial Internal Space Base Select, Bits 13–15

0–2	Internal Base Address
000	0xF0000000
001	0xF0F00000
010	0xFF000000
011	0xFFF00000
100	Reserved. Do not use.
101	Reserved. Do not use.
110	0x0F000000
111	0x0FF00000

MMR – Mask Masters Requests, Bit 18	
0	No masking on bus request lines
1	All external bus requests masked

TTPC – Transfer Type Pin Configuration, Bit 20	
0	TT0/Res. is TT0 and TT[2–4]/CS[5–7] is TT[2–4]
1	TT0/Res. is Res. and TT[2–4]/CS[5–7] is CS[5–7]

CS5PC – Chip Select 5 Pin Configuration, Bit 21	
0	CS5/BCTL1 is CS5
1	CS5/BCTL1 is BCTL1

TCPC – Transfer Code Pin Configuration, Bits 22–23	
00	TC[0–2]/BNKSEL[0–2] is TC[0–2]
01	Reserved
10	TC[0–2]/BNKSEL[0–2] is BNKSEL[0–2]
11	Reserved

LTLEND – Host Little-Endian Mode, Bit 24	
0	Big-endian mode
1	Little-endian mode

PPCLE – Munged Little-Endian Mode, Bit 25	
0	True little-endian mode
1	Munged little-endian mode

DLLDIS – DLL Disable, Bit 27	
0	DLL enabled, no bypass
1	DLL disabled, bypass

MODCK_H — High-Order Bits MODCK, Bits 28–30
See Clock Configuration

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EARB	EXMC	INT OUT	EBM	BPS		SCDIS	ISPS	IRPC	* 0	DPPC		NMI OUT	ISBSEL			* 0	BBD	MMR	* 0	TTPC	CS5 PC	TCPC		LTL END	MLE	* 0	DLL DIS	MODCK_H			* 0

* = Reserved. Write to 0 for future compatibility

Reset

RSR

Reset Status Register

QBus/System Bus Address: ___________ See **page 5-16**
DSI Address: 0x1D0C90
Reset: 0x00000003
Read/Write

SWRS – Software Watchdog Reset Status, Bit 28	
0	No software watchdog reset event occurred
1	A software watchdog reset occurred

BMRS – Bus Monitor Reset Status, Bit 29	
0	No bus monitor reset event occurred
1	Bus monitor reset occurred

JTRS – JTAG Reset Status, bit 26	
0	No host reset command occurred through JTAG
1	A host reset command was issued through JTAG

ESRS – External Soft Reset Status, Bit 30	
0	No external soft reset event occurred
1	External soft reset occurred

EHRS – External Hard Reset Status, Bit 31	
0	No external hard reset event occurred
1	External hard reset occurred

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	JTRS	* **0**	SWRS	BMRS	ESRS	EHRS

* = Reserved. Write to 0 for future compatibility

Note: The specified bits are set by events when they occur. The bits remain set until cleared by software. Writing a one to a bit clears the bits. A read has no effect.

System Interface Unit (SIU)

BCR

Bus Configuration Register

QBus/System Bus Address: ____________ See **page 4-10**
DSI Address: 0x1D00024
Reset: Depends on reset configuration sequence
Read/Write

EAV – Enable Address Visibility, Bit 11	
0	Bank select signals are driven on 60x-compatible system bus address lines. There is no full address visibility
1	Bank select signals are not driven on address bus. During READ and WRITE commands to SDRAM devices, the full address is driven on 60x-compatible system bus address lines

DSBI – Disable System Bus on Internal Access, Bit 9	
0	Internal system bus accesses can be snooped externally
1	Internal system bus accesses cannot be snooped

PLDP – Pipeline Maximum Depth, Bit 8	
0	Pipeline maximum depth is one
1	Pipeline maximum depth is zero

APD[0–2] – Address Phase Delay, Bits 1–3

Specifies the minimum number of address tenure wait states for address operations initiated by a 60x-compatible system bus master

EBM – External Bus Mode, Bit 0	
0	Single-master bus mode
1	Multi-master bus mode

ETM – Compatibility Mode Enable, Bit 12	
0	Strict 60x-compatible system bus mode. Extended transfer mode is disabled
1	Extended transfer mode is enabled

LETM – Local Bus Compatibility Mode Enable, Bit 13	
0	Extended transfer mode is disabled in the local bus
1	Extended transfer mode is enabled on the local bus

EPAR – Even Parity, Bit 14	
0	Odd parity
1	Even parity

NPQM[0–2] – Non-MSC8113 Master, Bits 16–18	
0	The bus master connected to the arbitration lines is an MSC8113
1	The bus master connected to the arbitration lines is not an MSC8113

EXDD – External Master Delay Disable, Bit 21	
0	The memory controller inserts one wait state between the assertion of $\overline{TS}$ and the assertion of $\overline{CS}$ when external master accesses an address space controlled by the memory controller
1	The memory controller asserts $\overline{CS}$ on the cycle following the assertion of $\overline{TS}$ by external master accessing an address space controlled by the memory controller

ISPS – Internal Space Port Size, Bit 27	
0	MSC8113 acts as a 64-bit slave to external master accesses to its internal space
1	MSC8113 acts as a 32-bit slave to external master accesses to its internal space

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EBM	APD0	APD1	APD2	* 0	* 0	* 0	* 0	PLDP	DSBI	* 0	EAV	ETM	LETM	EPAR	* 0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NPQM0	NPQM1	NPQM2	* 0	* 0	EXDD	* 0	* 0	* 0	* 0	* 0	ISPS	* 0	* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

PPC_ACR

System Bus Arbiter Configuration Register

QBus/System Bus Address: ___________ See **page 4-13**

DSI Address: 0x1D0028

Reset: 0b000x0010; the Boot program changes this to 0x000x0101.

Read/Write

EARB – External Arbitration, Bit 3	
0	Internal arbitration
1	External arbitration

Note: The reset value is determined by the EARB bit in the Hard Reset Configuration Word.

DBGD – Data Bus Grant Delay, Bit 2	
0	$\overline{\text{DBG}}$ asserted with $\overline{\text{TS}}$ if data bus is not busy
1	$\overline{\text{DBG}}$ asserted 1 cycle after $\overline{\text{TS}}$ if data bus is not busy

PRKM – Parking Master, Bits 4–7	
0000–0011	Reserved
0100	DSI
0101	SC140 core interface
0110	Reserved
0111	External master 1
1000	External master 2
1001	External master 3
1010	DMA priority 0
1011	DMA priority 1
1100	DMA priority 2
1101–1111	Reserved

0	1	2	3	4	5	6	7
*	*	DBGD	EARB	PRKM			
0	**0**						

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

Note: PPC_ALRH and PPC_ALRL assign arbitration priorities for sixteen potential system bus masters. Priority 0 is the highest and Priority 15 is the lowest. The system bus master index number defines each master uniquely. Assign the priority for a system bus master by entering its index number in the appropriate field in PPC_ALRH or PPC_ALRL. The reset value is the recommended configuration.

System Bus Master Indices (See PPC_ACR)	
0x0–0x3	Reserved
0x4	DSI
0x5	SC140 core interface
0x6	Reserved
0x7	External master 1
0x8	External master 2
0x9	External master 3
0xA	DMA priority 0
0xB	DMA priority 1
0xC	DMA priority 2
0xD–0xF	Reserved

PPC_ALRH

System Bus Arbitration-Level Register High

QBus/System Bus Address: ___________ See **page 8-55**
DSI Address: 0x1D002C
Reset: 0x01234567; the Boot program changes this to 0x0A547891
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Priority Field 0				Priority Field 1				Priority Field 2				Priority Field 3				Priority Field 4				Priority Field 5				Priority Field 6				Priority Field 7			

PPC_ALRL

System Bus Arbitration-Level Register Low

QBus/System Bus Address: ___________ See **page 8-55**
DSI Address: 0x1D0030
Reset: 0x89ABCDEF; the Boot program changes this to 0xB2C36DEF
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Priority Field 8				Priority Field 9				Priority Field 10				Priority Field 11				Priority Field 12				Priority Field 13				Priority Field 14				Priority Field 15			

System Interface Unit (SIU)

LCL_ACR

Local Bus Arbiter Configuration Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0034
Reset: 0x02; the Boot program changes this to 0x03
Read/Write

DBGD – Data Bus Grant Delay, Bit 2	
0	$\overline{\text{DBG}}$ asserted with $\overline{\text{TS}}$ if data bus is not busy
1	$\overline{\text{DBG}}$ asserted 1 cycle after $\overline{\text{TS}}$ if data bus is not busy

PRKM – Parking Master, Bits 4–7	
0000	TDM high priority
0001	Reserved
0010	TDM low priority
0011	Host bridge
0100	DSI
0101–1001	Reserved
1010	DMA priority 0
1011	DMA priority 1
1100	DMA priority 2
1101–1111	Reserved

0	1	2	3	4	5	6	7
*	*	DBGD	*	PRKM			
0	**0**		**0**				

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

Note: LCL_ALRH and LCL_ALRL assign arbitration priorities for sixteen potential local bus masters. Priority 0 is the highest and Priority 15 is the lowest. The local bus master index number defines each master uniquely. Assign the priority for a local bus master by entering its index number in the appropriate field in LCL_ALRH or LCL_ALRL. The reset value is the recommended configuration.

Local Bus Master Indices (See LCL_ACR)	
0000	Reserved
0001	Reserved
0010	TDM
0011	Host bridge
0100	DSI
0101–1001	Reserved
1010	DMA high priority
1011	DMA middle priority
1100	DMA low priority
1101	Ethernet high priority
1110	Ethernet middle priority
1111	Ethernet low priority

LCL_ALRH

Local Bus Arbitration-Level Register High

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0038
Reset: 0x01234567; the Boot program changes this to 0x041A53B2
Read/Write

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Priority Field 0</td><td colspan="4">Priority Field 1</td><td colspan="4">Priority Field 2</td><td colspan="4">Priority Field 3</td><td colspan="4">Priority Field 4</td><td colspan="4">Priority Field 5</td><td colspan="4">Priority Field 6</td><td colspan="4">Priority Field 7</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
</table>

LCL_ALRL

Local Bus Arbitration-Level Register Low

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D003C
Reset: 0x89ABCDEF; the Boot program changes this to 0x6C789DEF
Read/Write

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Priority Field 8</td><td colspan="4">Priority Field 9</td><td colspan="4">Priority Field 10</td><td colspan="4">Priority Field 11</td><td colspan="4">Priority Field 12</td><td colspan="4">Priority Field 13</td><td colspan="4">Priority Field 14</td><td colspan="4">Priority Field 15</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
</table>

System Interface Unit (SIU)

SIUMCR

SIU Module Configuration Register (page 1 of 2)

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0000

Reset: 0

Read/Write: Depends on reset configuration sequence

PBSE – Parity Byte Select Enable, Bit 2	
0	Parity byte select is disabled
1	Parity byte select is enabled

ESE – External Snoop Enable, Bit 1	
0	External snooping disabled (GBL/IRQ1 pin is IRQ1)
1	External snooping enabled (GBL/IRQ1 pin is GBL)

BBD – Bus Busy Disable, Bit 0	
0	ABB/IRQ2 pin is ABB, DBB/IRQ3 pin is DBB
1	ABB/IRQ2 pin is IRQ2, DBB/IRQ3 pin is IRQ3

INTOUT – IRQ7 or INT_OUT Selection, Bit 3	
0	IRQ7/INT_OUT pin is IRQ7
1	IRQ7/INT_OUT pin is INT_OUT

DPPC[0–1] – Data Parity Pin Configuration, Bits 4–5

Pin	00	01	10	11
RES/DP0/EXT_BR2	RES	DP0	RES	EXT_BR2
DP1/IRQ1/EXT_BG2	IRQ1	DP1	IRQ1	EXT_BG2
DP2/IRQ2/EXT_DBG2	IRQ2	DP2	RES	EXT_DBG2
DP3/IRQ3/EXT_BR3	IRQ3	DP3	RES	EXT_BR3
DP4/IRQ4/EXT_BG3/DREQ3	IRQ4	DP4	DREQ3	EXT_BG3
DP5/IRQ5/EXT_DBG3/DREQ4	IRQ5	DP5	DREQ4	EXT_DBG3
DP6/IRQ6/DACK3	IRQ6	DP6	DACK3	IRQ6
DP7/IRQ7/DACK4	IRQ7	DP7	DACK4	IRQ7

IRPC – Interrupt Pin Configuration, Bit 6	
0	BADDR29/IRQ5 pin is IRQ5 BADDR30/IRQ2 pin is IRQ2 BADDR31/IRQ3 pin is IRQ3
1	BADDR29/IRQ5 pin is BADDR29 BADDR30/IRQ2 pin is BADDR30 BADDR31/IRQ3 pin is BADDR31

See Sheet 2 – System Interface Unit – SIUMCR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BBD	ESE	PBSE	INTOUT	DPPC0	DPPC1	IRPC	BM0	BM1	BM2	TCPC0	TCPC1	BC1PC0	BC1PC1	BCTLC0	BCTLC1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MMR0	MMR1	* 0	CLKOD	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility. RES = Configuration is reserved.

System Interface Unit (SIU)

SIUMCR

SIU Module Configuration Register (page 2 of 2)

QBus/System Bus Address: ____________ See **page 8-56**
DSI Address: 0x1D0000
Reset: 0
Read/Write: Depends on reset configuration sequence

TCPC[0–1] – Transfer Code Pin Configuration, Bits 10–11				
Pin	**00**	**01**	**10**	**11**
MODCK1/BNKSEL0/TC0	TC0	Reserved	BNKSEL0	Reserved
MODCK2/BNKSEL1/TC1	TC1	Reserved	BNKSEL1	Reserved
MODCK3/BNKSEL2/TC2	TC2	Reserved	BNKSEL2	Reserved

BM[0–2] – Boot Mode, Bits 7–9	
000	MSC8113 boots from external memory
001	MSC8113 boots from external host (DSI or system bus)
010	MSC8113 boots from TDM
011	MSC8113 boots from UART
1xx	Reserved

See Sheet 1 – System Interface Unit – SIUMCR

CS5PC – Buffer Control 1 or $\overline{\text{CS5}}$ Pin Configuration, Bit 12	
0	$\overline{\text{CS5}}$/$\overline{\text{BCTL1}}$ is $\overline{\text{CS5}}$
1	$\overline{\text{CS5}}$/$\overline{\text{BCTL1}}$ is $\overline{\text{BCTL1}}$

TTPC– Transfer Type Pin Configuration, Bit 13		
Pin	**0**	**1**
TT0/Reserved	TT0	Reserved
TT2/$\overline{\text{CS5}}$	TT2	$\overline{\text{CS5}}$
TT3/$\overline{\text{CS6}}$	TT3	$\overline{\text{CS6}}$
TT4/$\overline{\text{CS7}}$	TT4	$\overline{\text{CS7}}$

BCTLC[0–1] – Buffer Control Configuration, Bits 14–15	
00	$\overline{\text{BCTL0}}$: W/$\overline{\text{R}}$ – $\overline{\text{BCTL1}}$: $\overline{\text{OE}}$
01	$\overline{\text{BCTL0}}$: $\overline{\text{W}}$/R – $\overline{\text{BCTL1}}$: $\overline{\text{OE}}$
10	$\overline{\text{BCTL0}}$: $\overline{\text{WE}}$ – $\overline{\text{BCTL1}}$: $\overline{\text{RE}}$
11	Reserved

MMR – Mask Masters Requests, Bit 16	
0	No masking on bus request lines
1	All external bus requests masked (boot master internal core)

INTODC – $\overline{\text{INT\_OUT}}$ Drive Control, Bit 17	
0	$\overline{\text{INT\_OUT}}$ is open-drain output
1	$\overline{\text{INT\_OUT}}$ driven using full drive

CLKOD – CLKOUT Disable, Bit 19	
0	CLKOUT driven with system clock
1	CLKOUT not driven

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BBD	ESE	PBSE	INTOUT	DPPC0	DPPC1	IRPC	BM0	BM1	BM2	TCPC0	TCPC1	CS5PC	TTPC	BCTLC0	BCTLC1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MMR	INTODC	*	CLKOD	*	*	*	*	*	*	*	*	*	*	*	*
		0		0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility. RES = Configuration is reserved.

System Interface Unit (SIU)

IMMR

Internal Memory Map Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D01A8

Reset: Depends on HRCW ISB field,
Bits 0–15 Read/Write
Bits 16–31 Read Only

ISB – Internal Space Base, Bits 0–14

Defines the base address of the internal memory space. The ISBSEL bits in the HRCW select one of the following initial addresses:

ISBSEL Value	Internal Base Address
000	0xF0000000
001	0xF0F00000
010	0xFF000000
011	0xFFF00000
100	Reserved
101	Reserved
110	0x0F000000
111	0x0FF00000

Note: The internal base address can be changed after reset.

PARTNUM – Part Number, Bits 16–23

A read-only field containing a code corresponding to the device part number.

MASKNUM – Mask Number, Bits 24–31

A read-only field containing a code corresponding to the device mask number.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ISB															*	PARTNUM								MASKNUM							
															0																

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

SYPCR

System Protection Control Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0004

Reset: 0b1111111111111111111111111100000x11

Read/Write

PBME – System Bus Monitor Enable, Bit 24	
0	System bus monitor is disabled
1	System bus monitor is enabled

BMT – Bus Monitor Timing, Bits 16–23
Defines the time-out period for the bus monitor. The granularity of this field is eight bus clocks. A value of 0 is invalid and generates an error.

SWTC – Software Watchdog Timer Count, Bits 0–15
Contains the count value for the software watchdog timer

LBME – Local Bus Monitor Enable, Bit 25	
0	Local bus monitor is disabled
1	Local bus monitor is enabled

SWE – Software Watchdog Enable, Bit 29	
0	Software watchdog timer is disabled
1	Software watchdog timer is enabled

Note: Value at reset determined by sampling SWTE.

SWRI – Software Watchdog Reset/Interrupt Select, Bit 30	
0	Software watchdog timer and bus monitor time-out cause a machine check interrupt to the SC140 core
1	Software watchdog timer and bus monitor time-out cause a hard reset

SWP – Software Watchdog Prescale, Bit 31	
0	Software watchdog timer is not prescaled
1	Software watchdog timer clock is prescaled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SWTC																BMT								PBME	LBME	*	*	*	SWE	SWRI	SWP
																										0	0	0			

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

SWSR

Software Service Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D000C

Reset: Unknown
Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SWSR															

Reading this register returns 0x0000. Writing can be done at any time. If the correct sequence is written to this register, it initializes the software watchdog timer. The user should write the value 0x556C to this register, followed by writing the value 0xAA39 to initialize the timer after reset. The register feeds a state machine that controls the watchdog timer. Once the state machine is reset, it enters State 0, waiting for 0x556C. Receipt of any other value causes the state machine to remain in State 0.
Writing 0x556C to the SWSR causes the state machine to go to State 1, where it waits for a write of 0xAA39 to the SWSR.
If any other value is written, the state machine returns to State 0 without reloading the watchdog timer. If the next write to the SWSR is 0xAA39, the state machine reloads the cnfigured initial value (from SYPCR[SWTC]) to the software watchdog timer and returns to State 0.

System Interface Unit (SIU)

TESCR1

System Bus Transfer Error Status and Control Register 1

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0040
Reset: 0x00000000
Read/Write

WP – Write Protect Error, Bit 5	
0	No write protect error occurred
1	Attempted write to a read-only area error occurred

ECC1 – Single ECC Error, Bit 4	
0	No single bit ECC error occurred
1	TEA asserted due to single-bit ECC error on system bus

ECC2 – Double ECC Error, Bit 3	
0	No double ECC error occurred
1	TEA asserted due to double ECC error on system bus

PAR – System Bus Parity Error, Bit 2	
0	No system bus parity error
1	NMI asserted due to system bus parity error

ISBE – Internal Space Bus Error, Bit 1	
0	No internal memory access error
1	TEA asserted due to internal memory access error

BM – System Bus Monitor Time-Out, Bit 0	
0	No system bus monitor time-out
1	TEA asserted due to system bus monitor time-out

EXT – External Error, Bit 6	
0	No external error occurred
1	TEA asserted by an external bus slave

TC – Transfer Code, Bits 7–9	
000	Reserved
001	DSI
010	Reserved
011	Reserved
100	Reserved
101	SC140 cores
110	DMA
111	DMA

TT – Transfer Type, Bits 11–15	
00000–00001	Reserved
00010	Single-beat or burst write
00011–01001	Reserved
01010	Single-beat or burst read
01011–11111	Reserved

DMD– Data Errors Disable, Bit 17	
0	Data errors on system bus enabled
1	Data errors on system bus disabled

ECNT – Single ECC Error Counter
Bits 24–31

Indicates the number of single ECC errors occurring in the system. Write a value to ECNT to select an error threshold (0x00–0xFF).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BM	ISBE	PAR	ECC2	ECC1	WP	EXT	TC			*	TT					*	DMD	*	*	*	*	*	*	ECNT							
										0						0		0	0	0	0	0	0								

* = Reserved. Write 0 for future compatibility

System Interface Unit (SIU)

TESCR2

System Bus Transfer Error Status and Control Register 2

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0044
Reset: 0x00000000
Read/Write

LCL – Local Bus Bridge Error, Bit 7	
0	No system-to-local bus error
1	System-to-local bus bridge error occurred

REGS – Internal Register Transaction Error, Bit 1	
0	No internal register transaction error
1	Internal register transaction error occurred

PB – Parity Error on Byte Bits 8–15	
Each bit is a status bit for a byte lane on the system bus	
0	No error occurred on the byte lane
1	Parity error detected on the byte lane

Note: Bit 8 = byte 0, bit 9 = byte 1, and so forth to bit 15 = byte 7.

BNK – Memory Controller Bank Bits 16–23	
Each bit is a status bit for a memory controller external bank	
0	No error occurred on memory controller bank
1	Parity or ECC error detected on memory controller bank

Note: Bit 16 = Bank 0, bit 17 = Bank 1, and so forth to bit 23 = Bank 7.
This bit is invalid if the error was not parity or ECC.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	REGS	*	*	*	*	*	LCL	PB							
0		0	0	0	0	0									

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BNK								*	*	*	*	*	*	*	*
								0	0	0	0	0	0	0	0

* = Reserved. Write 0 for future compatibility

System Interface Unit (SIU)

L_TESCR1

Local Bus Transfer Error Status and Control Register 1

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0048
Reset: 0x00000000
Read/Write

WP – Write Protect Error, Bit 5	
0	No write protect error occurred
1	Attempted write to a local bus read-only area error occurred

BM – Local Bus Monitor Time-Out, Bit 0	
0	No local bus monitor time-out
1	TEA asserted due to local bus monitor time-out

TC – Transfer Code, Bits 7–9	
000	System-to-local bus bridge
001	DSI
010	TDM
011	Reserved
100	Reserved
101	Reserved
110	DMA
111	DMA

TT – Transfer Type, Bits 11–15	
00000–00001	Reserved
00010	Single-beat or burst write
00011–01001	Reserved
01010	Single-beat or burst read
01011–11111	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BM	*	*	*	*	WP	*	TC			*	TT					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0		0				0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write 0 for future compatibility

System Interface Unit (SIU)

TMCNTSC

Timer Counter Status and Control Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0220
Reset: 0x0000
Read/Write

SIE – Once per Second Interrupt Enable, Bit 12	
0	The time counter does not generate an interrupt when SEC is set
1	The time counter generates an interrupt when SEC is set

ALR – Alarm Interrupt, Bit 9	
0	TMCNT value ≠ TMCNTAL value
1	TMCNT value = TMCNTAL value

Note: Write a 1 to this bit to clear it.

SEC – Once Per Second Interrupt, Bit 8
This status bit is set every second and should be cleared by software. Write a 1 to this bit to clear it.

ALE – Alarm Interrupt Enable, Bit 13	
0	No alarm interrupt
1	The time counter generates an interrupt when ALR is set

TCF – Time Counter Frequency, Bit 14	
0	The input clock to the time counter is 4 MHz
1	The input clock to the time counter is 32 kHz

TCE – Time Counter Enable, Bit 15	
0	The time counter is disabled
1	The time counter is enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	SEC	ALR	* **0**	* **0**	SIE	ALE	TCF	TCE

* = Reserved. Write to 0 for future compatibility

System Interface Unit (SIU)

TMCNT

Time Counter Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0224

Reset: 0x00000000

Read/Write

TMCNT – Timer Counter, Bits 0–31
Contains the current timer counter value

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MD																															

TMCNTAL

Time Counter Alarm Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D022C

Reset: 0x00000000

Read/Write

ALARM – Alarm Value, Bits 0–31
An alarm interrupt is generated when ALARM = TMCNT. The ALARM resolution is 1 second.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ALARM																															

System Interface Unit (SIU)

PISCR

Periodic Interrupt Status and Control Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0240
Reset: 0x0000
Read/Write

PIE – Periodic Interrupt Enable, Bit 13	
0	Periodic timer interrupt is disabled
1	Periodic timer interrupt is enabled

PTF – Periodic Interrupt Frequency, Bit 14	
0	PIT input is 4 MHz
1	PIT input is 32 KHz

PTE – Periodic Timer Enable, Bit 15	
0	PIT is disabled
1	PIT is enabled

PS – Periodic Interrupt Status, Bit 8	
0	No periodic interrupt is pending
1	Periodic interrupt was generated by the PIT

Note: Writing a 1 to this bit clears it. The PIT interrupt handler should write a 1 to this bit when the interrupt is processed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	PS	*	*	*	*	PIE	PTF	PTE
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**		**0**	**0**	**0**	**0**			

* = Reserved. Write to 0 for future compatibility

PITC

Periodic Interrupt Timer Count Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0244
Reset: 0x00000000
Read/Write

PITC – Periodic Interrupt Timing Count, Bits 0–15
Contains the count for the periodic timer. Setting PITC to 0xFFFF selects the maximum count period.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PITC																*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

* = Reserved. Write to 0 for future compatibility

Memory Controller

BR[0–7, 9, 11]

Base Register: BR______________ (enter number = *n*)

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0100 + (8 × *n*) = ______________

Reset: 0–31 depends on reset configuration sequence,
After a system reset, the V bit is set in BR0 and clear in BR[1–7, 9, 11]
but the Boot program sets the V bit in BR9 and BR11.
For BR9, the boot program also writes 0b11 to PS and 0b001 to MS.
For BR11, the boot program also writes 0b110 to MS.
Read/Write

DECC[0–1] – Data Error Correction and Checking, Bits 21–22	
00	Data errors checking disabled
01	Normal parity checking
10	Read-modify-write parity checking
11	ECC correction and checking

PS[0–1] – Port Size, Bits 19–20			
01	8-bit	11	32-bit
10	16-bit	00	64-bit

BA – Base Address, Bits 0–16

Upper 17 bits of each base address register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. BRx[BA] is used with ORx[AM]

WP – Write Protect, Bit 23	
0	Read and write accesses are allowed
1	Only read access is allowed

MSEL[0–2] – Machine Select, Bits 24–26			
000	GPCM system bus	100	UPMA
001	GPCM local bus	101	UPMB
010	SDRAM system bus	110	UPMC
011	Reserved	111	Reserved

EMEMC – External MEMC Enable, Bit 27	
0	Accesses are handled by the memory controller
1	Accesses are handled by an external memory controller

ATOM[0–1] – Atomic Operation, Bits 28–29	
00	Address space controlled by the memory controller bank is not used for atomic operations
01	Read-after-write-atomic (RAWA)
10	Write-after-read-atomic (WARA)
11	Reserved

DR – Data Pipelining, Bit 30	
0	No data pipelining is done
1	Data beats of accesses to the address space controlled by the memory controller bank are delayed by 1 cycle

V – Valid Bit, Bit 31			
0	Bank is invalid	1	Bank is valid

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BA																	*	*	PS		DECC		WP	MS			EMEMC	ATOM		DR	V
																	0	0													

* = Reserved. Write to 0 for future compatibility

Memory Controller

OR[0–7, 9, 11]

GPCM Mode

Option Register: OR_____________ **(enter number = *n*)**

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0104 + (8 × *n*) = _____________

Reset: 0–15 1111_1110_0000_0000,
16–31 0000_1110_1111_0100
For OR9, the boot program writes 0xFFFC0008.

Read/Write

CSNT – Chip-Select Negation Time, Bit 20	
0	$\overline{\text{CS}}/\overline{\text{PWE}}$ is deasserted normally
1	$\overline{\text{CS}}/\overline{\text{PWE}}$ is deasserted a quarter-clock early

BCTLD – Data Buffer Control Disable, Bit 19	
0	$\overline{\text{BCTLx}}$ is asserted upon access to the memory bank
1	$\overline{\text{BCTLx}}$ is not asserted upon access to the memory bank

AM – Address Mask, Bits 0–16	
0	Corresponding address bits are masked
1	Corresponding address bits are used in comparison with address pins. Address mask bits can be set or cleared in any order in the field

ACS – Address to Chip-Select Setup, Bits 21–22	
00	$\overline{\text{CS}}$ is output at the same time as the address lines
01	Reserved
10	$\overline{\text{CS}}$ is output a quarter of a clock after the address lines
11	$\overline{\text{CS}}$ is output half a clock after the address lines

SCY – Cycle Length in Clocks, Bits 24–27	
0000	0 clock cycle wait states
	• • • • • •
1111	15 clock cycles wait states

SETA – External Access Termination, Bit 28	
0	$\overline{\text{PSDVAL}}$ generated internally unless $\overline{\text{GTA}}$ is asserted externally
1	$\overline{\text{PSDVAL}}$ is generated after external logic asserts $\overline{\text{GTA}}$

TRLX – Timing Relaxed, Bit 29	
0	Normal timing generated by the GPCM
1	Relaxed timing generated by the GPCM

EHTR – Extended Hold Time on Read Accesses, Bit 30

TRLX	EHTR	Description
0	0	No additional cycles are inserted
0	1	One idle clock cycle is inserted
1	0	Four idle clock cycles are inserted (default)
1	1	Eight idle clock cycles are inserted

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AM																	*	*	BCTLD	CSNT	ACS		*	SCY				SETA	TRLX	EHTR	*
																	0	0					0								0

* = Reserved. Write to 0 for future compatibility

Memory Controller

OR[0–7, 9, 11]

SDRAM Mode

Option Register: OR______________ **(enter number = *n*)**

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0104 + (8 × *n*) = ______________

Reset: 0
Read/Write

BPD – Banks Per Device, Bits 17–18	
00	2 internal banks per device
01	4 internal banks per device
10	8 internal banks per device (not valid for 128-Mb SDRAMs)
11	Reserved

LSDAM – Lower SDRAM Address Mask, Bits 12–16
Reset LSDAM to 0x0 to implement a minimum size of 1 Mb

SDAM – SDRAM Address Mask, Bits 0–11
Clearing bits masks the corresponding address bit. Setting bits causes the corresponding address bit to be compared with the address pins

ROWST[0–3] – Row Start Address Bit, Bits 19–22			
PSDMR[PBI] = 0			
0010	A7	1010	A11
0100	A8	1100	A12
0110	A9	1110	A13
1000	A10	Other values are reserved	
PSDMR[PBI] = 1			
0000	A0	0001	A1
• • • • • •			
1100	A12	1101–1111	Reserved

NUMR – Number of Row Address Lines, Bits 23–25			
000	9 row address lines	100	13 row address lines
001	10 row address lines	101	14 row address lines
010	11 row address lines	110	15 row address lines
011	12 row address lines	111	16 row address lines

PMSEL – Page Mode Select, Bit 26	
0	Page is closed when the bus becomes idle
1	Page is kept open until a page miss or refresh occurs

IBID – Internal Bank Interleaving Within Same Device Disable, Bit 27	
0	Enables bank interleaving
1	Disables bank interleaving

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SDAM												LSDAM					BPD		ROWST				NUMR			PMSEL	IBID				
																												* 0	* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility

Memory Controller

OR[0–7, 9, 11]

UPM Mode

Option Register: OR______________ **(enter number = *n*)**

QBus/System Bus Address: ____________ See **page 8-56**

DSI Address: 0x1D0104 + (8 × *n*) = ______________

Reset: 0

For OR11, the Boot program writes 0xFFE00000.

Read/Write

BCTLD – Data Buffer Control Disable, Bit 19	
0	$\overline{\text{BCTLx}}$ is asserted upon access to the memory bank
1	$\overline{\text{BCTLx}}$ is not asserted upon access to the memory bank

AM – Address Mask, Bits 0–16	
0	Corresponding address bits are masked
1	Corresponding address bits are used in comparison with address pins. Address mask bits can be set or cleared in any order in the field

BI – Burst Inhibit, Bit 23	
0	Bank supports burst accesses
1	Bank does not support burst accesses. UPMx executes burst accesses as series of single accesses

EHTR – Extended Hold Time on Read Accesses, Bits 29–30	
00	No additional cycles are inserted, normal timing generated
01	One idle clock cycle is inserted
10	Four idle clock cycles are inserted
11	Eight idle clock cycles are inserted

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AM																	*	*	BCTLD	*	*	*	BI	*	*	*	*	*	EHTR		*
																	0	0		0	0	0		0	0	0	0	0			0

* = Reserved. Write to 0 for future compatibility

Memory Controller
MAMR, MBMR, MCMR

Machine A/B/C Mode Registers

MAMR: QBus/System Bus Address: ____________See **page 8-56**
DSI Address: 0x1D0170
MBMR: QBus/System Bus Address: ____________See **page 8-56**
DSI Address: 0x1D0174
MCMR: QBus/System Bus Address: ____________See **page 8-56**
DSI Address: 0x1D0178
Reset: 0x00000000 (Boot rewrites MCMR as 0x30011240)
Read/Write

AMx – Address Multiplex Size, Bits 5–7

AMx	External PPC Bus Address Pin	Signal Driven on External Pin	AMx	External PPC Bus Address Pin	Signal Driven on External Pin
000	A[16–31]	A[8–23]	011	A[16–31]	A[5–20]
001	A[16–31]	A[7–22]	100	A[17–31]	A[5–19]
010	A[16–31]	A[6–21]	101	A[18–31]	A[5–18]

OP – Command Opcode, Bits 2–3

00	Normal Operation	10	Read from array
01	Write to array	11	Run pattern

RFEN – Refresh Enable, Bit 1

0	Refresh services are not required
1	Refresh services are required

BSEL – Bus Select, Bit 0

0	Banks that select UPMx are assigned to the system bus
1	Banks that select UPMx are assigned to the local bus

DSx[0–1] – Disable Timer Period, Bits 8–9

00	1 cycle disable period	10	3 cycle disable period
01	2 cycle disable period	11	4 cycle disable period

G0CLx[0–2] – General Line 0 Control, Bits 10–12

000	A12	010	A10	100	A8	110	A6
001	A11	011	A9	101	A7	111	A5

GPL_x4DIS – GPL_A4 Output Line Disable, Bit 13

0	$\overline{\text{PGTA}}$/PUPMWAIT/PGPL4/$\overline{\text{PPBS}}$ behaves as PGPL4 UPMx[G4T4/DLT3] is interpreted as G4T4 UPMx[G4T3/WAEN] is interpreted as G4T3
1	$\overline{\text{PGTA}}$/PUPMWAIT/PGPL4/$\overline{\text{PPBS}}$ behaves asPUPMWAIT UPMx[G4T4/DLT3] is interpreted as DLT3 UPMx[G4T3/WAEN] is interpreted as WAEN

RLFx[0–3] – Read Loop Field, Bits 14–17

0001	Loop executes 1 time	0010	Loop executes 2 times
	• • • •		
1111	Loop executes 15 times	0000	Loop executes 16 times

WLFx[0–3] – Write Loop Field, Bits 18–21

0001	Loop executes 1 time	0010	Loop executes 2 times
	• • • •		
1111	Loop executes 15 times	0000	Loop executes 16 times

TLFx[0–3] – Refresh Loop Field, Bits 22–25

0001	Loop executes 1 time	0010	Loop executes 2 times
	• • • •		
1111	Loop executes 15 times	0000	Loop executes 16 times

MAD[0–5] – Machine Address, Bits 26–31

RAM address pointer for the command executed

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BSEL	RFEN	OP		*	AMx			DSx		G0CLx			GPL_ x4DIS	RLFx				WLFx				TLFx				MAD					
				0																											

* = Reserved. Write to 0 for future compatibility

Memory Controller

PSDMR (page 1 of 2)

System Bus SDRAM Mode Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0190

Reset: 0
Read/Write

OP – SDRAM Operation, Bits 2–4

000	Normal operation	100	Precharge bank (debug)
001	CBR refresh (SDRAM)	101	Precharge all banks (SDRAM)
010	Self refresh (debug)	110	Activate bank (debug)
011	Mode register write (SDRAM)	111	Read/write (debug)

RFEN – Refresh Enable, Bit 1

0	Refresh services are not required
1	Refresh services are required

PBI – Page-Based Interleaving, Bit 0

0	Bank-based interleaving
1	Page-based interleaving (normal operation)

SDAM – Address Multiplex Size, Bits 5–7

SDAM	External System Bus Address Pin	Signal Driven on External Pin	SDAM	External System Bus Address Pin	Signal Driven on External Pin
000	A[13–31]	A[5–23]	011	A[16–31]	A[5–20]
001	A[14–31]	A[5–22]	100	A[17–31]	A[5–19]
010	A[15–31]	A[5–21]	101	A[18–31]	A[5–18]

BSMA – Bank Select Multiplexed Address Line, Bits 8–10

000	A1[2–14]	100	A[16–18]
001	A[13–15]	101	A[17–19]
010	A[14–16]	110	A[18–20]
011	A[15–17]	111	A[19–21]

SDA10 – A10 Control, Bits 11–13

PBI = 0							
000	A12	010	A10	100	A8	110	A6
001	A11	011	A9	101	A7	111	A5
PBI = 1							
000	A10	010	A8	100	A6	110	A4
001	A9	011	A7	101	A5	111	A3

See Sheet 2 – Memory Controller – PSDMR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PBI	RFEN	OP			SDAM			BSMA			SDA10			RFRC			PRETOACT			ACTTORW			BL	LDOTOPRE		WRC		EAMUX	BUFCMD	CL	

Memory Controller

PSDMR (page 2 of 2)

60x Bus SDRAM Mode Register

QBus/System Bus Address: __________ See **page 8-56**

DSI Address: 0x1D0190

Reset: 0
Read/Write

PRETOACT – Precharge to Activate Interval, Bits 17–19			
001	1 clock-cycle wait states	010	2 clock-cycle wait states
• • • •			
111	7 clock-cycle wait states	000	8 clock-cycle wait states

RFRC– Refresh Recovery, Bits 14–16			
000	Reserved	100	6 clock cycles
001	3 clock cycles	101	7 clock cycles
010	4 clock cycles	110	8 clock cycles
011	5 clock cycles	111	16 clock cycles

See Sheet 1 – Memory Controller – PSDMR

ACTTORW – Activate to Read/Write Interval, Bits 20–22			
001	1 clock cycle	010	2 clock cycles
• • • •			
111	7 clock cycles	000	8 clock cycles

BL – Burst Length, Bit 23			
0	SDRAM burst length is 4	1	SDRAM burst length is 8

LDOTOPRE– Last Data Out to Precharge, Bits 24–25							
00	0 cycles	01	-1 cycle	10	-2 cycles	11	Reserved

WRC– Write Recovery Time, Bits 26–27							
01	1 cycle	10	2 cycles	11	3 cycles	00	4 cycles

EAMUX – External Address Multiplexing Enable/Disable, Bit 28	
0	No external address multiplexing. Fastest timing
1	Memory controller asserts SDAMUX for an extra cycle before issuing an ACTIVATE command to the SDRAM

BUFCMD – Command Buffer, Bit 29	
0	Normal timing for control lines
1	All control lines except $\overline{CS}$ are asserted for 2 cycles

CL – CAS Latency, Bits 30–31							
00	Reserved	01	1	10	2	11	3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PBI	RFEN	OP			SDAM			BSMA			SDA10			RFRC			PRETOACT			ACTTORW			BL	LDOTOPRE		WRC		EAMUX	BUFCMD	CL	

Memory Controller

MDR

Memory Data Register

QBus/System Bus Address: ____________ See **page 8-56**

DSI Address: 0x1D0188

Reset: 0x00000000

Read/Write

MD – Memory Data, Bits 0–31

Data to be read from or written to in the RAM array when a WRITE or READ command is supplied to the UPM.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MD																															

MAR

Memory Address Register

QBus/System Bus Address: ____________ See **page 8-56**

DSI Address: 0x1D0168

Reset: 0x00000000

Read/Write

A – Memory Address, Bits 0–31

Address generated under the control of the AMx bits in the UPM.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
A																															

Memory Controller

PURT

System Bus Assigned UPM Refresh Timer

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0198
Reset: 0x00
Read/Write

PURT – UPM Refresh Timer, Bits 0–7

Used to select the UPM refresh rate:
PURT < ((Bus frequency × required refresh rate)/(MPTPR[PTP] + 1)) – 1

0	1	2	3	4	5	6	7
PURT							

PSRT

System Bus Assigned SDRAM Refresh Timer

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D019C
Reset: 0x00
Read/Write

PSRT – SDRAM Refresh Timer, Bits 0–7

Used to select the UPM refresh rate:
PSRT < ((Bus frequency × required refresh rate)/(MPTPR[PTP] + 1)) – 1

0	1	2	3	4	5	6	7
PSRT							

MPTPR

Memory Refresh Timer Prescaler Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0184
Reset: 0b0000001x00000000
Read/Write

PTP – Timer Prescaler, Bits 0–7

Used to divide the bus clock to determine the refresh period base value (prescaler clock frequency = bus frequency / (PTP + 1))

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PTP								*	*	*	*	*	*	*	*
								0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

* = Reserved. Write to 0 for future compatibility

Interrupt Scheme

VIGR

Virtual Interrupt Generation Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC000

DSI Address: 0x1BC000

Write Only

VIRQNUM – Virtual Source Interrupt Number, Bits 29–31

000	VS Interrupt 0
001	VS Interrupt 1
010	VS Interrupt 2
011	VS Interrupt 3
100	VS Interrupt 4
101	VS Interrupt 5
110	VS Interrupt 6
111	VS Interrupt 7

CORENUM – Core Number, Bits 22–23

00	Interrupt to core 0
01	Interrupt to core1
10	Interrupt to core 2
11	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CORENUM		*	*	*	*	*	VIRQNUM		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0			

* = Reserved. Write to 0 for future compatibility

VISR

Virtual Interrupt Status Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC008

DSI Address: 0x1BC008

Reset: 0x00000000

Read/Write

VS[31–24] – Virtual Source Interrupt Status, Bits 0–7
(Reserved)

VS31	Reserved
VS30	Reserved
VS29	Reserved
VS28	Reserved
VS27	Reserved
VS26	Reserved
VS25	Reserved
VS24	Reserved

VS[23–16] – Virtual Source Interrupt Status, Bits 8–15
Used for Core 2

VS23	VS Interrupt 7
VS22	VS Interrupt 6
VS21	VS Interrupt 5
VS20	VS Interrupt 4
VS19	VS Interrupt 3
VS18	VS Interrupt 2
VS17	VS Interrupt 1
VS16	VS Interrupt 0

VS[15–8] – Virtual Source Interrupt Status, Bits 16–23
Used for Core 1

VS15	VS Interrupt 7
VS14	VS Interrupt 6
VS13	VS Interrupt 5
VS12	VS Interrupt 4
VS11	VS Interrupt 3
VS10	VS Interrupt 2
VS9	VS Interrupt 1
VS8	VS Interrupt 0

VS[7–0] – Virtual Source Interrupt Status, Bits 24–31
Used for Core 0

VS7	VS Interrupt 7
VS6	VS Interrupt 6
VS5	VS Interrupt 5
VS4	VS Interrupt 4
VS3	VS Interrupt 3
VS2	VS Interrupt 2
VS1	VS Interrupt 1
VS0	VS Interrupt 0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
VS31	VS30	VS29	VS28	VS27	VS26	VS25	VS24	VS23	VS22	VS21	VS20	VS19	VS18	VS17	VS16	VS15	VS14	VS13	VS12	VS11	VS10	VS9	VS8	VS7	VS6	VS5	VS4	VS3	VS2	VS1	VS0

Note: The VISR bits are set by the VIGR selections. Writing a one to a VISR bit clears the bit; writing a zero has no effect.

