MPC5200 Users Guide

Document Number: MPC5200UG
Rev. 3.1
03/2006

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

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

Document Number: MPC5200UG
Rev. 3.1
03/2006

Table of Contents

Chapter 5 Clocks and Power Management

Chapter 6 G2_LE Processor Core

Paragraph Number | Page Number

Chapter 7 System Integration Unit (SIU)

Chapter 8 SDRAM Memory Controller

Chapter 9 LocalPlus Bus (External Bus Interface)

Chapter 10 PCI Controller

Paragraph Number | Page Number

Chapter 11 ATA Controller

Chapter 12 Universal Serial Bus (USB)

chapter 13 BestComm

Chapter 14 Fast Ethernet Controller (FEC)

Chapter 15 Programmable Serial Controllers (PSC)

Chapter 16 XLB Arbiter

Paragraph Number | Page Number

Chapter 17 Serial Peripheral Interface (SPI)

Chapter 18 Inter-Integrated Circuit (I^2C)

Chapter 19 Motorola Scalable CAN (MSCAN)

Paragraph Number | Page Number

Chapter 20 Byte Data Link Controller (BDLC)

Chapter 21 Debug Support and JTAG Interface

Appendix A Acronyms and Terms

Appendix B List of Registers

List of Figures

Figure Number		Page Number

Notes

List of Tables

Table Number		Page Number

Table Number		Page Number

Revision History

Release	Date	Author	Summary of Changes
0	01Jul03	Various	First Version of User's Manual.
1	27Oct03	AS/TB/RM/CM	Errata fixes all chapters
2	22Jul04	AS/TB/PL	Enhancements and corrections, change to Freescale format
3	26Jan04	AS/TB/PL	Updates to PSC, SPI, MSCAN, LPC, SDRAM, Signals and SystemIntegration chapters
3.1	24Mar06	KL/AE	New title page (no Launched by Motorola, added back page); updated bit 19's description on pg. 15-27.

Chapter 1
Introduction

1.1 Overview

The digital communication networking and consumer markets require significant processor performance to enable operating systems and applications such as VxWorks™, QNX™, JAVA and soft modems. High integration is essential to reducing device and systems costs. The MPC5200 is specifically designed to meet these market needs while building on the family of microprocessors that use PowerPC™ architecture. For more information on PowerPC architecture, see "*The Programming Environments Manual for 32-bit Implementations of the PowerPC Architecture*".

MPC5200 integrates a high performance 603e G2_LE core with a rich set of peripheral functions focused on communications and systems integration. The 603e G2_LE core design is based on the PowerPC™ core architecture. The MPC5200 incorporates an innovative I/O subsystem, which isolates routine maintenance of peripheral functions from the embedded 603e G2_LE core.

The MPC5200 supports a dual external bus architecture. It has a high speed SDRAM Bus interface that connects directly to the 603e G2_LE core. In addition, the MPC5200 has a LocalPlus Bus used as a generalized interface to system level peripheral devices and debug environments.

1.1.1 Features

Key features are shown below.

- 603e G2_LE core
  - Superscalar architecture
  - 760MIPS at 400MHz (-40 to +85 ^{o}C)
  - 16k Instruction cache, 16k Data cache
  - Double precision FPU
  - Instruction and Data MMU
  - Standard & Critical interrupt capability
- SDRAM / DDR Memory Interface
  - up to 132MHz operation
  - SDRAM and DDR SDRAM support
  - 256-MByte addressing range per Chip Select (Two CS lines available)
  - 32-bit data bus
  - Built-in initialization and refresh
- Flexible multi-function External Bus Interface
  - Supports interfacing to ROM/Flash/SRAM memories or other memory mapped devices
  - 8 programmable Chip Selects
  - Non multiplexed data access using 8/16/32 bit databus with up to 26 bit address
  - Short or Long Burst capable
  - Multiplexed data access using 8/16/32 bit databus with up to 25 bit address
- Peripheral Component Interconnect (PCI) Controller
  - Version 2.2 PCI compatibility
  - PCI initiator and target operation
  - 32-bit PCI Address/Data bus
  - 33 and 66 MHz operation
  - PCI arbitration function
- ATA Controller
  - Version 4 ATA compatible external interface—IDE Disk Drive connectivity
- BestComm DMA subsystem
  - Intelligent virtual DMA Controller
  - Dedicated DMA channels to control peripheral reception and transmission
  - Local memory (SRAM 16kBytes)
- 6 Programmable Serial Controllers (PSC), configurable for:
  - UART or RS232 interface
  - CODEC interface for Soft Modem, Master/Slave CODEC Mode, I^2S and AC97

  - Full duplex SPI mode
  - IrDA mode from 2400 bps to 4 Mbps
- Fast Ethernet Controller (FEC)
  - Supports 100Mbps IEEE 802.3 MII, 10Mbps IEEE 802.3 MII, 10Mbps 7-wire interface
- Universal Serial Bus Controller (USB)
  - USB Revision 1.1 Host
  - Open Host Controller Interface (OHCI)
  - Integrated USB Hub, with two ports.
- Two Inter-Integrated Circuit Interfaces (I^2C)
- Serial Peripheral Interface (SPI)
- Dual CAN 2.0 A/B Controller (MSCAN)
  - Motorola Scalable CAN (MSCAN) architecture
  - Implementation of version 2.0A/B CAN protocol
  - Standard and extended data frames
- J1850 Byte Data Link Controller (BDLC)
  - J1850 Class B data communication network interface compatible and ISO compatible for low speed (<125kbps) serial data communications in automotive applications.
  - Supports 4X mode, 41.6 kbps
  - In-frame response (IFR) types 0, 1, 2, and 3 supported
- Systems level features
  - Interrupt Controller supports 4 external interrupt request lines and 47 internal interrupt sources
  - GPIO/Timer functions
    - Up to 56 total GPIO pins (depending on functional multiplexing selections) that support a variety of interrupt/Wake Up capabilities.
    - 8 GPIO pins with timer capability supporting input capture, output compare and pulse width modulation (PWM) functions
  - Real-time Clock with 1 second resolution
  - Systems Protection (watch dog timer, bus monitor)
  - Individual control of functional block clock sources
  - Power management: Nap, Doze, Sleep, Deep Sleep modes
  - Support of Wake Up from low power modes by different sources (GPIO, RTC, CAN)
- Test/Debug features
  - JTAG (IEEE 1149.1 test access port)
  - Common On-Chip Processor (COP) debug port
- On-board PLL and clock generation
- Software
  - QNX
  - VXWorks
  - Linux
  - Software Modem capable
  - JAVA

1.2 Architecture

The following areas comprise the MPC5200 system architecture:

- Embedded G2_LE Core
- BestComm I/O Subsystem
- Controller Area Network (CAN)
- Byte Data Link Controller - Digital BDLC-D
- System Level Interfaces
- SDRAM Controller and Interface
- Multi-Function External LocalPlus Bus
- Power Management
- Systems Debug and Test
- Physical Characteristics

A dynamically managed external pin multiplexing scheme minimizes overall pin count. The result is low cost packaging and board assembly costs.

Figure 1-1 shows a simplified MPC5200 block diagram.