Interrupt Scheme

VNMIGR

Virtual NMI Generation Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC010

DSI Address: 0x1BC010

Write Only

CORENUM – Core Number, Bits 22–23	
00	NMI to core 0
01	NMI to core1
10	NMI to core 2
11	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CORENUM		*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

GICR

GIC Interrupt Configuration Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC018

DSI Address: 0x1BC018

Reset: 0x00000000 (Boot writes to 0x0F000000)

Read/Write

EL[4–7. 12–30]– Edge/Level, Bits 4–7, 12–30	
0	Level-mode interrupt
1	Edge-mode interrupt

EL[4–7] – Edge/Level, Bits 0–7	
EL4	VS Interrupts 24
EL5	VS Interrupts 16
EL6	VS Interrupts 8
EL7	VS Interrupts 0

EL[12–15] – Edge/Level, Bits 12–15	
EL12	UART Interrupt
EL13	TMCNT Interrupt
EL14	PIT Interrupt
EL15	DMA Interrupt

EL[16–23] – Edge/Level, Bits 16–23	
EL16	$\overline{\text{IRQ15}}$
EL17	$\overline{\text{IRQ14}}$
EL18	$\overline{\text{IRQ13}}$
EL19	$\overline{\text{IRQ12}}$
EL20	$\overline{\text{IRQ11}}$
EL21	$\overline{\text{IRQ10}}$
EL22	$\overline{\text{IRQ9}}$
EL23	$\overline{\text{IRQ8}}$

EL[24–30] – Edge/Level, Bits 24–30	
EL24	$\overline{\text{IRQ7}}$
EL25	$\overline{\text{IRQ6}}$
EL26	$\overline{\text{IRQ5}}$
EL27	$\overline{\text{IRQ4}}$
EL28	$\overline{\text{IRQ3}}$
EL29	$\overline{\text{IRQ2}}$
EL30	$\overline{\text{IRQ1}}$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	EL4	EL5	EL6	EL7	*	*	*	*	EL12	EL13	EL14	EL15	EL16	EL17	EL18	EL19	EL20	EL21	EL22	EL23	EL24	EL25	EL26	EL27	EL28	EL29	EL30	*
0	0	0	0					0	0	0	0																				0

Note: The EL bits configure the trigger mode for the specified interrupts.

* = Reserved. Write to 0 for future compatibility

Interrupt Scheme

GEIER

GIC External Interrupt Enable Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC020
DSI Address: 0x1BC020

Reset: 0x00000000
Read/Write

EN[4–7. 12–30]– Enable, Bits 4–7, 12–30	
0	Disables interrupt external output
1	Enables interrupt and routes output to INT_OUT

EN[4–7] – Enable, Bits 0–7	
EN4	VS Interrupts 24
EN5	VS Interrupts 16
EN6	VS Interrupts 8
EN7	VS Interrupts 0

EN[12–15] – Enable, Bits 12–15	
EN12	UART Interrupt
EN13	TMCNT Interrupt
EN14	PIT Interrupt
EN15	DMA Interrupt

EN[16–23] – Enable, Bits 16–23	
EN16	$\overline{\text{IRQ15}}$
EN17	$\overline{\text{IRQ14}}$
EN18	$\overline{\text{IRQ13}}$
EN19	$\overline{\text{IRQ12}}$
EN20	$\overline{\text{IRQ11}}$
EN21	$\overline{\text{IRQ10}}$
EN22	$\overline{\text{IRQ9}}$
EN23	$\overline{\text{IRQ8}}$

EN[24–30] – Enable, Bits 24–30	
EN24	$\overline{\text{IRQ7}}$
EN25	$\overline{\text{IRQ6}}$
EN26	$\overline{\text{IRQ5}}$
EN27	$\overline{\text{IRQ4}}$
EN28	$\overline{\text{IRQ3}}$
EN29	$\overline{\text{IRQ2}}$
EN30	$\overline{\text{IRQ1}}$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	* 0	* 0	* 0	EN4	EN5	EN6	EN7	* 0	* 0	* 0	* 0	EN12	EN13	EN14	EN15

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EN16	EN17	EN18	EN19	EN20	EN21	EN22	EN23	EN24	EN25	EN26	EN27	EN28	EN29	EN30	* 0

Note: The EN bits configure the specified interrupts as external.

* = Reserved. Write to 0 for future compatibility

Interrupt Scheme

GCIER

GIC Core Interrupt Enable Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC028

DSI Address: 0x1BC028

Reset: 0x00000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

IRQ[15–8] – IRQ[15-8] Core Enable, Bits 16–23	
0	Disable interrupt for core
1	Enable interrupt and route to PIC IRQ16

* = Reserved. Write to 0 for future compatibility

GISR

GIC Interrupt Status Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBC030

DSI Address: 0x1BC030

Reset: 0x00000000

Read/Write

GIS[4–7, 12–30]– GIC Interrupt Status, Bits 4–7, 12–30	
0	No interrupt
1	Interrupt generated

GIS[4–7] – Edge/Level, Bits 0–7	
GIS4	VS Interrupt 24
GIS5	VS Interrupt 16
GIS6	VS Interrupt 8
GIS7	VS Interrupt 0

GIS[12–15] – Edge/Level, Bits 12–15	
GIS12	UART Interrupt
GIS13	TMCNT Interrupt
GIS14	PIT Interrupt
GIS15	DMA Interrupt

GIS[16–23] – Edge/Level, Bits 16–23	
GIS16	$\overline{\text{IRQ15}}$
GIS17	$\overline{\text{IRQ14}}$
GIS18	$\overline{\text{IRQ13}}$
GIS19	$\overline{\text{IRQ12}}$
GIS20	$\overline{\text{IRQ11}}$
GIS21	$\overline{\text{IRQ10}}$
GIS22	$\overline{\text{IRQ9}}$
GIS23	$\overline{\text{IRQ8}}$

GIS[24–30] – Edge/Level, Bits 24–30	
GIS24	$\overline{\text{IRQ7}}$
GIS25	$\overline{\text{IRQ6}}$
GIS26	$\overline{\text{IRQ5}}$
GIS27	$\overline{\text{IRQ4}}$
GIS28	$\overline{\text{IRQ3}}$
GIS29	$\overline{\text{IRQ2}}$
GIS30	$\overline{\text{IRQ1}}$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	GIS4	GIS5	GIS6	GIS7	*	*	*	*	GIS12	GIS13	GIS14	GIS15	GIS16	GIS17	GIS18	GIS19	GIS20	GIS21	GIS22	GIS23	GIS24	GIS25	GIS26	GIS27	GIS28	GIS29	GIS30	*
0	0	0	0					0	0	0	0																				0

Note: The GIS bits reflect the current interrupt status for the specified interrupts. Write a one to the bit to clear it. Writing a zero has no effect.

* = Reserved. Write to 0 for future compatibility

Interrupt Scheme

LICAICR[0–3]

Local Interrupt Controller Group A Registers 0–3

Address: LICAICR0 = 0x00F0AC00
LICAICR1 = 0x00F0AC08
LICAICR2 = 0x00F0AC10
LICAICR3 = 0x00F0AC18

Reset: LICAICR0 = 0x00000000
LICAICR1 = 0x00000000 (boot rewrites as 0x44044044)
LICAICR2 = 0x00000000 (boot rewrites as 0x04404404)
LICAICR3 = 0x00000000 (boot rewrites as 0x40440440)

Read/Write

EM[31–0] – Edge-Mode Selection for LIC Group A IRQs, Bits 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	
00	Level mode
01	Single-edge mode
10	Second-edge-detection mode
11	Reserved

IMAP[31–0] – MAP Selection for LIC Group A IRQs, Bits 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	
00	Route enabled interrupt through IRQOUTA0 into the PIC
01	Route enabled interrupt through IRQOUTA1 into the PIC
10	Route enabled interrupt through IRQOUTA2 into the PIC
11	Route enabled interrupt through IRQOUTA3 into the PIC

LICAICR0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EM31		IMAP31		EM30		IMAP30		EM29		IMAP29		EM28		IMAP28		EM27		IMAP27		EM26		IMAP26		EM25		IMAP25		EM24		IMAP24	

LICAICR1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EM23		IMAP23		EM22		IMAP22		EM21		IMAP21		EM20		IMAP20		EM19		IMAP19		EM18		IMAP18		EM17		IMAP17		EM16		IMAP16	

LICAICR2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EM15		IMAP15		EM14		IMAP14		EM13		IMAP13		EM12		IMAP12		EM11		IMAP11		EM10		IMAP10		EM9		IMAP9		EM8		IMAP8	

LICAICR3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EM7		IMAP7		EM6		IMAP6		EM5		IMAP5		EM4		IMAP4		EM3		IMAP3		EM2		IMAP2		EM1		IMAP1		EM0		IMAP0	

Interrupt Scheme

LICBICR[0–3]

Local Interrupt Controller Group A Registers 0–3

Address: LICBICR0 = 0x00F0AC40
LICBICR1 = 0x00F0AC48
LICBICR2 = 0x00F0AC50
LICBICR3 = 0x00F0AC58

Reset: 0 (Boot rewrites LICBICR0 as 0x00004444; LICBICR1 as 0x44440000; and LICBICR2 as 0x44444444)
Read/Write

EM[31–0] – Edge-Mode Selection for LIC Group B IRQs, Bits 0–1, 4–5, 8–9, 12–13, 16–17, 20–21, 24–25, 28–29	
00	Level mode
01	Single-edge mode
10	Second-edge-detection mode
11	Reserved

IMAP[31–0] – MAP Selection for LIC Group B IRQs, Bits 2–3, 6–7, 10–11, 14–15, 18–19, 22–23, 26–27, 30–31	
00	Route enabled interrupt through IRQOUTB0 into the PIC
01	Route enabled interrupt through IRQOUTB1 into the PIC
10	Route enabled interrupt through IRQOUTB2 into the PIC
11	Route enabled interrupt through IRQOUTB3 into the PIC

LICBICR0

0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15	16–17	18–19	20–21	22–23	24–25	26–27	28–29	30–31
EM31	IMAP31	EM30	IMAP30	EM29	IMAP29	EM28	IMAP28	EM27	IMAP27	EM26	IMAP26	EM25	IMAP25	EM24	IMAP24

LICBICR1

0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15	16–17	18–19	20–21	22–23	24–25	26–27	28–29	30–31
EM23	IMAP23	EM22	IMAP22	EM21	IMAP21	EM20	IMAP20	EM19	IMAP19	EM18	IMAP18	EM17	IMAP17	EM16	IMAP16

LICBICR2

0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15	16–17	18–19	20–21	22–23	24–25	26–27	28–29	30–31
EM15	IMAP15	EM14	IMAP14	EM13	IMAP13	EM12	IMAP12	EM11	IMAP11	EM10	IMAP10	EM9	IMAP9	EM8	IMAP8

LICBICR3

0–1	2–3	4–5	6–7	8–9	10–11	12–13	14–15	16–17	18–19	20–21	22–23	24–25	26–27	28–29	30–31
EM7	IMAP7	EM6	IMAP6	EM5	IMAP5	EM4	IMAP4	EM3	IMAP3	EM2	IMAP2	EM1	IMAP1	EM0	IMAP0

Interrupt Scheme

LICAIER/LICBIER

LIC Group A/B Interrupt Enable Register

Address: LICAIER = 0x00F0AC20
LICBIER = 0x00F0AC60
Reset: 0x00000000
Read/Write

E[31–0] – Enable, Bits 0–31	
0	Disable interrupt for PIC
1	Enable interrupt and route by IMAP to PIC interrupt input

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16	E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0

LICAISR/LICBISR

LIC Group A/B Interrupt Status Register

Address: LICAISR = 0x00F0AC28
LICBISR = 0x00F0AC68
Reset: 0x00000000
Read/Write

S[31–0] – Status, Bits 0–31	
0	No interrupt for specified source
1	Interrupt occurred for specified source

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Note: The S bits reflect the current interrupt status for the specified interrupts. Write a one to the bit to clear it. Writing a zero has no effect.

LICAIESR/LICBIESR

LIC Group A/B Interrupt Error Status Register

Address: LICAIESR = 0x00F0AC30
LICBIESR = 0x00F0AC70
Reset: 0x00000000
Read/Write

ES[31–0] – Error Status, Bits 0–31	
0	No second-edge interrupt for specified source
1	Second-edge detected while primary status bit set

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ES31	ES30	ES29	ES28	ES27	ES26	ES25	ES24	ES23	ES22	ES21	ES20	ES19	ES18	ES17	ES16	ES15	ES14	ES13	ES12	ES11	ES10	ES9	ES8	ES7	ES6	ES5	ES4	ES3	ES2	ES1	ES0

Note: The ES bits reflect the current interrupt error status for the specified interrupts. Write a one to the bit to clear it. Writing a zero has no effect.

Interrupt Scheme

ELIRA

PIC Edge/Level-Triggered Interrupt Priority Register A

Address: 0x00F09C00
Reset: 0
Read/Write

PED[3, 2, 1, 0] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[30–32, 20–22, 10–12, 0–2] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED3	PIL30	PIL31	PIL32	PED2	PIL20	PIL21	PIL22	PED1	PIL10	PIL11	PIL12	PED0	PIL00	PIL01	PIL02

ELIRB

PIC Edge/Level-Triggered Interrupt Priority Register B

Address: 0x00F01C08
Reset: 0
Read/Write

PED[7, 6, 5, 4] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[70–72, 60–62, 50–52, 40–42] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED7	PIL70	PIL71	PIL72	PED6	PIL60	PIL61	PIL62	PED5	PIL50	PIL51	PIL52	PED4	PIL40	PIL41	PIL42

Interrupt Scheme

ELIRC

PIC Edge/Level-Triggered Interrupt Priority Register C
Address: 0x00F01C10
Reset: 0
Read/Write

PED[11, 10, 9, 8] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[110–112, 100–102, 90–92, 80–82] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED11	PIL110	PIL111	PIL112	PED10	PIL100	PIL101	PIL102	PED9	PIL90	PIL91	PIL92	PED8	PIL80	PIL81	PIL82

ELIRD

PIC Edge/Level-Triggered Interrupt Priority Register D
Address: 000F0x1C18
Reset: 0
Read/Write

PED[15, 14, 13, 12] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[150–152, 140–142, 130–132, 120–122] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED15	PIL150	PIL151	PIL152	PED14	PIL140	PIL141	PIL142	PED13	PIL130	PIL131	PIL132	PED12	PIL120	PIL121	PIL122

Interrupt Scheme

ELIRE

PIC Edge/Level-Triggered Interrupt Priority Register E

Address: 0x00F01C20
Reset: 0x4000
Read/Write

PED[19, 18, 17, 16] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[190–192, 180–182, 170–172, 160–162] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED19	PIL190	PIL191	PIL192	PED18	PIL180	PIL181	PIL182	PED17	PIL170	PIL171	PIL172	PED16	PIL160	PIL161	PIL162

ELIRF

PIC Edge/Level-Triggered Interrupt Priority Register F

Address: 0x00F01C28
Reset: 0x0000 (Boot rewrites to 0x0008)
Read/Write

PED[23, 22, 21, 20] – Trigger Mode for IRQ Input xx, Bits 0, 4, 8, 12	
0	Level-triggered mode
1	Edge-triggered mode

PIL[230–232, 220–222, 210–212, 200–202] – Priority Level for IRQ Input xx, Bits 1–3, 5–7, 9–11, 13–15			
000	Interrupts disabled	100	IPL4
001	IPL0 (lowest priority)	101	IPL5
010	IPL1	110	IPL6
011	IPL2/IPL3	111	IPL7 (highest priority)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PED23	PIL230	PIL231	PIL232	PED22	PIL220	PIL221	PIL222	PED21	PIL210	PIL211	PIL212	PED20	PIL200	PIL201	PIL202

Interrupt Scheme

IPRA

PIC Interrupt Pending Register A
Address: 0x00F09C30
Reset: 0x00
Read/Write

IP – Interrupt Pending Status, Bits 0–15	
Level-Triggered Mode	
0	No interrupt request pending
1	Interrupt request pending
Edge-Triggered Mode	
0	No interrupt request pending
1	Interrupt request acknowledged by SC140 core

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IP15	IP14	IP13	IP12	IP11	IP10	IP9	IP8	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0

Note: IPRA reflects the status of PIC IRQ[0–15].

IPRB

PIC Interrupt Pending Register B
Address: 0x00F09C38
Reset: 0x00
Read/Write

IP – Interrupt Pending Status, Bits 0–15	
Level-Triggered Mode	
0	No interrupt request pending
1	Interrupt request pending
Edge-Triggered Mode	
0	No interrupt request pending
1	Interrupt request acknowledged by SC140 core

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IP31	IP30	IP29	IP28	IP27	IP26	IP25	IP24	IP23	IP22	IP21	IP20	IP19	IP18	IP17	IP16

Note: IPRB reflects the status of PIC IRQ[16–23] on bits 8–15 and the status of PIC NMI[0–7] on bits 0–7.

Instruction Cache

ICCR

Instruction Cache Control Register
Address: 0x00F0FC00
Reset: 0xF001
Read/Write

DM – Debug Mode, Bit 12	
0	Normal mode
1	Debug mode

LM – Lock Mode, Bit 14	
0	ICache not locked
1	ICache locked

ON – On/Off, Bit 15	
0	ICache off
1	ICache on

LB – Lower Boundary, Bits 4–7
Specifies the cache region lower boundary.

UB – Upper Boundary, Bits 0–3
Specifies the cache region upper boundary.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UB				LB				*	*	*	*	DM	*	LM	ON
								0	0	0	0		0		

* = Reserved. Write to 0 for future compatibility

ICCMR

Instruction Cache Command Register
Address: 0x00F0FC02
Reset: 0x0000
Write Only

C – Command, Bits 0–3	
0000	Flush cache
0001	Flush cache between boundaries
1000	Initialize state registers
1001	Clear line defined by DA bits
All others	Reserved

DA – Destination Address Field, Bits 10–15
Specifies the line to clear.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C[3–0]				*	*	*	*	*	*	DA[5–0]					
				0	0	0	0	0	0						

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

DCHCR[0–15]

DMA Channel Configuration Register (page 1 of 2)

DCHCR______________ **(enter number = *n*)**

QBus/System Bus Address: ___________See **page 8-56**
DSI Address: 0x1D0700 + (4 × *n*) = ______________

Reset: 0
Read/Write

PPC – 60x-compatible Bus, Bit 1	
0	Channel is assigned to the local bus
1	Channel is assigned to the 60x-compatible system bus

ACTV – Active DMA Channel x, Bit 0	
0	Channel is disabled
1	Channel is enabled

EXP[0–2] – Expiration Timer, Bits 5–7
The channel will ignore level request to ‘EXP+1’ bus cycles after the assertion of DRACK or DACK signal, as defined by DRACK bit

DRS – DREQ Sensitivity Mode, Bit 8	
0	DREQ is edge-triggered
1	DREQ is level-triggered

DPL – DREQ Polarity, Bit 9	
0	DREQ is active high or rising edge-triggered, according to DRS value
1	DREQ is active low or falling edge-triggered, according to DRS value

BDPTR[0–5] – Buffer Pointer, Bits 10–15
Pointer to the line in the DCPRAM assigned to this channel

See Sheet 2 – Direct Memory Access – DCHCR[0–15]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ACTV	PPC	* 0	* 0	* 0	EXP0	EXP1	EXP2	DRS	DPL	BD-PTR0	BD-PTR1	BD-PTR2	BD-PTR3	BD-PTR4	BD-PTR5

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DRACK	FLY	* 0	RQ-NUM0	RQ-NUM1	RQ-NUM2	RQ-NUM3	RQ-NUM4	FRZ	INT	* 0	* 0	PRIO0	PRIO1	PRIO2	PRIO3

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

DCHCR[0–15]

DMA Channel Configuration Register (page 2 of 2)

DCHCR_______________ **(enter number = *n*)**

QBus/System Bus Address: ____________ See **page 8-56**
DSI Address: 0x1D0700 + (4 × *n*) = _______________

Reset: 0
Read/Write

RQNUM[0–4] – Requestor Number, Bits 19–23			
00000	Reserved	01000	External request 1, DREQ1
00001	Reserved	01001	External request 2, DREQ2
00010	Reserved	01010	External request 3, DREQ3
00011	Reserved	01011	External request 4, DREQ4
00100	Core0 flyby-a counter req.	01100	Core0 flyby-b counter req.
00101	Core1 flyby-a counter req.	01101	Core1 flyby-b counter req.
00110	Core2 flyby-a counter req.	01110	Core2 flyby-b counter req.
00111	Reserved	01111	Reserved
		1xxxx	Reserved

FRZ – Freezes Channel, Bit 24	
0	Channel operates normally
1	Channel is frozen

INT – Internal Requestor, Bit 25	
0	External request. Transaction is initiated by a peripheral
1	Internal request. Transaction between memory and DMA is initiated by DMA

PRIO[0–3] – Channel Priority, Bits 28–31	
0000	Highest priority
1111	Lowest priority

FLY – Flyby Transaction, Bit 17	
0	Dual access transaction
1	Flyby mode. Single access transaction

DRACK – $\overline{\text{DRACK}}$ Protocol, Bit 16	
0	Channel does not use $\overline{\text{DRACK}}$. Expiration timer starts counting after DACK assertion
1	Channel uses DRACK. Expiration timer starts counting after DRACK assertion

See Sheet 1 – Direct Memory Access – DCHCR[0–15]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ACTV	PPC	*	*	*	EXP0	EXP1	EXP2	DRS	DPL	BD-PTR0	BD-PTR1	BD-PTR2	BD-PTR3	BD-PTR4	BD-PTR5	DRACK	FLY	*	RQ-NUM0	RQ-NUM1	RQ-NUM2	RQ-NUM3	RQ-NUM4	FRZ	INT	*	*	PRIO0	PRIO1	PRIO2	PRIO3
		0	0	0														0								0	0				

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

DPCR

DMA Pin Configuration Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D078C

Reset: 0
Read/Write

0	1	2	3	4	5	6	7
*	*	*	*	SDN0	SDN1	*	*
0	0	0	0			0	0

* = Reserved. Write to 0 for future compatibility

SDN0 – Select DONE0, Bit 4	
0	$\overline{\text{DONE1}}$/$\overline{\text{DRACK1}}$ is $\overline{\text{DONE1}}$
1	$\overline{\text{DONE1}}$/$\overline{\text{DRACK1}}$ is $\overline{\text{DRACK1}}$

SDN1 – Select DONE1, Bit 5	
0	$\overline{\text{DONE2}}$/$\overline{\text{DRACK2}}$ is $\overline{\text{DONE2}}$
1	$\overline{\text{DONE2}}$/$\overline{\text{DRACK2}}$ is $\overline{\text{DRACK2}}$

DSTR

DMA Status Register

QBus/System Bus Address: ___________ See **page 8-56**

DSI Address: 0x1D0784

Reset: 0
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15

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

I[0–15] – Interrupt from DMA Channel, Bits 0–15	
0	No interrupt
1	Channel requires interrupt servicing

Note: The I bits reflect the current interrupt status for the respective DMA channels. Write a one to the bit to clear it. Writing a zero has no effect.

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

DCPRAM

DMA Channel Parameters RAM

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0800
Reset: Undefined
Read/Write

BD_ADDR – Buffer Address, Bits 0–31

Contains the buffer’s current address

BD_ADDR

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

BD_SIZE – Buffer Size, Bits 32–63

Contains the size of the buffer remaining for transfer

BD_SIZE

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

BD_ATTR – Buffer Attributes and Temporary Data, Bits 64–95

See the BD_ATTR programming sheets

BD_ATTR

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
INTRPT	CYC	CONT	*	NO_INC	BP0	BP1	*	*	NBUS	NBD0	NBD1	NBD2	NBD3	NBD4	NBD5	*	*	*	*	*	*	TSZ0	TSZ1	TSZ2	*	FLS	RD	*	TC	*	GBL
			0				0	0								0	0	0	0	0	0				0			0		0	

* = Reserved. Write to 0 for future compatibility

BD_BSIZE – Buffer Size, Bits 96–127

Contains the buffer base size

BD_BSIZE

96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

Direct Memory Access (DMA)

BD_ATTR

Buffer Attributes Parameter (page 1 of 2)
Reset: Undefined
Read/Write

CYC – Cyclic Address, Bit 65	
0	Sequential address. BD_ADDR is incremented
1	Cyclic address. BD_ADDR is restored to original value

INTRPT – Interrupt, Bit 64	
0	Do not issue interrupt
1	Issue interrupt when size reaches zero

CONT – Continuous Buffer Mode, Bit 66	
0	Buffer is closed when BD_SIZE reaches zero
1	Buffer continues operating when BD_SIZE reaches zero

NO_INC – Increments Address, Bit 68	
0	Increment address after request is serviced
1	Do not increment address after request is serviced

BP[0–1] – Bus Priority, Bits 69–70	
00	Arbitrate for bus mastership with request 1100
01	Arbitrate for bus mastership with request 1011
10	Arbitrate for bus mastership with request 1010
11	Reserved

NBUS – Next Bus, Bit 73	
0	Local bus
1	60x-compatible system bus

See Sheet 2 – Direct Memory Access – BD_ATTR

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
INTRPT	CYC	CONT	*	NO_INC	BP0	BP1	*	*	NBUS	NBD0	NBD1	NBD2	NBD3	NBD4	NBD5
			0				0	0							

80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
*	*	*	*	*	*	TSZ0	TSZ1	TSZ2	*	FLS	RD	*	TC	*	GBL
0	0	0	0	0	0				0			0		0	

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

BD_ATTR

Buffer Attributes Parameter (page 2 of 2)
Reset: Undefined
Read/Write

TSZ[0–2] – Transfer Size, Bits 86–88			
001	Max transfer size is 8 bits	100	Max transfer size is one burst
010	Max transfer size is 16 bits		
011	Max transfer size is 32 bits	101	Reserved
000	Max transfer size is 64 bits	11x	Reserved

NBD[0–5] – Next Buffer, Bits 74–79
When size reaches zero and CONT is set, the next request will call the buffer pointed to by NBD

See Sheet 1 – Direct Memory Access – BD_ATTR

FLS – Flush FIFO, Bit 90	
0	Do not flush FIFO
1	Flush FIFO

RD – Read Channel, Bit 91	
0	Write transaction
1	Read transaction

TC – Transfer Code, Bit 93	
0	TC[0–2] value is 110
1	TC[0–2] value is 111

GBL – Global Transaction, Bit 95	
0	Non global transaction
1	Global transaction

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
INTRPT	CYC	CONT	*	NO_INC	BP0	BP1	*	*	NBUS	NBD0	NBD1	NBD2	NBD3	NBD4	NBD5
			0				0	0							

80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
*	*	*	*	*	*	TSZ0	TSZ1	TSZ2	*	FLS	RD	*	TC	*	GBL
0	0	0	0	0	0				0			0		0	

* = Reserved. Write to 0 for future compatibility

Direct Memory Access (DMA)

DIMR

DMA Internal Mask Register

QBus/System Bus Address: ___________See **page 8-56**
DSI Address: 0x1D0780

Reset: 0
Read/Write

M[0–15] – Internal Interrupt Mask, Bits 0–15	
0	LIC interrupt disabled
1	Interrupt to LIC enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: All internal DMA interrupts are disabled after reset. The user must enable a DMA channel interrupt request.

* = Reserved. Write to 0 for future compatibility

DEMR

DMA External Mask Register

QBus/System Bus Address: ___________See **page 8-56**
DSI Address: 0x1D0790

Reset: 0
Read/Write

M[0–15] – GIC/External Interrupt Mask, Bits 0–15	
0	GIC interrupt disabled
1	Interrupt to GIC for external host enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: All external DMA interrupts are disabled after reset. The user must enable a DMA channel interrupt request.

* = Reserved. Write to 0 for future compatibility

CAUTION: Enabling a channel in DIMR and DEMR results in undefined system behavior. Therefore, for each channel, enable the interrupt in the DIMR or the DEMR, but not both.

Direct Memory Access (DMA)

DTEAR

DMA Transfer Error Address Register

QBus/System Bus Address: ___________ See **page 8-56**
DSI Address: 0x1D0788

Reset: 0
Read/Write

SDN0 – DMA Channel System Bus Error, Bit 0	
0	No system bus DMA transfer error
1	System bus error: Error address in PDMTEA, Channel in PDMTER

SDN1 – DMA Channel Local Bus Error, Bit 1	
0	No local bus DMA transfer error
1	Local bus error: Error address in LDMTEA, Channel in LDMTER

0	1	2	3	4	5	6	7
DBER_P	DBER_L	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**

* = Reserved. Write to 0 for future compatibility

Note: Write a one to the bit to clear it. Writing a zero has no effect.

Direct Slave Interface (DSI)

DCR

DSI Control Register

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE000

DSI Address: 0x1BE000

Reset: 0x00000000

Read/Write

SLDWA – Sliding Window Active, Bit 0	
0	Sliding window is not active
1	Sliding window is active

BRSTP – Burst Signal Polarity, Bit 1	
0	Burst signal is active low (HBRST)
1	Burst signal is active high (HBRST)

BEM – Byte Enable Multiple, Bit 2	
0	Single byte enable signal is used
1	Multiple byte enable signals are used

SNGLM – Single Strobe Mode, Bit 3	
0	Dual strobe mode
1	Single strobe mode

HTAAD – $\overline{\mathrm{HTA}}$ Actively Driven, Bit 4	
0	HTA is released in logic 0 (HTADT = 00)
1	HTA is released in logic 1 (HTADT = 01 or 10 or 11)

LEDS – Little-Endian Structure, Bits 5–6	
00	8-bit data structure
01	16-bit data structure
10	32-bit data structure
11	64-bit data structure

DSRFA – Data Structure Register Field Active, Bit 7	
0	Data structure is from the HDST signals
1	Data structure is from the LEDS field

RPE – Read Prefetch Enable, Bit 8	
0	Read prefetch disabled
1	Read prefetch enabled

HTADT – $\overline{\mathrm{HTA}}$ Drive Time, Bits 9–10	
00	No drive time
01	HTA driven for 0.5–1 internal bus clock cycles
10	HTA driven for 1–1.5 internal bus clock cycles
11	HTA driven for 1.5–2.5 internal bus clock cycles

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SLDWA	BRSTP	BEM	SNGLM	HTAAD	LEDS		DSRFA	RPE	HTADT		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write a 0 to all reserved bits for future compatibility

Direct Slave Interface (DSI)

DSWBAR

DSI Sliding Window Base Address Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE008

DSI Address: 0x1BE008

Reset: 0

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Value Ignored by DSI											BAVAL					*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BAVAL – Base Address Value, Bits 11–14

Any value between 0b0000–0b1111

* = Reserved. Write to 0 for future compatibility

DIBAR[9,11]

DSI Internal Base Address Registers 9 and 11

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE010, 0x1FBE020

DSI Address: 0x1BE010, 0x1BE020

Reset: 0

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BA																	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BA – Base Address, Bits 0–16

Must be set to the same value as the BA field in BR9 and BR11, respectively

* = Reserved. Write to 0 for future compatibility

DIAMR[9,11]

DSI Internal Address Mask Registers 9 and 11

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE028, 0x01FEB038

DSI Address: 0x1BE028, 0x1BE038

Reset: 0

For DIAMR9, the Boot program writes 0xFFFC0000.

For DIAMR11, the Boot program writes 0xFFE00000.

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
AM																	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AM – Address Mask, Bits 0–16

Must be set to the same value as the AM field in OR9 in GPCM mode or OR11 in UPM mode, respectively

* = Reserved. Write to 0 for future compatibility

Direct Slave Interface (DSI)

DCIR

DSI Chip ID Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE040

DSI Address: 0x1BE040

Reset: 0

Read/Write

CHIPID – Chip ID Value, Bits 0–3
Any value between 0b0000–0b1111. Initialized at PORESET deassertion from CHIP_ID[0–3].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CHIPID				*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

DDR

DSI Disable Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE048

DSI Address: 0x1BE048

Reset: 0

Read/Write from IPBus. Read-only from host side.

DSISTP – DSI Stop, Bit 0	
0	DSI cannot enter Stop mode
1	DSI can enter Stop mode

DSIDIS – DSI Disable, Bit 1	
0	DSI enabled
1	DSI disabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DSISTP	DSIDIS	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

DER

DSI Error Register

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBE808

DSI Address: 0x1BE808

Reset: 0

Read/Write

OVF – Overflow, Bit 0	
0	No overflow occurred
1	Overflow occurred

Note: This is a sticky bit. Write a 1 to this bit to clear it.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OVF	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Internal Peripheral Bus (IPBus)

SCR

Stop Control Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBB000
DSI Address: 0x1BB000
Reset: 0x00000000
Read/Write

UART_STC – UART Stop, Bit 31	
0	Normal UART operation
1	UART Stop Mode

DSI_STC – DSI Stop, Bit 30	
0	Normal DSI operation
1	DSI Stop Mode

GIC_STC – GIC Stop, Bit 29	
0	Normal GIC operation
1	GIC Stop Mode

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	GIC_ STC	DSI_ STC	UART_ STC

* = Reserved. Write 0 for future compatibility

Hardware Semaphores

HSMPR[0–7]

Hardware Semaphore Registers 0–7

HSMPR____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC100 + (8 × *n*) = ______________
DSI Address: 0x1BC100 + (8 × *n*) = ______________

Reset: 0x00000000
Bits 0–23 Read only
Bits 24–31 Read/Write

SMPVAL – Semaphore Value, Bits 24–31

If SMPVAL = 0, the semaphore is free. Any other value indicates that the semaphore is locked and the value is the lock code. Each master device should have a unique lock code to indicate which device locked the semaphore. If SMPVAL is not zero, it is locked. The master device that locked it must write a zero to SMPVAL to free the semaphore.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	SMPVAL							
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**								

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]GIR

TDM 0–3 General Interface Registers

TDM_____________ (enter number = n) GIR

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FF8 + (16384 × *n*) = _______________

DSI Address: 0x183FF8 + (16384 × *n*) = _______________

Reset: 0x00000000
Read/Write

RTSAL – Receive and Transmit Sharing and Active Links, Bits 28–31	
0000	Receive and transmit are independent. The TDM receives on data link and transmits one data link.
0001	Receive and transmit are independent. The TDM receives two data links and transmits two data links (valid only if CTS = 1).
0010	Reserved
0011	Reserved
0100	Receive and transmit share frame clock and frame sync. The TDM receives on data link and transmits one data link.
0101	Receive and transmit share frame clock and frame sync. The TDM receives two data links and transmits two data links (valid only if CTS = 1).
0110	Reserved
0111	Reserved
1000	Reserved
1001	Reserved
1010	Reserved
1011	Reserved
1100	Receive and transmit share frame clock, frame sync, and one full duplex data link.
1101	Receive and transmit share frame clock, frame sync, and two full duplex data links.
1110	Reserved
1111	Receive and transmit share frame clock, frame sync, and four full duplex data links.

CTS – Common TDM Signals, Bit 27	
0	Does not share TDM signals with other TDM modules
1	Shares TDM signals with other TDM modules

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CTAS	RTSAL			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]RIR

TDM 0–3 Receiver Interface Registers

TDM_____________ (enter number = n) RIR

Local Bus Address: ____________
See **page 8-28**

QBus (IPBus) Address: 0x01F83FF0 + (16384 × *n*) = _______________
DSI Address: 0x183FF0 + (16384 × *n*) = _______________
Reset: 0x00010000
Read/Write

RRDO – Receive Reversed Data Order, Bit 31	
0	First received bit stored as msb in memory
1	First received bit stored as lsb in memory

RFSE – Receive Frame Sync Edge, Bit 30	
0	RSYN sampled on RCLK rising edge
1	RSYN sampled on RCLK falling edge

RDE – Receive Data Edge, Bit 29	
0	RDAT sampled on RCLK rising edge
1	RDAT sampled on RCLK falling edge

RSL – Receive Sync Level, Bit 28	
0	RSYN is active on logic 1
1	RSYN is active on logic 0

RFSD – Receive Frame Sync Delay, Bits 26–27	
00	0 (RFSE = RDE) or 0.5 (RFSE ¼ RDE) RCLKs
01	1 (RFSE = RDE) or 1.5 (RFSE ¼ RDE) RCLKs
10	2 (RFSE = RDE) or 2.5 (RFSE ¼ RDE) RCLKs
11	3 (RFSE = RDE) or 3.5 (RFSE ¼ RDE) RCLKs

RSTL – Receive Second Threshold Level, Bit 17	
0	Receive second threshold interrupt is pulse
1	Receive second threshold interrupt is level

RFTL – Receive First Threshold Level, Bit 16	
0	Receive first threshold interrupt is pulse
1	Receive first threshold interrupt is level

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RFTL	RSTL	*	*	*	*	*	*	*	*	RFSD		RSL	RDE	RFSE	RRDO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			0	0	0	0	0	0	0	0						

* = Reserved. Write a 1 to bit 15 and 0 to all other reserved bits for future compatibility

TDM

TDM[0–3]TIR

TDM 0–3 Transmitter Interface Registers

TDM_____________ (enter number = n) TIR

Local Bus Address: ___________
See **page 8-28**

QBus (IPBus) Address: 0x01F83FE8 + (16384 × *n*) = _______________
DSI Address: 0x183FE8 + (16384 × *n*) = _______________

Reset: 0x00010000
Read/Write

SOE – Sync Out Edge, Bit 21	
0	TSYN output driven on TCLK rising edge
1	TSYN output driven on TCLK falling edge

SOL – Sync Out Length, Bit 20	
0	TSYN output is one bit wide
1	TSYN output is one channel wide

TAO – Transmit Always Output, Bit 19	
0	TDM TX does not drive TDAT for inactive channels
1	TDM TX drives all TDAT channels

TSO – Transmit Sync Output, Bit 18	
0	Transmit sync is input
1	Transmit sync is output

TSTL – Receive Second Threshold Level, Bit 17	
0	Transmit second threshold interrupt is pulse
1	Transmit second threshold interrupt is level

TFTL – Receive First Threshold Level, Bit 16	
0	Transmit first threshold interrupt is pulse
1	Transmit first threshold interrupt is level

TRDO – Transmit Reversed Data Order, Bit 31	
0	First transmitted bit is msb
1	First transmitted bit is lsb

TFSE – Transmit Frame Sync Edge, Bit 30	
0	TSYN sampled on TCLK rising edge
1	TSYN sampled on TCLK falling edge

TDE – Transmit Data Edge, Bit 29	
0	TDAT sampled on TCLK rising edge
1	TDAT sampled on TCLK falling edge

TSL – Transmit Sync Level, Bit 28	
0	TSYN is active on logic 1
1	TSYN is active on logic 0

TFSD – Transmit Frame Sync Delay, Bits 26–27	
00	–1 (TFSE = TDE) or –0.5 (TFSE ¼ TDE) TCLKs
01	0 (TFSE = TDE) or 0.5 (TFSE ¼ TDE) TCLKs
10	1 (TFSE = TDE) or 1.5 (TFSE ¼ TDE) TCLKs
11	2 (TFSE = TDE) or 2.5 (TFSE ¼ TDE) TCLKs

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TFTL	TSTL	TSO	TAO	SOL	SOE	*	*	*	*	TFSD		TSL	TDE	TFSE	TRDO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1							0	0	0	0						

* = Reserved. Write a 1 to bit 15 and 0 to all other reserved bits for future compatibility

TDM

TDM[0–3]RFP

TDM 0–3 Receive Frame Parameters

TDM____________ (enter number = *n*) RFP

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FE0 + (16384 × *n*) = ______________

DSI Address: 0x183FE0 + (16384 × n) = ______________

Reset: 0x00000000

Read/Write

RNCF – Receive Number of Channels in a TDM Frame, Bits 8–15							
0x00 = R	0x20 = R	0x40 = R	0x60 = R	0x80 = R	0xA0 = R	0xC0 = R	0xE0 = R
0x01 = 2	0x21 = 34	0x41 = 66	0x61 = 98	0x81 = 130	0xA1 = 162	0xC1 = 194	0xE1 = 226
0x02 = R	0x22 = R	0x42 = R	0x062 = R	0x82 = R	0xA2 = R	0xC2 = R	0xE2 = R
0x03 = 4	0x23 = 36	0x43 = 68	0x63 = 100	0x83 = 132	0xA3 = 164	0xC3 = 196	0xE3 = 228
0x04 = R	0x24 = R	0x44 = R	0x64 = R	0x84 = R	0xA4 = R	0xC4 = R	0xE4 = R
0x05 = 6	0x25 = 38	0x45 = 70	0x65 = 102	0x85 = 134	0xA5 = 166	0xC5 = 198	0xE5 = 230
0x06 = R	0x26 = R	0x46 = R	0x66 = R	0x86 = R	0xA6 = R	0xC6 = R	0xE6 = R
0x07 = 8	0x27 = 40	0x47 = 72	0x67 = 104	0x87 = 136	0xA7 = 168	0xC7 = 200	0xE7 = 232
0x08 = R	0x28 = R	0x48 = R	0x68 = R	0x88 = R	0xA8 = R	0xC8 = R	0xE8 = R
0x09 = 10	0x29 = 42	0x49 = 74	0x69 = 106	0x89 = 138	0xA9 = 170	0xC9 = 202	0xE9 = 234
0x0A = R	0x2A = R	0x4A = R	0x6A = R	0x8A = R	0xAA = R	0xCA = R	0xEA = R
0x0B = 12	0x2B = 44	0x4B = 76	0x6B = 108	0x8B = 140	0xAB = 172	0xCB = 204	0xEB = 236
0x0C = R	0x2C = R	0x4C = R	0x6C = R	0x8C = R	0xAC = R	0xCC = R	0xEC = R
0x0D = 14	0x2D = 46	0x4D = 78	0x6D = 110	0x8D = 142	0xAD = 174	0xCD = 206	0xED = 238
0x0E = R	0x2E = R	0x4E = R	0x6E = R	0x8E = R	0xAE = R	0xCE = R	0xEE = R
0x0F = 16	0x2F = 48	0x4F = 80	0x6F = 112	0x8F = 144	0xAF = 176	0xCD = 208	0xEF = 240
0x10 = R	0x30 = R	0x50 = R	0x70 = R	0x90 = R	0xB0 = R	0xD0 = R	0xF0 = R
0x11 = 18	0x31 = 50	0x51 = 82	0x71 = 114	0x91 = 146	0xB1 = 178	0xD1 = 210	0xF1 = 242
0x12 = R	0x32 = R	0x52 = R	0x72 = R	0x92 = R	0xB2 = R	0xD2 = R	0xF2 = R
0x13 = 20	0x33 = 52	0x53 = 84	0x73 = 116	0x93 = 148	0xB3 = 180	0xD3 = 212	0xF3 = 244
0x14 = R	0x34 = R	0x54 = R	0x74 = R	0x94 = R	0xB4 = R	0xD4 = R	0xF4 = R
0x15 = 22	0x35 = 54	0x55 = 86	0x75 = 118	0x95 = 150	0xB5 = 182	0xD5 = 214	0xF5 = 246
0x16 = R	0x36 = R	0x56 = R	0x76 = R	0x96 = R	0xB6 = R	0xD6 = R	0xF6 = R
0x17 = 24	0x37 = 56	0x57 = 88	0x77 = 120	0x97 = 152	0xB7 = 184	0xD7 = 216	0xF7 = 248
0x18 = R	0x38 = R	0x58 = R	0x78 = R	0x98 = R	0xB8 = R	0xD8 = R	0xF8 = R
0x19 = 26	0x39 = 58	0x59 = 90	0x79 = 122	0x99 = 154	0xB9 = 186	0xD9 = 218	0xF9 = 250
0x1A = R	0x3A = R	0x5A = R	0x7A = R	0x9A = R	0xBA = R	0xDA = R	0xFA = R
0x1B = 28	0xeB = 60	0x5B = 92	0x7B = 124	0x9B = 156	0xBB = 188	0xDB = 220	0xFB = 252
0x1C = R	0x3C = R	0x5C = R	0x7C = R	0x9C = R	0xBC = R	0xDC = R	0xFC = R
0x1D = 30	0x3D = 62	0x5D = 94	0x7D = 126	0x9D = 158	0xBD = 190	0xDD = 222	0xFD = 254
0x1E = R	0x3E = R	0x5E = R	0x7E = R	0x9E = R	0xBE = R	0xDE = R	0xFE = R
0x1F = 32	0x3F = 64	0x5F = 96	0x7F = 128	0x9F = 160	0xBF = 192	0xDF = 224	0xFF = 256

Note: Each cell above lists an RNCF value and number of receive channels indicated by that value. R indicates that the RNCF value is reserved (all even values).

RUBM – Receive Unified Buffer Mode, Bit 31	
0	Each channel written to its own local bus data buffer
1	All channels written to the same local bus data buffer

RT1 – Receive T1 Frame, Bit 30	
0	Receive frame is not a T1 frame
1	Receive frame is a T1 frame

RCS – Receive Channel Size, Bits 26–29	
0000	Reserved
0001	Receiver channel size is 2 bits
0010	Reserved
0011	Receiver channel size is 4 bits
0100–0110	Reserved
0111	Receiver channel size is 8 bits
1000–1110	Reserved
1111	Receiver channel size is 16 bits

RCDBL – Receive Channel Data Bits Latency, Bits 21–23	
000	Maximum 64 channel bits
001	Maximum 128 channel bits
010	Maximum 256 channel bits
011	Maximum 512 channel bits
100	Maximum 1024 channel bits
101	Maximum 2048 channel bits
110	Reserved
111	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	RNCF								*	*	*	*	*	RCDBL			*	*	RCS				RT1	RUBM
0	0	0	0	0	0	0	0									0	0	0	0	0				0	0						

* = Reserved. Write 0 to all reserved bits for future compatibility

TDM

TDM[0–3]TFP

TDM 0–3 Transmit Frame Parameters

TDM_____________ (enter number = *n*) TFP

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FD8 + (16384 × *n*) = ______________

DSI Address: 0x183FD8 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

TNCF – Transmit Number of Channels in a TDM Frame, Bits 8–15

0x00 = R	0x20 = R	0x40 = R	0x60 = R	0x80 = R	0xA0 = R	0xC0 = R	0xE0 = R
0x01 = 2	0x21 = 34	0x41 = 66	0x61 = 98	0x81 = 130	0xA1 = 162	0xC1 = 194	0xE1 = 226
0x02 = R	0x22 = R	0x42 = R	0x062 = R	0x82 = R	0xA2 = R	0xC2 = R	0xE2 = R
0x03 = 4	0x23 = 36	0x43 = 68	0x63 = 100	0x83 = 132	0xA3 = 164	0xC3 = 196	0xE3 = 228
0x04 = R	0x24 = R	0x44 = R	0x64 = R	0x84 = R	0xA4 = R	0xC4 = R	0xE4 = R
0x05 = 6	0x25 = 38	0x45 = 70	0x65 = 102	0x85 = 134	0xA5 = 166	0xC5 = 198	0xE5 = 230
0x06 = R	0x26 = R	0x46 = R	0x66 = R	0x86 = R	0xA6 = R	0xC6 = R	0xE6 = R
0x07 = 8	0x27 = 40	0x47 = 72	0x67 = 104	0x87 = 136	0xA7 = 168	0xC7 = 200	0xE7 = 232
0x08 = R	0x28 = R	0x48 = R	0x68 = R	0x88 = R	0xA8 = R	0xC8 = R	0xE8 = R
0x09 = 10	0x29 = 42	0x49 = 74	0x69 = 106	0x89 = 138	0xA9 = 170	0xC9 = 202	0xE9 = 234
0x0A = R	0x2A = R	0x4A = R	0x6A = R	0x8A = R	0xAA = R	0xCA = R	0xEA = R
0x0B = 12	0x2B = 44	0x4B = 76	0x6B = 108	0x8B = 140	0xAB = 172	0xCB = 204	0xEB = 236
0x0C = R	0x2C = R	0x4C = R	0x6C = R	0x8C = R	0xAC = R	0xCC = R	0xEC = R
0x0D = 14	0x2D = 46	0x4D = 78	0x6D = 110	0x8D = 142	0xAD = 174	0xCD = 206	0xED = 238
0x0E = R	0x2E = R	0x4E = R	0x6E = R	0x8E = R	0xAE = R	0xCE = R	0xEE = R
0x0F = 16	0x2F = 48	0x4F = 80	0x6F = 112	0x8F = 144	0xAF = 176	0xCD = 208	0xEF = 240
0x10 = R	0x30 = R	0x50 = R	0x70 = R	0x90 = R	0xB0 = R	0xD0 = R	0xF0 = R
0x11 = 18	0x31 = 50	0x51 = 82	0x71 = 114	0x91 = 146	0xB1 = 178	0xD1 = 210	0xF1 = 242
0x12 = R	0x32 = R	0x52 = R	0x72 = R	0x92 = R	0xB2 = R	0xD2 = R	0xF2 = R
0x13 = 20	0x33 = 52	0x53 = 84	0x73 = 116	0x93 = 148	0xB3 = 180	0xD3 = 212	0xF3 = 244
0x14 = R	0x34 = R	0x54 = R	0x74 = R	0x94 = R	0xB4 = R	0xD4 = R	0xF4 = R
0x15 = 22	0x35 = 54	0x55 = 86	0x75 = 118	0x95 = 150	0xB5 = 182	0xD5 = 214	0xF5 = 246
0x16 = R	0x36 = R	0x56 = R	0x76 = R	0x96 = R	0xB6 = R	0xD6 = R	0xF6 = R
0x17 = 24	0x37 = 56	0x57 = 88	0x77 = 120	0x97 = 152	0xB7 = 184	0xD7 = 216	0xF7 = 248
0x18 = R	0x38 = R	0x58 = R	0x78 = R	0x98 = R	0xB8 = R	0xD8 = R	0xF8 = R
0x19 = 26	0x39 = 58	0x59 = 90	0x79 = 122	0x99 = 154	0xB9 = 186	0xD9 = 218	0xF9 = 250
0x1A = R	0x3A = R	0x5A = R	0x7A = R	0x9A = R	0xBA = R	0xDA = R	0xFA = R
0x1B = 28	0xeB = 60	0x5B = 92	0x7B = 124	0x9B = 156	0xBB = 188	0xDB = 220	0xFB = 252
0x1C = R	0x3C = R	0x5C = R	0x7C = R	0x9C = R	0xBC = R	0xDC = R	0xFC = R
0x1D = 30	0x3D = 62	0x5D = 94	0x7D = 126	0x9D = 158	0xBD = 190	0xDD = 222	0xFD = 254
0x1E = R	0x3E = R	0x5E = R	0x7E = R	0x9E = R	0xBE = R	0xDE = R	0xFE = R
0x1F = 32	0x3F = 64	0x5F = 96	0x7F = 128	0x9F = 160	0xBF = 192	0xDF = 224	0xFF = 256

Note: Each cell above lists an TNCF value and number of transmit channels indicated by that value. R indicates that the TNCF value is reserved (all even values).

RUBM – Transmit Unified Buffer Mode, Bit 31

0	Each channel is read from its own local bus data buffer
1	All channels are read from the same local bus data buffer

RT1 – Transmit T1 Frame, Bit 30

0	Transmit frame is not a T1 frame
1	Transmit frame is a T1 frame

TCS – Transmit Channel Size, Bits 26–29

0000	Reserved
0001	Transmitter channel size is 2 bits
0010	Reserved
0011	Transmitter channel size is 4 bits
0100–0110	Reserved
0111	Transmitter channel size is 8 bits
1000–1110	Reserved
1111	Transmitter channel size is 16 bits

TCDBL – Transmit Channel Data Bits Latency, Bits 21–23

000	Maximum 64 channel bits
001	Maximum 128 channel bits
010	Maximum 256 channel bits
011	Maximum 512 channel bits
100	Maximum 1024 channel bits
101	Maximum 2048 channel bits
110	Reserved
111	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	TNCF								*	*	*	*	*	TCDBL			*	*	TCS				TT1	TUBM
0	0	0	0	0	0	0	0									0	0	0	0	0				0	0						

* = Reserved. Write 0 to all reserved bits for future compatibility

TDM

TDM[0–3]RDBS

TDM 0–3 Receive Data Buffer Size

TDM_____________ (enter number = *n*) RDBS

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FD0 + (16384 × *n*) = _______________

DSI Address: 0x183FD0 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

RDBS – Receive Data Buffer Size, Bits 8–31

Value is receive data buffers size in bytes minus 1. Because the buffer size is 8-byte aligned, bits 29–31 must be set to 111.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	RDBS																							
0	0	0	0	0	0	0	0																						1	1	1

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset. For correct operation, you must write 1s to these bits.

TDM[0–3]TDBS

TDM 0–3 Transmit Data Buffer Size

TDM_____________ (enter number = *n*) TDBS

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FC8 + (16384 × *n*) = _______________

DSI Address: 0x183FC8 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

TDBS – Transmit Data Buffer Size, Bits 8–31

Value is transmit data buffers size in bytes minus 1. Because the buffer size is 8-byte aligned, bits 29–31 must be set to 111.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	TDBS																							
0	0	0	0	0	0	0	0																						1	1	1

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset. For correct operation, you must write 1s to these bits.

TDM

TDM[0–3]RGBA

TDM 0–3 Receive Global Base Address

TDM_____________ (enter number = *n*) RGBA

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FC0 + (16384 × *n*) = _______________

DSI Address: 0x183FC0 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

RGBA – Receive Global Base Address, Bits 16–31

Value forms the 16 msbs for the receiver data buffers global base address. The actual address for each buffer is RCDBA + (RBGA<<16)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RGBA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write 0 for future compatibility

TDM[0–3]TGBA

TDM 0–3 Transmit Global Base Address

TDM_____________ (enter number = *n*) TGBA

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FB8 + (16384 × *n*) = _______________

DSI Address: 0x183FB8 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

TGBA – Transmit Global Base Address, Bits 16–31

Value forms the 16 msbs for the transmit data buffers global base address. The actual address for each buffer is TCDBA + (TBGA<<16)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TGBA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]ACR

TDM 0–3 Adaptation Control Registers

TDM_____________ (enter number = *n*) ACR

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FB0 + (16384 × *n*) = _______________

DSI Address: 0x183FB0 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

LTS – Learn Transmit Sync, Bit 31	
0	Adaptation machine learns the receive sync
1	Adaptation machine learns the transmit sync

AME – Adaptation Machine Enable, Bit 30	
0	Adaptation machine is disabled
1	Adaptation machine is enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	AME	LTS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]RCR

TDM 0–3 Receive Control Registers

TDM_____________ (enter number = *n*) RCR

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FA8 + (16384 × *n*) = _______________

DSI Address: 0x183FA8 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	REN

REN – Receive Enable, Bit 31	
0	Receiver is disabled
1	Receiver is enabled

* = Reserved. Write 0 for future compatibility

TDM[0–3]TCR

TDM 0–3 Transmit Control Registers

TDM_____________ (enter number = *n*) TCR

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83FA0 + (16384 × *n*) = _______________

DSI Address: 0x183FA0 + (16384 × *n*) = _______________

Reset: 0x00000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	TEN

TEN – Transmit Enable, Bit 31	
0	Transmitter is disabled
1	Transmitter is enabled

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]RDBFT

TDM 0–3 Receive Data Buffer First Threshold

TDM_____________ (enter number = *n*) RDBFT

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F98 + (16384 × *n*) = ______________

DSI Address: 0x183F98 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

RDBFT – Receive Data Buffer First Threshold, Bits 8–31

Value is receive data buffer first threshold location. Because the register value has a granularity of 8 bytes, bits 29–31 must be 000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	RDBFT																							
0	0	0	0	0	0	0	0																						0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset.
For correct operation, you
must always write 0s to these bits.

TDM[0–3]TDBFT

TDM 0–3 Transmit Data Buffer First Threshold

TDM_____________ (enter number = *n*) TDBFT

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F90 + (16384 × *n*) = ______________

DSI Address: 0x183F90 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

TDBFT – Transmit Data Buffer First Threshold, Bits 8–31

Value is transmit data buffer first threshold location. Because the register value has a granularity of 8 bytes, bits 29–31 must be 000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	TDBFT																							
0	0	0	0	0	0	0	0																						0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset.
For correct operation, you
must always write 0s to these bits.

TDM

TDM[0–3]RDBST

TDM 0–3 Receive Data Buffer Second Threshold

TDM_____________ (enter number = *n*) RDBST

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F88 + (16384 × *n*) = ______________

DSI Address: 0x183F88 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

RDBST – Receive Data Buffer Second Threshold, Bits 8–31

Value is receive data buffer second threshold location. Because the register value has a granularity of 8 bytes, bits 29–31 must be 000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	RDBST																							
0	0	0	0	0	0	0	0																						0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset.
For correct operation, you
must always write 0s to these bits.

TDM[0–3]TDBST

TDM 0–3 Transmit Data Buffer Second Threshold

TDM_____________ (enter number = *n*) TDBST

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F80 + (16384 × *n*) = ______________

DSI Address: 0x183F80 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

TDBST – Transmit Data Buffer Second Threshold, Bits 8–31

Value is transmit data buffer second threshold location. Because the register value has a granularity of 8 bytes, bits 29–31 must be 000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	TDBST																							
0	0	0	0	0	0	0	0																						0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 29–31 are 0s after reset.
For correct operation, you
must always write 0s to these bits.

TDM

TDM[0–3]RCPR[0–255]

TDM 0–3 Receive Channel Parameter Registers 0–255

TDM_____________ (enter number = *n*) RCPR_________ (enter channel number = *c*)

Local Bus Address: base + (4 × *c*) = ___________See **page 8-28**

QBus (IPBus) Address: 0x01F81000 + (16384 × *n*) + (4 × *c*) = _______________

DSI Address: 0x181000 + (16384 × *n*) + (4 × *c*) = _______________

Reset: Unknown
Read/Write

RACT – Receive Channel Active, Bit 0	
0	Channel is non-active
1	Channel is active

RCONV – Receive Channel Convert, Bits 1–2	
00	Receive channel is in transparent mode
01	Receive channel is a μ-law channel
10	Receive channel is an A-law channel
11	Reserved

RCDBA – Receive Channel Data Buffer Base Address, Bits 8–31

Determines receive data buffer base address offset from the RGBA. The RCDBA must be 16-byte aligned, so bits 28–31 must be 0000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RACT	RCONV		*	*	*	*	*	RCDBA																							
			0	0	0	0	0																					0	0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 28–31 are unknown after reset.
For correct operation, you
must write 0s to these bits.

TDM[0–3]TCPR[0–255]

TDM 0–3 Transmit Channel Parameter Registers 0–255

TDM_____________ (enter number = *n*) RCPR_________ (enter channel number = *c*)

Local Bus Address: base + (4 × *c*) = ___________See **page 8-28**

QBus (IPBus) Address: 0x01F82800 + (16384 × *n*) + (4 × *c*) = _______________

DSI Address: 0x182800 + (16384 × *n*) + (4 × *c*) = _______________

Reset: Unknown
Read/Write

TACT – Transmit Channel Active, Bit 0	
0	Channel is non-active
1	Channel is active

TCONV – Transmit Channel Convert, Bits 1–2	
00	Transmit channel is in transparent mode
01	Transmit channel is a μ-law channel
10	Transmit channel is an A-law channel
11	Reserved

TCDBA – Transmit Channel Data Buffer Base Address, Bits 8–31

Determines transmit data buffer base address offset from the TGBA. The TCDBA must be 16-byte aligned, so bits 28–31 must be 0000.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TACT	TCONV		*	*	*	*	*	TCDBA																							
			0	0	0	0	0																					0	0	0	0

* = Reserved. Write 0 for future compatibility

Note: Bits 28–31 are unknown after reset.
For correct operation, you
must write 0s to these bits.

TDM

TDM[0–3]RIER

TDM 0–3 Receive Interrupt Enable Registers

TDM_____________ (enter number = *n*) RIER

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F78 + (16384 × *n*) = ___________

DSI Address: 0x183F78 + (16384 × *n*) = _______________

Reset: 0x00000000
Read/Write

RSTEE – Receive Second Threshold Event Enable, Bit 31	
0	Receive second threshold interrupt disabled
1	Receive second threshold interrupt enabled

RFTEE – Receive First Threshold Event Enable, Bit 30	
0	Receive first threshold interrupt disabled
1	Receive first threshold interrupt enabled

OLBEE – Overrun Local Buffer Event Enable, Bit 29	
0	Overrun local buffer interrupt masked
1	Overrun local buffer interrupt enabled

RSEEE – Receive Sync Error Event Enable, Bit 28	
0	Receive sync error interrupt masked
1	Receive sync error interrupt enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RSEEE	OLBEE	RFTEE	RSTEE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

* = Reserved. Write 0 for future compatibility

TDM[0–3]TIER

TDM 0–3 Transmit Interrupt Enable Registers

TDM_____________ (enter number = *n*) TIER

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F70 + (16384 × *n*) = ___________

DSI Address: 0x183F70 + (16384 × *n*) = _______________

Reset: 0x00000000
Read/Write

TSTEE – Transmit Second Threshold Event Enable, Bit 31	
0	Transmit second threshold interrupt disabled
1	Transmit second threshold interrupt enabled

TFTEE – Transmit First Threshold Event Enable, Bit 30	
0	Transmit first threshold interrupt disabled
1	Transmit first threshold interrupt enabled

ULBEE – Underrun Local Buffer Event Enable, Bit 29	
0	Underrun local buffer interrupt masked
1	Underrun local buffer interrupt enabled

TSEIE – Transmit Sync Error Interrupt Enable, Bit 28	
0	Transmit sync error interrupt masked
1	Transmit sync error interrupt enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TSEIE	ULBEE	TFTEE	TSTEE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]RER

TDM 0–3 Receive Event Registers

TDM_____________ (enter number = *n*) RER

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F40 + (16384 × *n*) = ___________

DSI Address: 0x183F40 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

RSTE – Receive Second Threshold Event, Bit 31

0	No receive second threshold event occurred
1	Receive second threshold event occurred

RFTE – Receive First Threshold Event, Bit 30

0	No receive first threshold event occurred
1	Receive first threshold event occurred

OLBE – Overrun Local Buffer Event, Bit 29

0	No overrun local buffer event occurred
1	Overrun local buffer event occurred

RSE – Receive Sync Error Event Enable, Bit 28

0	No receive sync error occurred
1	Receive sync error occurred

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RSE	OLBE	RFTE	RSTE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

* = Reserved. Write 0 for future compatibility

TDM[0–3]TER

TDM 0–3 Transmit Event Registers

TDM_____________ (enter number = *n*) TER

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F38 + (16384 × *n*) = ___________

DSI Address: 0x183F38 + (16384 × *n*) = ______________

Reset: 0x00000000

Read/Write

TSTE – Transmit Second Threshold Event, Bit 31

0	No transmit second threshold event occurred
1	Transmit second threshold event occurred

TFTE – Transmit First Threshold Event, Bit 30

0	No transmit first threshold event occurred
1	Transmit first threshold event occurred

ULBE – Underrun Local Buffer Event, Bit 29

0	No underrun local buffer event occurred
1	Underrun local buffer event occurred

TSE – Transmit Sync Error, Bit 28

0	No transmit sync error occurred
1	Transmit sync error occurred

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TSE	ULBE	TFTE	TSTE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

* = Reserved. Write 0 for future compatibility

TDM

TDM[0–3]ASR

TDM 0–3 Adaptation Status Registers

TDM_____________ (enter number = n) ACR

Local Bus Address: ____________ See **page 8-28**

QBus (IPBus) Address: 0x01F83F30 + (16384 × n) = _______________

DSI Address: 0x183F30 + (16384 × n) = _______________

Reset: 0x00000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	AMS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

* = Reserved. Always 0.

AMS – Adaptation Machine Status, Bit 31	
0	No sync arrived and TDM[0–3]ASDR was not updated
1	Sync arrived and TDM[0–3]ASDR was updated

Note: Write a 1 to this bit to clear it. Writing a 0 has no effect.

Serial Communication Interface (SCI)

SCIBR

SCI Baud-Rate Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBD000
DSI Address: 0x1BD000

Reset: 0x00000000
Read/Write

SBR – SCI Baud Rate, Bits 19–31
Value = SCI clock/(16 × required baud rate) Value is in the range 1–8191.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	SBR												
0	0	0													

* = Reserved. Write 0 for future compatibility

SCIDR

SCI Data Register

Local Bus Address: ___________
See **page 8-28**
QBus (IPBus) Address: 0x01FBD018
DSI Address: 0x1BD018
Reset: 0x00000000
Read/Write

T8 – Transmit Bit 8, Bit 17
Write-only ninth bit when the SCI is in 9-bit data mode

R8 – Receive Bit 8, Bit 16
Read-only ninth bit when the SCI is in 9-bit data mode

T[7–0] – Transmit Bits 7–0, Bits 24–31
Write-only transmit bits for all modes Note: When read, these are R[7–0].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R8	T8	*	*	*	*	*	*	T[7–0]							
		0	0	0	0	0	0								

* = Reserved. Write 0 for future compatibility

SCIDDR

SCI Data Direction Register

Local Bus Address: ___________See **page 8-28**
QBus (IPBus) Address: 0x01FBD028
DSI Address: 0x1BD028
Reset: 0x00000000
Read/Write

DDRTX – Data Direction TX, Bit 22	
0	TXD not driven when transmitter disabled or loop enabled
1	If enabled, the transmitter drives TXD; otherwise TXD = 0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	DDRTX	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0		0	0	0	0	0	0	0	0	0

* = Reserved. Write 0 for future compatibility

Serial Communication Interface (SCI)

SCICR

SCI Control Register 1

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBD008
DSI Address: 0x1BD008
Reset: 0x00000000
Read/Write

PT – Parity Type, Bit 23	
0	Even parity
1	Odd parity

PE – Parity Enable, Bit 22	
0	Parity function disabled
1	Parity function enabled

ILT – Idle Line Type, Bit 21	
0	Idle character bit count begins after start bit
1	Idle character bit count begins after stop bit

WAKE – Wake, Bit 20	
0	Idle line wake-up
1	Address mark wake-up

M – Data Format Mode, Bit 19	
0	1 start bit, 8 data bits, 1 stop bit
1	1 start bit, 9 data bits, 1 stop bit

RSRC – Receiver Source, Bit 18	
0	Receiver input connected to transmitter output internally
1	Receiver input connected to TXD

LOOPS – Loop Select, Bit 16	
0	Normal operation enabled
1	Loop operation enabled

SBK – Send Break, Bit 31	
0	No break characters
1	Transmit break characters

RWU – Receiver Wake-Up, Bit 30	
0	Normal operation
1	Receiver Wake-Up from Standby state

RE – Receiver Enable, Bit 29	
0	Receiver disabled
1	Receiver enabled

TE – Transmitter Enable, Bit 28	
0	Transmitter disabled
1	Transmitter enabled

ILIE – Idle Line Interrupt Enable, Bit 27	
0	IDLE interrupt source disabled
1	IDLE interrupt source enabled

RIE – Receiver Full Interrupt Enable, Bit 26	
0	RDRF and OR interrupt sources disabled
1	RDRF and OR interrupt sources enabled

TCIE – Transmission Complete Interrupt Enable, Bit 25	
0	TC interrupt source disabled
1	TC interrupt source enabled

TIE – Transmitter Interrupt Enable, Bit 24	
0	TDRE interrupt source disabled
1	TDRE interrupt source enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	LOOPS	*	RSRC	M	WAKE	ILT	PE	PT	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0														

* = Reserved. Write a 0 to all reserved bits for future compatibility

Timers

TGCRA

Timer General Configuration Register A

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF380
DSI Address: 0x1BF380

Reset: 0x00000000
Read/Write

DIR4 – TIMER1 Pin Direction, Bit 26	
0	TIMER1 is input
1	TIMER1 is output

TOG4 – TIMER1 Pulse/Toggle, Bit 25	
0	TIMER1 output toggles
1	TIMER1 output is asserted for one clock

POL4 – TIMER1 Polarity, Bit 24	
0	TIMER1 (Timer A4) output signal polarity unchanged
1	TIMER1 (Timer A4) output signal polarity inverted

DIR0 – TIMER0 Pin Direction, Bit 31	
0	TIMER0 is input
1	TIMER0 is output

TOG0 – TIMER0 Pulse/Toggle, Bit 30	
0	TIMER0 output toggles
1	TIMER0 output is asserted for one clock

POL0 – TIMER0 Polarity, Bit 29	
0	TIMER0 (Timer A0) output signal polarity unchanged
1	TIMER0 (Timer A0) output signal polarity inverted

INTP – Interrupt Pulse/Level, Bit 28	
0	Timer A interrupts asserted for one clock
1	Timer A interrupts are level-type

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	POL4	TOG4	DIR4	*	INTP	POL0	TOG0	DIR0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0				

* = Reserved. Write a 0 to all reserved bits for future compatibility

MSC8113 Reference Manual, Rev. 0

Timers

TGCRB

Timer General Configuration Register B

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF780
DSI Address: 0x1BF780

Reset: 0x00000000
Read/Write

TOG0 – TIMER2 Pulse/Toggle, Bit 30	
0	TIMER2 output toggles
1	TIMER2 output is asserted for one clock

POL0 – TIMER2 Polarity, Bit 29	
0	TIMER2 (Timer A0) output signal polarity unchanged
1	TIMER2 (Timer A0) output signal polarity inverted

INTP – Interrupt Pulse/Level, Bit 28	
0	Timer B interrupts asserted for one clock
1	Timer B interrupts are level-type

TOG4 – TIMER3 Pulse/Toggle, Bit 25	
0	TIMER3 output toggles
1	TIMER3 output is asserted for one clock

POL4 – TIMER3 Polarity, Bit 24	
0	TIMER3 (Timer B4) output signal polarity unchanged
1	TIMER3 (Timer B4) output signal polarity inverted

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	POL4	TOG4	*	*	INTP	POL0	TOG0	*
0	0	0	0	0	0	0	0			0	0				0

* = Reserved. Write a 0 to all reserved bits for future compatibility

Timers

TIERA

Timer Interrupt Enable Register A
Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF390
DSI Address: 0x1BF390
Reset: 0x000000000
Read/Write

IE[0–15] – Timer A Interrupt Enables, Bits 16–31	
0	Timer An interrupt disabled
1	Timer An interrupt enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**																

* = Reserved. Write to 0 for future compatibility

TIERB

Timer Interrupt Enable Register B
Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF790
DSI Address: 0x1BF790
Reset: 0x000000000
Read/Write

IE[0–15] – Timer B Interrupt Enables, Bits 16–31	
0	Timer Bn interrupt disabled
1	Timer Bn interrupt enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**																

* = Reserved. Write to 0 for future compatibility

Timers

TCFRA[0–15]

Timer Configuration Register A for Timers 0–15

TCFRA____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF000 + (8 × *n*) = ______________

DSI Address: 0x1BF000 + (8 × *n*) = ______________

Reset: 0x000000000
Read/Write

CYC – Cyclic/One-Shot, Bit 31	
0	Timer operates in one-shot mode
1	Timer operates in cycle mode

IPOL – Input Clock Polarity, Bit 30	
0	Counter changes on clock rising edge
1	Counter changes on clock falling edge

INSEL – Input Select, Bits 25–28	
0000	Timer receives its clock from TIMER0
0001	Timer receives its clock from TIMER1
0010	Timer receives its clock from TDM0RCLK
0011	Timer receives its clock from TDM1RCLK
0100	Timer receives its clock from TDM0TCLK
0101	Timer receives its clock from TDM1TCLK
0110	Timer receives its clock from the internal BUSES_CLOCK
0111	Reserved
1000	Timer receives its clock from the output of timer A8
1001	Timer receives its clock from the output of timer A9
1010	Timer receives its clock from the output of timer A10
1011	Timer receives its clock from the output of timer A11
1100	Timer receives its clock from the output of timer A12
1101	Timer receives its clock from the output of timer A13
1110	Timer receives its clock from the output of timer A14
1111	Timer receives its clock from the output of timer A15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	INSEL				*	IPOL	CYC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0		

* = Reserved. Write to 0 for future compatibility

Timers

TCFRB[0–15]

Timer Configuration Register B for Timers 0–15

TCFRB____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF400 + (8 × *n*) = ______________

DSI Address: 0x1BF400 + (8 × *n*) = ______________

Reset: 0x000000000

Read/Write

CYC – Cyclic/One-Shot, Bit 31	
0	Timer operates in one-shot mode
1	Timer operates in cycle mode

IPOL – Input Clock Polarity, Bit 30	
0	Counter changes on clock rising edge
1	Counter changes on clock falling edge

INSEL – Input Select, Bits 25–28	
0000	Timer receives its clock from TIMER2
0001	Timer receives its clock from TIMER3
0010	Timer receives its clock from TDM2RCLK
0011	Timer receives its clock from TDM3RCLK
0100	Timer receives its clock from TDM2TCLK
0101	Timer receives its clock from TDM3TCLK
0110	Timer receives its clock from the internal BUSES_CLOCK
0111	Reserved
1000	Timer receives its clock from the output of timer B8
1001	Timer receives its clock from the output of timer B9
1010	Timer receives its clock from the output of timer B10
1011	Timer receives its clock from the output of timer B11
1100	Timer receives its clock from the output of timer B12
1101	Timer receives its clock from the output of timer B13
1110	Timer receives its clock from the output of timer B14
1111	Timer receives its clock from the output of timer B15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	INSEL				*	IPOL	CYC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0		

* = Reserved. Write to 0 for future compatibility

Timers

TCMPA[0–15]

Timer Compare Register A for Timers 0–15

TCMPA_____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF080 + (8 × *n*) = ______________
DSI Address: 0x1BF080 + (8 × *n*) = ______________

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	COMPVAL															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

COMPVAL – Compare Value, Bits 16–31

Contains the value to compare to the counter value.

TCMPB[0–15]

Timer Compare Register B for Timers 0–15

TCMPB_____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF480 + (8 × *n*) = ______________
DSI Address: 0x1BF480 + (8 × *n*) = ______________

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	COMPVAL															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

COMPVAL – Compare Value, Bits 16–31

Contains the value to compare to the counter value.

Timers

TCRA[0–15]

Timer Control Register A for Timers 0–15

TCRA____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF100 + (8 × *n*) = ______________

DSI Address: 0x1BF100 + (8 × *n*) = ______________

Reset: 0x000000000
Read/Write

TE – Timer Enable, Bit 31	
0	Timer disabled
1	Timer enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

* = Reserved. Write to 0 for future compatibility

TCRB[0–15]

Timer Control Register B for Timers 0–15

TCRB____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-28**

QBus (IPBus) Address: 0x01FBF500 + (8 × *n*) = ______________

DSI Address: 0x1BF500 + (8 × *n*) = ______________

Reset: 0x000000000
Read/Write

TE – Timer Enable, Bit 31	
0	Timer disabled
1	Timer enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TE
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

* = Reserved. Write to 0 for future compatibility

Timers

TERA

Timer Event Register A

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF388
DSI Address: 0x1BF388
Reset: 0x000000000
Read/Write

CF[0–15] – Timer A Compare Flags, Bits 16–31	
0	Timer An ≠ TCMPAn[COMPVAL]
1	Timer An = TCMPAn[COMPVAL]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

TERB

Timer Event Register B

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBF788
DSI Address: 0x1BF788
Reset: 0
Read/Write

CF[0–15] – Timer B Compare Flags, Bits 16–31	
0	Timer Bn ≠ TCMPBn[COMPVAL]
1	Timer Bn = TCMPBn[COMPVAL]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

General-Purpose Input/Output (GPIO)

PODR

Pin Open-Drain Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC200
DSI Address: 0x1BC200

Reset: 0x000000000
Read/Write

OD[31–0] – Open Drain, Bits 0–31	
0	Pin driven actively as output
1	Pin driven by open-drain driver

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OD31	OD30	OD29	OD28	OD27	OD26	OD25	OD24	OD23	OD22	OD21	OD20	OD19	OD18	OD17	OD16	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0

PDAT

Pin Data Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC208
DSI Address: 0x1BC208

Reset: 0x000000000
Read/Write

D[31–0] – Data, Bits 0–31	
0	Pin value is 0
1	Pin value is 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

PDIR

Pin Data Direction Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC210
DSI Address: 0x1BC210

Reset: 0x000000000
Read/Write

DR[31–0] – Direction, Bits 0–31	
0	Pin is an input
1	Pin is an output

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DR31	DR30	DR29	DR28	DR27	DR26	DR25	DR24	DR23	DR22	DR21	DR20	DR19	DR18	DR17	DR16	DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

General-Purpose Input/Output (GPIO)

PAR

Pin Assignment Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC218
DSI Address: 0x1BC218

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DD31	DD30	DD29	DD28	DD27	DD26	DD25	DD24	DD23	DD22	DD21	DD20	DD19	DD18	DD17	DD16	DD15	DD14	DD13	DD12	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0

DD[31–0] – Dedicated, Bits 0–31	
0	Pin is GPIO
1	Pin is dedicated function

PSOR

Pin Special Options Register

Local Bus Address: ___________ See **page 8-28**
QBus (IPBus) Address: 0x01FBC220
DSI Address: 0x1BC220

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SO31	SO30	SO29	SO28	SO27	SO26	SO25	SO24	SO23	SO22	SO21	SO20	SO19	SO18	SO17	SO16	SO15	SO14	SO13	SO12	SO11	SO10	SO9	SO8	SO7	SO6	SO5	SO4	SO3	SO2	SO1	SO0

SO[31–0] – Data, Bits 0–31	
0	Dedicated peripheral function 1
1	Dedicated peripheral function 2

Ethernet Controller

IEVENT

Interrupt Event Register

Local Bus Address: ___________ See **page 8-36**
QBus (IPBus) Address: 0x01FB8010
DSI Address: 0x1B8010
Reset: 0x00000000
Read/Write

TXC – Transmit Control Interrupt, Bit 8	
0	Even parity
1	Odd parity

BABT – Babbling Transmit Error, Bit 7	
0	Normal operation
1	Babbling transmit error

GTSC – Graceful Transmit Stop Complete, Bit 6	
0	No interrupt
1	Graceful stop complete interrupt

MSRO – MSTAT Register Overflow, Bit 5	
0	No MSTAT Register overflow
1	MSTAT Register overflow

EBERR – Ethernet Bus Error, Bit 3	
0	No bus error
1	System bus error

BSY – Busy Condition Interrupt, Bit 2	
0	No frame discarded
1	Frame discarded

RXC – Receive Control Interrupt, Bit 1	
0	No control frame
1	Control frame received

TXE – Transmit Error, Bit 9	
0	No transmit error
1	Transmit error halted the transmitter

TXB – Transmit Buffer, Bit 10	
0	Normal operation
1	Transmit buffer

TXF – Transmit Frame Interrupt, Bit 11	
0	No frame transmitted
1	Frame transmitted and last BD updated

IE – Insertion Error, Bit 12	
0	No insertion error
1	Insertion error

LC – Late Collision, Bit 13	
0	No collision
1	Collision occurred beyond collision window

CRL – Collision Retry Limit, Bit 14	
0	No excessive transmission collisions
1	Successive transmissions exceeded maximum count.