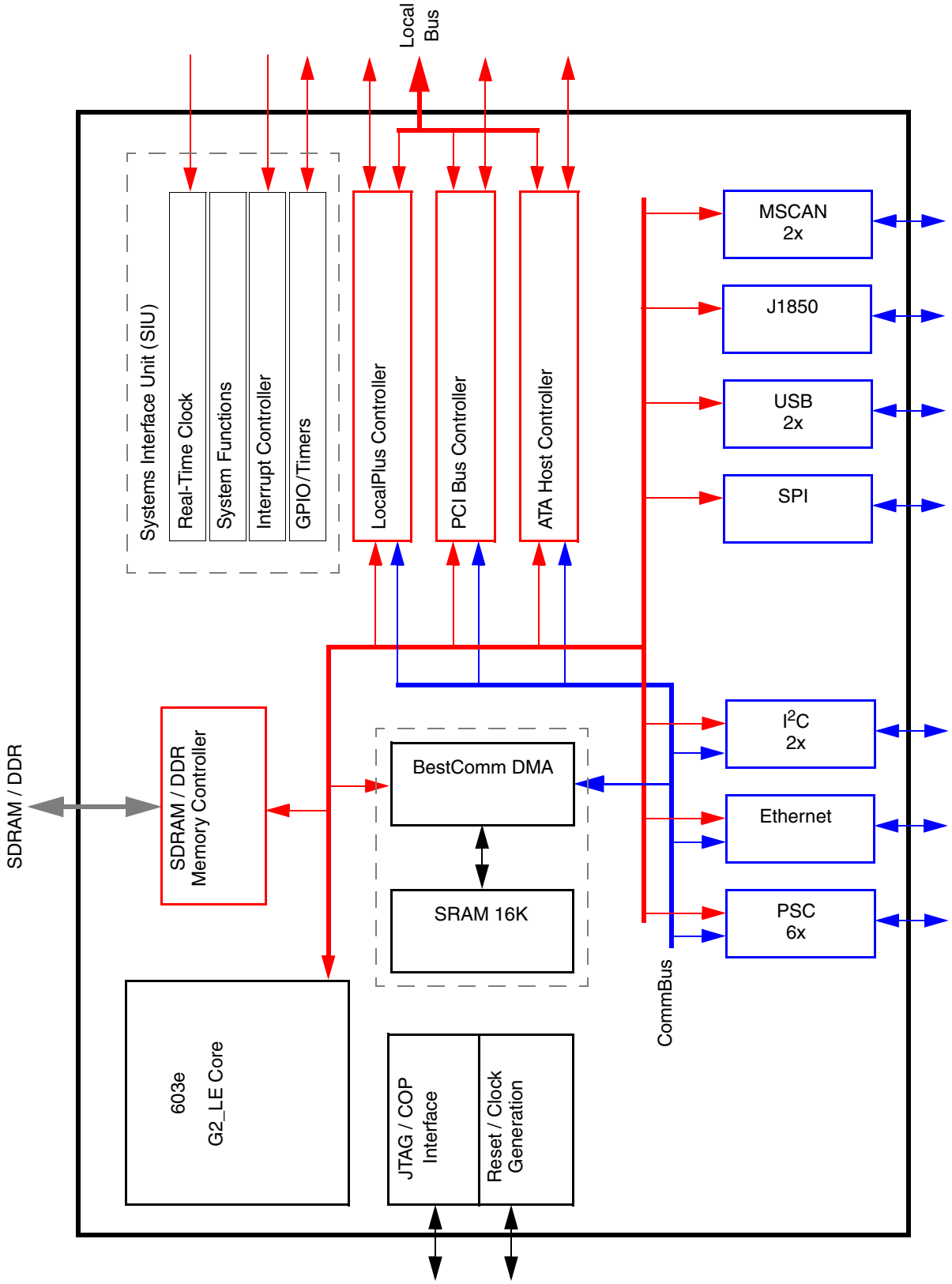


Figure 1-1. Simplified Block Diagram—MPC5200

The MPC5200 supports a dual external bus architecture consisting of:

1. an SDRAM Bus
2. a multi-function LocalPlus Bus

The SDRAM Bus has a Memory Controller interface supports standard SDRAM and Double Data Rate (DDR) SDRAM devices. The Memory Controller has 13-bit Memory Address (MA) lines multiplexed with 32-bit Data Bus lines. Standard SDRAM control signals are included.

The high-speed Memory Controller SDRAM interface connects directly to the microprocessor, allowing optimized instruction and data bursting. The dedicated memory interface, coupled with on-chip 16Kilobyte instruction and 16Kilobyte data caches, enables high performance, computer intensive applications, such as Java and soft modems. Still, plenty of processing power remains for peripheral management and system control tasks.

The LocalPlus Bus provides for connection of external peripheral devices, disk storage, and slower speed memory. The LocalPlus Bus supports:

- an external Boot ROM/FLASH/SRAM interface

The MPC5200 integrates a high performance 603e G2_LE core with an I/O subsystem containing an intelligent Direct Memory Access (DMA) unit. The MPC5200 is capable of:

- responding to peripheral interrupts, independent of the 603e G2_LE core.
- providing low level peripheral management, protocol processing, and peripheral data movement functions.

The MPC5200 has an optimized peripheral mix to support today's embedded automotive and telematics requirements.

Figure 1-2 shows an MPC5200-based system.

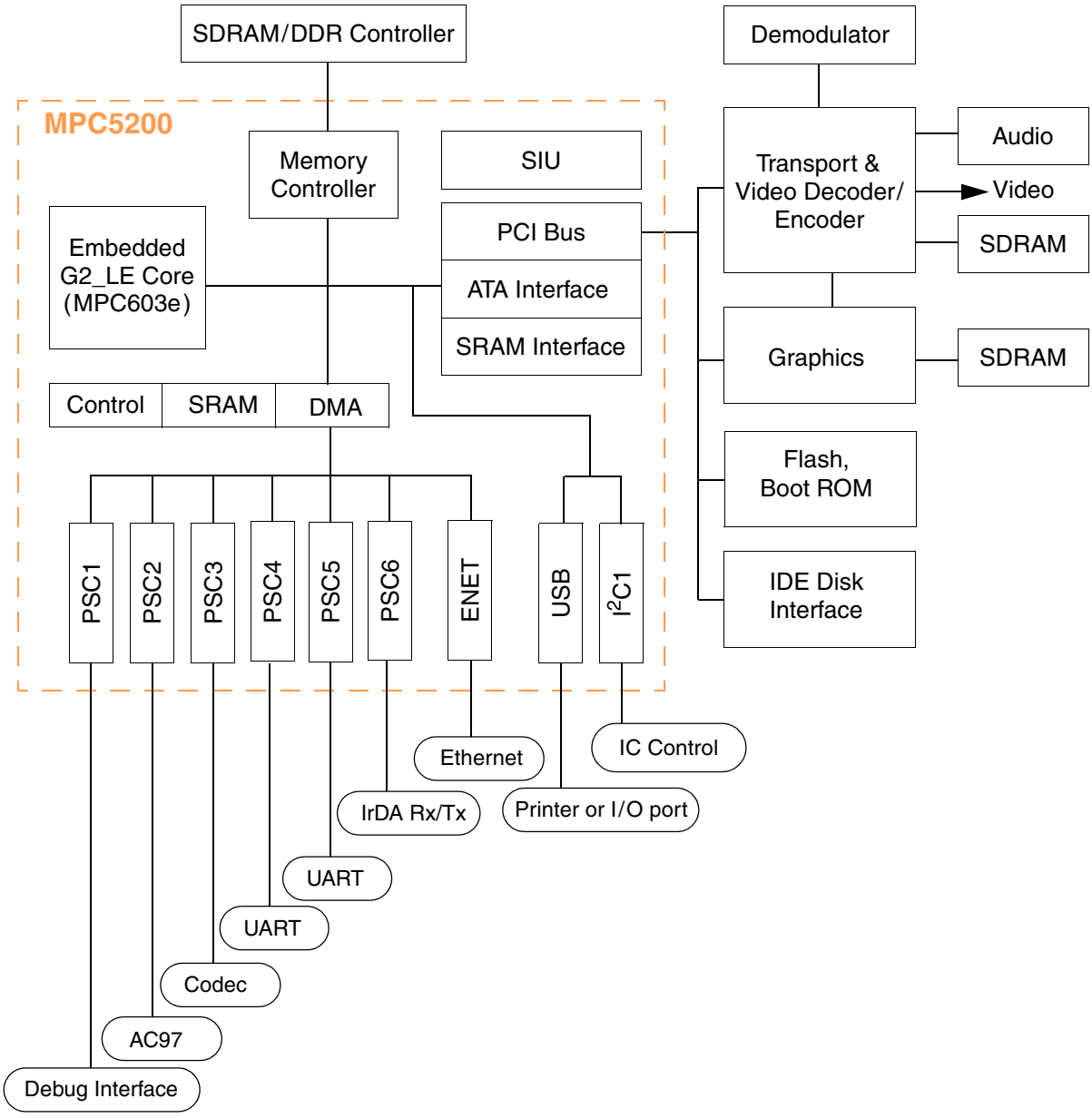


Figure 1-2. MPC5200-Based System

1.2.1 Embedded G2_LE Core

The MPC5200 embedded 603e G2_LE core is derived from Freescale's (formerly Motorola) MPC603e family of Reduced Instruction Set Computer (RISC) microprocessors. The 603e G2_LE core is a high-performance low-power implementation of the PowerPC superscalar architecture. The MPC5200 603e G2_LE core contains:

- 16KBytes of instruction cache
- 16KBytes of data cache

Caches are 4-way set associative and use the Least Recently Used (LRU) replacement algorithm.

Four independent execution units are used:

1. Branch Processing Unit (BPU)
2. Integer Unit (IU)
3. Load/Store Unit (LSU)
4. System Register Unit (SRU)

Up to 3 instructions can be issued and retired per clock. Most instructions execute in a single cycle. The core contains an integrated Floating Point Unit (FPU), a Data Cache Memory Management Unit and an Instruction Cache Memory Management Unit.. The core implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addressing and integer data types of 8-, 16-, and 32-bits.

Enhancements in this core version, specific to embedded automotive/telematics include:

- Improved interrupt latency (critical interrupt)
- New MMU with additional 8 BAT (16 total) registers and 1KByte page management

The 603e G2_LE core performance for SPEC95 benchmark integer operations, ranges between 4.4 and 5.1 at 200MHz. In Drystone 2.1MIPS, the 603e G2_LE core is 280MIPS at 200MHz.

1.2.2 BestComm I/O Subsystem

BestComm contains an intelligent DMA unit. This unit provides a front-line interrupt control and data movement interface via a separate peripheral bus to the on-chip peripheral functions. This leaves the 603e G2_LE core free for higher level activities. The concurrent operation enables a significant boost in overall systems performance.

BestComm supports up to 16 simultaneously enabled DMA tasks from up to 32 DMA requestors. Also included is:

- a hardware logic unit
- a hardware CRC unit

BestComm uses internal buffers for prefetched reads and post writes. Bursting is used whenever possible. This optimizes both internal and external bus activity.

1.2.2.1 Programmable Serial Controllers (PSCs)

The MPC5200 supports six PSCs. Each can be configured to operate in different modes. PSCs support both synchronous and asynchronous protocols. They are used to interface to external full-function modems or external CODECs for soft modem support. 8, 16, 24 and 32-bit data widths are supported. PSCs can be configured to support 1200 baud POTS modem, SPI, I^2S, V.34 or V.90 protocols. The standard UART interface supports connection to an external terminal/computer for debug support.

1.2.2.2 10/100 Ethernet Controller

The Ethernet Controller supports the following standard MAC-PHY interfaces:

- 100Mbps IEEE 802.3 MII
- 10Mbps IEEE 802.3 MII
- 10Mbps 7-wire interface

The controller is full duplex, supports a programmable maximum frame length and retransmission from the Tx FIFO following a collision.

1.2.2.3 Universal Serial Bus (USB)

The MPC5200 supports two USB channels. The USB Controller implements the USB Host Controller/Root Hub in compliance with the USB1.1 specification. The user may choose to have either one or two USB ports on the root hub, each of which can interface to an off-chip USB transceiver. The Host Controller supports the Open Host Controller Interface (OHCI) standard.

1.2.2.4 Infrared Support

The MPC5200 supports the IrDA format. All three IrDA modes are supported (SIR, MIR, FIR) to 4.0Mbps. The required 48MHz clock can be generated internally or supplied externally on an input pin.

1.2.2.5 Inter-Integrated Circuit (I^2C)

The MPC5200 supports two I^2C channels. Both master and slave interfaces can be controlled directly by the processor or can use the BestComm Controller to buffer Tx/Rx data when the I^2C data rate is high.

1.2.2.6 Serial Peripheral Interface (SPI)

The SPI module allows full-duplex, synchronous, serial communication between the MPC5200 and peripheral devices. It supports master and slave mode, double-buffered operation and can operate in a polling or interrupt driven environment.

1.2.3 Controller Area Network (CAN)

The MPC5200 supports two CAN channels. The CAN is an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1Mbps), short distance, priority based protocol that runs on a variety of mediums. For example, transmission media of fiber optic cable or unshielded twisted wire pairs can be used.

MSCAN supports both standard and extended identifier (ID) message formats specified in BOSCH CAN protocol specification, revision 2.0, part B. Each MSCAN module contains:

- 4 receive buffers (with FIFO storage scheme)
- 3 transmit buffers
- flexible mask able identifier filters

1.2.4 Byte Data Link Controller - Digital BDLC-D

The MPC5200 supports J1850 Class B data communication network interface compatible and ISO compatible for low speed (<125kbps) serial data communications in automotive applications.

- Hardware cyclical redundancy check (CRC) generation and checking
- Two power saving modes with automatic wake up on network activity
- Polling and CPU interrupt available
- Block mode receive/transmit supported
- Supports 4X mode, 41.6 kbps
- In-frame response (IFR) types 0, 1, 2, and 3 supported
- Wake up on J1850 message

1.2.5 System Level Interfaces

System Level Interfaces are listed below and described in the sections that follow:

- Chip Selects
- Interrupt Controller
- Timers
- General Purpose Input/Outputs (GPIO)
- Functional Pin Multiplexing
- Real-Time Clock (RTC)

1.2.5.1 Chip Selects

The MPC5200 integrates the most common system integration interfaces and signals. There are 8 fully programmable external chip selects, which are independent of the SDRAM interface. LP_CS0 has special features to support a Boot ROM. Two of the chip selects may be used by the IDE disk drive interface, when enabled.

1.2.5.2 Interrupt Controller

The Interrupt Controller has 4 external interrupt signals and manages both external and internal interrupts. All interrupt levels and priorities are programmable.