XFUN – Transmit FIFO Underrun, Bit 15	
0	No underrun
1	Transmit FIFO underrun

RXB[0–3] – Receive Buffer 0–3, Bits 16–19	
0	Normal operation
1	Receive buffer for queue n

GRSC – Graceful Receive Stop Complete, Bit 23	
0	No graceful receive stop
1	Graceful receive stop completed

RXF[0–3] – Receive Frame Interrupt 0–3, Bits 24–27	
0	No receive interrupt for queue n
1	Receive frame interrupt for queue n

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	RXC	BSY	EBERR	*	MSCO	GTSC	BABT	TXC	TXE	TXB	TXF	IE	LC	CRL	XFUN
0				0											

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RXB0	RXB1	RXB2	RXB3	*	*	*	GRSC	RXF0	RXF1	RXF2	RXF3	*	*	*	*
				0	0	0						0	0	0	0

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

IMASK

Interrupt Mask Register

Local Bus Address: ___________ See **page 8-36**
QBus (IPBus) Address: 0x01FB8014
DSI Address: 0x1B8014
Reset: 0x00000000
Read/Write

TXCEN – Transmit Control Interrupt Enable, Bit 8	
0	Disabled
1	Enabled

BTEN – Babbling Transmitter Interrupt Enable, Bit 7	
0	Disabled
1	Enabled

GTSCEN – Graceful Tx Stop Complete Int. En., Bit 6	
0	Disabled
1	Enabled

MSROEN – MSTAT Register Overflow Int. En., Bit 5	
0	Disabled
1	Enabled

EBERREN – Ethernet Bus Error Enable, Bit 3	
0	Disabled
1	Enabled

BSYEN – Busy Interrupt Enable, Bit 2	
0	Disabled
1	Enabled

RXCEN – Receive Control Interrupt Enable, Bit 1	
0	Disabled
1	Enabled

TXEEN – Transmit Error Interrupt Enabled, Bit 9	
0	Disabled
1	Enabled

TXBEN – Transmit Buffer Interrupt Enabled, Bit 10	
0	Disabled
1	Enabled

TXFEN – Transmit Frame Interrupt Enable, Bit 11	
0	Disabled
1	Enabled

IEEN – Insertion Error Interrupt Enable, Bit 12	
0	Disabled
1	Enabled

LCEN – Late Collision Enable, Bit 13	
0	Disabled
1	Enabled

CRLEN – Collision Retry Limit Enable, Bit 14	
0	Disabled
1	Enabled

XFUNEN – Transmit FIFO Underrun Int. Enable, Bit 15	
0	Disabled
1	Enabled

RXBEN[0–3] – Receive Buffer Queue 0–3 Int. En., Bits 16–19	
0	Disabled
1	Enabled

GRSCEN – Graceful Receive Stop Complete, Bit 23	
0	Disabled
1	Enabled

RXFEN[0–3] – Receive Frame Queue 0–3 Int. En., Bits 24–27	
0	Disabled
1	Enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	RXCEN	BSYEN	EBERREN	* 0	MSROEN	GTSCEN	BTEN	TXCEN	TXEEN	TXBEN	TXFEN	IEEN	LCEN	CRLEN	XFUNEN

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RXBEN0	RXBEN1	RXBEN2	RXBEN3	* 0	* 0	* 0	GRSCEN	RXENF0	RXFEN1	RXFEN2	RXFEN3	* 0	* 0	* 0	* 0

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

ECNTRL

Ethernet Control Register

Local Bus Address: ___________ See **page 8-36**
QBus (IPBus) Address: 0x01FB8020
DSI Address: 0x1B8020

Reset: 0x000000000
Read/Write

CLRCNT – Clear All Statistics Counters, Bit 17	
0	MSTAT counters update normally
1	Reset all MSTAT counters

AUTOZ – Zero Written Automatically to Counter, Bit 18	
0	User must write a zero to the counter after read
1	System writes a zero automatically after read

STEN – Statistics Enabled, Bit 19	
0	Statistics disabled
1	Statistics enabled

DBDS – Data Buffer Descriptor Size, Bit 24	
0	8-byte BD format
1	32-byte BD format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	CLRCNT	AUTOZ	STEN	* 0	* 0	* 0	* 0	DBDS	* 0	* 0	* 0	* 0	* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility

MINFLR

Minimum Framelength Register

Local Bus Address: ___________ See **page 8-36**
QBus (IPBus) Address: 0x01FB8024
DSI Address: 0x1B8024

Reset: 0x000000000
Read/Write

MINFLR – Min. Rx Frame Length, Bits 25–31
Specifies the smallest packet to place into a receive buffer specified by RxBD.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	MINFLR						

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

PTV

Pause Time Value Register

Local Bus Address: ___________ See **page 8-36**

QBus (IPBus) Address: 0x01FB8028
DSI Address: 0x1B8028

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PTE																PT															

PTE – Extended Pause Control, Bits 0–15
Specifies a 16-bit additional control parameter for the pause frame if TCTRL[TFCP] = 1. **Note**: **IEEE** Std 802.3 requires this value to be 0.

PT – Pause Time Value, Bits 16–31
Specifies the pause time used when TCTRL[TFCP] = 1. The period is measured in pause_quanta.

* = Reserved. Write to 0 for future compatibility

DMACTRL

DMA Control Register

Local Bus Address: ___________ See **page 8-36**

QBus (IPBus) Address: 0x01FB802C
DSI Address: 0x1B802C

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	GRS	GTS	*	**	WOP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	1	

GRS – Graceful Receive Stop, Bit 27	
0	Stop receiving after current frame
1	Resume receiving frames

GTS – Graceful Transmit Stop, Bit 28	
0	Resume transmitting frames
1	Stop transmitting after current frame

WOP – Wait or Poll, Bit 31	
0	Poll TxBD per DMAMR[PCNT] value
1	Do not poll, wait for TSTAT[THLT] write

* = Reserved. Write to 0 for future compatibility
** = Reserved. Always write a 1 to this bit after any reset or configuration.

Ethernet Controller

DMAMR

DMA Maintenance Register

Local Bus Address: ___________ See **page 8-36**

QBus (IPBus) Address: 0x01FB8038
DSI Address: 0x1B8038

Reset: 0x000D00000
Read/Write

PCNT – Polling Count, Bits 4–5	
00	512 clocks
01	257 clocks
10	128 clocks
11	64 clocks

APR – Alarm Mode Priority, Bits 12–13	
00	Low priority
01	Mid priority
10	Mid priority
11	High priority

DOOS – Disable Out-of Sequence BD, Bit 7	
0	Buffer enabled
1	Buffer disabled

BDPR – BD Transaction Priority, Bits 14–15	
00	Low priority
01	Mid priority
10	Mid priority
11	High priority

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	PCNT		*	DOOS	*	*	*	*	APR		BDPR	
0	0	0	0			0		0	0	0	0				

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

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

FRXSTATR

FIFO Receive Status Register

Local Bus Address: ___________ See **page 8-36**

QBus (IPBus) Address: 0x01FB8048
DSI Address: 0x1B8048

Reset: 0x000000002
Read/Write

EMPTY – Rx FIFO Empty, Bit 30	
0	Rx FIFO not empty
1	Rx FIFO empty

FULL – Rx FIFO Full, Bit 29	
0	Rx FIFO not full
1	Rx FIFO full

PFS – Pause Frame Sent, Bit 27	
0	No pause frame sent
1	Pause frame sent

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	PFS	*	FULL	EMPTY	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0			0

* = Reserved. Write to 0 for future compatibility

FRXCTRLR

FIFO Receive Control Register

Local Bus Address: ___________ See **page 8-36**

QBus (IPBus) Address: 0x01FB804C
DSI Address: 0x1B804C

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	**	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

* = Reserved. Write to 0 for future compatibility
** = Reserved. Write to 1 after any reset or configuration

Ethernet Controller

FRXALAR

FIFO Receive Alarm Register

Local Bus Address: ___________ See **page 8-36**
QBus (IPBus) Address: 0x01FB8050
DSI Address: 0x1B8050

Reset: 0x000000100
Read/Write

FRXALAR – FIFO Rx Alarm, Bits 25–31

Contains the value used to trigger the receive alarm function.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FRXALAR								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

FRXSHR

FIFO Receive Alarm Shutoff Register

Local Bus Address: ___________ See **page 8-37**
QBus (IPBus) Address: 0x01FB8054
DSI Address: 0x1B8054

Reset: 0x000000100
Read/Write

FRXSH – FIFO Rx Alarm Shutoff, Bits 23–31

Contains the value used to turn off the alarm state.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FRXSH								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

FRXPAR

FIFO Receive Panic Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8058
DSI Address: 0x1B8058

Reset: 0x000000180
Read/Write

FRXPA – FIFO Rx Panic, Bits 25–31

Contains the value used to trigger the panic function.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FRXPA								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

FRXPSR

FIFO Receive Panic Shutoff Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB805C
DSI Address: 0x1B805C

Reset: 0x000000100
Read/Write

FRXPS – FIFO Rx Panic Shutoff, Bits 23–31

Contains the value used to turn off the panic state.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FRXSH								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

FTXSTATR

FIFO Transmit Status Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8078
DSI Address: 0x1B8078

Reset: 0x000000002
Read/Write

EMPTY – Tx FIFO Empty, Bit 30	
0	Tx FIFO not empty
1	Tx FIFO empty

FULL – Tx FIFO Full, Bit 29	
0	Tx FIFO not full
1	Tx FIFO full

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FULL	EMPTY	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0

* = Reserved. Write to 0 for future compatibility

FRXTHR

FIFO Transmit Threshold Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB808C
DSI Address: 0x1B808C

Reset: 0x000000100
Read/Write

FTT – FIFO Tx Threshold, Bits 23–31
Specifies the number of FIFO Tx entries that trigger frame data unloading to the MAC.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FTT								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

FTXSPR

FIFO Transmit Space Available Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8094
DSI Address: 0x1B8094
Reset: 0x000000010
Read/Write

FTXSP – FIFO Tx Space Available, Bits 25–31

Contains a value to define available space in the transmit FIFO. Before using half duplex mode in MII, write 0x25 to this field.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FTXSP								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

FTXSR

FIFO Transmit Starve Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8098
DSI Address: 0x1B8098
Reset: 0x000000100
Read/Write

FTXS – FIFO Tx Starve, Bits 23–31

Contains the value used to trigger the starve function.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FRXSH								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

FTXSSR

FIFO Transmit Starve Shutoff Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB809C
DSI Address: 0x1B809C
Reset: 0x000000100
Read/Write

FTXSS – FIFO Tx Starve Shutoff, Bits 23–31

Contains the value used to turn off the panic state.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	FTXSS								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TCTRL

Transmit Control Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8100
DSI Address: 0x1B8100

Reset: 0x000000000
Read/Write

TFCP – Tx Flow Control Pause Frame, Bit 28	
0	No pause
1	Stop data frame Tx for specified time

RFCP – Rx Flow Control Pause Frame, Bit 27	
0	No flow control pause frame
1	Flow control pause frame received

THDF – Tx Half Duplex Flow, Bit 20	
0	Disable back pressure
1	Back pressure applied to media

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	THDF	*	*	*	*	*	*	RFCP	TFCP	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	0

* = Reserved. Write to 0 for future compatibility

TSTAT

Transmit Status Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8104
DSI Address: 0x1B8104
Reset: 0x000000000
Read/Write

THLT – Tx Halt, Bit 0	
0	No hardware Tx halt
1	Ethernet controller halted

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
THLT	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TBDLEN

TxBD Data Length Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB810C
DSI Address: 0x1B810C
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TBDLEN															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

TBDLEN – TxBD Data Length, Bits 16–31

Contains a value to define the length of the transmit or insert buffer.

CTBPTR

Current TxBD Pointer

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8124
DSI Address: 0x1B8124
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CTBPTR																													*	*	*
																													0	0	0

* = Reserved. Write to 0 for future compatibility

CTBPTR – Current TxBD Pointer, Bits 0–28

Contains the pointer to the current TxBD.

Ethernet Controller

TBPTR

TxBD Pointer

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8184
DSI Address: 0x1B8184
Reset: 0x000000000
Read/Write

TBPTR – TxBD Pointer, Bits 0–28

Used by the DMA controller to point to the TxBD.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TBPTR																													* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility

TBASE

TxBD Base Address Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB8204
DSI Address: 0x1B8204
Reset: 0x000000000
Read/Write

TBASE – TxBD Base Address, Bits 0–28

Contains a value to define the base address for the Ethernet controller TxBDs.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TBASE																													* 0	* 0	* 0

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

OSTBD

Out-of-Sequence TxBD Register

Local Bus Address: ___________ See **page 8-37**
QBus (IPBus) Address: 0x01FB82B0
DSI Address: 0x1B82B0

Reset: 0x08000000
Read/Write

DEF – Defer Indication, Bit 6	
0	Current frame is not deferred
1	Current is late because it was deferred

TC – Tx Cyclic Redundancy Check, Bit 5	
0	End Tx after last byte unless TxBD[PAD] = 1
1	Transmit the CRC after the last data byte

L – Last in Frame, Bit 4

Hardwired to 1, because the OSTBD is always last in frame

I – Interrupt, Bit 3	
0	Do not generate interrupt after buffer is serviced
1	IEVENT[TXF} is set after buffer is serviced

W – Wrap, Bit 2	
0	Next TxBD is consecutive
1	Next TxBD is at user-defined location

PAD – Padding for Short Frames, Bit 1	
0	Do not add padding unless TxBD[TC] is set
1	Add padding to short frames

R – Ready, Bit 0	
0	TxBD is not ready for transmission
1	Prepared data buffer is ready/being transmitted

LC – Late Collision, Bit 8	
0	No late collision
1	Collision after 64 bytes

RL – Retransmission Limit, Bit 9	
0	Tx has not exceeded limit
1	Tx failed due to repeated collisions, exceeded limit

RC – Retry Count, Bits 10–13

Indicates the number of retries necessary to send the frame.
A zero indicates that the frame was sent correctly the first time.

UN – Underrun, Bit 14	
0	No underrun
1	Transmitter underrun detected

OSTBDLEN – Out-of-Sequence TxBD Data Length, Bits 16–31

Contains a the number of octets that the Ethernet controller should transmit from the BD data buffer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PAD	W	I	L	TC	DEF	*	LC	RL	RC				UN	*	OSTBDLEN															
				1			0								0																

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

OSTBDP

Out-of-Sequence Tx Data Buffer Pointer

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB82B4
DSI Address: 0x1B82B4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OSTBDP																															

* = Reserved. Write to 0 for future compatibility

OSTBDP – Out-of-Sequence Tx Data Buffer Pointer, Bits 0–31

Contains the address of the associated data buffer.

OS32TBDP

Out-of-Sequence 32 Byte TxData Buffer Pointer Register

Local Bus Address: ___________ See **page 8-37**

QBus (IPBus) Address: 0x01FB82B8
DSI Address: 0x1B82B8
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OS32TBDP																															

* = Reserved. Write to 0 for future compatibility

OS32TBDP – Out-of-Sequence 32 Byte Tx Data Buffer Pointer, Bits 0–31

Contains the address of the associated data buffer.

Ethernet Controller

OS32IPTR

Out-of-Sequence 32 Byte Tx BD Pointer Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB82C0
DSI Address: 0x1B82C0
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OS32IPTR																															

* = Reserved. Write to 0 for future compatibility

OS32IPTR – Out-of-Sequence Tx Insert Buffer Pointer, Bits 0–31
Contains the address of the buffer to contain the inserted data.

OS32TBDR

Out-of-Sequence 32 Byte TxBD Reserved Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB82C4
DSI Address: 0x1B82C4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IE	IT	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

* = Reserved. Write to 0 for future compatibility

IT – Insertion Type, Bits 30–31	
00	No insertion
01	Replacement
10	Expansion
11	Reserved

IE – Insertion Error, Bit 29	
0	No insertion error
1	Error during insertion

Ethernet Controller

OS32IIL

Out-of-Sequence 32 Byte Tx BD Insert/Length Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB82C8
DSI Address: 0x1B82C8
Reset: 0x000000000
Read/Write

OS32INX – Out-of-Sequence 32 Byte TxBD Insert Index, Bits 0–15

Contains the number of bytes to jump in the TxBD before inserting data.

OS32ILEN – Out-of-Sequence 32 Byte TxBD Insert Length, Bits 0–31

Contains the address of the buffer to contain the inserted data.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OS32INX																OS32ILEN															

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RCTRL

Receive Control Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB8300

DSI Address: 0x1B8300

Reset: 0x000000000

Read/Write

BCREJ – Broadcast Frame Reject, Bit 27	
0	No effect
1	Reject frames with address 0xFFFFFFFFFFFF

PROM – Promiscuous Mode, Bit 28	
0	No effect
1	Accept all frames

PMEN – Pattern Match Enabled, Bit 25	
0	Pattern match disabled
1	Pattern match enabled

RSF – Receive Short Frame Mode, Bit 29	
0	No effect
1	Rx frames shorter than MINFLR

FA – Reject All Mode, Bit 30	
0	No effect
1	No frames accepted for DA hit

CRCLSEL – CRC LSB Select, Bit 20	
0	Use CRC remainder[0–8] to map DA to hash table
1	Use CRC remainder[1–9] to map DA to hash table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	CRCLSEL	*	*	*	*	PMEN	*	BCREJ	PROM	RSF	RA	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0		0					0

* = Reserved. Write to 0 for future compatibility

RSTAT

Receive Status Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB8304

DSI Address: 0x1B8304

Reset: 0x000000000

Read/Write

RHLT – Receive Halted, Bit 0	
0	No receive queues halted
1	At least one receive queue halted

Q0HLT – RxBD Queue 0 Halt, Bit 8	
0	RxBD queue 0 enabled
1	RxBD queue 0 halted

Q1HLT – RxBD Queue 1 Halt, Bit 9	
0	RxBD queue 1 enabled
1	RxBD queue 1 halted

Q2HLT – RxBD Queue 2 Halt, Bit 10	
0	RxBD queue 2 enabled
1	RxBD queue 2 halted

Q3HLT – RxBD Queue 3 Halt, Bit 11	
0	RxBD queue 3 enabled
1	RxBD queue 3 halted

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RHLT	*	*	*	*	*	*	*	Q0HLT	Q1HLT	Q2HLT	Q3HLT	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RBDLEN

RxBD Data Length Register

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB830C

DSI Address: 0x1B830C

Reset: 0x000000000

Read/Write

RBDLEN – RxBD Data Length, Bits 16–31

Contains a value to define the length of the receive buffer. Writing a zero stops all receive channels.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RBDLEN															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

CRBPTRL

Current RxBD Pointer

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB8324

DSI Address: 0x1B8324

Reset: 0x000000000

Read/Write

CTBPTR – Current RxBD Pointer, Bits 0–28

Contains the pointer to the current RxBD.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CRBPTR																													*	*	*
																													0	0	0

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MRBLR0R1

Maximum Receive Buffer Length R0R1 Register

Local Bus Address: ___________ See **page 8-38**
QBus (IPBus) Address: 0x01FB8340
DSI Address: 0x1B8340

Reset: 0x000000000
Read/Write

MRBLR1 – Maximum Rx Buffer Length Ring 1, Bits 0–9

Specifies the number of writes to buffer ring 1.

MRBLR0 – Maximum Rx Buffer Length Ring 0, Bits 16–25

Specifies the number of writes to buffer ring 0.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MRBLR1										*	*	*	*	*	*	MRBLR0										*	*	*	*	*	*
										0	0	0	0	0	0											0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

MRBLR2R3

Maximum Receive Buffer Length R2R3 Register

Local Bus Address: ___________ See **page 8-38**
QBus (IPBus) Address: 0x01FB8344
DSI Address: 0x1B8344

Reset: 0x000000000
Read/Write

MRBLR3 – Maximum Rx Buffer Length Ring 3, Bits 0–9

Specifies the number of writes to buffer ring 3.

MRBLR2 – Maximum Rx Buffer Length Ring 2, Bits 16–25

Specifies the number of writes to buffer ring 2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MRBLR3										*	*	*	*	*	*	MRBLR2										*	*	*	*	*	*
										0	0	0	0	0	0											0	0	0	0	0	0

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RBPTR[0–3]

RxBD Pointer 0–3

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB8384 + (8 × *n*) = ______________

DSI Address: 0x1B8384 + (8 × *n*) = ______________

Reset: 0x00000000
Read/Write

RBPTRn – RxBD Pointer n, Bits 0–28

The value points to the current RxBD.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RBPTRn																													*	*	*
																													0	0	0

* = Reserved. Write 0 for future compatibility

RBASE[0–3]

Receive Descriptor Base Address 0–3

Local Bus Address: ___________ See **page 8-38**

QBus (IPBus) Address: 0x01FB8404 + (8 × *n*) = ______________

DSI Address: 0x1B9404 + (8 × *n*) = ______________

Reset: 0x00000000
Read/Write

RBASEn – Receive Descriptor Base Address n, Bits 0–28

Value is the starting location for the RxBD memory map.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RBASEn																													*	*	*
																													0	0	0

* = Reserved. Write 0 for future compatibility

Ethernet Controller

MACCFG1R

MAC Configuration 1 Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB8500
DSI Address: 0x1B8500
Reset: 0x00000000
Read/Write

TXEN – Transmit Enable, Bit 31	
0	The MAC may not transmit frames from the system
1	The MAC may transmit frames from the system

SYTXEN – Synchronized Tx Enable, Bit 30	
0	Frame transmission is not enabled
1	Frame transmission is enabled

RXEN – Receive Enable, Bit 29	
0	The MAC may not receive frames from the PHY
1	The MAC may receive frames from the PHY

SYRXEN – Synchronized Rx Enable, Bit 28	
0	Frame reception not enabled
1	Frame reception enabled

TXFL – Tx Flow, Bit 27	
0	No transmit pause flow control frames
1	Detect and act on transmit pause control frames

RXFL – Rx Flow, Bit 26	
0	Ignore receive pause control frames
1	Detect and act on receive control frames

MIILB – MII Loopback, Bit 23	
0	Normal operation
1	MII loopback mode

RTXF – Reset Tx Function, Bit 15	
0	Normal operation
1	Reset the transmit function block

RRXF – Reset Rx Function, Bit 14	
0	Normal operation
1	Reset the receive function block

RTXM – Reset Tx MAC, Bit 13	
0	Normal operation
1	Reset the PETMC transmit control block

RRXM – Reset Rx MAC, Bit 12	
0	Normal operation
1	Reset the MAC receive block

SRESET – Soft Reset, Bit 0	
0	Normal operation
1	Soft reset

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SRESET	*	*	*	*	*	*	*	*	*	*	*	RRXM	RTXM	RRXF	RTXF
	0	0	0	0	0	0	0	0	0	0	0				

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	MIILB	*	*	RXFL	TXFL	SYRXEN	RXEN	SYTXEN	TXEN
0	0	0	0	0	0	0		0	0						

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

MACCFG2R

MAC Configuration 2 Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB8504
DSI Address: 0x1B8504

Reset: 0x00007100
Read/Write

FDUP – Full Duplex, Bit 31	
0	Half-duplex mode
1	Full-duplex mode

CRCEN – CRC Enable, Bit 30	
0	Valid frame and valid CRC
1	Frame and/or CRC not valid

PADCRC – Padding/CRC, Bit 29	
0	No padding or CRC
1	MAC pads all Tx short frames and appends CRC

LENC – Length Check, Bit 27	
0	No length field checking
1	MAC checks the frame length field

PREAL – Preamble Length, Bits 16–19

The value specifies the length of the preamble field in bytes. The default value is 0x7. A value of 0x0 is not supported.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	PREAL				*	*	*	**	*	*	*	LENC	*	PADCRC	CRCEN	FDUP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	1	0	0	0		0			

* = Reserved. Write a 0 to all reserved bits for future compatibility
** = Reserved. Write a 1 to all reserved bits for future compatibility

Ethernet Controller

IPGIFGR

Inter-Packet Gap/Inter-Frame Gap Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB8508
DSI Address: 0x1B8508

Reset: 0x40605060
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	NBBIPG1							*	NBBIPG2							MIFGE								*	BBIPG						
0								0																0							

NBBIPG1 – Non-Back-to-Back Inter-Packet Gap 1, Bits 1–7

Defines the optional carrier sense window.

NBBIPG2 – Non-Back-to-Back Inter-Packet Gap 2, Bits 9–15

Represents the non-back-to-back inter-packet gap size in bits.

MIFGE – Minimum IFG Enforcement, Bits 16–23

The minimum number of bits of the IFG to enforce between frames.

BPIPG – Back-to-Back Interpacket Gap, Bits 25–31

The IPG between back-to-back packets.

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

HAFDUPR

Half-Duplex Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB850C
DSI Address: 0x1B850C

Reset: 0x0051F037
Read/Write

CW – Collision Window, Bits 26–31
Specifies the number of frames in the collision window

RTXM – Retransmission Maximum, Bits 16–19
Specifies the number of retransmission attempts after a collision before the packet is aborted

ED – Excess Defer, Bit 15	
0	Abort an excessively deferred packet
1	Transmit an excessively deferred packet

NB – No Back-off, Bit 14	
0	Tx MAC uses the binary exponential back-off rule
1	No back-off

BPNB – Back Pressure No Back-off, Bit 13	
0	Tx MAC uses the binary exponential back-off rule
1	No backoff during a back pressure operation

ABEB – Alternate Binary Exponential Back-off Enable, Bit 12	
0	Tx MAC uses the binary exponential back-off rule
1	Tx MAC uses the alternate binary exponential back-off rule

ABEBT – Alternate Binary Exponential Back-off Truncation, Bits 8–12
A value substituted for the Ethernet standard value of 10

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	ABEBT				ABEB	BPNB	NB	ED	RTXM				*	*	*	*	*	*	CW					
0	0	0	0	0	0	0	0													0	0	0	0	0	0						

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

MAXFRMR

Maximum Frame Length Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB8510
DSI Address: 0x1B8510
Reset: 0x000000600
Read/Write

MF – Maximum Frame, Bits 16–31

Contains a value that sets the maximum frame size for receive and transmit channels

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MF															

* = Reserved. Write to 0 for future compatibility

IFSTATR

Interface Status Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB853C
DSI Address: 0x1B853C
Reset: 0x000000001
Read/Write

EXD – Excess Deferral, Bit 22

Set to indicate excessive deferral. Read bit to clear.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	EXD	*	*	*	*	*	*	*	*	**
0	0	0	0	0	0		0	0	0	0	0	0	0	0	1

* = Reserved. Write to 0 for future compatibility
** = Reserved. Write a 1 to all reserved bits for future compatibility

Ethernet Controller

MACSTADDR1R

MAC Station Address Part 1 Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB8540
DSI Address: 0x1B8540

Reset: 0x00000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SA1								SA2								SA3								SA4							

SA1 – Station Address Octet 1, Bits 0–7

Defines the first octet of the station address.

SA2 – Station Address Octet 2, Bits 8–15

Defines the second octet of the station address.

SA3 – Station Address Octet 3, Bits 16–23

Defines the third octet of the station address.

SA4 – Station Address Octet 4, Bits 24–31

Defines the fourth octet of the station address.

Ethernet Controller

MACSTADDR2R

MAC Station Address Part 2 Register

Local Bus Address: ___________ See **page 8-39**
QBus (IPBus) Address: 0x01FB8544
DSI Address: 0x1B8544
Reset: 0x00000000
Read/Write

SA5 – Station Address Octet 5, Bits 0–7

Defines the first octet of the station address.

SA6 – Station Address Octet6, Bits 8–15

Defines the second octet of the station address.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SA5								SA6								*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

MIIMCFGR

MII Management Configuration Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB8520
DSI Address: 0x1B8520

Reset: 0x000000003
Read/Write

MGTCS – Management Clock Select, Bits 29–31	
000	BUSES_CLOCK/4
001	BUSES_CLOCK/4
010	BUSES_CLOCK/6
011	BUSES_CLOCK/8
100	BUSES_CLOCK/10
101	BUSES_CLOCK/14
110	BUSES_CLOCK/20
111	BUSES_CLOCK/28

RMGT – Reset Management, Bit 0	
0	MII management enabled
1	MII management reset

NOPRE – No Preamble, Bit 27	
0	Preamble generated
1	No preamble

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RMGT	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	NOPRE	*	MGTCS		
0	0	0	0	0	0	0	0	0	0	0		0			

* = Reserved. Write to 0 for future compatibility

MIIMCOMR

MII Management Command Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB8524
DSI Address: 0x1B8524
Reset: 0x000000000
Read/Write

RCYC – Read Cycle, Bit 31	
0	Normal operation
1	Perform single read cycle

SCYC – Scan Cycle, Bit 30	
0	Normal operation
1	Continuous read cycles

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	SCYC	RCYC
0	0	0	0	0	0	0	0	0	0	0	0	0	0		

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MIIMADDR

MII Management Configuration Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB8528
DSI Address: 0x1B8528

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	PHYADDR					*	*	*	RADDR				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0					

RADDR – Register Address, Bits 27–31

Management cycle register address field addresses up to 32 registers

PHYADDR – Register Address, Bits 27–31

Management cycle PHY address field addresses up to 31 PHYs

* = Reserved. Write to 0 for future compatibility

MIIMCONR

MII Management Command Register

Local Bus Address: ___________ See **page 8-39**

QBus (IPBus) Address: 0x01FB852C
DSI Address: 0x1B852C
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	PHYC															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

PHYC – PHY Control, Bits 16–31

Contains 16-bit value to write to the register and PHY specified by MIIMADDR

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MIIGSK_CFGR

MIIGSK Configuration Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB9000
DSI Address: 0x1B9000

Reset: 0x000000000
Read/Write

IFMODE – Interface Mode, Bits 30–31	
00	MII mode
01	RMII mode
10	SMII mode
11	Reserved

EMODE – Echo Mode, Bit 28	
0	Normal operation
1	MII inputs looped to Tx outputs

LBMODE – Loopback Mode, Bit 27	
0	Normal operation
1	RMII/SMII outputs looped back to RMII/SMII inputs.

FRCONT – Frequency Control, Bit 25	
0	100 Mbps operation
1	10 Mbps operation

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	FRCONT	*	LBMODE	EMODE	*	IFMODE	
0	0	0	0	0	0	0	0	0		0			0		

* = Reserved. Write to 0 for future compatibility

MIIGSK_GPR

MIIGSK General-Purpose Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB9004
DSI Address: 0x1B9004
Reset: 0x000000000
Read/Write

DS – Drive Strength, Bit 31	
0	Low drive
1	High drive

IR – Ethernet Controller Internal Reset, Bit 27	
0	Normal operation
1	Internal reset

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	IR	*	*	*	DS
0	0	0	0	0	0	0	0	0	0	0		0	0	0	

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MIIGSK_ENR

MIIGSK Enable Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB9008
DSI Address: 0x1B9008

Reset: 0x000000000
Read/Write

EN – Enable, Bit 31	
0	Ethernet controller disabled
1	Ethernet controller enabled

READY – Ready, Bit 30	
0	Ethernet controller not ready
1	Ethernet controller ready for use

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	READY	EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

* = Reserved. Write to 0 for future compatibility

MIIGSK_SMII_SYNCDIR

MIIGSK SMII SYNC Direction Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB900C
DSI Address: 0x1B900C
Reset: 0x000000000
Read/Write

SYNC – SYNC Enable, Bit 31	
0	ETHSYNC output disabled
1	ETHSYNC output enabled

SINC_IN – SYNC_IN Enable, Bit 30	
0	ETHSYNC_IN input disabled
1	ETHSYNC_IN input enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	SYNC_IN	SYNC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MIIGSK_TIFBR

MIIGSK SMII Transmit Inter-Frame Bits Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB9010
DSI Address: 0x1B9010

Reset: 0x000000000
Read/Write

TXD[7–0] – IFG Transmit Segment Data Bits 7–0, Bits 24–31

Part of the 10-bit data segments transferred in the inter-packet frame gap.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

* = Reserved. Write to 0 for future compatibility

MIIGSK_ERIFBR

MIIGSK SMII Expected Receive Inter-Frame Bits Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB9018
DSI Address: 0x1B9018
Reset: 0x000000000
Read/Write

ERXD[7–0] – Expected IFG Receive Segment Bits 7–0, Bits 24–31

The expected values of the received inter-frame segment bits.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	ERXD7	ERXD6	ERXD5	ERXD4	ERXD3	ERXD2	ERXD1	ERXD0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

MIIGSK_IEVENT

MIIGSK SMII Interrupt Event Register

Local Bus Address: ___________ See **page 8-46**

QBus (IPBus) Address: 0x01FB901C
DSI Address: 0x1B901C

Reset: 0x000000000
Read/Write

IE[7–0] – Interrupt Event 7–0, Bits 24–31	
0	No difference
1	Difference detected in bit value

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

* = Reserved. Write to 0 for future compatibility

MIIGSK_IMASK

MIIGSK SMII Interrupt Mask Register

Local Bus Address: ___________ See **page 8-47**

QBus (IPBus) Address: 0x01FB9020
DSI Address: 0x1B9020
Reset: 0x000000000
Read/Write

IE[7–0]EN – Interrupt Event 7–0 Enable, Bits 24–31	
0	Interrupt disabled
1	Interrupt enabled

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IE7EN	IE6EN	IE5EN	IE4EN	IE3EN	IE2EN	IE1EN	IE0EN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TR64

Transmit and Receive 64-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8680
DSI Address: 0x1B8680

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TR64																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TR64 – Tx and Rx 64-Byte Frame Counter, Bits 10–31

The RMON MIB counter that counts the number of 64-byte frames.

TR127

Transmit and Receive 65- to 127-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8684
DSI Address: 0x1B8684
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TR127																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TR127 – Tx and Rx 65- to 127-Byte Counter, Bits 10–31

The RMON MIB counter that counts the number of 65- to 127-byte frames.

Ethernet Controller

TR255

Transmit and Receive 128- to 255-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8688
DSI Address: 0x1B8688

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TR255																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TR255 – Tx and Rx 128- to 255-Byte Frame Counter, Bits 10–31

The RMON MIB counter that counts the number of 128- to 255-byte frames.

TR511

Transmit and Receive 256- to 511-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB868C
DSI Address: 0x1B868C
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TR511																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TR511 – Tx and Rx 255- to 512-Byte Counter, Bits 10–31

The RMON MIB counter that counts the number of 256- to 511-byte frames.

Ethernet Controller

TR1K

Transmit and Receive 512- to 1023-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8690

DSI Address: 0x1B8690

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TR1K																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TR1K – Tx and Rx 512- to 1023-Byte Frame Counter, Bits 10–31

The RMON MIB counter that counts the number of 512- to 1023-byte frames.

TRMAX

Transmit and Receive 1024- to 1518-Byte Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8694

DSI Address: 0x1B8694

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TRMAX																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TRMAX – Tx and Rx 1024- to 1518-Byte Counter, Bits 10–31

The RMON MIB counter that counts the number of 1024- to 1518-byte frames.

Ethernet Controller

TRMGV

Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB8698
DSI Address: 0x1B8698

Reset: 0x000000000
Read/Write

TRMGV – Tx and Rx 1519- to 1522-Byte VLAN Frame Counter, Bits 10–31

The RMON MIB counter that counts the number of 1519- to 1522-byte frames.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TRMGV																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

RBYT

Receive Byte Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB869C
DSI Address: 0x1B869C
Reset: 0x000000000
Read/Write

RBYT – Rx Byte Counter, Bits 1–31

The RMON MIB counter that counts the number of received bytes.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	TRMAX																														
0																															

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RPKT

Receive Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86A0
DSI Address: 0x1B86A0

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	RPKT																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

RPKT – Receive Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of frame received packets.

RFCS

Receive FCS Error Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86A4
DSI Address: 0x1B86A4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RFCS															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

RFCS – Receive FCS Error Counter, Bits 16–31

The RMON MIB counter that counts the number of received 64- to 1518-byte frames with a check sequence error.

Ethernet Controller

RMCA

Receive Multicast Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86A8

DSI Address: 0x1B86A8

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	RMCA																					

* = Reserved. Write to 0 for future compatibility

RMCA – Rx Multicast Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of non-VLAN and VLAN multicast good frames.

RBCA

Receive Broadcast Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86AC

DSI Address: 0x1B86AC

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	RBCA																					

* = Reserved. Write to 0 for future compatibility

RBCA – Rx Broadcast Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of non-VLAN and VLAN broadcast good frames.

Ethernet Controller

RXCF

Receive Control Frame Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86B0
DSI Address: 0x1B86B0

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	RXCF																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

RXCF – Rx Control Frame Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of MAC control frames.

RXPF

Receive Pause Frame Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86B4
DSI Address: 0x1B86B4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	RXPF																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

RXPF – Rx Pause Frame Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of valid pause MAC control frames.

Ethernet Controller

RXUO

Receive Unknown Opcode Packet Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86B8
DSI Address: 0x1B86B8

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RXUO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

RXUO – Rx Unknown Opcode Packet Counter, Bits 16–31

The RMON MIB counter that counts the number of MAC control frames with non-pause opcode.

* = Reserved. Write to 0 for future compatibility

RALN

Receive Alignment Error Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86BC
DSI Address: 0x1B86BC
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RALN															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

RALN – Rx Alignment Error Counter, Bits 16–31

The RMON MIB counter that counts the number of received frames with an invalid FCS and a non-integral number of bytes.

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RFLR

Receive Frame Length Error Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86C0
DSI Address: 0x1B86C0

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RFLR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

RFLR – Rx Frame Length Error Counter, Bits 16–31

The RMON MIB counter that counts the number of frames in which the 802.3 length field does not match actual bytes.

* = Reserved. Write to 0 for future compatibility

RCDE

Receive Code Error Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86C4
DSI Address: 0x1B86C4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RCDE															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

RCDE – Rx Code Error Counter, Bits 16–31

The RMON MIB counter that counts when a valid carrier is present and at least one invalid data symbol is detected.

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RCSE

Receive Carrier Sense Error Counter

Local Bus Address: ___________ See **page 8-40**

QBus (IPBus) Address: 0x01FB86C8
DSI Address: 0x1B86C8

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RCSE															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

RCSE – Rx Carrier Sense Error Counter, Bits 16–31

The RMON MIB counter that counts the number of times a false carrier is detected.