The Interrupt Controller takes advantage of the new critical interrupt feature defined by the PowerPC architecture. This allows 603e G2_LE core interrupts outside operating system boundaries, for critical functions such as real-time packet processing.

1.2.5.3 Timers

MPC5200 integrates several timer functions required by most embedded systems:

- Two internal Slice timers can create short-cycle periodic interrupts.
- A WatchDog timer can interrupt the processor if not regularly serviced, catching software hang-ups.

A bus monitor monitors bus cycles and provides an interrupt if transactions take longer than a prescribed time.

1.2.5.4 General Purpose Input/Outputs (GPIO)

A total of 56 pins on the MPC5200 can be programmed as GPIOs.

- 8 pins can interrupt the processor.
- 8 pins can support a “Wake Up” capability that lets the MPC5200 be brought out of low power modes.
- 8 pins are “output only” GPIOs.

The remaining GPIO pins support a simple “set the output level” or “detect the input level” type GPIO function. Eight I/Os can be connected to one of eight general purpose timers to support input capture, output compare or pulse width modulation functions.

The number of GPIOs available in the various modes depends on the peripheral functionality required. See pin descriptions and I/O port maps below for more information.

1.2.5.5 Functional Pin Multiplexing

Many serial/parallel port pins serve multiple functions, allowing flexibility in optimizing the system to meet a specific set of integration requirements. For example, when PSC3 interfaces to a full function external modem, 10 pins are required:

- PSC3_TXD—Transmit Data
- PSC3_RXD—Receive Data
- PSC3_$\overline{\text{RTS}}$—Ready to Send
- PSC3_$\overline{\text{CTS}}$—Clear to Send
- PSC3_CD—Carrier Detect
- MODEM_RI—Ring Indicator
- MODEM_DSR—Hook Switch
- MODEM_IO—Control I/O (A0 gain)
- MODEM_IO—Control I/O (Mode 1)
- MODEM_IO—Control I/O (Mode 2)

If PSC3 connects to a simple UART, only the first four signals (shown above) are required. The remaining 6 signals can be used as GPIOs.

If a 7-wire Ethernet connection is adequate, the additional 11 Ethernet I/Os can be used as GPIOs.

1.2.5.6 Real-Time Clock (RTC)

An RTC is included on the MPC5200. The RTC provides a 2-pin interface to an external 32.768 KHz crystal. This allows internal time-of-day/calendar tracking, as well as clock based periodic interrupts.

1.2.6 SDRAM Controller and Interface

The MPC5200 high speed SDRAM Controller supports both standard SDRAM and Double Data Rate (DDR) SDRAM devices. It supports up to 256 MBytes per chip select (2 Chip Select lines available) with a 32-bit interface. Memory sizes of 64-Mbit, 128-Mbit, 256-Mbit and 512-Mbit are supported.

1.2.7 Multi-Function External LocalPlus Bus

The MPC5200 supports a multi-function external LocalPlus Bus to allow connections to PCI and ATA compliant devices, as well as external ROM/SRAM.

The MPC5200 integrates a 3.3 V, PCI V2.2 compatible external LocalPlus Bus controller and interface. This bus is a 32-bit multiplexed address/data bus.

The external LocalPlus Bus provides support for an ATA disk drive interface. ATA control signals (chip selects, write/read, etc.) are provided independent of the PCI control signals. This prevents bus contention. However, the 32-bit data bus is shared. When The MPC5200 recognizes an external LocalPlus Bus access meant for the ATA Controller, ATA control logic arbitrates for PCI interface control. The 32-bit address/data bus function is transformed into 16 bits of ATA data and 3 bits of ATA address.

The external LocalPlus Bus also allows connection to external memory or peripheral devices that adhere to a ROM or SRAM-like interface. These devices occupy a separate location in the memory map and have independent control signals. When an internal access is decoded to fall in the SRAM/ROM memory space, the 32-bit PCI address/data bus is transformed into either:

- 24 bits of address and 8 bits of data
- 16 bits of address and 16 bits of data.

The MPC5200 supports a reset configuration mode common on the family of processors that use the PowerPC architecture. 16 bits of configuration information is driven and sampled during reset to establish the initial processor configuration.

1.2.8 Power Management

The MPC5200 is processed in a low-power static CMOS technology. In addition, it supports the dynamic power management modes available on the MPC603e series processors. These modes include:

- nap
- dose
- sleep
- deep sleep

In deep sleep, all internal clocks can be disabled, thus, reducing the power draw to CMOS leakage levels.

A Wake Up capability is supported by CAN, RTC, several GPIOs and the interrupt lines. Therefore, the MPC5200 can be shut down to a low-power standby mode, then re-enabled by one of the Wake Up inputs without resetting the MPC5200.

1.2.9 Systems Debug and Test

The MPC5200 supports the Common On-chip Processor (COP) debug capability common on other microprocessors that use the PowerPC architecture. The COP interface supports features such as:

- memory down load
- single step instruction execution
- break/watch point capability
- access to internal registers
- pipeline tracking, etc.

The MPC5200 also supports a JTAG IEEE 1149.1 controller and test access port (TAP).

1.2.10 Physical Characteristics

- 1.5 V internal, 3.3 V external operation (2.5 v for DDR interface)
- TTL compatible I/O pins
- 272-pin Plastic Ball Grid Array (PBGA)

Chapter 2
Signal Descriptions

2.1 Overview

The MPC5200 contains a 603e G2_LE CPU core, an internal DMA engine, BestComm, multiple functional blocks and associated I/O ports. There are two external data/address bus structures, the LocalPlus bus and SDRAM bus. A block diagram of the MPC5200 structure is shown in Figure 1-1.

In general, the LocalPlus bus connects to external SRAM, FLASH, peripheral devices, etc. The LocalPlus bus is capable of executing standard memory cycles, PCI cycles and ATA cycles. In addition to the data and address bus pins on the LocalPlus bus, there are pins specifically dedicated to ATA transactions, PCI transactions and standard memory transactions. When the MPC5200 is released from reset, Chip Select 0 is the only active chip select. Program execution must always start from the “boot device” on the LocalPlus bus. There are 8 chip select signals associated with the LocalPlus bus. It’s possible to execute from every CS. Also every CS can address “data space”.

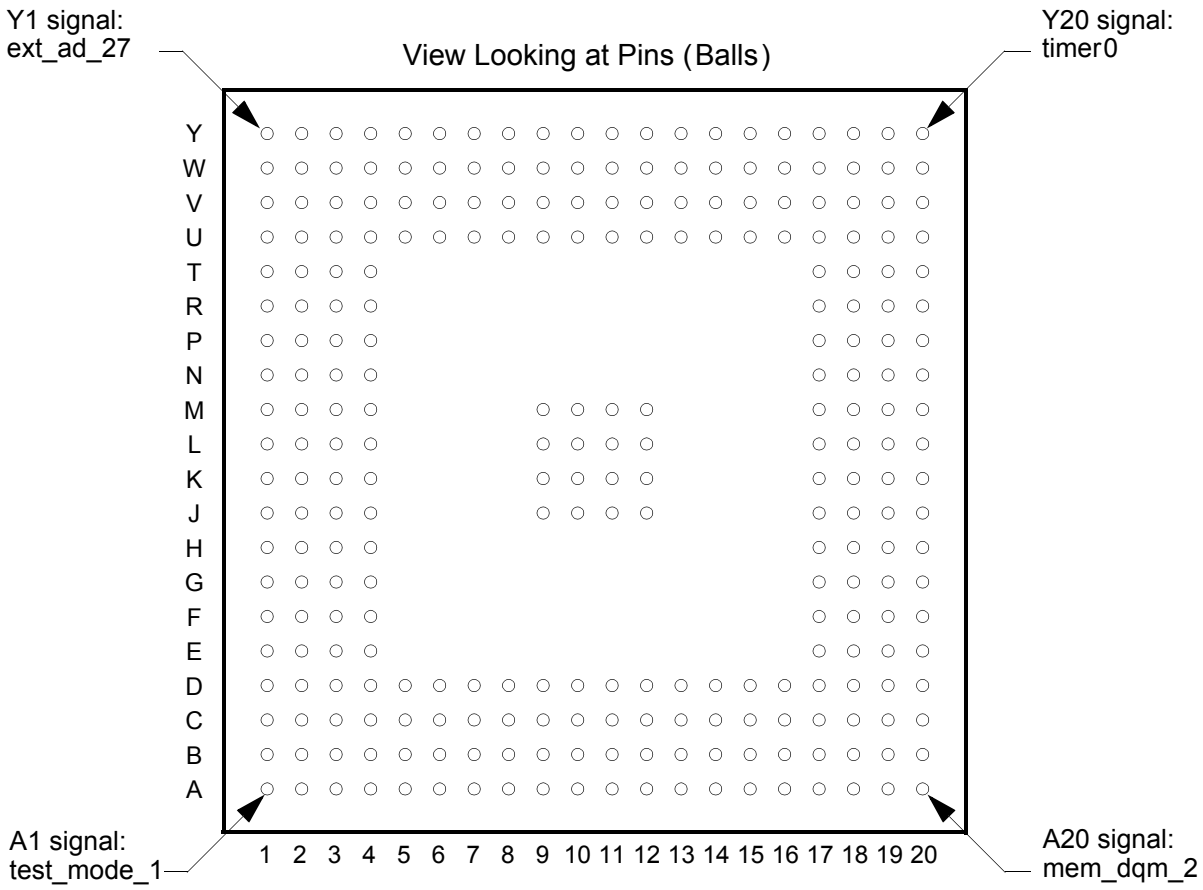
The SDRAM bus interfaces to Synchronous DRAM. Both Single Data Rate and Double Data Rate DRAMs are supported. Executable programs are generally loaded into memory residing on the SDRAM bus. The SDRAM bus has a 32-bit wide data/address bus structure and is capable of burst accesses. It is possible to execute program code over the LocalPlus bus. However, the data transfer rate on the SDRAM bus is many times faster than LocalPlus.

There are 16 peripheral functional blocks on the MPC5200. These are General Purpose I/O, I2C, TIMER, PSC1, PSC2, PSC3, PSC4, PSC5, PSC6, Ethernet, USB, MSCAN, SPI and J1850. Each of these functional blocks are routed to one or more I/O ports through a system of multiplexers. A functional block can only be routed to one I/O port at a time and in many cases, several functional blocks can be routed to the same I/O port.

The I/O ports are Dedicated GPIO Group, I^2C Group, Timer Group, PSC1 Group, PSC2 Group, PSC3 Group, PSC6 Group, Ethernet Group, and the USB Group.

Figures 2-2 through 2-10 present detailed on the multiplexing options for each I/O port.

MPC5200 is packaged in a 272-pin Plastic Ball Gate Array (PBGA). Package ball locations are shown in Figure 2-1. See Appendix D, for case diagram.



Note: Table 2-1 and Table 2-2 give the signals on each pin/ball.

Figure 2-1. 272-Pin PBGA Pin Detail

Table 2-1 gives a list of MPC5200 I/O signals sorted by package ball name. Table 2-2 gives the same list sorted by signal name.

Many signal pins can have multiple functions depending on internal register settings. These additional functions are described in Table 2-3 through Table 2-31.