RUND

Receive Undersize Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86CC
DSI Address: 0x1B86CC
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RUND															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

RUND – Rx Undersize Packet Counter, Bits 16–31

The RMON MIB counter that counts the number of frames less than 64 bytes long with a valid FCS.

Ethernet Controller

RVOR

Receive Oversize Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86D0

DSI Address: 0x1B86D0

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	ROVR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

ROVR – Rx Oversize Packet Counter, Bits 16–31

The RMON MIB counter that counts the number of oversized frames with a valid FCS.

* = Reserved. Write to 0 for future compatibility

RFRG

Receive Fragments Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86D4

DSI Address: 0x1B86D4

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RFRG															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

RFRG – Rx Fragments Counter, Bits 16–31

The RMON MIB counter that counts the number of frames less than 64 bytes long with an invalid FCS.

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

RJBR

Receive Jabber Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86D8
DSI Address: 0x1B86D8

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RJBR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

RJBR – Rx Jabber Counter, Bits 16–31

The RMON MIB counter that counts the number of oversized frames with an invalid FCS.

RDRP

Receive Dropped Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86DC
DSI Address: 0x1B86DC
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RDRP															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

RDRP – Rx Dropped Packet Counter, Bits 16–31

The RMON MIB counter that counts the number of frames streamed and later dropped from lack of system resources.

Ethernet Controller

TBYT

Transmit Byte Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86E0
DSI Address: 0x1B86E0

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	TBYT																														
0																															

* = Reserved. Write to 0 for future compatibility

TBYT – Tx Byte Counter, Bits 1–31

The RMON MIB counter that counts the number of transmitted bytes.

TPKT

Transmit Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86E4
DSI Address: 0x1B86E4
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TPKT																					
0	0	0	0	0	0	0	0	0	0																						

* = Reserved. Write to 0 for future compatibility

TPKT – Tx Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of transmitted packets.

Ethernet Controller

TMCA

Transmit Multicast Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86E8
DSI Address: 0x1B86E8

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TMCA																					
0	0	0	0	0	0	0	0	0	0																						

TMCA – Tx Multicast Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of valid multicast transmitted frames.

* = Reserved. Write to 0 for future compatibility

TBCA

Transmit Broadcast Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86EC
DSI Address: 0x1B86EC
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	TBCA																					
0	0	0	0	0	0	0	0	0	0																						

TBCA – Tx Broadcast Packet Counter, Bits 10–31

The RMON MIB counter that counts the number of valid broadcast transmitted frames.

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TXPF

Transmit Pause Control Frame Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86F0
DSI Address: 0x1B86F0

Reset: 0x000000000
Read/Write

TXPF – Tx Pause Frame Packet Counter, Bits 16–31

The RMON MIB counter that counts the number of valid pause MAC control frames.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	RJBR															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

* = Reserved. Write to 0 for future compatibility

TDFR

Transmit Deferral Packets Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86F4
DSI Address: 0x1B86F4
Reset: 0x000000000
Read/Write

TDFR – Tx Deferral Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames deferred on the first transmission attempt.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TDFR											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TEDF

Transmit Excessive Deferral Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86F8
DSI Address: 0x1B86F8

Reset: 0x000000000
Read/Write

TEDF – Tx Excessive Deferral Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames aborted for excessive deferral.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TEDF											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TSCL

Transmit Single Collision Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB86FC
DSI Address: 0x1B86FC
Reset: 0x000000000
Read/Write

TSCL – Tx Single Collision Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames with exactly one collision during transmit.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TSCL											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TMCL

Transmit Multiple Collision Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8700
DSI Address: 0x1B8700

Reset: 0x000000000
Read/Write

TMCL – Tx Multiple Collision Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames with 2–15 collisions.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TMCL											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TLCL

Transmit Late Collision Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8704
DSI Address: 0x1B8704
Reset: 0x000000000
Read/Write

TLCL – Tx Late Collision Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames with a late collision during transmit.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TLCL											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TXCL

Transmit Excessive Collision Packet Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8708
DSI Address: 0x1B8708

Reset: 0x000000000
Read/Write

TXCL – Tx Excessive Collision Packet Counter, Bits 20–31

The RMON MIB counter that counts the number of frames with 16 collisions and were aborted.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TXCL											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TNCL

Transmit Total Collision Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB870C
DSI Address: 0x1B870C
Reset: 0x000000000
Read/Write

TNCL – Tx Total Collision Packet Counter, Bits 20–31

The RMON MIB counter that counts the total number of transmit frame collisions.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TNCL											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TJBR

Transmit Jabber Frame Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8718
DSI Address: 0x1B8718
Reset: 0x000000000
Read/Write

TJBR – Tx Jabber Frame Counter, Bits 20–31

The RMON MIB counter that counts the total number of oversized transmitted frames with an incorrect FCS value.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TJBR											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

TFCS

Transmit FCS Error Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB871C
DSI Address: 0x1B871C

Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TFCS											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TFCS – Tx FCS Error Counter, Bits 20–31

The RMON MIB counter that counts the number of transmit packet with a valid size and incorrect FCS value.

TXCF

Transmit Control Frame Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8720
DSI Address: 0x1B8720
Reset: 0x000000000
Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TXCF											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TXCF – Tx Control Frame Counter, Bits 20–31

The RMON MIB counter that counts the total number of oversized transmitted frames with an incorrect FCS value.

Ethernet Controller

TOVR

Transmit Oversize Frame Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8724

DSI Address: 0x1B8724

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TOVR											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TOVR – Tx Oversize Frame Counter, Bits 20–31

The RMON MIB counter that counts the number of oversized transmitted frames with a correct FCS value.

TUND

Transmit Undersize Frame Counter

Local Bus Address: ___________ See **page 8-41**

QBus (IPBus) Address: 0x01FB8728

DSI Address: 0x1B8728

Reset: 0x000000000

Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TUND											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

TUND – Tx Undersize Frame Counter, Bits 20–31

The RMON MIB counter that counts the total number of transmitted frames less than 64 bytes with a correct FCS value.

Ethernet Controller

TFRG

Transmit Fragment Counter

Local Bus Address: ___________ See **page 8-42**

QBus (IPBus) Address: 0x01FB872C
DSI Address: 0x1B872C
Reset: 0x000000000
Read/Write

TFRG – Tx Fragment Counter, Bits 20–31

The RMON MIB counter that counts the total number of frames less than 64 bytes with an incorrect FCS value.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	TFRG											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

CAR1

Carry Register One

Local Bus Address: ___________ See **page 8-42**
QBus (IPBus) Address: 0x01FB8730
DSI Address: 0x1B8730

Reset: 0x00000000
Read/Write

C1MGV – CAR1 TRMGV Carry, Bit 6	
0	No carry
1	Carry

C1MAX – CAR1 TRMAX Carry, Bit 5	
0	No Carry
1	Carry

C11K – CAR1 TR1K Carry, Bit 4	
0	No carry
1	Carry

C1511 – CAR1 TR511 Carry, Bit 3	
0	No carry
1	Carry

C1255 – CAR1 TR255 Carry, Bit 2	
0	No carry
1	Carry

C1127 – CAR1 TR127 Carry, Bit 1	
0	No carry
1	Carry

C164 – CAR1 TR64 Carry, Bit 0	
0	No carry
1	Carry

C1RAL – CAR1 RALN Carry, Bit 23	
0	No carry
1	Carry

C1RXU – CAR1 RXUO Carry, Bit 22	
0	No carry
1	Carry

C1RXP – CAR1 RXPF Carry, Bit 21	
0	No carry
1	Carry

C1RXC – CAR1 RXCF Carry, Bit 20	
0	No carry
1	Carry

C1RBC– CAR1 RBCA Carry, Bit 19	
0	No carry
1	Carry

C1RMC – CAR1 RMCA Carry, Bit 18	
0	No carry
1	Carry

C1RFC – CAR1 RFCS Carry, Bit 17	
0	No carry
1	Carry

C1RPK – CAR1 RPKT Carry, Bit 16	
0	No carry
1	Carry

C1RBY – CAR1 RBYT Carry, Bit 15	
0	No carry
1	Carry

C1RDR – CAR1 RDRP Carry, Bit 31	
0	No carry
1	Carry

C1RJB – CAR1 RJBR Carry, Bit 30	
0	No carry
1	Carry

C1RFR – CAR1 RFRG Carry, Bit 29	
0	No carry
1	Carry

C1ROV – CAR1 ROVR Carry, Bit 28	
0	No carry
1	Carry

C1RFL– CAR1 RFLR Carry, Bit 24	
0	No carry
1	Carry

C1RCD – CAR1 RCDE Carry, Bit 25	
0	No carry
1	Carry

C1RCS – CAR1 RCSE Carry, Bit 26	
0	No carry
1	Carry

C1RUN – CAR1 RUND Carry, Bit 27	
0	No carry
1	Carry

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C164	C1127	C1255	C1511	C11K	C1MAX	C1MGV	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	* **0**	C1RBY

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
C1RPK	C1RFC	C1RMC	C1RBC	C1RXC	C1RXP	C1RXU	C1RAL	C1RFL	C1RCD	C1RCS	C1RUN	C1ROV	C1RFR	C1RJB	C1RDR

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

CAR2

Carry Register Two

Local Bus Address: ___________ See **page 8-42**
QBus (IPBus) Address: 0x01FB8734
DSI Address: 0x1B8734
Reset: 0x00000000
Read/Write

C2CTF – CAR2 TXCF Carry, Bit 14

0	No carry
1	Carry

C2TFC – CAR2 TFCS Carry, Bit 13

0	No carry
1	Carry

C2CJB – CAR2 TJBR Carry, Bit 12

0	No carry
1	Carry

C2TDF – CAR2 TDFR Carry, Bit 23

0	No carry
1	Carry

C2TPF – CAR2 TXPF Carry, Bit 22

0	No carry
1	Carry

C2TBC – CAR2 TBCA Carry, Bit 21

0	No carry
1	Carry

C2TMC – CAR2 TMCA Carry, Bit 20

0	No carry
1	Carry

C2TPK– CAR2 TPKT Carry, Bit 19

0	No carry
1	Carry

C2TBY – CAR2 TBYT Carry, Bit 18

0	No carry
1	Carry

C2TFG – CAR2 TFRG Carry, Bit 17

0	No carry
1	Carry

C2TUN – CAR2 TUND Carry, Bit 16

0	No carry
1	Carry

C2TOV – CAR2 TOVR Carry, Bit 15

0	No carry
1	Carry

C2TDP – CAR2 TDRP Carry, Bit 31

0	No carry
1	Carry

C2TNC – CAR2 TNCL Carry, Bit 29

0	No carry
1	Carry

C2TXC – CAR2 TXCL Carry, Bit 28

0	No carry
1	Carry

C2TED – CAR2 TEDF Carry, Bit 24

0	No carry
1	Carry

C2TSC – CAR2 TSCL Carry, Bit 25

0	No carry
1	Carry

C2TMA – CAR2 TMCL Carry, Bit 26

0	No carry
1	Carry

C2TLC – CAR2 TLCL Carry, Bit 27

0	No carry
1	Carry

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	C2CJB	C2TFC	C2CTF	C2TOV	C2TUN	C2TFG	C2TBY	C2TPK	C2TMC	C2TBC	C2TPF	C2TDF	C2TED	C2TSC	C2TMA	C2TLC	C2TXC	C2TNC	* 0	C2TDP

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

CAM1

Carry Register One Mask

Local Bus Address: ___________ See **page 8-42**
QBus (IPBus) Address: 0x01FB8738
DSI Address: 0x1B8738

Reset: 0xFE01FFFF
Read/Write

M1MGV – CAR1 TRMGV Carry Mask, Bit 6

0	Not masked
1	Masked

M1MAX – CAR1 TRMAX Carry Mask, Bit 5

0	No Masked
1	Masked

M11K – CAR1 TR1K Carry Mask, Bit 4

0	Not masked
1	Masked

M1511 – CAR1 TR511 Carry Mask, Bit 3

0	Not masked
1	Masked

M1255 – CAR1 TR255 Carry Mask, Bit 2

0	Not masked
1	Masked

M1127 – CAR1 TR127 Carry, Bit 1

0	Not masked
1	Masked

M164 – CAR1 TR64 Carry Mask, Bit 0

0	Not masked
1	Masked

M1RAL – CAR1 RALN Carry Mask, Bit 23

0	Not masked
1	Masked

M1RXU – CAR1 RXUO Carry Mask, Bit 22

0	Not masked
1	Masked

M1RXP – CAR1 RXPF Carry Mask, Bit 21

0	Not masked
1	Masked

M1RXC – CAR1 RXCF Carry Mask, Bit 20

0	Not masked
1	Masked

M1RBC– CAR1 RBCA Carry Mask, Bit 19

0	Not masked
1	Masked

M1RMC – CAR1 RMCA Carry Mask, Bit 18

0	Not masked
1	Masked

M1RFC – CAR1 RFCS Carry Mask, Bit 17

0	Not masked
1	Masked

M1RPK – CAR1 RPKT Carry Mask, Bit 16

0	Not masked
1	Masked

M1RBY – CAR1 RBYT Carry Mask, Bit 15

0	Not masked
1	Masked

M1RDR – CAR1 RDRP Carry Mask, Bit 31

0	Not masked
1	Masked

M1RJB – CAR1 RJBR Carry Mask, Bit 30

0	Not masked
1	Masked

M1RFR – CAR1 RFRG Carry Mask, Bit 29

0	Not masked
1	Masked

M1ROV – CAR1 ROVR Carry Mask, Bit 28

0	Not masked
1	Masked

M1RFL– CAR1 RFLR Carry Mask, Bit 24

0	Not masked
1	Masked

M1RCD – CAR1 RCDE Carry Mask, Bit 25

0	Not masked
1	Masked

M1RCS – CAR1 RCSE Carry Mask, Bit 26

0	Not masked
1	Masked

M1RUN – CAR1 RUND Carry Mask, Bit 27

0	Not masked
1	Masked

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M164	M1127	M1255	M1511	M11K	M1MAX	M1MGV	* 0	* 0	* 0	* 0	* 0	* 0	* 0	* 0	M1RBY

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
M1RPK	M1RFC	M1RMC	M1RBC	M1RXC	M1RXP	M1RXU	M1RAL	M1RFL	M1RCD	M1RCS	M1RUN	M1ROV	M1RFR	M1RJB	M1RDR

* = Reserved. Write a 0 to all reserved bits for future compatibility

MSC8113 Reference Manual, Rev. 0

Ethernet Controller

CAM2

Carry Register Two Mask

Local Bus Address: ___________ See **page 8-42**
QBus (IPBus) Address: 0x01FB8734
DSI Address: 0x1B8734
Reset: 0x00000000
Read/Write

M2TXC – CAR2 TXCF Carry Mask, Bit 14	
0	Not masked
1	Masked

M2TFC – CAR2 TFCS Carry Mask, Bit 13	
0	Not masked
1	Masked

M2TJB – CAR2 TJBR Carry Mask, Bit 12	
0	Not masked
1	Masked

M2TDF – CAR2 TDFR Carry Mask, Bit 23	
0	Not masked
1	Masked

M2TPF – CAR2 TXPF Carry Mask, Bit 22	
0	Not masked
1	Masked

M2TBC – CAR2 TBCA Carry Mask, Bit 21	
0	Not masked
1	Masked

M2TMC – CAR2 TMCA Carry Mask, Bit 20	
0	Not masked
1	Masked

M2TPK– CAR2 TPKT Carry Mask, Bit 19	
0	Not masked
1	Masked

M2TBY – CAR2 TBYT Carry Mask, Bit 18	
0	Not masked
1	Masked

M2TFG – CAR2 TFRG Carry Mask, Bit 17	
0	Not masked
1	Masked

M2TUN – CAR2 TUND Carry Mask, Bit 16	
0	Not masked
1	Masked

M2TOV – CAR2 TOVR Carry Mask, Bit 15	
0	Not masked
1	Masked

M2TDP – CAR2 TDRP Carry Mask, Bit 31	
0	Not masked
1	Masked

M2TNC – CAR2 TNCL Carry Mask, Bit 29	
0	Not masked
1	Masked

M2TXC – CAR2 TXCL Carry Mask, Bit 28	
0	Not masked
1	Masked

M2TED – CAR2 TEDF Carry Mask, Bit 24	
0	Not masked
1	Masked

M2TSC – CAR2 TSCL Carry Mask, Bit 25	
0	Not masked
1	Masked

M2TMA – CAR2 TMCL Carry Mask, Bit 26	
0	Not masked
1	Masked

M2TLC – CAR2 TLCL Carry Mask, Bit 27	
0	Not masked
1	Masked

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	M2TJB	M2TFC	M2TXC	M2TOV
0	0	0	0	0	0	0	0	0	0	0	0				

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
M2TUN	M2TFG	M2TBY	M2TPK	M2TMC	M2TBC	M2TPF	M2TDF	M2TED	M2TSC	M2TMA	M2TLC	M2TXC	M2TNC	*	M2TDP
														0	

* = Reserved. Write a 0 to all reserved bits for future compatibility

Ethernet Controller

IADDR[0–7]

Individual Address Registers 0–7

IADDR_____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-42**

QBus (IPBus) Address: 0x01FB8800 + (4 × *n*) = ______________

DSI Address: 0x1B8800 + (4 × *n*) = ______________

Reset: 0x000000000

Read/Write

IADDR – Individual Address, Bits 0–31

Contains the 32-bit value associated with the corresponding register.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IADDR																															

* = Reserved. Write to 0 for future compatibility

GADDR[0–7]

Group Address Registers 0–7

GADDR_____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-42**

QBus (IPBus) Address: 0x01FB8880 + (4 × *n*) = ______________

DSI Address: 0x1B8880 + (4 × *n*) = ______________

Reset: 0x000000000

Read/Write

GADDR – Group Address, Bits 0–31

Contains the value used to generate a pointer to the group address hash table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
GADDR																															

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

PMD[0–15]

Pattern Match Data 0–15

PMD_____________ (enter number = *n*)

Local Bus Address: ____________ See **page 8-42**

QBus (IPBus) Address: 0x01FB8900 + (32 × *n*) = _______________
DSI Address: 0x1B8900 + (32 × *n*) = _______________

Reset: 0x000000000
Read/Write

PMD – Pattern Match Data, Bits 0–31

Contains the data to compare to the frame data.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PMD																															

* = Reserved. Write to 0 for future compatibility

PMASK[0–15]

Pattern Mask Data 0–15

PMASK_____________ (enter number = *n*)

Local Bus Address: ____________ See **page 8-42**

QBus (IPBus) Address: 0x01FB8908 + (32 × *n*) = _______________
DSI Address: 0x1B8908 + (32 × *n*) = _______________

Reset: 0x000000000
Read/Write

PMASK – Pattern Mask Data, Bits 0–31

Contains the 32-bit mask used for pattern matching.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PMASK																															

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

PCNTRL[0–15]

Pattern Match Control Register 0–15
PCNTRL____________ (enter number = *n*)

Local Bus Address: ___________ See **page 8-42**
QBus (IPBus) Address: 0x01FB8910 + (32 × *n*) = ______________
DSI Address: 0x1B8910 + (32× *n*) = ______________

Reset: 0x000000000
Read/Write

PMC – Pattern Match Control, Bits 30–31	
00	Entry disabled
01	Pattern match
10	Pattern match accept
11	Pattern match reject

CP – Concatenated Pattern, Bit 25	
0	No pattern concatenation
1	Concatenate following pattern to current one

CSE – Continue Search Enable, Bit 24	
0	If pattern matches, discontinue search
1	If pattern matches, continue search to end

MI – Matching Index, Bits 18–23
Contains the index used for starting the pattern matching.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	MI						CSE	CP	*	*	*	*	PMC	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0		

* = Reserved. Write to 0 for future compatibility

Ethernet Controller

PATTRB[0–15]

Pattern Match Attributes Register 0–15

PATTRB__________ (enter number = *n*)

Local Bus Address: __________ See **page 8-43**

QBus (IPBus) Address: 0x01FB8918 + (32 × *n*) = ______________

DSI Address: 0x1B8918 + (32× *n*) = ______________

Reset: 0x000000000
Read/Write

QC – Queue Classification, Bits 30–31	
00	Queue 0
01	Queue 1
10	Queue 2
11	Queue 3

RBDSEN – RxBD Snoop Enable, Bit 25	
0	Disables snooping of Rx BD memory accesses
1	Enables snooping of Rx BD memory accesses

RDSEN – Rx Data Snoop Enable, Bit 24	
0	Disables snooping of all rx frame data
1	Enables snooping of all rx frame data

PMF – Pattern Match File, Bit 22	
0	Use DATTR[QC] to file matched frame
1	Use PATTRB[QC] to file matched frame

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	PMF	*	RDSEN	RBDSEN	*	*	*	*	QC	
0	0	0	0	0	0		0			0	0	0	0		

* = Reserved. Write to 0 for future compatibility

DATTR

Default Attributes Register

Local Bus Address: __________ See **page 8-46**

QBus (IPBus) Address: 0x01FB8BF8

DSI Address: 0x1B8BF8

Reset: 0x000000000
Read/Write

RBDSEN – RxBD Snoop Enable, Bit 25	
0	Disables snooping of Rx BD memory accesses
1	Enables snooping of Rx BD memory accesses

QC – Queue Classification, Bits 30–31	
00	Queue 0
01	Queue 1
10	Queue 2
11	Queue 3

RDSEN – Rx Data Snoop Enable, Bit 24	
0	Disables snooping of all rx frame data
1	Enables snooping of all rx frame data

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	*	*	*	*	*	*	*	RDSEN	RBDSEN	*	*	*	*	QC	
0	0	0	0	0	0	0	0			0	0	0	0		

* = Reserved. Write to 0 for future compatibility

MSC8113 Dictionary B

This appendix presents an alphabetical list of terms, phrases, and abbreviations that are used in this manual. Many of the terms are defined in the context of how they are used in this manual—that is, in the context of the MSC8113. Some of the definitions are derived from *Newton’s Telecom Dictionary: The Official Dictionary of Telecommunications*, © 1998 by Harry Newton.

$\overline{\text{AACK}}$ Address acknowledge signal. Asserted by a slave to acknowledge address tenure by the master.

AAU Address arithmetic unit. On the SC140 core, there are two identical AAUs. Each contains a 32-bit full adder called an offset adder that can add or subtract two AGU registers, add immediate value, increment or decrement an AGU register, add PC, or add with reverse-carry. The offset adder also performs compare or test operations and arithmetic and logical shifts. The offset values added in this adder are pre-shifted by 1, 2, or 3, according to the access width. In reverse-carry mode, the carry propagates in the opposite direction. A second full adder, called a modulo adder, adds the summed result of the first full adder to a modulo value, M or minus M, where M is stored in the selected modifier register. In modulo mode, the modulo comparator tests whether the result is inside the buffer, by comparing the results to the B register, and chooses the correct result from between the offset adder and the modulo adder.

$\overline{\text{ABB}}$ Address bus busy signal. The MSC8113 asserts this pin as an output for the duration of its address bus tenure. An external master asserts this signal as an input to the MSC8113 to maintain bus tenure.

ABIST Autonomous built-in self test.

ADS Application development system.

AGU Address generation unit.

ALE Address latch enable signal. Controls the external address latch used on the external system bus.

ALU Arithmetic logic unit. The part of the CPU that performs the arithmetic and logical operations. The SC140 is the four-ALU version of the StarCore SC100 DSP core family.

AM Address mode.

ANSI American National Standards Institute.

$\overline{\textbf{ARTRY}}$ Address retry signal. Asserted by a slave device to indicate that the master should retry the bus transaction.

ASIC Application-specific integrated circuit. An integrated circuit that performs a particular function by defining the interconnection of a set of basic circuit building blocks taken from a library provided by a circuit manufacturer.

atomic A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address. The MSC8113 initiates the read and write separately, but it signals the memory system that it is attempting an atomic operation. If the operation fails, status is kept so that MSC8113 can try again.

A*x* Address bus signals (A[0–31]) for the external system bus.

BADDR*x* Burst address signal (BADDR[27–31]). These five burst address pins are outputs of the SIU memory controller. They connect directly to burstable memory devices.

bandwidth A measure of the carrying capacity, or size, of a communications channel. For an analog circuit, the bandwidth is the difference between the highest and lowest frequencies that a medium can transmit and is expressed in hertz (Hz). Hz is equal to one cycle per second.

baseband The original band of frequencies of a signal before it is modulated for transmission at a higher frequency. The signal is typically multiplexed and sent on a carrier with other signals at the same time.

BBW Bus bandwidth.

BCI Binary code instrumentation.

BCR Bus Configuration Register.

$\overline{\textbf{BCTL}}$*x* Buffer control signals ($\overline{\text{BCTL}}$[0–1]). Control buffers on the system bus.

BD Buffer descriptor.

BD_ATTR Buffer attribute parameters.

beat A single state on the MSC8113 interface that may extend across multiple bus cycles. An MSC8113 transaction can be composed of multiple address or data beats.

BFU Bit-field unit.

$\overline{\text{BG}}$ Bus grant signal. The MSC8113 asserts this pin as an output to grant system bus ownership to an external bus master. An external arbiter asserts this pin as an input to grant bus ownership to the MSC8113.

big-endian For big-endian scalars, the most significant byte (MSB) is stored at the lowest, or starting, address while the least significant byte (LSB) is stored at the highest, or ending, address. This memory structure is called “big-endian” because the big end of the scalar comes first in memory. The MSC8113 supports big-endian. *See also* little-endian and munged little-endian.

BMU Bit manipulation unit. On the SC140 core, performs bit manipulation operations, such as setting, clearing, changing, or testing a destination, according to an immediate operand. All bit manipulation instructions typically execute in two cycles and work on 16-bit data. This data can be a memory location, or a portion (high or low) of a register. Only a single bit manipulation instruction is allowed in any single execution set, since only one execution unit exists for these instructions.

BM*x* Boot mode signals (BM[0–2]). Sampled on the deassertion of $\overline{\text{PORESET}}$.

BNKSEL*x* Bank select signals (BNKSEL[0–2]). Used to select SDRAM banks on the system bus.

bootloader Loads and executes source code that initializes the after it completes a reset sequence and programs its registers for the required mode of operation. The bootloader program, which is provided in the on-chip ROM of the MSC8113, loads and executes source programs received from a host processor, an EPROM, or a standard memory device.

BOOTROM MSC8113 boot ROM.

BR*x* Base Registers 0–7, 9–11.

$\overline{\text{BR}}$ Bus request signal. The MSC8113 asserts this pin as an output to request ownership of the system bus. An external master should assert this pin as an input to request system bus ownership from the internal arbiter.

broadband Also called wideband. A type of data transmission in which a single medium (wire) can carry several channels at once. Cable TV, for example, uses broadband transmission. In contrast, baseband transmission allows only one signal at a time. Most communications between computers, including the majority of local-area networks, use baseband communications. An exception is B-ISDN networks, which employ broadband transmission.

BSR MSC8113 Boundary Scan Register.

BT Burst tolerance.

BTAM Block transfer acknowledge message.

BTM Block transfer message.

BUFCMD Command buffers.

buffer descriptor (BD) Each DMA channel uses the specifications in its associated buffer descriptors to define operation of the channel.

burst A multiple-beat data transfer in the MSC8113 whose total size is equal to 32 bytes or 4 data beats at 8 bytes per beat.

CAM Content-addressable memory.

CCITT Consultative Committee on International Telegraphy and Telephony.

CHIP_ID*x* Chip ID signals (CHIP_ID[0–3]).

CIDR Core ID Register.

CLKIN Clock in signal.

CLKOUT Clock out signal.

CNFGS Configuration signal.

CRC Cyclic redundancy check.

$\overline{\mathbf{CS}\boldsymbol{x}}$ Chip select signal ($\overline{\text{CS}[0–7]}$). Enables specific memory devices or peripherals connected to external system bus.

DABR Data address breakpoint register.

$\overline{\mathbf{DACK}\boldsymbol{x}}$ Data acknowledge signals ($\overline{\text{DACK}[1–4]}$). Asserted to acknowledge a DMA transaction on the specified DMA channel.

DALU Data ALU. Performs arithmetic and logical operations on data operands in the MSC8113. The source operands for the Data ALU, which may be 16, 32, or 40 bits, originate either from data registers or from immediate data. The results of all Data ALU operations are stored in the data registers. All Data ALU operations are performed in one clock cycle. Up to four parallel arithmetic operations can be performed in each cycle. The destination of every arithmetic operation can be used as a source operand for the operation immediately following, without any time penalty.

DAR Data Address Register

$\overline{\textbf{DBB}}$ Data bus busy signal. The MSC8113 asserts this pin as an output for the duration of its data bus tenure. An external master asserts this signal as an input to the MSC8113 to maintain data bus tenure.

$\overline{\textbf{DBG}}$ Data bus grant signal. The MSC8113 asserts this pin as an output to grant data bus ownership to an external bus master. The external arbiter asserts this pin as an input to grant data bus ownership to the MSC8113.

DBR*x* Data area Registers [0–3]

DCHCR*x* DMA Channel [0–15] Configuration Registers.

DCIR DSP chip ID register.

DCPRAM DMA Channel Parameters RAM.

DCR DSI Control Register.

DDR DSI Disable Register.

Debug mode On the MSC8113, the JTAG and **IEEE** Std. 1149.1 Test Access Port gives entry to the debug mode of operation. With the EOnCE real-time debugging capability, users can read the chip’s internal resources without having to stop the device and go into debug mode. The benefits range from faster debugging to reduced system development costs and improved field diagnostics. The SC140 core has a debug mode that is enabled at reset by pulling up the DBREQ/EE0 pin. *See also* EOnCE.

DEC Decrementer register.

DEMR DMA External Mask Register.

DER DSI Error Register.

DF Division factor.

DIAMR*x* DSI Internal Address Mask Registers (DIAMR[9, 11]).

DIBAR*x* DSI Internal Base Address Registers (DIBAR[9, 11]).

DIMR DMA Internal Mask Register.

DMA Direct memory access. A fast method of moving data from a storage device to RAM, which speeds up processing. The MSC8113 multi-channel DMA controller supports up to 16 time-multiplexed channels and buffer alignment by hardware. The DMA controller connects to both the 60x-compatible system bus and the local bus and can function as a bridge between both buses. The MSC8113 DMA controller supports flyby transactions on either bus. The DMA controller enables hot swap between channels, by time-multiplexed channels with no cost in clock cycles. Sixteen priority levels support synchronous and asynchronous transfers on the bus and give a varying bus bandwidth per channel. The DMA controller can service multiple requestors. A requestor can be any one of four external peripherals or sixteen internal requests generated by the DMA FIFO itself. *See also* flyby transfer.

Double word For the 16-bit SC140 core, a double word is 32 bits. For the CPM and 60x-compatible bus, a double word is 64 bits.

$\overline{\text{DONE}x}$ Done 1–2 signals ($\overline{\text{DONE[1–2]}}$).

DPCR DMA Pin Configuration Register.

DPLL Digital phase-lock loop.

DPR Dual-port RAM.

DPRAM Dual-port RAM.

DP*x* Data parity signal (DP[0–7]). Used by the bus master to generate an odd parity value for the respective byte being transferred.

$\overline{\text{DRACK}x}$ Data request acknowledge 1–2 signals ($\overline{\text{DRACK[1–2]}}$).

DRAM Dynamic random-access memory. Dynamic memory is solid-state memory in which the stored information decays over a period of time. The decay time can range from milliseconds to seconds depending on the device and its physical environment. The memory cells must undergo refresh operations often enough to maintain the integrity of the stored information. The dynamic nature of the circuits for DRAM require data to be written back after being read, hence the difference between access time and cycle time. DRAM memory is organized as a rectangular matrix addressed by rows and columns.

DREQ*x* Data request signals (DREQ[1–4]). Used by an external peripheral to request DMA service from the specified channel.

DSI Direct Slave Interface.

DSI64 DSI size select (32/64 bits) signal.

DSISYNC Indicates the DSI mode of operation as Synchronous or Asynchronous Mode.

DSP Digital signal processor.

DSR DSI Status Register.

DSTR DMA Status Register.

DSWBAR DSI Sliding Window Base Address Register.

DTEAR DMA Transfer Error Address Status Register.

D*x* Data bus signals (D[0–63]) for the external system bus.

E1 The European equivalent of the North American T1, except that E1 carries information at the rate of 2.048 Mbps. This is a telephony standard. Its size is based on the number of channels, each of which carries 64 Kbps. *See also* T1.

EA Effective address.

ECC Error checking and correction.

EDCA_CTRL EDCA Control Register.

EE_CTRL EE Pin Control Register.

EE*x* EOnCE event signals (EE[0–1]).

ELIR*x* PIC Edge/Level-Triggered Interrupt Priority Registers A–F.

EMR SC140 core Exception and Mode Register.

ENQ Enquiry character.

EOB End-of-burst (data).

EOnCE Enhanced on-chip emulation. Allows nonintrusive interaction with the MSC8113 and its peripherals so that a user can examine registers, memory, or on-chip peripherals, define various breakpoints, and read the trace-FIFO. These interactions facilitate hardware and software development on the MSC8113 processor. The EOnCE module interfaces with the debugging system through on-chip JTAG TAP controller pins.

EPROM Erasable programmable read-only memory.

EQBS Extended QBus system.

EQBUSBR EQBus Bank Register.

ESEL_DM Event Selector Mask Debug Mode Register.

ESP Exception stack pointer.

ETB End-of-block.

ETX End-of-text character.

EVM Evaluation module.

$\overline{\text{EXT\_BG}x}$ External bus grant signals ($\overline{\text{EXT\_BG[2–3]}}$). Used to grant bus mastership to the requesting bus master.

$\overline{\text{EXT\_BR}x}$ External bus request signal ($\overline{\text{EXT\_BR[2–3]}}$). Used by an external master to request bus mastership.

$\overline{\text{EXT\_DBG}n}$ External data bus grant signal ($\overline{\text{EXT\_DBG[2–3]}}$). Used to grant data bus mastership to the requesting bus master.

FC-PBGA package Flip Chip-Plastic Ball Grid Array. The MSC8113 FC-PBGA package has 431 balls.

FDMA Frequency division multiple access. A method of allowing multiple users to share the radio frequency spectrum by assigning each active user an individual frequency channel. In this practice, users are dynamically allocated a group of frequencies so that the apparent availability is greater than the number of channels.

FEPROM Flash EPROM.

FFT Fast Fourier transform.

FIFO First-in, first-out buffer.

FIR Finite impulse response. A type of filter. FIR filters are characterized by transfer functions that are polynomials, where the coefficients are directly the impulse response of the filter. The form of an FIR filter gives rise to the terminology of tapped delay line and the coefficients as tap weights. The length of an FIR filter is the number of taps, N, and thus the convention of using indices from 0 through (N-1) for the coefficients.

flyby transfer Also known as a “single access transaction.” The data path is between a peripheral and memory with the same port size, located on the same bus. On the MSC8113, flyby transactions can occur only between external peripherals and external memories located on the 60x-compatible system bus. Flyby operations do not require access to the DMA FIFO. See also DMA.

FU Fetch unit.

full duplex Transmission in two directions simultaneously—that is, simultaneous two-way communications. Such communications occur on four-wire circuits. In contrast, half duplex communications occur in only one direction at one time.

GB Gigabyte.

GBL Global signal. Assertion of this pin by the bus master indicates that the transfer is global and should be snooped by caches in the system.

GBps Gigabyte per second.

GCIER GIC Core Interrupt Enable Register.

GEIER GIC External Interrupt Enable Register.

GIC Global interrupt controller.

GICR GIC Interrupt Configuration Register.

GISR GIC Interrupt Status Register.

GND Ground signal for V_{DD}.

GND$_H$ Ground signal for V_{DDH}.

GND$_{SYN}$ Ground signal for V_{CCSYN}.

GPCM General-purpose chip-select machine. Part of the memory controller in the SIU. The GPCM provides interfacing for simpler, lower-performance memory resources and memory-mapped devices. The GPCM has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals. The GPCM controls Bank 11, which is assigned for DSP peripherals. Banks 0–7 can be assigned to the GPCM as well.

GPIO*x* General-purpose input/output signals (GPIO[0–31]).

GPR JTAG General-Purpose Register.

GPR*x* General-Purpose Registers 0–1.

GUI Graphical user interface.

HA*x* DSI host address line signal (HA[11–29]).

half-word For the 16-bit SC140 core, a half-word is 8 bits. For the 60x-compatible bus, a half-word is 16 bits.

$\overline{\text{HBCS}}$ Host Broadcast Chip Select. DSI chip select for broadcast mode. Enables more than one DSI to share the same host chip-select pin for broadcast write accesses.

$\overline{\text{HBRST}}$ Host Burst. The host asserts this pin to indicate that the current DSI transaction is a burst transaction in synchronous mode only.

HCID*x* Host Chip ID 0–3. Carries the chip ID of the DSI. The DSI is accessed only if $\overline{\text{HCS}}$ is asserted and HCID[0–3] matches the Chip_ID, or if $\overline{\text{HBCS}}$ is asserted.

HCLKIN Host clock input signal.

$\overline{\text{HCS}}$ DSI chip select signal.

HD*x* DSI data signals (HD[0–63]).

$\overline{\text{HDBE}x}$ DSI data byte enable signals ($\overline{\text{HDBE[4–7]}}$).

$\overline{\text{HDBS}x}$ DSI data byte strobe signals ($\overline{\text{HDBS[4–7]}}$).

$\overline{\text{HDST}x}$ DSI data structure (HDST[0–1]).

HRCW Hard reset configuration word.

$\overline{\textbf{HRDE}}$ Host Read Data Enable (in Synchronous dual mode). Indicates valid data for host read accesses.

$\overline{\textbf{HRDS}}$ Host Read Data Strobe (in Asynchronous dual mode). Used as a strobe for host read accesses.

$\overline{\textbf{HRESET}}$ Hard reset signal.

HRW DSI read/write select signal.

HS Hardware semaphore.

HSMPR*x* Hardware Semaphore Registers 0–7

$\overline{\textbf{HTA}}$ Host transfer acknowledge signal.

HW Hardware reset.

$\overline{\textbf{HWBEn}}$ signals ($\overline{\text{HWBE[4–7]}}$).

$\overline{\textbf{HWBS}x}$ DSI write byte strobe signals ($\overline{\text{HWBS[0–7]}}$).

Hz Hertz.

ICache Instruction Cache

ICACR Instruction Cacheable Area Control Register.

ICBR Instruction Cacheable Area Base Register.

ICCMR ICache Command Register.

ICCR ICache Control Register.

ID Identification Register.

IDE Integrated development environment.

IEEE Institute of Electrical and Electronics Engineers.

IFUR Instruction FU Configuration Register.

IIR Infinite impulse response. A type of filter. *See* FIR.

IMMR Internal Memory Map Register.

INTMSK Interrupt mask.

$\overline{\textbf{INT\_OUT}}$ Interrupt output signal. When asserted, the pending interrupt must be handled by an external device.

I/O	Input/output.
IPBus	Internal peripheral bus.
IPL	Interrupt priority level.
IPR*x*	PIC Interrupt Pending Registers A–B.
IR	Individual reset.
IR	Instruction Register.
$\overline{\mathbf{IRQ}\boldsymbol{x}}$	Interrupt request signals ($\overline{\mathrm{IRQ[1–15]}}$).
ISDN	Integrated services digital network.
ISR	Interrupt service routine. In the MSC8113, the SC140 core handles pending unmasked interrupts in order of priority. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt.
ITU	International Telecommunication Union.
IU	Integer unit.
JTAG	Joint Test Action Group
JTAGID	JTAG Identification (ID) Register.
KB	Kilobyte.
Kb	Kilobit.
Kbps	Kilobits per second.
KHz	Kilohertz.
lane	A sub-grouping of signals within a bus. An 8-bit section of the address or data bus may be referred to as a byte lane for that bus.
L_TESCR1	Local Bus Transfer Error Status and Control Register 1
LC	Loop Counter Register.
LCL_ACR	Local Bus Arbiter Configuration Register.
LCL_ALRH	Local Bus Arbitration Level Register (bus masters 0–7).
LCL_ALRL	Local Bus Arbitration Level Register (bus masters 8–15).
LDMTEA	DMA Transfer Error Address Register.

LDMTER DMA Transfer Error Requestor Number Register.

LGTDTEA Local Bus GTD Transfer Error Address.

LGTDTEM GTD Transfer Error TDMNUM_TR Register.

LIC Local interrupt controller.

LICAICR*n* LIC Group A Interrupt Configuration Registers 0–3

LICAIER LIC Group A Interrupt Enable Register

LICAIESR LIC Group A Interrupt Error Status Register

LICAISR LIC Group A Interrupt Status Register

LICBICR*n* LIC Group B Interrupt Configuration Registers 0–3

LICBIER LIC Group B Interrupt Enable Register

LICBIESR LIC Group B Interrupt Error Status Register

LICBISR LIC Group B Interrupt Status Register

little-endian For little-endian scalars, the least-significant byte (LSB) is stored at the lowest (or starting) address. This is called “little-endian” because the little end of the scalar comes first in memory. *See also* big-endian and munged little-endian.

LRU Least recently used.

LRUSR LRU Status Register.

lsb Least-significant bit.

LSB Least-significant byte.

LSU Load/store unit.

M1 Internal SC140 core memory (224 KB).

M2 Device shared memory (476 KB).

MAC Multiply and accumulate. On the SC140 core, the MAC unit is the main arithmetic processing unit. It performs all the calculations on data operands. The MAC unit outputs one 40-bit result in the form of [Extension:Most Significant Portion:Least Significant Portion] (EXT:MSP:LSP). The multiplier executes 16-bit x 16-bit fractional or integer multiplication between two's complement signed, unsigned, or mixed operands. The 32-bit product is right-justified and added to the 40-bit contents of one of the 16 data registers.

MAMR Machine A Mode Register.

MAR Memory Address Register.

maskable interrupt A hardware interrupt that can be enabled or disabled through software.

master The device that owns the address or data bus, the device that initiates or requests the transaction.

Mb Megabit.

MB Megabyte.

MBMR Machine B Mode Register.

MBS Maximum burst size.

MCMR Machine C Mode Register

MCP Machine check interrupt signal.

MCTL Modifier Control Register.

MDR Memory Data Register.

memory controller A unit whose main function is to control the bus memories and I/O devices. The MSC8113 memory controller is located in the SIU portion of the MSC8113. It controls a maximum of 10 memory banks shared by a high-performance SDRAM machine, a general-purpose chip-select machine (GPCM), and three user-programmable machines (UPMs). It supports a glueless interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.

MF Multiplication factor.

MHz Megahertz.

MINFLR Minimum Frame Length Register.

MIPS Millions of instructions per second. A rough measure of processor performance, measuring the number of instructions that can be executed in one second. However, different instructions require more or less time than others, and performance can be limited by other factors, such as memory and I/O speed.

MMACS Million multiply-accumulates per second.

MMU Memory management unit.

MODCKn Clock mode signals (MODCK[1–2]). Sampled when $\overline{\text{PORESET}}$ is deasserted to select the clock mode.

modulo An arithmetic term to designate an operation that uses the remainder value from a division operation.

MPTPR Memory Refresh Timer Prescaler Register

MPU Microprocessor unit.

MQBus Memory bus for the QBus connecting the SC140 cores to the M2 shared memory and Boot ROM.

msb Most-significant bit.

MSB Most-significant byte.

Multi-Master Bus Mode The multi-master bus mode can include one or more potential bus masters external to the MSC8113. The other bus masters can, for example, be ASIC DMAs, high-end PowerQUICC IIs, or other MSC8113 devices. Also see Single Master Bus Mode.

multiplexing A method by which two or more signals share a physical pin connection or a single data stream.

munged little-endian Although data is stored in big-endian order, the data address is modified so that the memory structure appears to be in little-endian format to the executing processor. The address modification is called munging. *See also* big-endian and little-endian.

mux Multiplexer.

M*x* Modifier registers (M[0–3]) in the SC140 cores.

M*x*MR Machine A/B/C Mode Registers.

NLMS Normalized least-mean square.

$\overline{\text{NMI}}$ Non-maskable interrupt. Asserted as an input to request handling by the MSC8113.

$\overline{\text{NMI\_OUT}}$ Non-maskable interrupt signal. An open drain output from the MSC8113 used to request handling by an external host.

NOP No operation.

NRZ Non-return-to-zero.

NSP Normal stack pointer.

Nx Offset Registers (N[0–3]) in the SC140 cores.

OE Output enable signal.

opcode Operation code.

operation code Also known as “opcode.” The command part of a machine instruction. That is, in most cases, the first byte of the machine code that describes the type of operation and combination of operands to the central processing unit (CPU).

OR*x* Option Registers 0–7, 9–11. The definitions depend on the mode used: SDRAM, GPCM, or UPM.

PAG SC140 core program address generator.

PAR Pin Assignment Register.

parameter RAM The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SMCs, SPI, and I^{2}C channels. The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM.

parking Granting potential bus mastership without requiring a prior bus request from that device. This eliminates the arbitration delay associated with the bus request.

PBSn Bus UPM byte select signal ($\overline{\text{PBS[0–7]}}$).

PC Program Counter Register used with the SC140 core program sequencer unit.

PCM Pulse-code modulation.

PCU SC140 core program control unit.

PDAT Pin Data Register.

PDF Pre-division factor.

PDIR Pin Data Direction Register.

PDMTEA DMA Transfer Error Address Register.

PDMTER DMA Transfer Error Requestor Number Register.

PDU SC140 core program dispatch unit.

PGPLn Bus UPM general-purpose lines 0–5 (PGPL[0–5]).

PGTA System GPCM transaction termination signal.

PIC Programmable interrupt controller. A peripheral module to serve all the interrupt requests ($\overline{\text{IRQ}}$s) and non-maskable interrupts ($\overline{\text{NMI}}$s) received from MSC8113 peripherals and I/O pins. The PIC is memory mapped to the SC140 core and is accessed via the SC140 core QBus. The PIC not only handles incoming interrupts from internal and external devices, but also generates interrupts to other devices. This capability enables the MSC8113 to be used as a companion chip complementing an external CPU such as a 60x-compatible processor. For example, the MSC8113 might be used to provide protocol handling services, sending an interrupt to notify the central processor each time it finishes processing a batch of data.

pipelining Initiating a bus transaction before the current one finishes. This involves running an address tenure for a new bus transaction before the data tenure for a current bus transaction completes.

PIR Processor Identification Register.

PIREG JTAG Parallel Input Register.

PISCR Periodic Interrupt Status and Control Register.

PIT Periodic interrupt timer.

PITC Periodic Interrupt Timer Count Register.

PITR Periodic Interrupt Timer Register.

PLL Phase lock loop. An electronic circuit that controls an oscillator so that it maintains a constant phase angle relative to a reference signal. A PLL can be used to multiply or divide an input clock frequency to generate a different output frequency.

PODR Pin Open-Drain Register.

POE Bus output enable signal.

$\overline{\text{PORESET}}$ Power-on reset signal.

PPBS Bus parity byte select signal.

PPC_ACR Bus Arbiter Configuration Register.

PPC_ALRH Bus Arbitration-Level Register (bus masters 0–7).

PPC_ALRL Bus Arbitration-Level Register (bus masters 8–15).

PS Port size.

PSDA10 Bus SDRAM A10 signal

PSDAMUX Bus SDRAM address multiplexer signal.

$\overline{\text{PSDCAS}}$ Bus SDRAM column address strobe.

$\overline{\text{PSDDQM}x}$ Bus SDRAM DQM signals ($\overline{\text{PSDDQM[0–7]}}$).

PSDMR Bus SDRAM Mode Register.

$\overline{\text{PSDRAS}}$ Bus SDRAM row address strobe signal.

$\overline{\text{PSDVAL}}$ Data valid signal. Indicates that a valid data beat is on the data bus. It must be used in conjunction with the $\overline{\text{TA}}$ signal on the last data movement to terminate the transfer.

$\overline{\text{PSDWE}}$ Bus SDRAM write enable signal.

PSEQ SC140 core program sequencer.

PSOR Pin Special Options Register.

PSRT Bus assigned SDRAM Refresh Timer.

PUPMWAIT Bus UPM wait signal.

PURT Bus assigned UPM Refresh Timer.

PVR Processor Version Register.

$\overline{\mathrm{PWE}x}$ Bus write enable signals ($\overline{\mathrm{PWE[0–7]}}$). The GPCM uses these signals to select byte lanes for write operations on the system bus.

Q2PPC Quartz bus to 60x-compatible bus.

QBC QBus control unit.

QBus Internal SC140 core bus to extended core devices.

QBUSBR*x* QBus Base Address Register 0–2.

QBUSMR*x* QBus Mask Register 0–2.

quad word For the 16-bit SC140 core, a quad word is 64 bits. For the CPM and 60x-compatible bus, a quad word is 128 bits.

RAM Word RAM word.

RAWA Read-after-write atomic bus operation.

RCLK Receive clock.

RDBS Receive data buffer size.

requestor An external peripheral or an internal request generated by the DMA FIFO. A peripheral interfaces with the DMA by placing a request for service. The request can be external or internal, depending on its origin.

reset A means to bring a device and its components to a known state by setting the registers and control bits to predetermined values and signaling execution to start at a specified address.

RISC Reduced instruction set computing.

RMW Read-modify-write.

RNC Receive number of channels.

RQNUM Requestor number.

RSR Reset Status Register.

RSTCONF Reset configuration signal.

RTOS Real-time operating system.

RWITM read with intent to modify.

R*x* SC140 core address registers (R[0–15]).

Rx Receiver.

SA Start Address Register.

SC140 StarCore 140 core.

SCI Serial communication interface (see UART).

SCIBR SCI Baud-Rate Register.

SCICR SCI Control Register.

SCIDDR SCI Direction Register.

SCIDR SCI Data Register.

SCISR SCI Status Register.

SCMSR System Clock Mode Status Register.

SCR Stop Control Register.

SDDQM SDRAM DQM signals. The SDRAM control machine uses these signals (SDDQM[0–3]) to select specific byte lanes for SDRAM devices on the system bus.

SDRAM A type of DRAM that can deliver bursts of data at very high speeds using a synchronous interface.

set To write a non-zero value to a bit or bit field; the opposite of *clear*. The term *set* can also more generally describe the updating of a bit or bit field.

SIMM Signed immediate value.

Single-Master Bus Mode This mode uses the MSC8113 memory controller as the only 60x-compatible bus master to connect external devices to the bus. In single-master bus mode, the MSC8113 uses the address bus as a memory address bus. Slaves cannot use the 60x-compatible system bus signals because the addresses use memory timing, not address tenure timing. Also see Multi-Master Bus Mode.

SIU System interface unit. Controls system startup and initialization, as well as operation, protection, and the external system bus. The system configuration and protection functions provide various monitors and timers, including the bus monitor, software watchdog timer, periodic interrupt timer, and time counter. The clock synthesizer generates the clock signals for the SIU and other MSC8113 modules.

SIUMCR SIU Module Configuration Register.

slave The device addressed by the master. The slave is identified in the address tenure and is responsible for sourcing or sinking the requested data for the master during the data tenure.

SN Sequence number.

snooping Monitoring addresses driven by a bus master to detect the need for coherency actions.

SNR Signal-to-noise ratio.

SP Stack pointer.

split-transaction A transaction with separate request and response tenures.

SPLL System PLL.

SPLL MF SPLL multiplication factor.

SPLL PDF SPLL pre-division factor.

SQBus System bus for the QBus connecting the SC140 cores to the system bus, local bus, and IPBus.

SR Status Register

SRAM Static random access memory. Contrast with dynamic random access memory (DRAM). The dynamic nature of the circuits for DRAM require data to be written back after being read, hence the difference between the access time and the cycle time and also the need to refresh. SRAMs use more circuits per bit to prevent the information from being disturbed when read. Thus, unlike DRAMs, there is no difference between access time and cycle time, and there is no need to refresh SRAM. In DRAM designs, the emphasis is on capacity, while SRAM designs are concerned with both capacity and speed.

$\overline{\text{SRESET}}$ Soft reset signal.

SWR Software Watchdog Register.

SWSR Software Service Register.

SWT Software watchdog timer.

SWTE Software watchdog timer enable signal.

SYNC	Synchronization character.
SYPCR	System Protection Control Register.
T1	Digital transmission link with a capacity of 1.544 Mbps. *See also* E1.
TA	Transfer acknowledge signal. Assertion of this signal indicates that a data beat is valid on the system bus.
TAP	Test access port (**IEEE** Std. 1149.1).
TASR	Tag Array Status Register.
TBST	Bus transfer burst signal. The bus master asserts this pin to indicate that the current transaction is a burst transaction (transfers eight words).
TCn	Transfer code signals (TC[0–2]).
TCFRAn	Timer Configuration Registers A
TCFRBn	Timer Configuration Registers B
TCK	Test clock signal for the TAP.
TCMPA*x*	Timer Compare Registers A 0–15
TCMPB*x*	Timer Compare Registers B 0–15
TCNRA*x*	Timer Count Register A 0–15
TCNRB*x*	Timer Count Register B 0–15
TCRA*x*	Timer Control Registers A 0–15
TCRB*x*	Timer Control Registers B 0–15
TDI	Test data input signal.
TDM	Time-division multiplexing.
TDM*x*ACR	TDM 0–3 Adaptation Control Register.
TDM*x*ASDR	TDM 0–3 Adaptation Sync Distance Registers.
TDM*x*ASR	TDM 0–3 Adaptation Status Registers.
TDM*x*GIR	TDM 0–3 General Interface Registers.
TDM*x*RCLK	TDM 0–3 receive clocks (TDM0RCLK–TDM3RCLK).
TDM*x*RCPR_n	TDM 0–3 Receive Channel Parameter Register n

TDM*x*RCR	TDM 0–3 Receive Control Registers.
TDM*x*RDAT	TDM 0–3 receive data signals (TDM0RDAT–TDM3RDAT).
TDM*x*RDBDR	TDM 0–3 Receive Data Buffers Displacement Register.
TDM*x*RDBFT	TDM 0–3 Receive Data Buffers First Threshold.
TDM*x*RDBS	TDM 0–3 Receive Data Buffer Size.
TDM*x*RDBST	TDM 0–3 Receive Data Buffers Second Threshold.
TDM*x*RER	TDM 0–3 Receive Event Register.
TDM*x*RFP	TDM 0–3 Receive Frame Parameters.
TDM*x*RGBA	TDM 0–3 Receive Global Base Addresses.
TDM*x*RIER	TDM 0–3 Receive Interrupt Enable Registers
TDM*x*RIR	TDM 0–3 Receive Interface Registers.
TDM*x*RNB	TDM 0–3 Receive Number of Buffers.
TDM*x*RSR	TDM 0–3 Receive Status Registers.
TDM*x*RSYN	TDM 0–3 receive sync signals (TDM0RSYN–TDM3RSYN).
TDM*x*TCLK	TDM 0–3 transmit clocks (TDM0TCLK–TDM3TCLK).
TDM*x*TCPR_n	TDM 0–3 Transmit Channel Parameter Register n
TDM*x*TCR	TDM 0–3 Transmit Control Registers.
TDM*x*TDAT	TDM 0–3 transmit data signals (TDM0TDAT–TDM3TDAT).
TDM*x*TDBDR	TDM 0–3 Transmit Data Buffers Displacement Register.
TDM*x*TDBFT	TDM 0–3 Transmit Data Buffers First Threshold.
TDM*x*TDBS	TDM 0–3 Transmit Data Buffer Size.
TDM*x*TDBST	TDM 0–3 Transmit Data Buffers Second Threshold.
TDM*x*TER	TDM 0–3 Transmit Event Register.
TDM*x*TFP	TDM 0–3 Transmit Frame Parameters.
TDM*x*TGBA	TDM 0–3 Transmit Global Base Address.
TDM*x*TIER	TDM 0–3 Transmit Interrupt Enable Registers.

TDM*x*TIR TDM 0–3 Transmit Interface Registers.

TDM*x*TNB TDM 0–3 Transmit Number of Buffers.

TDM*x*TSR TDM 0–3 Transmit Status Registers.

TDM*x*TSYN TDM 0–3 transmit sync signals (TDM0TSYN–TDM3TSYN).

TDO Test data output signal.

$\overline{\text{TEA}}$ Transfer error acknowledge signal. Assertion indicates a failure of the current data tenure transaction.

TEA Transfer error address.

tenure The period of bus mastership. For MSC8113, there can be separate address bus tenures and data bus tenures.

TERA Timer Event Register A

TERB Timer Event Register B

TESCR*x* System Bus Transfer Error Status and Control Register 1–2

TGCR*x* Timer General Configuration Registers A and B

TIER*x* Timer Interrupt Enable Registers A and B

TIMER*x* Timer signals (TIMER[0–3]).

TMCLK Timer clock signal.

TMCNT Time Counter Register.

TMCNTAL Time Counter Alarm Register.

TMCNTSC Time Counter Status and Control Register.

TMS Test mode select signal for the TAP.

transaction A complete exchange between two bus devices. A typical transaction is composed of an address tenure and a data tenure, which may overlap or occur separately from the address tenure. A transaction can minimally consist of an address tenure alone.

$\overline{\text{TRST}}$ Test reset signal.

$\overline{\text{TS}}$ Bus transfer start signal. The current bus master asserts this signal to indicate the start of a new bus tenure.

TSRA Timer Status Register A

TSRB Timer Status Register B

TSZ*x* Transfer size signal (TSZ[0–3]). The system bus master drives these pins with a value indicating the amount of bytes transferred in the current transaction.

TT*x* Bus transfer type signal (TT[0–4]). The system bus master drives these pins during the address tenure to specify the type of the transaction.

Tx Transmit.

UART Universal asynchronous receiver/transmitter. A serial communications interface.

UPM User-programmable machine. The MSC8113 memory controller has three UPMs. The UPMs support address multiplexing of the 60x-compatible system bus, refresh timers, and generation of programmable control signals for row address and column address strobes to allow for a glueless interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can generate different timing patterns for the control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access request. Refresh timers are also available to periodically generate user-defined refresh cycles.

URXD UART receive data line.

UTXD UART transmit data line.

VA Virtual address.

VAB Vector address bus.

VBA Vector Base Address Register

VBASR Valid Bit Array Status Register

V_{CCSYN} Input power for PLLs.

V_{DD} Input power for the SC140 cores.

V_{DDH} Input power for I/O lines.

VIGR Virtual Interrupt Generation Register.

VISR Virtual Interrupt Status Register.

VLES	Variable-length execution set.
VNMIGR	Virtual NMI Generation Register.
VR	Version Register.
VRC	Vertical redundancy checking.
wait state	A period of time when a bus does nothing but wait. Wait states are used to synchronize circuitry or devices operating at different speeds so that they seem to be operating at the same speed.
WARA	Write-after-read atomic bus operation.
WB	Write buffer.
WBCR	Write-Back Control Register.
WBFR	Write-Back Flush Register.
word	The MSC8113 DSP core is a 16-bit processor, so a word in the core portion of the MSC8113 is 16 bits. For the SIU portion of the MSC8113 device, a word equals 32 bits.
XDBA	Internal SC140 core memory data bus 1.
XDBB	Internal SC140 core memory data bus 2.
***x*DMTEA**	DMA Transfer Error Address Registers.
***x*DMTER**	DMA Transfer Error Requestor Number Registers.