A01 TEST_MODE_1	A02 JTAG_TDO	A03 JTAG_TDI	A04 JTAG_TMS	A05 PSC3_8	A06 PSC3_5	A07 PSC3_2	A08 PSC2_4	A09 PSC2_2	A10 PSC1_4	A11 PSC1_1	A12 PSC6_2	A13 PORRESET	A14 SRESET	A15 SYS_XTAL_IN	A16 MEM_MA_1	A17 MEM_MBA_1	A18 MEM_RAS	A19 MEM_WE	A20 MEM_DQM_2
B01 TEST_SEL_0	B02 TEST_MODE_0	B03 JTAG_TRST	B04 JTAG_TCK	B05 PSC3_7	B06 PSC3_4	B07 PSC3_1	B08 PSC2_3	B09 PSC2_1	B10 PSC1_3	B11 PSC1_0	B12 PSC6_0	B13 HRESET	B14 SYS_PLL_AVDD	B15 SYS_PLL_TPA	B16 MEM_MA_2	B17 MEM_MA_10	B18 MEM_CS_0	B19 MEM_CAS	B20 MEM_MA_4
C01 RTC_XTAL_OUT	C02 RTC_XTAL_IN	C03 TEST_SEL_1	C04 PSC3_9	C05 PSC3_6	C06 PSC3_3	C07 PSC3_0	C08 CORE_PLL_AVDD	C09 PSC2_0	C10 PSC1_2	C11 PSC6_1	C12 GPIO_WKUP_7	C13 PSC6_3	C14 SYS_PLL_AVSS	C15 GPIO_WKUP_6	C16 MEM_MA_3	C17 MEM_MA_0	C18 MEM_MBA_0	C19 MEM_MA_5	C20 MEM_MA_6
D01 TIMER_4	D02 TIMER_3	D03 TIMER_2	D04 VSS	D05 VDD_CORE	D06 VDD_IO	D07 VDD_CORE	D08 LP_OE	D09 VDD_IO	D10 VDD_CORE	D11 VDD_CORE	D12 VDD_MEM_IO	D13 VDD_MEM_IO	D14 SYS_XTAL_OUT	D15 VDD_MEM_IO	D16 VSS	D17 VDD_MEM_IO	D18 MEM_MDQS_2	D19 MEM_MA_7	D20 MEM_MA_8
E01 TIMER_7	E02 TIMER_6	E03 TIMER_5	E04 VDD_IO													E17 VDD_MEM_IO	E18 MEM_MDQ_16	E19 MEM_MA_9	E20 MEM_MA_11
F01 USB_7	F02 USB_8	F03 USB_9	F04 VDD_IO													F17 VDD_MEM_IO	F18 MEM_MDQ_17	F19 MEM_MA_12	F20 MEM_CLK_EN
G01 USB_3	G02 USB_4	G03 USB_5	G04 USB_6													G17 MEM_MDQ_18	G18 MEM_MDQ_19	G19 MEM_CLK	G20 MEM_CLK
H01 USB_0	H02 USB_1	H03 USB_2	H04 VDD_IO													H17 d VDD_MEM_IO	H18 203 MEM_MDQ_20	H19 204 MEM_DQM_1	H20 206 MEM_MDQS_1
J01 ETH_3	J02 ETH_4	J03 ETH_10	J04 ETH_17					J09 VSS	J10 VSS	J11 VSS	J12 VSS					J17 MEM_MDQ_22	J18 MEM_MDQ_21	J19 MEM_MDQ_8	J20 MEM_MDQ_9
K01 ETH_0	K02 ETH_1	K03 ETH_2	K04 VDD_CORE					K09 VSS	K10 VSS	K11 VSS	K12 VSS					K17 d VDD_MEM_IO	K18 193 MEM_MDQ_23	K19 194 MEM_MDQ_10	K20 196 MEM_MDQ_11
L01 ETH_9	L02 ETH_16	L03 ETH_5	L04 ETH_11					L09 VSS	L10 VSS	L11 VSS	L12 VSS					L17 MEM_DQM_3	L18 MEM_MDQS_3	L19 MEM_MDQ_12	L20 MEM_MDQ_13
M01 ETH_13	M02 ETH_12	M03 ETH_8	M04 VDD_CORE					M09 VSS	M10 VSS	M11 VSS	M12 VSS					M17 d VDD_MEM_IO	M18 182 MEM_MDQ_24	M19 184 MEM_MDQ_14	M20 186 MEM_MDQ_15
N01 ETH_7	N02 ETH_6	N03 ETH_15	N04 ETH_14													N17 MEM_MDQ_25	N18 MEM_MDQ_26	N19 MEM_DQM_0	N20 MEM_MDQS_0
P01 IRQ1	P02 IRQ2	P03 IRQ0	P04 VDD_CORE													P17 d VDD_MEM_IO	P18 172 MEM_MDQ_27	P19 174 MEM_MDQ_7	P20 176 MEM_MDQ_6
R01 IRQ3	R02 PCI_RESET	R03 EXT_AD_30	R04 PCI_GNT													R17 MEM_MDQ_28	R18 MEM_MDQ_29	R19 MEM_MDQ_5	R20 MEM_MDQ_4
T01 PCI_CLOCK	T02 EXT_AD_26	T03 EXT_AD_28	T04 VDD_IO													T17 VDD_MEM_IO	T18 MEM_MDQ_30	T19 MEM_MDQ_3	T20 MEM_MDQ_2
U01 PCI_REQ	U02 PCI_IDSEL	U03 EXT_AD_24	U04 VSS	U05 VDD_IO	U06 VDD_IO	U07 VDD_CORE	U08 EXT_AD_15	U09 VDD_IO	U10 VDD_IO	U11 EXT_AD_6	U12 VDD_CORE	U13 VDD_IO	U14 LP_ACK	U15 VDD_CORE	U16 VDD_IO	U17 VSS	U18 MEM_MDQ_31	U19 MEM_MDQ_1	U20 MEM_MDQ_0
V01 EXT_AD_31	V02 EXT_AD_20	V03 EXT_AD_22	V04 EXT_AD_18	V05 PCI_FRAME	V06 PCI_STOP	V07 PCI_PAR	V08 EXT_AD_13	V09 EXT_AD_11	V10 EXT_AD_9	V11 EXT_AD_4	V12 EXT_AD_2	V13 EXT_AD_0	V14 LP_ALE	V15 LP_CS2	V16 LP_CS5	V17 ATA_DRQ	V18 TIMER_1	V19 I2C_0	V20 I2C_2
W01 EXT_AD_29	W02 EXT_AD_25	W03 EXT_AD_23	W04 EXT_AD_16	W05 PCI_TRDY	W06 PCI_CBE_2	W07 PCI_DEVSEL	W08 PCI_SERR	W09 EXT_AD_14	W10 PCI_CBE_0	W11 EXT_AD_8	W12 EXT_AD_5	W13 EXT_AD_1	W14 LP_CS0	W15 LP_CS3	W16 LP_RW	W17 ATA_IOW	W18 ATA_IOCHRDY	W19 I2C_1	W20 I2C_3
Y01 EXT_AD_27	Y02 PCI_CBE_3	Y03 EXT_AD_21	Y04 EXT_AD_19	Y05 EXT_AD_17	Y06 PCI_IRDY	Y07 PCI_PERR	Y08 PCI_CBE_1	Y09 EXT_AD_12	Y10 EXT_AD_10	Y11 EXT_AD_7	Y12 EXT_AD_3	Y13 LP_TS	Y14 LP_CS1	Y15 LP_CS4	Y16 ATA_ISOLATION	Y17 ATA_IOR	Y18 ATA_DACK	Y19 ATA_INTRQ	Y20 TIMER_0

Key for IO Balls:

A6
PSC3_5

<-- Ball

<-- Signal Name

Key for PWR/GND Balls:

VSS	Core and IO VSS
VDD_CORE	1.5V Core VDD
VDD IO	3.3V IO VDD
VDD_MEM_IO	Memory VDD