MSC8113 Boot Code

C

```
STACK_ADDR      equ     $01076f40

BASE_QBUS_8 equ $00f08000
BASE_QBUS_C equ $00f0c000
BASE_IP     equ $01f80000
BASE_IP_8   equ $01f8c000
BASE_IP_B   equ $01fbc000

BASE_ROM_ADDRESS equ $01077000
BASE_EXEPTION_TABLE equ $01077000

EXTERNAL_MEM_BOOT_TABLE equ $fe000110

;; HOST , dsp side
;BASE_QBUS_C
CIDR     equ $3ff0
QBUSMR1  equ $3f04
IFUR     equ $3f60
ICCR     equ $3c00
ICCMR    equ $3c02
IFUR_ADDR equ $00f0ff60
WBCR_ADDR equ $00f0ff82

;BASE_QBUS_8
ELIRA    equ $1c00
ELIRD    equ $1c18
ELIRF    equ $1c28
IPRB     equ $1c38
IPRB_ADDR equ $00f09c38
LICAICR1 equ $2c08
LICAICR2 equ $2c10
LICAICR3 equ $2c18
LICBICR0 equ $2c40
LICBICR1 equ $2c48
LICBICR2 equ $2c50
LICAISR  equ $2c28
LICBISR  equ $2c68
LICBIER  equ $2c60

;; EONCE
EE_CTRL equ $effe18

;; SIU Registers , according to IMMR.ISB , r6+10000
SIUMCR equ $0
PPC_ACR equ $28
PPC_ALRH equ $2c
PPC_ALRL equ $30
LCL_ACR equ $34
LCL_ALRH equ $38
LCL_ALRL equ $3c
BR9    equ $148
OR9    equ $14c
BR10   equ $150
OR10   equ $154
BR11   equ $158
OR11   equ $15c
MAR    equ $168
MCMR   equ $178
MDR    equ $188
RSR    equ $c90
```

```
; BASE_IP_B
VIGR     equ    $0
VISR     equ    $8
GICR     equ    $18
TCRA0    equ    $1fbf100
TCRB0    equ    $1fbf500
TCFRA0   equ    $1fbf000
TCFRB0   equ    $1fbf400
TCMPA0   equ    $1fbf080
TCMPB0   equ    $1fbf480
;;; GPIO
PODR     equ    $200
PDAT     equ    $208
PDAT_ADDR equ   $01fbc208
PDIR     equ    $210
PAR      equ    $218
PSOR     equ    $220
;;; DSI
DIBAR9   equ    $2010
DIBAR10  equ    $2018
DIBAR11  equ    $2020
DIAMR9   equ    $2028
DIAMR10  equ    $2030
DIAMR11  equ    $2038
DCIR     equ    $2040
;;; UART
; SCI Baud-Rate Register
SCIBR    equ    $1000
; SCI Control Register
SCICR    equ    $1008
; SCI Status Register
SCISR     equ    $1010
; SCI Data Register
SCIDR     equ    $1018
; SCI Data Direction Register
SCIDDR    equ    $1028

;;; ETH
MACCFG2   equ    $3afc
DMA_MR    equ    $3fc8
MIIMCFG   equ    $3ae0

; BASE_IP_8
;TDM3 Receive/Transmit General Interface Register
TDM3GIR  equ    $3ff8
;TDMX Receive/Transmit Interface Registers
TDM3RIR  equ    $3ff0
TDM3TIR  equ    $3fe8

;TDM3 Receive/Transmit Frame Parametera , according to the adaption procedure
TDM3RFP  equ    $3fe0
TDM3TFP  equ    $3fd8

;TDM3 Recieve/Transmit Data Buffer size
TDM3RDBS  equ    $3fd0
TDM3TDBS  equ    $3fc8

;TDM3 Recieve/Transmit Global Base Address
TDM3RGBA  equ    $3fc0 ;;; set to 0x01076e00 + ISBSEL+2M
TDM3TGBA  equ    $3fb8 ;;; set to 0x01076f00 + ISBSEL+2M

;TDM3 Adaption Control Register
TDM3ACR   equ    $3fb0
;TDM3 Recieve/Transmit Control Register
TDM3RCR   equ    $3fa8
TDM3TCR   equ    $3fa0

;TDM3 Recieve/Transmit Data Buffers First Treshhold
TDM3RDBFT equ    $3f98
TDM3TDBFT equ    $3f90
;TDM3 Recieve/Transmit Data Buffers Second Treshhold
TDM3RDBST equ    $3f88
TDM3TDBST equ    $3f80
```

```
;TDM3 Recieve/Transmit Channel Parameter Register n
TDM3RCPR_0 equ  $1000
TDM3TCPR_BASE equ  $2800 ;;; depend on CHIP-ID
TDM3RCPR_BASE equ  $1000

TDM3LOCALMEM_BASE equ $1800

;TDM3 Recieve/Transmit Interrupt Enable Register
TDM3RIER equ  $3f78
TDM3TIER equ  $3f70

;TDM3 Adaption Sync Distance Register
TDM3ASDR equ  $3f68
;TDM3 Recieve/Transmit Data Buffers Displacement Register
TDM3RDBDR equ  $3f60
TDM3RDBDR_ADDR equ $01f8ff60
TDM3TDBDR equ  $3f58
TDM3TDBDR_ADDR equ $01f8ff58
;TDM3 Recieve/Transmit Number of Buffers
TDM3RNB equ  $3f50
TDM3TNB equ  $3f48

;TDM3 Recieve/Transmit Event Register
TDM3RER equ  $3f40
TDM3RER_ADDR equ $01f8ff40
TDM3TER equ  $3f38
TDM3TER_ADDR equ $01f8ff38

;TDM3 Adaption Status Register
TDM3ASR equ  $3f30

;TDM3 Recieve/Transmit Status Register
TDM3RSR equ  $3f28
TDM3TSR equ  $3f20

;;; LOGIC STATE defines , by state2equ script
LOGIC_STATE_1           equ  $7972
LOGIC_STATE_1_T         equ  $797A
LOGIC_STATE_2_1         equ  $7992
LOGIC_STATE_2_1_T       equ  $799E
LOGIC_STATE_2_2         equ  $79A8
LOGIC_STATE_2_2_T       equ  $79B4
LOGIC_STATE_2_3         equ  $79BE
LOGIC_STATE_2_3_T       equ  $79CA
LOGIC_STATE_3           equ  $79D4
LOGIC_STATE_3_T         equ  $79EC
LOGIC_STATE_4           equ  $79FA
LOGIC_STATE_4_T         equ  $7A0A
LOGIC_STATE_5           equ  $7A14
LOGIC_STATE_6_1         equ  $7A30
LOGIC_STATE_6_2         equ  $7A40
LOGIC_STATE_6_3         equ  $7A52
LOGIC_STATE_7_1         equ  $7A64
LOGIC_STATE_7_2         equ  $7A70
LOGIC_STATE_7_3         equ  $7A7E
LOGIC_STATE_7_4         equ  $7A8C
LOGIC_STATE_8_1         equ  $7AE8
LOGIC_STATE_8_2         equ  $7AF4
LOGIC_STATE_9           equ  $7B3C
LOGIC_STATE_9_C         equ  $7B28
LOGIC_STATE_10_1        equ  $7B98
LOGIC_STATE_10_2        equ  $7BA4
CALC_CRC_ADDR           equ  $1077B4E
CRC_TABLE_ADDR          equ  $1077C74
NEXT_BYTE_ADDR          equ  $1077B7A
I2C_MEM_WRITE_ADDR      equ  $1077F84

I2C_SCL_PERIOD_ADDR     equ  $01076f28
I2C_HEADER_ADDR         equ  $01076f10
I2C_START_ADDR          equ  $00070020
VIRTUAL_REG_ADDR        equ  $01076f20
SIUMCR_IMAGE_ADDR       equ  $01076f30
```

```
SCL_SDA_01          equ $4000
SCL_SDA_10          equ $8000
SCL_SDA_11          equ $c000

;;;;; 50KBps @ 500M ratio 1:3
SCL_HIGH_PERIOD     equ $0040
SCL_LOW_HALF_PERIOD equ $0100

HD_STA_TIME         equ $0021 ;;; 5uS   @ 500M ratio 1:3
BUF_TIME            equ $0042 ;;; 10uS  @ 500M ratio 1:3
BUS_FREE_TIME       equ $190b ;;; 1mS   @ 500M ratio 1:3
HALF_BUS_FREE_TIME  equ $c85  ;;; 500uS @ 500M ratio 1:3

 ;the registers to be used are :
                    ;      r2    BASE_IP_B equ $01fbc000
                    ;      r3    BASE_IP_8 equ $01f8c000
                    ;      r4    BASE_QBUS_8 equ $00f08000
                   ;      r5    for holding the SRAM BASE MEM , bank 11 , 0x0200000 + 2M*ISBSEL
                    ;      r6    for holding the IMMR.ISB
                    ;      r7    BASE_QBUS_C equ $00f0c000
P:01077000                    org     p:$0000+BASE_ROM_ADDRESS
                    start
P:01077000 31                 move.l #BASE_EXEPTION_TABLE,vba       ;  init vba
           63
           30
           00
           81
           07
   ; initilize EE1 pin to acknowledge debug
P:01077006 20                 move.w #$fff7,d0
           E0
           9F
           F7
P:0107700A 00                 move.w d0,EE_CTRL
           E0
           3E
           18
           80
           EF
   ; initilize QBUS bank 1 mask register
   ; init r7  to BASE_QBUS and BASE_IP
P:01077010 3F                 move.l #BASE_QBUS_C,r7
           C0
           20
           00
           80
           F0
P:01077016 20                 move.w #$ff80,d0
           E0
           9F
           80
P:0107701A 00                 move.w d0,(r7+QBUSMR1)
           27
           9F
           04
   ; Set Stack according to CORE-ID
P:0107701E 30                 move.l #STACK_ADDR,d0
           60
           2F
           40
           81
           07
P:01077024 C2                 move.w #$0030,d2
           B0
P:01077026 6E                 clr d4
           10
P:01077028 11                 move.w (r7+CIDR),d1
           27
           9F
           F0
P:0107702C 32                 extractu #2,#0,d1,d4
           CC
           80
           80
```

```
P:01077030 68                  imac d2,d4,d0                         ; (d0+d4.l*d2.l)->d0
           08
P:01077032 C8                  move.l d0,r0
           40
P:01077034 90                  nop
           C0
P:01077036 E7                  tfra r0,sp                            ; init ESP
           E8
P:01077038 6C                  clr d0
           10
P:0107703A 40                  move.l d0,(r0)
           90
   ;init r2 , r3 , r4 for BASE_QBUS and BASE_IP
P:0107703C 3C                  move.l #BASE_QBUS_8,r4
           80
           20
           00
           80
           F0
P:01077042 3A                  move.l #BASE_IP_B,r2
           C0
           20
           00
           81
           FB
P:01077048 3B                  move.l #BASE_IP,r3
           00
           20
           00
           81
           F8
   ; initialize LIC-TDM Interrupts to edge
P:0107704E 30                  move.l #$44044044,d0
           48
           20
           44
           84
           04
P:01077054 00                  move.l d0,(r4+LICAICR1)
           A4
           8C
           08
P:01077058 DC                  lsrr #$4,d0                           ; $04404404 -> d0
           24
P:0107705A 00                  move.l d0,(r4+LICAICR2)
           A4
           8C
           10
P:0107705E 30                  move.l #$40440440,d0
           08
           24
           40
           80
           44
P:01077064 00                  move.l d0,(r4+LICAICR3)
           A4
           8C
           18
   ; initialize LIC-TIMERS Interrupts to edge
P:01077068 30                  move.l #$44444444,d0
           48
           24
           44
           84
           44
P:0107706E 00                  move.l d0,(r4+LICBICR2)
           A4
           8C
           50
   ; initialize LIC-VIRQ Interrupts to edge
P:01077072 08                  bmclr #$ffff,d0.l                     ; $44440000 -> d0
           E0
           BF
           FF
```

```
P:01077076 00                  move.l d0,(r4+LICBICR1)
           A4
           8C
           48
P:0107707A 88                  bra <Fmain_cont_1
           0B
                     ;---------- illegal exeption offset 0x80 ----------
P:01077080                     org     p:$0080+BASE_EXEPTION_TABLE
                     illegal_exeption
P:01077080 9E                  debug
           70
P:01077082 9F                  rte
           73
                     Fmain_cont_1
P:01077084 D8                  asrw d0,d0                            ; $00004444 -> d0
           18
P:01077086 00                  move.l d0,(r4+LICBICR0)
           A4
           8C
           40
   ; initialize ELIRF - Edge/Level-Triggered Interrupt Register F
P:0107708A C0                  move.w #$0008,d0
           88
P:0107708C 00                  move.w d0,(r4+ELIRF)
           04
           9C
           28
P:01077090 89                  bra <Fmain_cont_2
           49
                     i2c_txrx_byte
P:01077092 6F                  clr d6
           10
P:01077094 24                  move.w #$80,d4
           00
           80
           80
                     byte_loop
P:01077098 23                  bsr i2c_txrx_bit
           18
           80
           6C
P:0107709C 92                  ift  &  rts
           C2
           9F
           71
P:010770A0 76                  asr d4,d4
           44
P:010770A2 6F                  asl d6,d6
           5E
P:010770A4 66                  tsteq d4
           69
P:010770A6 85                  bf <byte_loop
           F3
P:010770A8 0A                  bmchg #$1,d7.l
           07
           A0
           01
P:010770AC 23                  bsr i2c_txrx_bit
           18
           80
           58
P:010770B0 77                  asr d6,d6
           46
P:010770B2 0A                  bmchg #$1,d7.l
           07
           A0
           01
P:010770B6 9F                  rts
           71
                     ;---------- trap exeption offset 0xc0 ----------
P:010770C0                     org     p:$00c0+BASE_EXEPTION_TABLE
                     debug_exeption
P:010770C0 9E                  debug
           70
```

```
P:010770C2 9F                  rte
           73
                     i2c_sample_gpio
P:010770C4 01                  moveu.w PDAT_ADDR,d1
           C7
           22
           08
           81
           FB
P:010770CA 34                  and d8,d1
           00
           A8
           00
           DC
           80
                     loop_sample
P:010770D0 02                  moveu.w PDAT_ADDR,d2
           C7
           22
           08
           81
           FB
P:010770D6 34                  and d8,d2
           00
           A8
           00
           DD
           00
P:010770DC 71                  cmpeq d1,d2
           61
P:010770DE 74                  tfr d2,d1
           D2
P:010770E0 85                  bf <loop_sample
           F1
P:010770E2 9F                  rts
           71
                     i2c_assert_stop
P:010770E4 6E                  clr d4
           10
P:010770E6 6F                  clr d6
           10
P:010770E8 6F                  clr d7
           90
P:010770EA 23                  bsr i2c_txrx_bit
           18
           80
           1A
P:010770EE 92                  ift  &  rts
           C2
           9F
           71
P:010770F2 34                  move.w d8,(r9)
           A0
           A0
           00
           40
           11
P:010770F8 88                  bra <wait_sda_high
           C5
                     ;---------- overflow exeption offset 0x100 ----------
P:01077100                     org     p:$0100+BASE_EXEPTION_TABLE
                     overflow_exeption
P:01077100 9E                  debug
           70
P:01077102 9F                  rte
           73
                     i2c_txrx_bit
P:01077104 36                  bmclr.w #SCL_SDA_10,(r9)
           80
           A0
           00
           10
           89
           A0
           00
P:0107710C 34                  tfr d15,d0
           00
           A8
           00
           74
           57
                     lperiod_loop_1
P:01077112 64                  deceq d0
           6D
P:01077114 85                  bf <lperiod_loop_1
           FF
P:01077116 74                  tfr d5,d0
           55
P:01077118 DC                  and d4,d0
           04
P:0107711A DC                  or d7,d0
           1F
P:0107711C 64                  tsteq d0
           69
P:0107711E 6C                  clr d1
           90
P:01077120 94                  iff  &  move.w #SCL_SDA_01,d1
           C3
           21
           40
           80
           00
P:01077126 34                  move.w d1,(r9)
           80
           A0
           00
           41
           11
P:0107712C 34                  tfr d15,d0
           00
           A8
           00
           74
           57
                     lperiod_loop_2
P:01077132 64                  deceq d0
           6D
P:01077134 85                  bf <lperiod_loop_2
           FF
P:01077136 09                  or #SCL_SDA_10,d1.l
           81
           A0
           00
P:0107713A C8                  move.l d1,r0
           41
P:0107713C 34                  move.w d1,(r9)
           80
           A0
           00
           41
           11
                     wait_scl_high
P:01077142 83                  bsr i2c_sample_gpio
           83
P:01077144 35                  and #SCL_SDA_11,d2,d3
           DB
           80
           00
P:01077148 08                  and #SCL_SDA_10,d2.l
           62
           BF
           FF
P:0107714C 65                  tsteq d2
           69
P:0107714E 81                  bt <wait_scl_high
           F5
P:01077150 0A                  bmchg #SCL_SDA_01,d3.l
           43
           A0
           00
P:01077154 34                  tfr d14,d0
           00
           A8
           00
           74
           56
                     hperiod_loop
P:0107715A 83                  bsr i2c_sample_gpio
           6B
P:0107715C 0D                  bmtsts #SCL_SDA_10,d2.l
           82
           A0
           00
P:01077160 34                  and d8,d2
           00
           A8
           00
           DD
           00
P:01077166 92                  iff  &  rts
           C3
           9F
           71
P:0107716A 71                  cmpeq d3,d2
           63
P:0107716C 90                  nop
           C0
P:0107716E 92                  ift  &  rts
           C2
           9F
           71
P:01077172 34                  cmpeq d8,d2
           00
           A8
           00
           71
           60
P:01077178 90                  nop
           C0
P:0107717A 94                  ift  &  bmset #$1,d6.l
           C2
           09
           06
           A0
           01
P:01077180 64                  deceq d0
           6D
P:01077182 85                  bf <hperiod_loop
           D9
P:01077184 0C                  bmtstc #$1,d7.l
           07
           A0
           01
P:01077188 90                  nop
           C0
P:0107718A 92                  iff  &  rts
           C3
           9F
           71
P:0107718E 0C                  bmtstc #$1,d6.l
           06
           A0
           01
P:01077192 C8                  move.l r0,d1
           49
P:01077194 94                  iff  &  move.w #SCL_SDA_10,d0
           C3
           20
           80
           80
           00
P:0107719A 34                  ift  &  tfr d8,d0
```

```
           02
           A8
           00
           74
           50
P:010771A0 70                 cmpeq d0,d1
           E0
P:010771A2 9F                 rts
           71
                    i2c_assert_start
P:010771A4 36                 move.w #SCL_SDA_10,(r9)
           80
           A0
           00
           19
           89
           A0
           00
P:010771AC C0                 move.w #HD_STA_TIME,d0
           A1
                    start_loop
P:010771AE 83                 bsr i2c_sample_gpio
           17
P:010771B0 0C                 bmtstc #SCL_SDA_10,d2.l
           82
           A0
           00
P:010771B4 80                 bt <start_rts
           07
P:010771B6 64                 deceq d0
           6D
P:010771B8 85                 bf <start_loop
           F7
                    start_rts
P:010771BA 9F                 rts
           71
                    wait_sda_high
P:010771BC 83                 bsr i2c_sample_gpio
           09
P:010771BE 0C                 bmtstc #SCL_SDA_01,d2.l
           42
           A0
           00
P:010771C2 81                 bt <wait_sda_high
           FB
P:010771C4 20                 move.w #BUF_TIME,d0
           00
           80
           42
                    stop_loop
P:010771C8 2B                 bsr i2c_sample_gpio
           FF
           9E
           FD
P:010771CC 0C                 bmtstc #SCL_SDA_01,d2.l
           42
           A0
           00
P:010771D0 80                 bt <stop_rts
           07
P:010771D2 64                 deceq d0
           6D
P:010771D4 85                 bf <stop_loop
           F5
                    stop_rts
P:010771D6 9F                 rts
           71
                    Fmain_cont_2
   ; find SIU_BASE(r6) by ISBSEL at emr register
P:010771D8 C0                 move.l emr,d1
           69
P:010771DA 32                 extractu #3,#19,d1,d5
           CD
           80
           D3
   ; xor 3'b100 to recover original isbsel
P:010771DE 0A                  eor #$4,d5.l
           05
           A0
           04
P:010771E2 90                  nop
           C0
P:010771E4 D2                  cmpeq.w #$0,d5
           A0
P:010771E6 90                  nop
           C0
P:010771E8 94                  ift  &  moveu.w #$f000,d1.l
           C2
           19
           E1
           B0
           00
P:010771EE D2                  cmpeq.w #$1,d5
           A1
P:010771F0 90                  nop
           C0
P:010771F2 94                  ift  &  moveu.w #$f0f0,d1.l
           C2
           19
           E1
           B0
           F0
P:010771F8 D2                  cmpeq.w #$2,d5
           A2
P:010771FA 90                  nop
           C0
P:010771FC 94                  ift  &  moveu.w #$ff00,d1.l
           C2
           19
           E1
           BF
           00
P:01077202 D2                  cmpeq.w #$3,d5
           A3
P:01077204 90                  nop
           C0
P:01077206 94                  ift  &  moveu.w #$fff0,d1.l
           C2
           19
           E1
           BF
           F0
P:0107720C D2                  cmpeq.w #$6,d5
           A6
P:0107720E 90                  nop
           C0
P:01077210 94                  ift  &  moveu.w #$0f00,d1.l
           C2
           19
           01
           AF
           00
P:01077216 D2                  cmpeq.w #$7,d5
           A7
P:01077218 90                  nop
           C0
P:0107721A 94                  ift  &  moveu.w #$0ff0,d1.l
           C2
           19
           01
           AF
           F0
P:01077220 D2                  cmpeq.w #$4,d5
           A4
P:01077222 64                  inc d1
           EF
P:01077224 92                  ift  &  debug
           C2
```

```
                  9E
                  70
       P:01077228 D2                  cmpeq.w #$5,d5
                  A5
       P:0107722A D8                  aslw d1,d1
                  91
       P:0107722C CE                  move.l d1,r6                          ; r6 = IMMR.ISB + $10000
                  41
       P:0107722E 92                  ift  &  debug
                  C2
                  9E
                  70
       P:01077232 51                  move.l (r6),d1                        ; siumcr
                  96
       P:01077234 32                  extractu #3,#22,d1,d3
                  CB
                  80
                  D6
          ; SIU-MEMC INIT set Bank 11 base by ISBSEL
       P:01077238 21                  move.w #$0200,d1                      ; base address for sram is
0x0200_0000
                  00
                  82
                  00
       P:0107723C C2                  move.w #$0020,d2                      ; 2M gap
                  A0
       P:0107723E 68                  imac d2,d5,d1                         ; (d1+d5.l*d2.l)->d1
                  89
       P:01077240 D8                  aslw d1,d1                            ; d1<<16
                  91
       P:01077242 CD                  move.l d1,r5                          ; keep base address of bank
11 in r5
                  41
       P:01077244 D2                  cmpeq.w #$0,d4
                  20
       P:01077246 80                  bt <Fmain_core0
                  CD
                            wait_virq
       P:01077248 20                  move.w #$0500,d0
                  00
                  85
                  00
       P:0107724C 00                  move.l d0,(r4+ELIRD)
                  84
                  9C
                  18
       P:01077250 30                  move.l #$00200000,d0
                  00
                  20
                  00
                  80
                  20
       P:01077256 00                  move.l d0,(r4+LICBIER)
                  A4
                  8C
                  60
       P:0107725A 35                  move.l #$000c0000,sr
                  03
                  20
                  00
                  80
                  0C
          ; Lock ICACHE
       P:01077260 6D                  clr d2
                  10
       P:01077262 19                  moveu.w #$f003,d1.l
                  E1
                  B0
                  03
       P:01077266 01                  move.w d1,(r7+ICCR)
                  27
                  9C
                  00
       P:0107726A 90                  nop
```

```
           C0
P:0107726C 90                  nop
           C0
   ; Flush ICACHE
P:0107726E 02                  move.w d2,(r7+ICCMR)
           27
           9C
           02
P:01077272 90                  nop
           C0
P:01077274 90                  nop
           C0
P:01077276 9F                  wait
           78
P:01077278 90                  nop
           C0
P:0107727A 08                  bmclr #$2,d1.l
           01
           A0
           02
   ; Unlock ICACHE
P:0107727E 01                  move.w d1,(r7+ICCR)
           27
           9C
           00
P:01077282 90                  nop
           C0
P:01077284 00                  move.l d0,(r4+LICBISR)
           A4
           8C
           68
P:01077288 6C                  clr d0
           10
P:0107728A 00                  move.l d0,(r4+ELIRD)
           84
           9C
           18
P:0107728E 00                  move.l d0,(r4+LICBIER)
           A4
           8C
           60
P:01077292 35                  move.l #$00e40000,sr
           03
           20
           00
           80
           E4
P:01077298 90                  nop
           C0
P:0107729A D2                  cmpeq.w #$0,d4
           20
P:0107729C 80                  bt <wake_core123_soft_reset
           4D
P:0107729E 88                  bra <boot_jmp0
           65
                     wake_core123
P:010772A0 36                  bmtstc #$0010,d11.h
           20
           A0
           00
           0C
           13
           A0
           10
P:010772A8 80                  bt <ecc_off
           0D
P:010772AA 00                  moveu.b VIRTUAL_REG_ADDR,d0
           65
           2F
           20
           81
           07
P:010772B0 D0                  cmpeq.w #$f,d0
           2F
```

```
P:010772B2 80                    bt <wake_core123_soft_reset
           37
                       ecc_off
P:010772B4 01                    move.l SIUMCR_IMAGE_ADDR,d1
           66
           2F
           30
           81
           07
P:010772BA 0D                    bmtsts #$0200,d1.l
           01
           A2
           00
P:010772BE 80                    bt <wake_core123_soft_reset
           2B
P:010772C0 38                    move.l #BASE_QBUS_C,r0
           C0
           20
           00
           80
           F0
P:010772C6 6D                    clr d2
           10
   ; Lock ICACHE
P:010772C8 19                    moveu.w #$f003,d0.l
           E0
           B0
           03
P:010772CC 00                    move.w d0,(r0+ICCR)
           20
           9C
           00
P:010772D0 90                    nop
           C0
P:010772D2 90                    nop
           C0
   ; Flush ICACHE
P:010772D4 02                    move.w d2,(r0+ICCMR)
           20
           9C
           02
P:010772D8 90                    nop
           C0
P:010772DA 90                    nop
           C0
P:010772DC 41                    move.l d1,(r6)
           96
P:010772DE 08                    bmclr #$2,d0.l
           00
           A0
           02
   ; Unlock ICACHE
P:010772E2 00                    move.w d0,(r0+ICCR)
           20
           9C
           00
P:010772E6 90                    nop
           C0
                       wake_core123_soft_reset
P:010772E8 20                    move.w #$0101,d0
           00
           81
           01
P:010772EC B0                    move.l d0,(r2+VIGR)
           50
P:010772EE 20                    move.w #$0201,d0
           00
           82
           01
P:010772F2 B0                    move.l d0,(r2+VIGR)
           50
P:010772F4 20                    move.w #$0301,d0
           00
           83
           01
P:010772F8 B0                  move.l d0,(r2+VIGR)
           50
P:010772FA 30                  move.l #$02020200,d0
           00
           22
           00
           82
           02
P:01077300 B0                  move.l d0,(r2+VISR)
           52
                    boot_jmp0
P:01077302 6C                  clr d0
           10
P:01077304 00                  move.w d0,IFUR_ADDR
           E0
           3F
           60
           80
           F0
P:0107730A 90                  nop
           C0
P:0107730C 31                  jmp $0
           04
           20
           00
           80
           00
                    Fmain_core0
   ;initialize GIC-VIRQ Interrupts to edge
P:01077312 30                  move.l #$0f000000,d0
           00
           20
           00
           8F
           00
P:01077318 B0                  move.l d0,(r2+GICR)
           56
   ;initialize ETH
P:0107731A 20                  move.w #$7100,d0
           60
           91
           00
P:0107731E 00                  move.l d0,(r2-MACCFG2)
           C2
           85
           04
P:01077322 30                  move.l #$004d0000,d0
           00
           20
           00
           80
           4D
P:01077328 00                  move.l d0,(r2-DMA_MR)
           C2
           80
           38
P:0107732C C0                  move.w #$3,d0
           83
P:0107732E 00                  move.l d0,(r2-MIIMCFG)
           C2
           85
           20
   ;initialize TDM-RIR, TDM-TIR ;;; r3=01f80000 -> r3=01f8c000
P:01077332 C0                  move.w #$1,d0
           81
P:01077334 00                  move.w d0,(r3+TDM3RIR)
           23
           9F
           F0
P:01077338 00                  move.w d0,(r3+TDM3TIR)
           23
           9F
           E8
```

```
P:0107733C 2B                adda #$4000,r3,r3
           4B
           80
           00
P:01077340 00                move.w d0,(r3+TDM3RIR)
           23
           9F
           F0
P:01077344 00                move.w d0,(r3+TDM3TIR)
           23
           9F
           E8
P:01077348 2B                adda #$4000,r3,r3
           4B
           80
           00
P:0107734C 00                move.w d0,(r3+TDM3RIR)
           23
           9F
           F0
P:01077350 00                move.w d0,(r3+TDM3TIR)
           23
           9F
           E8
P:01077354 2B                adda #$4000,r3,r3
           4B
           80
           00
P:01077358 00                move.w d0,(r3+TDM3RIR)
           23
           9F
           F0
P:0107735C 00                move.w d0,(r3+TDM3TIR)
           23
           9F
           E8
   ; initialize SIU
P:01077360 10                move.w (r6+PPC_ACR),d0
           06
           80
           28
P:01077364 7C                asrr #$8,d0
           68
P:01077366 08                bmclr #$000f,d0.l
           00
           A0
           0F
P:0107736A 09                bmset #$0005,d0.l
           00
           A0
           05
P:0107736E 00                move.b d0,(r6+PPC_ACR)
           16
           80
           28
P:01077372 30                move.l #$da54789e,d0
           78
           38
           9E
           9A
           54
P:01077378 00                move.l d0,(r6+PPC_ALRH)
           86
           80
           2C
P:0107737C 30                move.l #$bfc36012,d0
           70
           20
           12
           BF
           C3
P:01077382 00                move.l d0,(r6+PPC_ALRL)
           86
           80
```

```
                   30
        P:01077386 C0                   move.w #$0003,d0
                   83
        P:01077388 00                   move.b d0,(r6+LCL_ACR)
                   16
                   80
                   34
        P:0107738C 30                   move.l #$04dae3b2,d0
                   E0
                   23
                   B2
                   84
                   DA
        P:01077392 00                   move.l d0,(r6+LCL_ALRH)
                   86
                   80
                   38
        P:01077396 30                   move.l #$fc789156,d0
                   98
                   31
                   56
                   BC
                   78
        P:0107739C 00                   move.l d0,(r6+LCL_ALRL)
                   86
                   80
                   3C
        P:010773A0 10                   move.l  (r6+RSR),d0
                   86
                   8C
                   90
        P:010773A4 0C                   bmtstc #$0001,d0.l
                   00
                   A0
                   01
        P:010773A8 81                   bt <wake_core123_soft_reset
                   41
           ; SIU-MEMC INIT
        P:010773AA 01                   move.l d1,(r2+DIBAR11)
                   A2
                   80
                   20
        P:010773AE 09                   bmset #$00c1,d1.l
                   01
                   A0
                   C1
        P:010773B2 37                   move.l #$ffe00000,d7
                   18
                   20
                   00
                   BF
                   E0
        P:010773B8 07                   move.l d7,(r2+DIAMR11)
                   A2
                   80
                   38
        P:010773BC 07                   move.l d7,(r6+OR11)                     ; SET OR11 ;
l2,l1c0,l1c1,l1c2,l1c3
                   86
                   81
                   5C
        P:010773C0 01                 move.l d1,(r6+BR11)                   ; SET BR11 ; upmc - local-bus
; 0x02000000-0x0217ffff ; 1.5M
                   86
                   81
                   58
        P:010773C4 09                   bmset #$001f,d7.h                       ; $ffff0000 -> d7
                   17
                   A0
                   1F
        P:010773C8 07                   move.l d7,(r2+DIAMR10)
                   A2
                   80
                   30
```

```
       P:010773CC 21                move.w #$021e,d1                      ; base address for efcop
0x021e0000
                  00
                  82
                  1E
       P:010773D0 68                imac d2,d5,d1                         ; (d1+d5.l*d2.l)->d1
                  89
       P:010773D2 D8                aslw d1,d1                            ; d1<<16
                  91
       P:010773D4 01                move.l d1,(r2+DIBAR10)
                  A2
                  80
                  18
       P:010773D8 09               bmset #$1820,d1.l                     ; Valid bit set at the end
of the boot code
                  01
                  B8
                  20
       P:010773DC 09               bmset #$0008,d7.l                     ; $ffff0008 -> d7 - SETA=1
;;; PSDVAL is generated after external logic asserts GTA .
                  07
                  A0
                  08
       P:010773E0 07                move.l d7,(r6+OR10)                   ; SET OR10
                  86
                  81
                  54
       P:010773E4 01              move.l d1,(r6+BR10)                  ; SET BR10 ; gpcm - local-bus
; 0x021e0000-0x021effff ; 64K
                  86
                  81
                  50
       P:010773E8 08                bmclr #$0003,d7.h                     ; $fffc0008 -> d7
                  17
                  A0
                  03
       P:010773EC 08                bmclr #$0008,d7.l                     ; $fffc0000 -> d7
                  07
                  A0
                  08
       P:010773F0 07                move.l d7,(r2+DIAMR9)
                  A2
                  80
                  28
       P:010773F4 09               bmset #$0008,d7.l                     ; $fffc0008 -> d7 - SETA=1
;;; PSDVAL is generated after external logic asserts GTA .
                  07
                  A0
                  08
       P:010773F8 21                move.w #$0218,d1                      ; base address for ip
0x02180000
                  00
                  82
                  18
       P:010773FC 68                imac d2,d5,d1                         ; (d1+d5.l*d2.l)->d1
                  89
       P:010773FE D8                aslw d1,d1                            ; d1<<16
                  91
       P:01077400 01                move.l d1,(r2+DIBAR9)
                  A2
                  80
                  10
       P:01077404 09                bmset #$1821,d1.l                     ; Port Size 32-Bit
                  01
                  B8
                  21
       P:01077408 07                move.l d7,(r6+OR9)                    ; SET OR9 ; ip
                  86
                  81
                  4C
       P:0107740C 01               move.l d1,(r6+BR9)                    ; SET BR9 ; gpcm - local-bus
; 0x02180000-0x021bffff ; 256K
                  86
                  81
```

```
                 48
P:01077410 37                    move.l #$90051240,d7                ; read single-beat
                 10
                 32
                 40
                 90
                 05
P:01077416 07                    move.l d7,(r6+MCMR)
                 86
                 81
                 78
P:0107741A 37                    move.l #$00030040,d7
                 00
                 20
                 40
                 80
                 03
P:01077420 07                    move.l d7,(r6+MDR)
                 86
                 81
                 88
P:01077424 19                    move.w #$0,(r5)
                 0D
                 A0
                 00
P:01077428 7B                    add #$5,d7                          ; $00030045 -> d7
                 C5
P:0107742A 07                    move.l d7,(r6+MDR)
                 86
                 81
                 88
P:0107742E 19                    move.w #$0,(r5)
                 0D
                 A0
                 00
P:01077432 37                    move.l #$90051248,d7                ; read burst
                 10
                 32
                 48
                 90
                 05
P:01077438 07                    move.l d7,(r6+MCMR)
                 86
                 81
                 78
P:0107743C 37                    move.l #$00030c48,d7
                 00
                 2C
                 48
                 80
                 03
P:01077442 07                    move.l d7,(r6+MDR)
                 86
                 81
                 88
P:01077446 19                    move.w #$0,(r5)
                 0D
                 A0
                 00
P:0107744A 7B                    add #$4,d7                          ; $00030c4c -> d7
                 C4
P:0107744C 07                    move.l d7,(r6+MDR)
                 86
                 81
                 88
P:01077450 19                    move.w #$0,(r5)
                 0D
                 A0
                 00
P:01077454 07                    move.l d7,(r6+MDR)
                 86
                 81
                 88
P:01077458 19                    move.w #$0,(r5)
```

```
              0D
              A0
              00
   P:0107745C 37                move.l #$00030044,d7
              00
              20
              44
              80
              03
   P:01077462 07                move.l d7,(r6+MDR)
              86
              81
              88
   P:01077466 19                move.w #$0,(r5)
              0D
              A0
              00
   P:0107746A 67                inc d7                            ; $00030045 -> d7
              EF
   P:0107746C 07                move.l d7,(r6+MDR)
              86
              81
              88
   P:01077470 19                move.w #$0,(r5)
              0D
              A0
              00
   P:01077474 37                move.l #$90051258,d7              ; write  single-beat
              10
              32
              58
              90
              05
   P:0107747A 07                move.l d7,(r6+MCMR)
              86
              81
              78
   P:0107747E 27                move.w #$0040,d7
              00
              80
              40
   P:01077482 07                move.l d7,(r6+MDR)
              86
              81
              88
   P:01077486 19                move.w #$0,(r5)
              0D
              A0
              00
   P:0107748A 7B                add #$5,d7                        ; $0045 -> d7
              C5
   P:0107748C 07                move.l d7,(r6+MDR)
              86
              81
              88
   P:01077490 19                move.w #$0,(r5)
              0D
              A0
              00
   P:01077494 37                move.l #$90051260,d7              ; write burst
              10
              32
              60
              90
              05
   P:0107749A 07                move.l d7,(r6+MCMR)
              86
              81
              78
   P:0107749E 27                move.w #$0c48,d7
              00
              8C
              48
   P:010774A2 07                move.l d7,(r6+MDR)
```

```
                86
                81
                88
    P:010774A6  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774AA  7B              add #$4,d7                       ; $0c4c -> d7
                C4
    P:010774AC  07              move.l d7,(r6+MDR)
                86
                81
                88
    P:010774B0  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774B4  07              move.l d7,(r6+MDR)
                86
                81
                88
    P:010774B8  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774BC  27              move.w #$0044,d7
                00
                80
                44
    P:010774C0  07              move.l d7,(r6+MDR)
                86
                81
                88
    P:010774C4  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774C8  67              inc d7                           ; $0045 -> d7
                EF
    P:010774CA  07              move.l d7,(r6+MDR)
                86
                81
                88
    P:010774CE  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774D2  37              move.l #$9005127c,d7             ; exception
                10
                32
                7C
                90
                05
    P:010774D8  07              move.l d7,(r6+MCMR)
                86
                81
                78
    P:010774DC  37              move.l #$ff000001,d7
                18
                20
                01
                BF
                00
    P:010774E2  07              move.l d7,(r6+MDR)
                86
                81
                88
    P:010774E6  19              move.w #$0,(r5)
                0D
                A0
                00
    P:010774EA  37              move.l #$80011240,d7             ; normal oper. work
                10
                32
```

```
           40
           80
           01
P:010774F0 07                   move.l d7,(r6+MCMR)
           86
           81
           78
                     check_boot
P:010774F4 D1                   cmpeq.w #$3,d3
           A3
P:010774F6 80                   bt <from_tdm_uart_i2c
           49
P:010774F8 D1                   cmpeq.w #$6,d3
           A6
P:010774FA 80                   bt <from_tdm_uart_i2c
           45
P:010774FC D1                   cmpeq.w #$2,d3
           A2
P:010774FE 80                   bt <from_tdm_uart_i2c
           41
P:01077500 D1                   cmpeq.w #$4,d3
           A4
P:01077502 80                   bt <from_tdm_uart_i2c
           3D
P:01077504 11                   move.l (r6+BR10),d1                  ; BR10 register
           86
           81
           50
P:01077508 09                   bmset #$1,d1.l                       ; Valid bit set
           01
           A0
           01
P:0107750C 01                   move.l d1,(r6+BR10)
           86
           81
           50
P:01077510 D1                   cmpeq.w #$0,d3
           A0
P:01077512 80                   bt <external_memory
           0B
P:01077514 D1                   cmpeq.w #$1,d3
           A1
P:01077516 2D                   bt wait_virq
           FF
           9D
           33
P:0107751A 9E                   debug
           70
                     external_memory
P:0107751C 6C                   clr d0
           10
P:0107751E 00                   move.w d0,IFUR_ADDR
           E0
           3F
           60
           80
           F0
P:01077524 90                   nop
           C0
   ;r3 <- d5  ;;; d5 = ISB
P:01077526 CB                   move.l d5,r3
           45
P:01077528 90                   nop
           C0
   ; FIND External memory boot table address FROM ISBSEL
   ;r3 *= 4 so the offset will be in long instead of bytes
   ;r3 = r3<<2
P:0107752A EB                   asl2a r3
           FE
   ;put the address of the external memory boot table in r4
P:0107752C 3C                   move.l #EXTERNAL_MEM_BOOT_TABLE,r4
           18
           21
           10
```

```
              BE
              00
   P:01077532 90                    nop
              C0
   P:01077534 EB                    adda r4,r3
              1C
   P:01077536 90                    nop
              C0
      ;move the address from the table into r3
   P:01077538 5B                    move.l (r3),r3
              93
   P:0107753A 90                    nop
              C0
      ;jump to that address
   P:0107753C 9B                    jmp r3
              61
                          from_tdm_uart_i2c
   P:0107753E 34                    clr d9
              00
              A0
              04
              6C
              90
   P:01077544 34                    clr d11
              00
              A0
              04
              6D
              90
   P:0107754A 38                    move.l d9,VIRTUAL_REG_ADDR
              20
              A0
              00
              01
              62
              2F
              20
              81
              07
   P:01077554 10                    move.l (r2+DCIR),d0
              A2
              80
              40
   P:01077558 36                    extractu #4,#28,d0,d9
              00
              A0
              04
              30
              C9
              81
              1C
   P:01077560 51                    move.l (r6),d1
              96
   P:01077562 01                    move.l d1,SIUMCR_IMAGE_ADDR
              62
              2F
              30
              81
              07
   P:01077568 0E                    bmtset #$0200,d1.l
              01
              A2
              00
   P:0107756C 80                    bt <ecc_cont
              27
      ; Lock ICACHE
   P:0107756E 19                    moveu.w #$f003,d0.l
              E0
              B0
              03
   P:01077572 00                    move.w d0,(r7+ICCR)
              27
              9C
              00
```

```
P:01077576 90                  nop
           C0
P:01077578 90                  nop
           C0
P:0107757A 36                  move.w d11,(r7+ICCMR)
           20
           A0
           00
           03
           27
           9C
           02
P:01077582 90                  nop
           C0
P:01077584 90                  nop
           C0
P:01077586 41                  move.l d1,(r6)
           96
P:01077588 08                  bmclr #$2,d0.l
           00
           A0
           02
   ; Unlock ICACHE
P:0107758C 00                  move.w d0,(r7+ICCR)
           27
           9C
           00
P:01077590 90                  nop
           C0
                     ecc_cont
P:01077592 D1                  cmpeq.w #$4,d3
           A4
P:01077594 25                  bt from_i2c
           18
           87
           E0
                     from_tdm_uart
P:01077598 6F                  clr d7
           90
P:0107759A 2C                  move.w #LOGIC_STATE_1,r4
           60
           99
           72
P:0107759E 38                  move.l #CRC_TABLE_ADDR,r10
           20
           A0
           00
           3A
           60
           3C
           74
           81
           07
P:010775A8 08                  bmclr #$0200,r5.h
           1D
           A2
           00
P:010775AC 36                  move.w #$7abf,d15
           20
           A0
           00
           27
           60
           9A
           BF
P:010775B4 34                  tfr d15,d0
           00
           A8
           00
           74
           57
P:010775BA 34                  eor d9,d0
           00
           A8
           00
           DC
           11
P:010775C0 0D                  bmtsts #$0080,d0.l
           00
           A0
           80
P:010775C4 31                  and #$0007f,d0,d0
           18
           80
           7F
P:010775C8 C8                  move.l d0,r0
           40
P:010775CA 34                  asrr #$8,d15
           00
           A0
           04
           7F
           E8
P:010775D0 34                  move.w (r10+r0),d0
           80
           A0
           00
           A0
           90
P:010775D6 34                  eor d0,d15
           00
           A0
           04
           DF
           90
P:010775DC 36                  ift  &  eor #$a001,d15.l
           22
           A0
           00
           0A
           A7
           A0
           01
P:010775E4 36                  and #$0ffff,d15,d15
           00
           A8
           04
           3F
           FF
           9F
           FF
P:010775EC 36                  move.l #$55660000,d6
           08
           20
           00
           95
           66
P:010775F2 36                  insert #$4,#$8,d9,d6
           00
           A8
           00
           32
           EE
           81
           08
P:010775FA 38                  move.l #CALC_CRC_ADDR,r15
           20
           A0
           00
           3F
           60
           3B
           4E
           81
           07
P:01077604 38                  move.l #NEXT_BYTE_ADDR,r12
           20
           A0
```

```
                00
                3C
                60
                3B
                7A
                81
                07
        P:0107760E D1                   cmpeq.w #$2,d3
                A2
        P:01077610 27                   bf from_uart
                18
                82
                B4
                          from_tdm
        P:01077614 20                   move.w #$01f8,d0
                00
                81
                F8
        P:01077618 00                   move.l d0,(r2+PAR)
                82
                82
                18
        P:0107761C 00                   move.l d0,(r2+PSOR)
                82
                82
                20
        P:01077620 20                   move.w #$03ff,d0
                00
                83
                FF
        P:01077624 00                   move.w d0,WBCR_ADDR
                E0
                3F
                82
                80
                F0
           ;TDM3GIR ; CTS=0;RTSAL=000 ; by_RESET
           ;TDM3RIR ; RFTL = pulse ;RSTL = pulse ;RFSD=01-One cycle delay
        P:0107762A C0                   move.w #$0010,d0
                90
        P:0107762C 00                   move.w d0,(r3+TDM3RIR+$2)
                23
                9F
                F2
           ;TDM3TIR ; TFTL = pulse ;TSTL = pulse ;TSO=0-sync input;TAO=0-Not drive data on inactive
channel
        P:01077630 78                   add #$4,d0                            ; $0014 -> d0
                44
        P:01077632 00                   move.w d0,(r3+TDM3TIR+$2)
                23
                9F
                EA
           ;TDM3RFP and TDM3TFP by adaption procedure
           ;TDM3TDBS - 0x017
        P:01077636 78                   add #$3,d0                            ; $0017 -> d0
                43
        P:01077638 00                   move.l d0,(r3+TDM3TDBS)
                A3
                9F
                C8
           ;TDM3RDBS - Buffer Size - 0xff
        P:0107763C 20                   move.w #$00ff,d0
                00
                80
                FF
        P:01077640 00                   move.l d0,(r3+TDM3RDBS)
                A3
                9F
                D0
           ;TDM3RGBA - to 0x0207
        P:01077644 20                   move.w #$0207,d0
                00
                82
                07
```

```
P:01077648 68                  imac d2,d5,d0                       ; (d1+d5.l*d2.l)->d1
           09
P:0107764A 00                  move.l d0,(r3+TDM3RGBA)
           A3
           9F
           C0
   ;TDM3TGBA - to 0x0207
P:0107764E 00                  move.l d0,(r3+TDM3TGBA)
           A3
           9F
           B8
                     ams_start
P:01077652 34                  clr d10
           00
           A0
           04
           6D
           10
P:01077658 00                  move.l (TDM3RER_ADDR),d0
           E6
           3F
           40
           81
           F8
P:0107765E 01                  move.l (TDM3TER_ADDR),d1
           E6
           3F
           38
           81
           F8
P:01077664 00                  move.l d0,(TDM3RER_ADDR)
           E2
           3F
           40
           81
           F8
P:0107766A 01                  move.l d1,(TDM3TER_ADDR)
           E2
           3F
           38
           81
           F8
P:01077670 C0                  move.w #$0001,d0
           81
P:01077672 00                  move.l d0,(r3+TDM3ACR)
           A3
           9F
           B0
P:01077676 39                  move.l #BASE_IP_8,r1
           C0
           20
           00
           81
           F8
                     ams_sub
P:0107767C 6C                  clr d1
           90
P:0107767E C2                  move.w #$11,d2
           91
                     loop_ams
P:01077680 10                  move.l (r3+TDM3ASR),d0
           A3
           9F
           30
P:01077684 0D                  bmtsts #$0001,d0.l
           00
           A0
           01
P:01077688 85                  bf <loop_ams
           F9
P:0107768A 14                  move.l (r3+TDM3ASDR),d4
           A3
           9F
           68
```

```
P:0107768E 00                  move.l d0,(r3+TDM3ASR)
           A3
           9F
           30
P:01077692 72                  cmpeq d1,d4
           61
P:01077694 80                  bt <asdr_set
           09
P:01077696 C1                  move.l d4,d1
           44
P:01077698 C2                  move.w #$11,d2
           91
P:0107769A 8F                  bra <loop_ams
           E7
                     asdr_set
P:0107769C 65                  deceq d2
           6D
P:0107769E 85                  bf <loop_ams
           E3
P:010776A0 D0                  cmpeq.w #$7,d1
           A7
P:010776A2 84                  bf <asdr_match_16
           0F
P:010776A4 34                  move.l #$0001010d,d4
           00
           21
           0D
           80
           01
P:010776AA 04                  move.l d4,(r1+TDM3RFP)
           A1
           9F
           E0
P:010776AE 88                  bra <check_ams_tx
           4B
                     asdr_match_16
P:010776B0 D0                  cmpeq.w #$f,d1
           AF
P:010776B2 84                  bf <asdr_match_192
           19
P:010776B4 0D                  bmtsts #$bff8,r1.l
           A9
           BF
           F8
P:010776B8 34                  move.l #$0001011d,d4
           00
           21
           1D
           80
           01
P:010776BE 94                  ift  &  bmclr #$0001,d4.l
           C2
           08
           04
           A0
           01
P:010776C4 04                  move.l d4,(r1+TDM3RFP)
           A1
           9F
           E0
P:010776C8 88                  bra <check_ams_tx
           31
                     asdr_match_192
P:010776CA 34                  cmpeq.w #$c0,d1
           11
           80
           C0
P:010776CE 84                  bf <asdr_match_16n
           0F
P:010776D0 34                  move.l #$0017011e,d4
           00
           21
           1E
           80
```

```
               17
P:010776D6  04                  move.l d4,(r1+TDM3RFP)
               A1
               9F
               E0
P:010776DA  88                  bra <check_ams_tx
               1F
                      asdr_match_16n
P:010776DC  64                  inc d1
               EF
P:010776DE  0C                  bmtstc #$000f,d1.l
               01
               A0
               0F
P:010776E2  85                  bf <ams_sub
               9B
P:010776E4  7C                  asrr #$3,d1
               E3
P:010776E6  36                  cmpgt.w #$ff,d1
               11
               80
               FF
P:010776EA  81                  bt <ams_sub
               93
P:010776EC  78                  sub #$1,d1
               E1
P:010776EE  D8                  aslw d1,d1
               91
P:010776F0  09                  bmset #$011c,d1.l
               01
               A1
               1C
P:010776F4  01                  move.l d1,(r1+TDM3RFP)
               A1
               9F
               E0
                      check_ams_tx
P:010776F8  10                  move.l (r3+TDM3ACR),d0
               A3
               9F
               B0
P:010776FC  0D                  bmtsts #$0002,d0.l
               00
               A0
               02
P:01077700  80                  bt <activ_chan
               0D
P:01077702  C0                  move.w #$0003,d0
               83
P:01077704  00                  move.l d0,(r3+TDM3ACR)
               A3
               9F
               B0
P:01077708  E9                  suba #$8,r1
               68
P:0107770A  8F                  bra <ams_sub
               73
                      activ_chan
P:0107770C  6C                  clr d0
               10
P:0107770E  6C                  clr d1
               90
P:01077710  00                  move.l d0,(r3+TDM3ACR)
               A3
               9F
               B0
P:01077714  19                  move.w (r3+TDM3RFP),r1
               23
               9F
               E0
P:01077718  36                  move.w (r3+TDM3TFP),r14
               20
               A0
               00
               1E
               23
               9F
               D8
    P:01077720 E9                  inca r1
               41
    P:01077722 34                  inca r14
               80
               A0
               00
               EE
               41
                         init_rcpr
    P:01077728 E9                  deceqa r1
               F6
    P:0107772A E8                  tfra r1,r0
               E9
    P:0107772C E8                  asl2a r0
               FE
    P:0107772E E8                  adda r3,r0
               1B
    P:01077730 00                  move.l d0,(r0+TDM3RCPR_BASE)
               80
               90
               00
    P:01077734 85                  bf <init_rcpr
               F5
                         init_tcpr
    P:01077736 34                  deceqa r14
               80
               A0
               00
               EE
               F6
    P:0107773C 34                  tfra r14,r0
               20
               A0
               00
               E8
               EE
    P:01077742 E8                  asl2a r0
               FE
    P:01077744 E8                  adda r3,r0
               1B
    P:01077746 00                  move.l d0,(r0+TDM3TCPR_BASE)
               A0
               88
               00
    P:0107774A 85                  bf <init_tcpr
               ED
       ;TDM3RCPR_0
    P:0107774C 32                  move.l #$80006e00,d2
               70
               2E
               00
               80
               00
    P:01077752 02                  move.l d2,(r3+TDM3RCPR_0)
               83
               90
               00
       ;TDM3TCPR_CHIP_ID , NEED to clear Transmit buffer to 0
    P:01077756 34                  move.l d9,r1
               80
               A0
               00
               C9
               41
    P:0107775C 90                  nop
               C0
    P:0107775E E9                  asl2a r1
               FE
    P:01077760 E8                  tfra r1,r0
               E9
```