Figure 2-2. 272-Pin PBGA — Top View

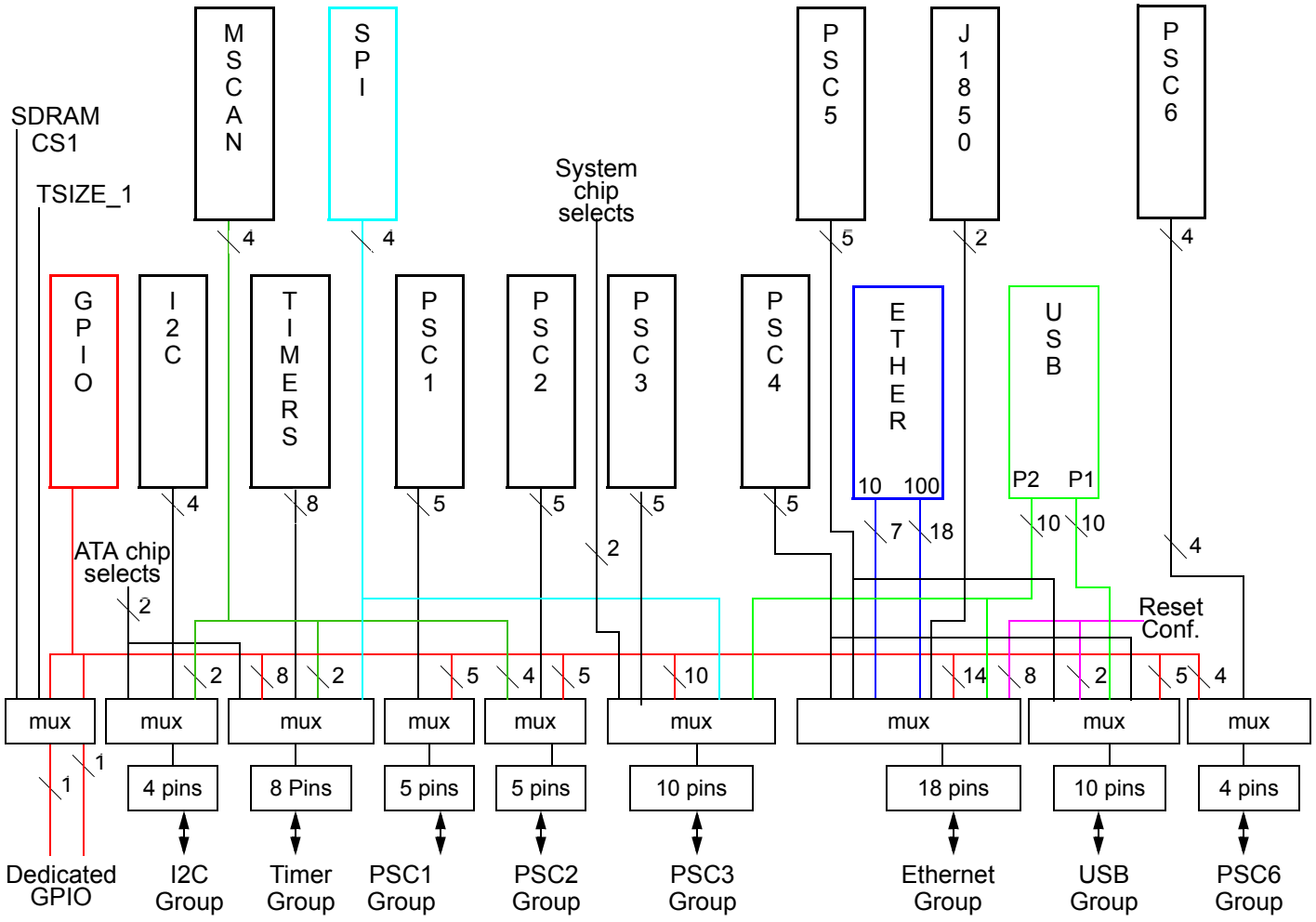


Figure 2-3. MPC5200 Peripheral Muxing

2.2 Pinout Tables

Table 2-1. Signals by Ball/Pin

Ball/Pin	Pin Name
A01	TEST_MODE_1
A02	JTAG_TDO
A03	JTAG_TDI
A04	JTAG_TMS
A05	PSC3_8
A06	PSC3_5
A07	PSC3_2
A08	PSC2_4
A09	PSC2_2
A10	PSC1_4

Ball/Pin	Pin Name
B16	MEM_MA_2
B17	MEM_MA_10
B18	MEM_CS_0
B19	MEM_CAS
B20	MEM_MA_4
C01	RTC_XTAL_OUT
C02	RTC_XTAL_IN
C03	TEST_SEL_1
C04	PSC3_9
C05	PSC3_6

Table 2-1. Signals by Ball/Pin (continued)

Ball/Pin	Pin Name
A11	PSC1_1
A12	PSC6_2
A13	PORRESET
A14	SRESET
A15	SYS_XTAL_IN
A16	MEM_MA_1
A17	MEM_MBA_1
A18	MEM_RAS
A19	MEM_WE
A20	MEM_DQM_2
B01	TEST_SEL_0
B02	TEST_MODE_0
B03	JTAG_TRST
B04	JTAG_TCK
B05	PSC3_7
B06	PSC3_4
B07	PSC3_1
B08	PSC2_3
B09	PSC2_1
B10	PSC1_3
B11	PSC1_0
B12	PSC6_0
B13	HRESET
B14	SYS_PLL_AVDD
B15	SYS_PLL_TPA
D11	VDD_CORE
D12	VDD_MEM_IO
D13	VDD_MEM_IO
D14	SYS_XTAL_OUT
D15	VDD_MEM_IO
D16	VSS_IO/CORE
D17	VDD_MEM_IO
D18	MEM_MDQS_2
D19	MEM_MA_7
D20	MEM_MA_8

Ball/Pin	Pin Name
C06	PSC3_3
C07	PSC3_0
C08	CORE_PLL_AVDD
C09	PSC2_0
C10	PSC1_2
C11	PSC6_1
C12	GPIO_WKUP_7
C13	PSC6_3
C14	SYS_PLL_AVSS
C15	GPIO_WKUP_6
C16	MEM_MA_3
C17	MEM_MA_0
C18	MEM_MBA_0
C19	MEM_MA_5
C20	MEM_MA_6
D01	TIMER_4
D02	TIMER_3
D03	TIMER_2
D04	VSS_IO/CORE
D05	VDD_CORE
D06	VDD_IO
D07	VDD_CORE
D08	LP_OE
D09	VDD_IO
D10	VDD_CORE
H04	VDD_IO
H17	VDD_MEM_IO
H18	MEM_MDQ_20
H19	MEM_DQM_1
H20	MEM_MDQS_1
J01	ETH_3
J02	ETH_4
J03	ETH_10
J04	ETH_17
J09	VSS_IO/CORE

Table 2-1. Signals by Ball/Pin (continued)

Ball/Pin	Pin Name
E01	TIMER_7
E02	TIMER_6
E03	TIMER_5
E04	VDD_IO
E17	VDD_MEM_IO
E18	MEM_MDQ_16
E19	MEM_MA_9
E20	MEM_MA_11
F01	USB_7
F02	USB_8
F03	USB_9
F04	VDD_IO
F17	VDD_MEM_IO
F18	MEM_MDQ_17
F19	MEM_MA_12
F20	MEM_CLK_EN
G01	USB_3
G02	USB_4
G03	USB_5
G04	USB_6
G17	MEM_MDQ_18
G18	MEM_MDQ_19
G19	MEM_CLK
G20	MEM_CLK
H01	USB_0
H02	USB_1
H03	USB_2
L17	MEM_DQM_3
L18	MEM_MDQS_3
L19	MEM_MDQ_12
L20	MEM_MDQ_13
M01	ETH_13
M02	ETH_12
M03	ETH_8
M04	VDD_CORE

Ball/Pin	Pin Name
J10	VSS_IO/CORE
J11	VSS_IO/CORE
J12	VSS_IO/CORE
J17	MEM_MDQ_22
J18	MEM_MDQ_21
J19	MEM_MDQ_8
J20	MEM_MDQ_9
K01	ETH_0
K02	ETH_1
K03	ETH_2
K04	VDD_CORE
K09	VSS_IO/CORE
K10	VSS_IO/CORE
K11	VSS_IO/CORE
K12	VSS_IO/CORE
K17	VDD_MEM_IO
K18	MEM_MDQ_23
K19	MEM_MDQ_10
K20	MEM_MDQ_11
L01	ETH_9
L02	ETH_16
L03	ETH_5
L04	ETH_11
L09	VSS_IO/CORE
L10	VSS_IO/CORE
L11	VSS_IO/CORE
L12	VSS_IO/CORE
R18	MEM_MDQ_29
R19	MEM_MDQ_5
R20	MEM_MDQ_4
T01	PCI_CLOCK
T02	EXT_AD_26
T03	EXT_AD_28
T04	VDD_IO
T17	VDD_MEM_IO

Table 2-1. Signals by Ball/Pin (continued)

Ball/Pin	Pin Name
M09	VSS_IO/CORE
M10	VSS_IO/CORE
M11	VSS_IO/CORE
M12	VSS_IO/CORE
M17	VDD_MEM_IO
M18	MEM_MDQ_24
M19	MEM_MDQ_14
M20	MEM_MDQ_15
N01	ETH_7
N02	ETH_6
N03	ETH_15
N04	ETH_14
N17	MEM_MDQ_25
N18	MEM_MDQ_26
N19	MEM_DQM_0
N20	MEM_MDQS_0
P01	IRQ1
P02	IRQ2
P03	IRQ0
P04	VDD_CORE
P17	VDD_MEM_IO
P18	MEM_MDQ_27
P19	MEM_MDQ_7
P20	MEM_MDQ_6
R01	IRQ3
R02	PCI_RESET
R03	EXT_AD_30
R04	PCI_GNT
R17	MEM_MDQ_28
V07	PCI_PAR
V08	EXT_AD_13
V09	EXT_AD_11
V10	EXT_AD_9
V11	EXT_AD_4
V12	EXT_AD_2