MSC8113 Reference Manual, Rev. 0

```
P:01077762 E8                          asla r0
           FC
P:01077764 E9                          adda r3,r1
           1B
P:01077766 E8                          adda r3,r0
           1B
P:01077768 32                          move.l #$80006f00,d2
           70
           2F
           00
           80
           00
P:0107776E 02                          move.l d2,(r1+TDM3TCPR_BASE)
           A1
           88
           00
P:01077772 34                          move.l r1,d14
           80
           A0
           00
           C9
           4E
P:01077778 39                          move.l #$01076f00,r1
           60
           2F
           00
           81
           07
P:0107777E 28                          adda #TDM3LOCALMEM_BASE,r0,r0
           08
           98
           00
P:01077782 40                          move.l d0,(r0)
           90
P:01077784 B0                          move.l d0,(r0+$4)
           41
P:01077786 28                          adda #$400,r0,r0
           08
           84
           00
P:0107778A 40                          move.l d0,(r0)
           90
P:0107778C B0                          move.l d0,(r0+$4)
           41
P:0107778E C0                          move.2l d0:d1,(r1)+
           19
P:01077790 C0                          move.2l d0:d1,(r1)+
           19
P:01077792 C0                          move.2l d0:d1,(r1)+
           19
P:01077794 C0                          move.w #$0001,d0
           81
P:01077796 00                          move.l d0,(r3+TDM3RCR)
           A3
           9F
           A8
P:0107779A 00                          move.l d0,(r3+TDM3TCR)
           A3
           9F
           A0
P:0107779E 38                          move.l #$01076e00,r14
           20
           A0
           00
           3E
           60
           2E
           00
           81
           07
P:010777A8 0F                          move.l (TDM3RDBDR_ADDR),r7
           E6
           3F
           60
           81
           F8
P:010777AE 08                  bmclr #$ff07,r7.l
           EF
           BF
           07
                     tdm_loop_data
P:010777B2 36                  bmtstc #$0003,d11.h
           20
           A0
           00
           0C
           13
           A0
           03
P:010777BA 84                  bf <tdm_check_transmit
           41
                     tdm_check_tx_ret
P:010777BC 38                  move.l (TDM3RDBDR_ADDR),r9
           20
           A0
           00
           09
           E6
           3F
           60
           81
           F8
P:010777C6 08                  move.l (TDM3RER_ADDR),r0
           E6
           3F
           40
           81
           F8
P:010777CC 0D                  bmtsts #$0008,r0.l
           08
           A0
           08
P:010777D0 2D                  bt ams_start
           FF
           9E
           83
                     tdm_next_byte_c
P:010777D4 34                  cmpeqa r9,r7
           20
           A0
           00
           EF
           A9
P:010777DA 81                  btd <tdm_loop_data
           D8
P:010777DC 34                  tfra r14,r0
           20
           A0
           00
           E8
           EE
P:010777E2 E8                  adda r7,r0
           1F
                     tdm_load_data
P:010777E4 C5                  move.2l (r0),d4:d5
           10
                     tdm_next_byte
P:010777E6 38                  extractu #$8,#$18,d4,d1
           C9
           82
           18
                               [
                               asll #$8,d4
                               bmset #$0107,r4.h
P:010777EA 3E                  ]
           48
           09
           1C
           A1
           07
P:010777F0 0D                  bmtsts #$0003,r7.l
           0F
           A0
           03
P:010777F4 9C                  jmpd r4
           60
P:010777F6 D2                  tfrt d5,d4
           55
P:010777F8 9E                  debug
           70
                     tdm_check_transmit
P:010777FA 08                  move.l (TDM3TER_ADDR),r0
           E6
           3F
           38
           81
           F8
P:01077800 0D                  bmtsts #$0008,r0.l
           08
           A0
           08
P:01077804 08                  move.l (TDM3TDBDR_ADDR),r0
           E6
           3F
           58
           81
           F8
P:0107780A 2D                  bt ams_start
           FF
           9E
           49
P:0107780E 36                  bmtstc #$0002,d11.h
           20
           A0
           00
           0C
           13
           A0
           02
P:01077816 84                  bf <tdm_check_tx_end
           15
                     tdm_tx_loop
P:01077818 34                  cmpeqa r0,r11
           80
           A0
           00
           EB
           A8
P:0107781E 81                  bt <tdm_check_tx_ret
           9F
P:01077820 8F                  brad <tdm_check_tx_ret
           9C
P:01077822 36                  bmset #$0002,d11.h
           20
           A0
           00
           09
           13
           A0
           02
                     tdm_check_tx_end
P:0107782A 34                  cmpeqa r0,r11
           80
           A0
           00
           EB
           A8
P:01077830 85                  bfd <tdm_check_tx_ret
           8C
                               [
                               clr d0
                               clr d1
P:01077832 20                  ]
           64
           6C
           90
P:01077836 C0                  move.2l d0:d1,(r1)
           11
P:01077838 36                  bmtsts #$10,d11.l
           20
           A0
           00
           0D
           03
           A0
           10
P:01077840 85                  bfd <tdm_check_tx_ret
           7C
P:01077842 36                  bmclr #$0003,d11.h
           20
           A0
           00
           08
           13
           A0
           03
                     tdm_wait
P:0107784A 08                  move.l (TDM3TDBDR_ADDR),r0
           E6
           3F
           58
           81
           F8
P:01077850 90                  nop
           C0
P:01077852 34                  cmpeqa r0,r11
           80
           A0
           00
           EB
           A8
P:01077858 81                  bt <tdm_wait
           F3
                     tdm_wait_to_end
P:0107785A 08                  move.l (TDM3TDBDR_ADDR),r0
           E6
           3F
           58
           81
           F8
P:01077860 90                  nop
           C0
P:01077862 34                  cmpeqa r0,r11
           80
           A0
           00
           EB
           A8
P:01077868 85                  bf <tdm_wait_to_end
           F3
P:0107786A 20                  move.w #$13ff,d0
           00
           93
           FF
P:0107786E 00                  move.w d0,WBCR_ADDR
           E0
           3F
           82
           80
           F0
P:01077874 6C                  clr d0
           10
P:01077876 34                  move.l d14,r0
           80
           A0
           00
```

```
              C8
              46
P:0107787C  00                 move.l d0,(r3+TDM3RCR)
              A3
              9F
              A8
P:01077880  00                 move.l d0,(r3+TDM3TCR)
              A3
              9F
              A0
P:01077884  00                 move.w d0,(r3+TDM3RIR+$2)
              23
              9F
              F2
P:01077888  00                 move.w d0,(r3+TDM3TIR+$2)
              23
              9F
              EA
P:0107788C  00                 move.l d0,(r3+TDM3RDBS)
              A3
              9F
              D0
P:01077890  00                 move.l d0,(r3+TDM3TDBS)
              A3
              9F
              C8
P:01077894  00                 move.l d0,(r3+TDM3RGBA)
              A3
              9F
              C0
P:01077898  00                 move.l d0,(r3+TDM3TGBA)
              A3
              9F
              B8
P:0107789C  00                 move.l d0,(r3+TDM3RCPR_0)
              83
              90
              00
P:010778A0  00                 move.l d0,(r0+TDM3TCPR_BASE)
              A0
              88
              00
                      tdm_uart_i2c_finish
P:010778A4  00                 move.l d0,(r2+SCIBR)
              82
              90
              00
P:010778A8  00                 move.l d0,(r2+SCICR)
              82
              90
              08
P:010778AC  00                 move.l d0,(r2+PAR)
              82
              82
              18
P:010778B0  00                 move.l d0,(r2+PSOR)
              82
              82
              20
P:010778B4  10                 move.l (r6+BR10),d0                 ; BR10 register
              86
              81
              50
P:010778B8  09                 bmset #$1,d0.l                      ; Valid bit set
              00
              A0
              01
P:010778BC  00                 move.l d0,(r6+BR10)
              86
              81
              50
P:010778C0  29                 bra wake_core123
              FF
              99
              E1
                        from_uart
   P:010778C4 30                  move.l #$18000000,d0
              00
              20
              00
              98
              00
   P:010778CA 00                  move.l d0,(r2+PAR)
              82
              82
              18
   P:010778CE 00                  move.l d0,(r2+PSOR)
              82
              82
              20
   P:010778D2 34                  clr d10
              00
              A0
              04
              6D
              10
   P:010778D8 34                  clr d14
              00
              A0
              04
              6F
              10
   P:010778DE 21                  move.w #$028b,d1
              00
              82
              8B
   P:010778E2 D1                  cmpeq.w #$3,d3
              A3
   P:010778E4 80                  bt <non_fast_mode
              05
   P:010778E6 C1                  move.w #$1,d1
              81
                        non_fast_mode
   P:010778E8 01                  move.l d1,(r2+SCIBR)
              82
              90
              00
   P:010778EC C1                  move.w #$0004,d1
              84
   P:010778EE 01                  move.l d1,(r2+SCICR)
              82
              90
              08
                        uart_loop_data
   P:010778F2 36                  bmtstc #$0002,d11.h
              20
              A0
              00
              0C
              13
              A0
              02
   P:010778FA 84                  bf <uart_check_transmit_end
              51
   P:010778FC 36                  bmtstc #$0001,d11.h
              20
              A0
              00
              0C
              13
              A0
              01
   P:01077904 84                  bf <uart_check_transmit
              17
                        uart_check_tx_ret
   P:01077906 10                  move.l (r2+SCISR),d0
              82
              90
```

```
           10
P:0107790A 0D                  bmtsts #$2000,d0.l
           20
           A0
           00
P:0107790E 85                  bf <uart_loop_data
           E5
P:01077910 09                  bmset #$0107,r4.h
           1C
           A1
           07
P:01077914 11                  move.l (r2+SCIDR),d1
           82
           90
           18
P:01077918 9C                  jmp r4
           61
                     uart_check_transmit
P:0107791A 10                  move.l (r2+SCISR),d0
           82
           90
           10
P:0107791E 0D                  bmtsts #$c000,d0.l
           C0
           A0
           00
P:01077922 85                  bf <uart_check_tx_ret
           E5
P:01077924 36                  extractu #$8,#$18,d14,d5
           00
           A8
           00
           3C
           CD
           82
           18
P:0107792C 05                  move.l d5,(r2+SCIDR)
           82
           90
           18
P:01077930 34                  asll #$8,d14
           00
           A0
           04
           7F
           48
P:01077936 D2                  cmpeq.w #$0003,d4
           23
P:01077938 34                  ift  &  tfr d7,d14
           02
           A0
           04
           77
           57
P:0107793E 66                  deceq d4
           6D
P:01077940 85                  bf <uart_check_tx_ret
           C7
P:01077942 36                  bmset #$0002,d11.h
           20
           A0
           00
           09
           13
           A0
           02
                     uart_check_transmit_end
P:0107794A 10                  move.l (r2+SCISR),d0
           82
           90
           10
P:0107794E 0D                  bmtsts #$c000,d0.l
           C0
           A0
```

```
            00
P:01077952  85                    bf <uart_check_tx_ret
            B5
P:01077954  C0                    move.w #$0004,d0
            84
P:01077956  00                    move.l d0,(r2+SCICR)
            82
            90
            08
P:0107795A  6C                    clr d0
            10
P:0107795C  36                    bmtsts #$10,d11.l
            20
            A0
            00
            0D
            03
            A0
            10
P:01077964  81                    btd <tdm_uart_i2c_finish
            40
P:01077966  36                    bmclr #$0003,d11.h
            20
            A0
            00
            08
            13
            A0
            03
P:0107796E  8F                    bra <uart_check_tx_ret
            99
P:01077970  9E                    debug
            70
                        state_1
P:01077972  D0                    cmpeq.w #$11,d1
            B1
P:01077974  34                    jf r12
            80
            A0
            00
            9C
            67
                        state_1_t
P:0107797A  36                    bmclr #$000f,d11.l
            20
            A0
            00
            08
            03
            A0
            0F
P:01077982  34                    clr d12
            00
            A0
            04
            6E
            10
P:01077988  34                    jmpd r15
            80
            A0
            00
            9F
            60
P:0107798E  2C                    move.w #LOGIC_STATE_2_1,r4
            60
            99
            92
                        state_2_1
P:01077992  34                    cmpeq.w #$22,d1
            11
            80
            22
P:01077996  80                    bt <state_2_1_t
            09
```

```
P:01077998 8F                 brad <state_1
           DA
P:0107799A 2C                 move.w #LOGIC_STATE_1,r4
           60
           99
           72
                    state_2_1_t
P:0107799E 34                 jmpd r15
           80
           A0
           00
           9F
           60
P:010779A4 2C                 move.w #LOGIC_STATE_2_2,r4
           60
           99
           A8
                    state_2_2
P:010779A8 34                 cmpeq.w #$33,d1
           11
           80
           33
P:010779AC 80                 bt <state_2_2_t
           09
P:010779AE 8F                 brad <state_1
           C4
P:010779B0 2C                 move.w #LOGIC_STATE_1,r4
           60
           99
           72
                    state_2_2_t
P:010779B4 34                 jmpd r15
           80
           A0
           00
           9F
           60
P:010779BA 2C                 move.w #LOGIC_STATE_2_3,r4
           60
           99
           BE
                    state_2_3
P:010779BE 34                 cmpeq.w #$44,d1
           11
           80
           44
P:010779C2 80                 bt <state_2_3_t
           09
P:010779C4 8F                 brad <state_1
           AE
P:010779C6 2C                 move.w #LOGIC_STATE_1,r4
           60
           99
           72
                    state_2_3_t
P:010779CA 34                 jmpd r15
           80
           A0
           00
           9F
           60
P:010779D0 2C                 move.w #LOGIC_STATE_3,r4
           60
           99
           D4
                    state_3
P:010779D4 34                 cmpeq d1,d9
           00
           A0
           04
           70
           E1
P:010779DA 80                 btd <state_3_t
           12
```

```
P:010779DC 2C                   move.w #LOGIC_STATE_4,r4
           60
           99
           FA
P:010779E0 0D                   bmtsts #$ff,d1.l
           01
           A0
           FF
P:010779E4 80                   bt <state_3_t
           09
P:010779E6 34                   jmp r15
           80
           A0
           00
           9F
           61
                      state_3_t
P:010779EC 34                   jmpd r15
           80
           A0
           00
           9F
           60
P:010779F2 36                   bmset #$01,d11.l
           20
           A0
           00
           09
           03
           A0
           01
                      state_4
P:010779FA 34                   cmpeq d1,d10
           00
           A0
           04
           71
           61
P:01077A00 84                   bf <state_4_t
           0B
P:01077A02 36                   bmset #$02,d11.l
           20
           A0
           00
           09
           03
           A0
           02
                      state_4_t
P:01077A0A 34                   jmpd r15
           80
           A0
           00
           9F
           60
P:01077A10 2C                   move.w #LOGIC_STATE_5,r4
           60
           9A
           14
                      state_5
P:01077A14 0D                   bmtsts #$ff,d1.l
           01
           A0
           FF
P:01077A18 34                   jfd r15
           80
           A0
           00
           9F
           66
P:01077A1E 2C                   move.w #LOGIC_STATE_6_1,r4
           60
           9A
           30
```

```
P:01077A22 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A28 36                  bmset #$08,d11.l
           20
           A0
           00
           09
           03
           A0
           08
                     state_6_1
P:01077A30 34                  tfr d1,d8
           00
           A0
           04
           74
           51
P:01077A36 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A3C 2C                  move.w #LOGIC_STATE_6_2,r4
           60
           9A
           40
                     state_6_2
P:01077A40 36                  insert #$8,#$8,d1,d8
           00
           A0
           04
           32
           E8
           82
           08
P:01077A48 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A4E 2C                  move.w #LOGIC_STATE_6_3,r4
           60
           9A
           52
                     state_6_3
P:01077A52 36                  insert #$8,#$10,d1,d8
           00
           A0
           04
           32
           E8
           82
           10
P:01077A5A 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A60 2C                  move.w #LOGIC_STATE_7_1,r4
           60
           9A
           64
                     state_7_1
P:01077A64 75                  tfr d1,d2
           51
P:01077A66 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A6C 2C                  move.w #LOGIC_STATE_7_2,r4
           60
           9A
           70
                     state_7_2
P:01077A70 32                  insert #$8,#$8,d1,d2
           EA
           82
           08
P:01077A74 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A7A 2C                  move.w #LOGIC_STATE_7_3,r4
           60
           9A
           7E
                     state_7_3
P:01077A7E 32                  insert #$8,#$10,d1,d2
           EA
           82
           10
P:01077A82 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077A88 2C                  move.w #LOGIC_STATE_7_4,r4
           60
           9A
           8C
                     state_7_4
                               [
                               move.w #LOGIC_STATE_8_1,r4
                               insert #$8,#$18,d1,d2
P:01077A8C 9A                  ]
           C0
           32
           EA
           82
           18
           90
           C0
           2C
           60
           9A
           E8
                     addr_update
                               [
                               asrw d2,d0
                               move.l d1,r0
P:01077A98 94                  ]
           C0
           D8
           1A
           C8
           41
P:01077A9E 34                  cmpeq.w #$01fc,d0
           10
           81
           FC
                               [
                               move.l #$00083b79,d1
                               asr d0,d0
P:01077AA2 34                  ]
           40
           31
           20
           3B
           79
           80
           08
                               [
                               ift  &  insert #$1b,#$5,d1,d2
                               ift  &  bmset #$0010,d11.h
P:01077AAA 3C                  ]
           22
           A0
           00
           09
           13
           A0
           10
           90
           C0
           32
           EA
           86
           C5
                               [
                               cmpeq.w #$010e,d0
                               move.l r6,d1
P:01077AB8 96                  ]
           C0
           34
           10
           81
           0E
           CE
           49
                               [
                               asrr #$11,d1
                               asrr #$4,d0
P:01077AC0 3C                  ]
           F1
           7C
           64
P:01077AC4 94                  ift  &  insert #$f,#$11,d1,d2
           C2
           32
           EA
           83
           D1
                               [
                               move.l d2,r8
                               move.l r0,d1
P:01077ACA 36                  ]
           20
           A0
           00
           C8
           42
           C8
           49
P:01077AD2 34                  ift  &  jmp r15
           82
           A0
           00
           9F
           61
P:01077AD8 D0                  cmpeq.w #$10,d0
           30
P:01077ADA 90                  nop
           C0
P:01077ADC 34                  jmpd r15
           80
           A0
           00
           9F
           60
P:01077AE2 34                  ift  &  adda r5,r8
```

```
           82
           A0
           00
           E8
           1D
                     state_8_1
P:01077AE8 75                  tfr d1,d2
           51
P:01077AEA 34                  jmpd r12
           80
           A0
           00
           9C
           60
P:01077AF0 2C                  move.w #LOGIC_STATE_8_2,r4
           60
           9A
           F4
                     state_8_2
P:01077AF4 32                  insert #$8,#$8,d1,d2
           EA
           82
           08
P:01077AF8 34                  cmpeq d2,d12
           00
           A0
           04
           72
           62
P:01077AFE 34                  jfd r12
           80
           A0
           00
           9C
           66
P:01077B04 2C                  move.w #LOGIC_STATE_1,r4
           60
           99
           72
P:01077B08 36                  bmtsts #$01,d11.l
           20
           A0
           00
           0D
           03
           A0
           01
P:01077B10 34                  clr d12
           00
           A0
           04
           6E
           10
P:01077B16 94                  ift  &  move.w #LOGIC_STATE_9,r4
           C2
           2C
           60
           9B
           3C
P:01077B1C 94                  iff  &  move.w #LOGIC_STATE_9_C,r4
           C3
           2C
           60
           9B
           28
P:01077B22 34                  jmp r12
           80
           A0
           00
           9C
           61
                     state_9_c
P:01077B28 34                  deceq d8
           00
```

```
             A0
             04
             64
             6D
P:01077B2E  90                       nop
             C0
P:01077B30  94                       ift   &  move.w #LOGIC_STATE_10_1,r4
             C2
             2C
             60
             9B
             98
P:01077B36  34                       jmp r12
             80
             A0
             00
             9C
             61
                          state_9
P:01077B3C  34                       deceq d8
             00
             A0
             04
             64
             6D
P:01077B42  34                       move.b d1,(r8)+
             80
             A0
             00
             91
             80
P:01077B48  94                       ift   &  move.w #LOGIC_STATE_10_1,r4
             C2
             2C
             60
             9B
             98
                          calc_crc
P:01077B4E  34                       eor d12,d1
             00
             A8
             00
             DC
             94
                                     [
                                     bmtsts #$0080,d1.l
                                     move.l d1,r0
P:01077B54  96                       ]
             C0
             C8
             41
             0D
             01
             A0
             80
P:01077B5C  08                       bmclr #$ff80,r0.l
             E8
             BF
             80
P:01077B60  34                       asrr #$8,d12
             00
             A0
             04
             7E
             68
P:01077B66  34                       move.w (r10+r0),d1
             80
             A0
             00
             A1
             90
P:01077B6C  34                       eor d1,d12
             00
             A0
```

```
              04
              DE
              11
   P:01077B72 36                  ift   &  eor #$a001,d12.l
              22
              A0
              00
              0A
              A4
              A0
              01
                        next_byte
                                  [
                                  and #$0ffff,d12,d12
                                  cmpeq.w #$2,d3
   P:01077B7A 38                  ]
              00
              A8
              04
              39
              FC
              9F
              FF
              D1
              A2
   P:01077B84 2E                  bfd uart_loop_data
              FF
              9D
              6F
   P:01077B88 0D                  bmtsts #$0007,r7.l
              0F
              A0
              07
   P:01077B8C EF                  inca r7
              41
   P:01077B8E 2E                  bfd tdm_next_byte
              FF
              9C
              59
   P:01077B92 08                  bmclr #$ff00,r7.l
              EF
              BF
              00
   P:01077B96 8C                  bra <tdm_next_byte_c
              3F
                        state_10_1
   P:01077B98 75                  tfr d1,d2
              51
   P:01077B9A 34                  jmpd r12
              80
              A0
              00
              9C
              60
   P:01077BA0 2C                  move.w #LOGIC_STATE_10_2,r4
              60
              9B
              A4
                        state_10_2
   P:01077BA4 36                  bmtsts #$1,d11.l
              20
              A0
              00
              0D
              03
              A0
              01
   P:01077BAC 34                  jfd r12
              80
              A0
              00
              9C
              66
   P:01077BB2 2C                  move.w #LOGIC_STATE_1,r4
```

```
               60
               99
               72
     P:01077BB6 32                     insert #$8,#$8,d1,d2
               EA
               82
               08
     P:01077BBA 34                     cmpeq d2,d12
               00
               A0
               04
               72
               62
     P:01077BC0 84                     bf <rn_not_inc
               13
     P:01077BC2 36                     bmtsts #$2,d11.l
               20
               A0
               00
               0D
               03
               A0
               02
     P:01077BCA 84                     bf <rn_not_inc
               09
     P:01077BCC 34                     inc d10
               00
               A0
               04
               65
               6F
                          rn_not_inc
     P:01077BD2 36                     iff   &  bmclr #$8,d11.l
               23
               A0
               00
               08
               03
               A0
               08
     P:01077BDA 34                     tfr d15,d2
               00
               A8
               00
               75
               57
     P:01077BE0 74                     tfr d2,d0
               52
     P:01077BE2 34                     eor d10,d0
               00
               A8
               00
               DC
               12
     P:01077BE8 0D                     bmtsts #$0080,d0.l
               00
               A0
               80
     P:01077BEC 31                     and #$0007f,d0,d0
               18
               80
               7F
     P:01077BF0 C8                     move.l d0,r0
               40
     P:01077BF2 7D                     asrr #$8,d2
               68
     P:01077BF4 34                     move.w (r10+r0),d0
               80
               A0
               00
               A0
               90
     P:01077BFA DD                     eor d0,d2
               10
```

```
P:01077BFC 94                   ift  &  eor #$a001,d2.l
           C2
           0A
           A2
           A0
           01
P:01077C02 34                   insert #$8,#$18,d2,d7
           EF
           82
           18
P:01077C06 7D                   asrr #$8,d2
           68
P:01077C08 34                   insert #$8,#$10,d2,d7
           EF
           82
           10
P:01077C0C 36                   bmtsts #$8,d11.l
           20
           A0
           00
           0D
           03
           A0
           08
P:01077C14 36                   insert #$8,#$0,d10,d6
           00
           A8
           00
           34
           EE
           82
           00
P:01077C1C 36                   bmset #$0001,d11.h
           20
           A0
           00
           09
           13
           A0
           01
P:01077C24 36                   ift  &  bmset #$10,d11.l
           22
           A0
           00
           09
           03
           A0
           10
P:01077C2C D1                   cmpeq.w #$2,d3
           A2
P:01077C2E 84                   bfd <uart_send_ack
           32
P:01077C30 39                   move.l #$01076f00,r1
           60
           2F
           00
           81
           07
P:01077C36 38                   move.l (TDM3TDBDR_ADDR),r11
           20
           A0
           00
           0B
           E6
           3F
           58
           81
           F8
P:01077C40 E9                   adda #$18,r1
           58
P:01077C42 34                   adda r11,r1
           20
           A0
           00
```

```
                  E9
                  1B
       P:01077C48 0C                   bmtstc #$0020,r1.l
                  09
                  A0
                  20
       P:01077C4C 08                   bmclr #$0020,r1.l
                  09
                  A0
                  20
       P:01077C50 94                   ift   &  bmclr #$0008,r1.l
                  C2
                  08
                  09
                  A0
                  08
       P:01077C56 90                   nop
                  C0
       P:01077C58 C6                   move.2l d6:d7,(r1)
                  11
       P:01077C5A 34                   jmp r12
                  80
                  A0
                  00
                  9C
                  61
                            uart_send_ack
       P:01077C60 C0                   move.w #$000c,d0
                  8C
       P:01077C62 00                   move.l d0,(r2+SCICR)
                  82
                  90
                  08
       P:01077C66 C4                   move.w #$6,d4
                  86
       P:01077C68 34                   jmpd r12
                  80
                  A0
                  00
                  9C
                  60
       P:01077C6E 34                   tfr d6,d14
                  00
                  A0
                  04
                  77
                  56
                            ;CRC-16 table
                            ; inital_crc = 0x0000
                            ;crc = (crc >> 8) ^ crc_table_16[(crc ^ BYTE_IN) & 0xff];
                          ;CRC_16_table - size of 512-Bytes -> size of 256-Bytes with 0xa001 operation
.
                            crc_table
       P:01077C74                      dcw     $0000
       P:01077C76                      dcw     $c0c1
       P:01077C78                      dcw     $c181
       P:01077C7A                      dcw     $0140
       P:01077C7C                      dcw     $c301
       P:01077C7E                      dcw     $03c0
       P:01077C80                      dcw     $0280
       P:01077C82                      dcw     $c241
       P:01077C84                      dcw     $c601
       P:01077C86                      dcw     $06c0
       P:01077C88                      dcw     $0780
       P:01077C8A                      dcw     $c741
       P:01077C8C                      dcw     $0500
       P:01077C8E                      dcw     $c5c1
       P:01077C90                      dcw     $c481
       P:01077C92                      dcw     $0440
       P:01077C94                      dcw     $cc01
       P:01077C96                      dcw     $0cc0
       P:01077C98                      dcw     $0d80
       P:01077C9A                      dcw     $cd41
       P:01077C9C                      dcw     $0f00
```

```
P:01077C9E                          dcw       $cfc1
P:01077CA0                          dcw       $ce81
P:01077CA2                          dcw       $0e40
P:01077CA4                          dcw       $0a00
P:01077CA6                          dcw       $cac1
P:01077CA8                          dcw       $cb81
P:01077CAA                          dcw       $0b40
P:01077CAC                          dcw       $c901
P:01077CAE                          dcw       $09c0
P:01077CB0                          dcw       $0880
P:01077CB2                          dcw       $c841
P:01077CB4                          dcw       $d801
P:01077CB6                          dcw       $18c0
P:01077CB8                          dcw       $1980
P:01077CBA                          dcw       $d941
P:01077CBC                          dcw       $1b00
P:01077CBE                          dcw       $dbc1
P:01077CC0                          dcw       $da81
P:01077CC2                          dcw       $1a40
P:01077CC4                          dcw       $1e00
P:01077CC6                          dcw       $dec1
P:01077CC8                          dcw       $df81
P:01077CCA                          dcw       $1f40
P:01077CCC                          dcw       $dd01
P:01077CCE                          dcw       $1dc0
P:01077CD0                          dcw       $1c80
P:01077CD2                          dcw       $dc41
P:01077CD4                          dcw       $1400
P:01077CD6                          dcw       $d4c1
P:01077CD8                          dcw       $d581
P:01077CDA                          dcw       $1540
P:01077CDC                          dcw       $d701
P:01077CDE                          dcw       $17c0
P:01077CE0                          dcw       $1680
P:01077CE2                          dcw       $d641
P:01077CE4                          dcw       $d201
P:01077CE6                          dcw       $12c0
P:01077CE8                          dcw       $1380
P:01077CEA                          dcw       $d341
P:01077CEC                          dcw       $1100
P:01077CEE                          dcw       $d1c1
P:01077CF0                          dcw       $d081
P:01077CF2                          dcw       $1040
P:01077CF4                          dcw       $f001
P:01077CF6                          dcw       $30c0
P:01077CF8                          dcw       $3180
P:01077CFA                          dcw       $f141
P:01077CFC                          dcw       $3300
P:01077CFE                          dcw       $f3c1
P:01077D00                          dcw       $f281
P:01077D02                          dcw       $3240
P:01077D04                          dcw       $3600
P:01077D06                          dcw       $f6c1
P:01077D08                          dcw       $f781
P:01077D0A                          dcw       $3740
P:01077D0C                          dcw       $f501
P:01077D0E                          dcw       $35c0
P:01077D10                          dcw       $3480
P:01077D12                          dcw       $f441
P:01077D14                          dcw       $3c00
P:01077D16                          dcw       $fcc1
P:01077D18                          dcw       $fd81
P:01077D1A                          dcw       $3d40
P:01077D1C                          dcw       $ff01
P:01077D1E                          dcw       $3fc0
P:01077D20                          dcw       $3e80
P:01077D22                          dcw       $fe41
P:01077D24                          dcw       $fa01
P:01077D26                          dcw       $3ac0
P:01077D28                          dcw       $3b80
P:01077D2A                          dcw       $fb41
P:01077D2C                          dcw       $3900
P:01077D2E                          dcw       $f9c1
P:01077D30                          dcw       $f881
```

```
       P:01077D32                      dcw     $3840
       P:01077D34                      dcw     $2800
       P:01077D36                      dcw     $e8c1
       P:01077D38                      dcw     $e981
       P:01077D3A                      dcw     $2940
       P:01077D3C                      dcw     $eb01
       P:01077D3E                      dcw     $2bc0
       P:01077D40                      dcw     $2a80
       P:01077D42                      dcw     $ea41
       P:01077D44                      dcw     $ee01
       P:01077D46                      dcw     $2ec0
       P:01077D48                      dcw     $2f80
       P:01077D4A                      dcw     $ef41
       P:01077D4C                      dcw     $2d00
       P:01077D4E                      dcw     $edc1
       P:01077D50                      dcw     $ec81
       P:01077D52                      dcw     $2c40
       P:01077D54                      dcw     $e401
       P:01077D56                      dcw     $24c0
       P:01077D58                      dcw     $2580
       P:01077D5A                      dcw     $e541
       P:01077D5C                      dcw     $2700
       P:01077D5E                      dcw     $e7c1
       P:01077D60                      dcw     $e681
       P:01077D62                      dcw     $2640
       P:01077D64                      dcw     $2200
       P:01077D66                      dcw     $e2c1
       P:01077D68                      dcw     $e381
       P:01077D6A                      dcw     $2340
       P:01077D6C                      dcw     $e101
       P:01077D6E                      dcw     $21c0
       P:01077D70                      dcw     $2080
       P:01077D72                      dcw     $e041
                             from_i2c
                             ;the registers to be used are :
                             ;      d0     - local use
[i2c_assert_stop,i2c_txrx_bit,i2c_assert_start,i2c_read_SequantialData]
                           ;      d1     - local use                      [i2c_sample_gpio,i2c_txrx_bit]
                             ;      d2     - sda,scl pin value
[i2c_sample_gpio,i2c_assert_stop,i2c_txrx_bit,i2c_assert_start]
                             ;      d3     - sda,scl pin value                [i2c_txrx_bit]
                             ;      d4     - bit position
[i2c_txrx_byte,i2c_assert_stop,i2c_txrx_bit]
                             ;      d5     - transmit byte
[i2c_txrx_bit,i2c_read_SequantialData]
                             ;      d6     - receive byte
[i2c_txrx_byte,i2c_txrx_bit,i2c_read_SequantialData]
                             ;      d7     - control byte , bit#0-tx=0/rx=1
[i2c_txrx_byte,i2c_assert_stop,i2c_txrx_bit,i2c_read_SequantialData]
                             ;      d8     - sda,scl  = 11 bits
[i2c_sample_gpio,i2c_assert_stop,i2c_txrx_bit]
                             ;      d9     - CHIP-ID value
                             ;      d10    - checksum                         [i2c_read_SequantialData]
                            ;      d11    - flag for ECC access,checksum byte [i2c_read_SequantialData]
                             ;      d12    - size of data bytes to read       [i2c_read_SequantialData]
                             ;      d14    - SCL_HIGH_PERIOD                  [i2c_txrx_bit]
                             ;      d15    - SCL_LOW_HALF_PERIOD              [i2c_txrx_bit]
                             ;      r2     - BASE_IP_B = $01fbc000
[i2c_sample_gpio,i2c_assert_stop,i2c_txrx_bit,i2c_assert_start]
                             ;      r3     - address at the serial EEPROM     [i2c_read_SequantialData]
                             ;      r4     - address at memory                [i2c_read_SequantialData]
                             ;      r7     - slave address                    [i2c_read_SequantialData]
                             ;      r8     - address at memory after update   [i2c_read_SequantialData]
                             ;      r9     - PDAT address
       P:01077D74 38                   move.l #PDAT_ADDR,r9
                  20
                  A0
                  00
                  39
                  C0
                  22
                  08
                  81
                  FB
```

```
P:01077D7E 38                  moveu.l #SCL_SDA_11,d8
           20
           A0
           00
           30
           C1
           20
           00
           80
           00
P:01077D88 36                  move.w #SCL_HIGH_PERIOD,d14
           20
           A0
           00
           26
           00
           80
           40
P:01077D90 36                  move.w #SCL_LOW_HALF_PERIOD,d15
           20
           A0
           00
           27
           00
           81
           00
                     i2c_start
P:01077D98 6C                  clr d0
           10
P:01077D9A 00                  move.w d0,(r2+PAR)
           02
           82
           18
P:01077D9E 00                  move.w d0,(r2+PSOR)
           02
           82
           20
P:01077DA2 34                  tfr d8,d0
           00
           A8
           00
           74
           50
P:01077DA8 34                  move.w d0,(r9)
           80
           A0
           00
           40
           11
P:01077DAE 00                  move.w d0,(r2+PDIR)
           02
           82
           10
P:01077DB2 00                  move.w d0,(r2+PODR)
           02
           82
           00
P:01077DB6 23                  bsr i2c_WaitFor_StartCond_BusFreeTime

           18
           82
           1C
P:01077DBA 34                  clr d10
           00
           A0
           04
           6D
           10
P:01077DC0 38                  move.l #I2C_SCL_PERIOD_ADDR,r12
           20
           A0
           00
           3C
           60
           2F
           28
           81
           07
P:01077DCA 39                  move.l #I2C_HEADER_ADDR,r1
           60
           2F
           10
           81
           07
P:01077DD0 38                  move.l #$ffffffff,r14
           20
           A0
           00
           3E
           F8
           3F
           FF
           BF
           FF
P:01077DDA 34                  move.2l d14:d15,(r12)
           A0
           A0
           00
           C6
           14
P:01077DE0 3B                  move.l #I2C_START_ADDR,r3
           00
           20
           20
           80
           07
                     next_block
P:01077DE6 34                  move.2l (r12),d14:d15
           A0
           A0
           00
           C7
           14
P:01077DEC 34                  tfra r3,r11
           80
           A0
           00
           EB
           EB
P:01077DF2 EC                  tfra r1,r4
           E9
P:01077DF4 34                  move.w #$c,d12
           20
           A0
           00
           C4
           8C
P:01077DFA 88                  bra <i2c_cont_1
           0B
                     ;---------- nmi 0 exeption offset 0xe00 ----------
P:01077E00                     org     p:$0e00+BASE_EXEPTION_TABLE
                     nmi0_exeption
P:01077E00 9E                  debug
           70
P:01077E02 9F                  rte
           73
                     i2c_cont_1
P:01077E04 36                  bmclr #$0100,d11.h                    ; write data to memory
           20
           A0
           00
           08
           13
           A1
           00
P:01077E0C 23                  bsr i2c_read_SequantialData
           18
           80
           DC
P:01077E10 92                  ift  &  bra <i2c_start
           C2
           8F
           89
P:01077E14 34                  move.l (r1),d12
           20
           A0
           00
           54
           91
P:01077E1A 90                  moveu.b (r1),d0
           B9
P:01077E1C 08                  and #$1f,d0.l
           E0
           BF
           E0
P:01077E20 0D                  bmtsts #$1f,d0.l
           00
           A0
           1F
P:01077E24 34                  move.l (r1+$4),r10
           20
           A0
           00
           BA
           C9
P:01077E2A 34                  iff  &  cmpeq d0,d9
           03
           A0
           04
           70
           E0
P:01077E30 BC                  move.l (r1+$8),r4
           CA
P:01077E32 36                  iff  &  bmset #$0100,d11.h
           23
           A0
           00
           09
           13
           A1
           00
P:01077E3A 88                  bra <i2c_cont_2
           0B
                     ;---------- nmi 1 exeption offset 0xe40 ----------
P:01077E40                     org     p:$0e40+BASE_EXEPTION_TABLE
                     nmi1_exeption
P:01077E40 9E                  debug
           70
P:01077E42 9F                  rte
           73
                     i2c_cont_2
P:01077E44 23                  bsr i2c_read_SequantialData
           18
           80
           A4
P:01077E48 92                  ift  &  bra <i2c_start
           C2
           8F
           51
P:01077E4C 36                  bmclr #$0100,d11.h
           20
           A0
           00
           08
           13
           A1
           00
P:01077E54 34                  move.w #$4,d12
           20
           A0
           00
           C4
           84
P:01077E5A 29                  adda #$c,r1,r4
           0C
           80
           0C
P:01077E5E 23                  bsr i2c_read_SequantialData
           18
           80
           8A
P:01077E62 92                  ift  &  bra <i2c_start
           C2
           8F
           37
P:01077E66 90                  moveu.b (r1),d0
           B9
P:01077E68 08                  and #$c0,d0.l
           E0
           BF
           3F
P:01077E6C 34                  cmpeq.w #$80,d0
           10
           80
           80
P:01077E70 84                  bf <next_block_addr
           3F
P:01077E72 B0                  move.l (r1+$c),d0
           CB
P:01077E74 64                  zxt.l d0
           60
P:01077E76 34                  not d10,d1
           00
           A8
           00
           D8
           82
P:01077E7C 88                  bra <i2c_cont_3
           09
                     ;---------- nmi 2 exeption offset 0xe80 ----------
P:01077E80                     org     p:$0e80+BASE_EXEPTION_TABLE
                     nmi2_exeption
P:01077E80 9E                  debug
           70
P:01077E82 9F                  rte
           73
                     i2c_cont_3
P:01077E84 36                  insert #$10,#$10,d1,d10
           00
           A0
           04
           32
           EA
           84
           10
P:01077E8C 34                  cmpeq d0,d10
           00
           A0
           04
           71
           60
P:01077E92 80                  bt <next_block_addr
           1D
P:01077E94 36                  bmchg #$0001,d11.l
           20
           A0
           00
           0A
           03
           A0
           01
P:01077E9C 36                  bmtstc #$0001,d11.l
           20
           A0
           00
           0C
           03
           A0
           01
P:01077EA4 85                  bfd <next_block
           42
P:01077EA6 34                  tfra r11,r3
           20
           A0
           00
           EB
           EB
P:01077EAC 9E                  debug
           70
                     next_block_addr
P:01077EAE 34                  clr d10
           00
           A0
           04
           6D
           10
P:01077EB4 34                  tsteqa.l r10
           80
           A0
           00
           EA
           F1
P:01077EBA 81                  bt <next_block
           2D
P:01077EBC 21                  bra i2c_cont_4
           18
           80
           08
                     ;---------- nmi 3 exeption offset 0xec0 ----------
P:01077EC0                     org     p:$0ec0+BASE_EXEPTION_TABLE
                     nmi3_exeption
P:01077EC0 9E                  debug
           70
P:01077EC2 9F                  rte
           73
                     i2c_cont_4
P:01077EC4 34                  cmpeqa r14,r10
           A0
           A0
           00
           EA
           AE
P:01077ECA 80                  bt <i2c_finish
           0B
P:01077ECC 34                  tfra r10,r3
           20
           A0
           00
           EB
           EA
P:01077ED2 8F                  bra <next_block
           15
                     i2c_finish
P:01077ED4 6C                  clr d0
           10
P:01077ED6 34                  move.l d0,(r9)
           80
           A0
           00
           40
           91
P:01077EDC 00                  move.l d0,(r2+PDIR)
           82
           82
           10
P:01077EE0 00                  move.l d0,(r2+PODR)
           82
           82
           00
P:01077EE4 29                  bra tdm_uart_i2c_finish
```

```
              FF
              99
              C1
                           i2c_read_SequantialData
P:01077EE8    6F                     clr d7
              90
P:01077EEA    30                     moveu.l #$00ffffff,d0
              E1
              3F
              FF
              80
              FF
P:01077EF0    34                     and d0,d12
              00
              A0
              04
              DE
              00
P:01077EF6    2B                     bsr i2c_assert_start
              FF
              92
              AF
P:01077EFA    21                     bra i2c_read_SequantialData_1
              18
              80
              0A
                           ;---------- nmi 4 exeption offset 0xf00 ----------
P:01077F00                           org     p:$0f00+BASE_EXEPTION_TABLE
                           nmi4_exeption
P:01077F00    9E                     debug
              70
P:01077F02    9F                     rte
              73
                           i2c_read_SequantialData_1
P:01077F04    6E                     clr d5
              90
P:01077F06    CB                     move.l r3,d0
              48
P:01077F08    30                     extractu #$3,#$10,d0,d5
              CD
              80
              D0
P:01077F0C    62                     asl d5,d5
              E2
P:01077F0E    09                     bmset #$a0,d5.l
              05
              A0
              A0
P:01077F12    CF                     move.l d5,r7
              45
P:01077F14    2B                     bsr i2c_txrx_byte
              FF
              91
              7F
P:01077F18    92                     ift  &  rts
              C2
              9F
              71
P:01077F1C    6E                     clr d5
              90
P:01077F1E    CB                     move.l r3,d0
              48
P:01077F20    30                     extractu #$8,#$8,d0,d5
              CD
              82
              08
P:01077F24    2B                     bsr i2c_txrx_byte
              FF
              91
              6F
P:01077F28    92                     ift  &  rts
              C2
              9F
              71
```

```
P:01077F2C CB                   move.l r3,d0
           48
P:01077F2E 30                   extractu #$8,#$0,d0,d5
           CD
           82
           00
P:01077F32 2B                   bsr i2c_txrx_byte
           FF
           91
           61
P:01077F36 92                   ift  &  rts
           C2
           9F
           71
P:01077F3A 2B                   bsr i2c_assert_stop
           FF
           91
           AB
P:01077F3E 88                   bra <i2c_read_SequantialData_cont_2
           07
                      ;---------- nmi 5 exeption offset 0xf40 ----------
P:01077F40                      org     p:$0f40+BASE_EXEPTION_TABLE
                      nmi5_exeption
P:01077F40 9E                   debug
           70
P:01077F42 9F                   rte
           73
                      i2c_read_SequantialData_cont_2
P:01077F44 2B                   bsr i2c_assert_start
           FF
           92
           61
P:01077F48 CF                   move.l r7,d5
           4D
P:01077F4A 09                   bmset #$1,d5.l
           05
           A0
           01
P:01077F4E 2B                   bsr i2c_txrx_byte
           FF
           91
           45
P:01077F52 92                   ift  &  rts
           C2
           9F
           71
                      read_byte_loop
P:01077F56 34                   deceq d12
           00
           A0
           04
           66
           6D
P:01077F5C C7                   move.w #$1,d7
           81
P:01077F5E 94                   ift  &  bmset #$2,d7.l
           C2
           09
           07
           A0
           02
P:01077F64 2B                   bsr i2c_txrx_byte
           FF
           91
           2F
P:01077F68 92                   ift  &  rts
           C2
           9F
           71
P:01077F6C CC                   move.l r4,d2
           4A
P:01077F6E 38                   move.l #I2C_MEM_WRITE_ADDR,r15
           20
           A0
```

```
               00
               3F
               60
               3F
               84
               81
               07
     P:01077F78 29                 bra addr_update
               FF
               9B
               21
     P:01077F7C 88                 bra <i2c_mem_write
               09
                          ;---------- nmi 6 exeption offset 0xf80 ----------
     P:01077F80                    org     p:$0f80+BASE_EXEPTION_TABLE
                          nmi6_exeption
     P:01077F80 9E                 debug
               70
     P:01077F82 9F                 rte
               73
                          i2c_mem_write
     P:01077F84 34                 tfra r8,r4
               20
               A0
               00
               EC
               E8
     P:01077F8A 36                 bmtsts #$0100,d11.h
               20
               A0
               00
               0D
               13
               A1
               00
     P:01077F92 EB                 inca r3
               41
     P:01077F94 92                 iff  &  move.b d6,(r4)
               C3
               96
               9C
     P:01077F98 EC                 inca r4
               41
     P:01077F9A 36                 bmtstc #$0001,d11.h
               20
               A0
               00
               0C
               13
               A0
               01
     P:01077FA2 90                 nop
               C0
     P:01077FA4 92                 ift  &  asll #$8,d6
               C2
               7F
               48
     P:01077FA8 34                 eor d6,d10
               00
               A0
               04
               DD
               16
     P:01077FAE 36                 bmchg #$0001,d11.h
               20
               A0
               00
               0A
               13
               A0
               01
     P:01077FB6 34                 tsteq d12
               00
               A0
```

```
           04
           66
           69
P:01077FBC 85                  bf <read_byte_loop
           9B
P:01077FBE 88                  bra <i2c_read_SequantialData_cont_3
           07
                     ;---------- nmi 7 exeption offset 0xfc0 ----------
P:01077FC0                     org     p:$0fc0+BASE_EXEPTION_TABLE
                     nmi7_exeption
P:01077FC0 9E                  debug
           70
P:01077FC2 9F                  rte
           73
                     i2c_read_SequantialData_cont_3
P:01077FC4 2B                  bsr i2c_assert_stop
           FF
           91
           21
P:01077FC8 36                  bmtsts #$f,d11.l
           20
           A0
           00
           0D
           03
           A0
           0F
P:01077FD0 9F                  rts
           71
                     i2c_WaitFor_StartCond_BusFreeTime
P:01077FD2 20                  move.w #HALF_BUS_FREE_TIME,d0
           00
           8C
           85
                     busfree_loop
P:01077FD6 2B                  bsr i2c_sample_gpio
           FF
           90
           EF
P:01077FDA 0D                  bmtsts #SCL_SDA_11,d2.l
           C2
           A0
           00
P:01077FDE 85                  bf i2c_WaitFor_StartCond_BusFreeTime
           F5
P:01077FE0 64                  deceq d0
           6D
P:01077FE2 85                  bf <busfree_loop
           F5
P:01077FE4 20                  move.w #BUS_FREE_TIME,d0
           00
           99
           0B
                     wait_loop
P:01077FE8 2B                  bsr i2c_sample_gpio
           FF
           90
           DD
P:01077FEC 0C                  bmtstc #SCL_SDA_10,d2.l
           82
           A0
           00
P:01077FF0 81                  bt <i2c_WaitFor_StartCond_BusFreeTime

           E3
P:01077FF2 0C                  bmtstc #SCL_SDA_01,d2.l
           42
           A0
           00
P:01077FF6 80                  bt <wait_rts
           07
P:01077FF8 64                  deceq d0
           6D
P:01077FFA 85                  bf <wait_loop
```

```
                EF
                          wait_rts
     P:01077FFC 9F                  rts
                71
```

Index