Ball/Pin	Pin Name
T18	MEM_MDQ_30
T19	MEM_MDQ_3
T20	MEM_MDQ_2
U01	PCI_REQ
U02	PCI_IDSEL
U03	EXT_AD_24
U04	VSS_IO/CORE
U05	VDD_IO
U06	VDD_IO
U07	VDD_CORE
U08	EXT_AD_15
U09	VDD_IO
U10	VDD_IO
U11	EXT_AD_6
U12	VDD_CORE
U13	VDD_IO
U14	LP_ACK
U15	VDD_CORE
U16	VDD_IO
U17	VSS_IO/CORE
U18	MEM_MDQ_31
U19	MEM_MDQ_1
U20	MEM_MDQ_0
V01	EXT_AD_31
V02	EXT_AD_20
V03	EXT_AD_22
V04	EXT_AD_18
V05	PCI_FRAME
V06	PCI_STOP
Y04	EXT_AD_19
Y05	EXT_AD_17
Y06	PCI_IRDY
Y07	PCI_PERR
Y08	PCI_CBE_1
Y09	EXT_AD_12

Table 2-1. Signals by Ball/Pin (continued)

Ball/Pin	Pin Name
V13	EXT_AD_0
V14	LP_ALE
V15	LP_CS2
V16	LP_CS5
V17	ATA_DRQ
V18	TIMER_1
V19	I2C_0
V20	I2C_2
W01	EXT_AD_29
W02	EXT_AD_25
W03	EXT_AD_23
W04	EXT_AD_16
W05	PCI_TRDY
W06	PCI_CBE_2
W07	PCI_DEVSEL
W08	PCI_SERR
W09	EXT_AD_14
W10	PCI_CBE_0
W11	EXT_AD_8
W12	EXT_AD_5
W13	EXT_AD_1
W14	LP_CS0
W15	LP_CS3
W16	LP_RW
W17	ATA_IOW
W18	ATA_IOCHRDY
W19	I2C_1
W20	I2C_3
Y01	EXT_AD_27
Y02	PCI_CBE_3
Y03	EXT_AD_21

Ball/Pin	Pin Name
Y10	EXT_AD_10
Y11	EXT_AD_7
Y12	EXT_AD_3
Y13	LP_TS
Y14	LP_CS1
Y15	LP_CS4
Y16	ATA_ISOLATION
Y17	ATA_IOR
Y18	ATA_DACK
Y19	ATA_INTRQ
Y20	TIMER_0

Table 2-2. Signals by Signal Name

Signal Name	Ball/Pin
ATA_DACK	Y18
ATA_DRQ	V17
ATA_INTRQ	Y19
ATA_IOCHRDY	W18
ATA_IOR	Y17
ATA_IOW	W17
ATA_ISOLATION	Y16
LP_CS0	W14
LP_CS1	Y14
LP_CS2	V15
LP_CS3	W15
LP_CS4	Y15
LP_CS5	V16
ETH_0	K01
ETH_1	K02
ETH_2	K03
ETH_3	J01
ETH_4	J02
ETH_5	L03
ETH_6	N02
ETH_7	N01
ETH_8	M03
ETH_9	L01
ETH_10	J03
ETH_11	L04
ETH_12	M02
ETH_13	M01
ETH_14	N04
ETH_15	N03
ETH_16	L02
ETH_17	J04
EXT_AD_0	V13
EXT_AD_1	W13
EXT_AD_2	V12
EXT_AD_3	Y12

Signal Name	Ball/Pin
EXT_AD_6	U11
EXT_AD_7	Y11
EXT_AD_8	W11
EXT_AD_9	V10
EXT_AD_10	Y10
EXT_AD_11	V09
EXT_AD_12	Y09
EXT_AD_13	V08
EXT_AD_14	W09
EXT_AD_15	U08
EXT_AD_16	W04
EXT_AD_17	Y05
EXT_AD_18	V04
EXT_AD_19	Y04
EXT_AD_20	V02
EXT_AD_21	Y03
EXT_AD_22	V03
EXT_AD_23	W03
EXT_AD_24	U03
EXT_AD_25	W02
EXT_AD_26	T02
EXT_AD_27	Y01
EXT_AD_28	T03
EXT_AD_29	W01
EXT_AD_30	R03
EXT_AD_31	V01
GPIO_WKUP_6	C15
GPIO_WKUP_7	C12
CORE_PLL_AVDD	C08
CORE_PLL_AVSS	NC (no connection)
HRESET	B13
I2C_0	V19
I2C_1	W19
I2C_2	V20
I2C_3	W20

Table 2-2. Signals by Signal Name (continued)

Signal Name	Ball/Pin
EXT_AD_4	V11
EXT_AD_5	W12
PSC6_3	C13
PSC6_1	C11
IRQ0	P03
IRQ1	P01
IRQ2	P02
IRQ3	R01
JTAG_TCK	B04
JTAG_TDI	A03
JTAG_TDO	A02
JTAG_TMS	A04
JTAG_TRST	B03
LP_ACK	U14
LP_ALE	V14
LP_OE	D08
LP_RW	W16
LP_TS	Y13
MEM_CAS	B19
MEM_CLK_EN	F20
MEM_CS_0	B18
MEM_DQM_0	N19
MEM_DQM_1	H19
MEM_DQM_2	A20
MEM_DQM_3	L17
MEM_MA_0	C17
MEM_MA_1	A16
MEM_MA_2	B16
MEM_MA_3	C16
MEM_MA_4	B20
MEM_MA_5	C19
MEM_MA_6	C20
MEM_MA_7	D19
MEM_MA_8	D20
MEM_MA_9	E19

Signal Name	Ball/Pin
PSC6_0	B12
PSC6_2	A12
MEM_MBA_1	A17
MEM_MDQ_0	U20
MEM_MDQ_1	U19
MEM_MDQ_2	T20
MEM_MDQ_3	T19
MEM_MDQ_4	R20
MEM_MDQ_5	R19
MEM_MDQ_6	P20
MEM_MDQ_7	P19
MEM_MDQ_8	J19
MEM_MDQ_9	J20
MEM_MDQ_10	K19
MEM_MDQ_11	K20
MEM_MDQ_12	L19
MEM_MDQ_13	L20
MEM_MDQ_14	M19
MEM_MDQ_15	M20
MEM_MDQ_16	E18
MEM_MDQ_17	F18
MEM_MDQ_18	G17
MEM_MDQ_19	G18
MEM_MDQ_20	H18
MEM_MDQ_21	J18
MEM_MDQ_22	J17
MEM_MDQ_23	K18
MEM_MDQ_24	M18
MEM_MDQ_25	N17
MEM_MDQ_26	N18
MEM_MDQ_27	P18
MEM_MDQ_28	R17
MEM_MDQ_29	R18
MEM_MDQ_30	T18
MEM_MDQ_31	U18

Table 2-2. Signals by Signal Name (continued)

Signal Name	Ball/Pin
MEM_MA_10	B17
MEM_MA_11	E20
MEM_MA_12	F19
MEM_MBA_0	C18
MEM_CLK	G19
MEM_CLK	G20
MEM_RAS	A18
MEM_WE	A19
PCI_CBE_0	W10
PCI_CBE_1	Y08
PCI_CBE_2	W06
PCI_CBE_3	Y02
PCI_CLOCK	T01
PCI_DEVSEL	W07
PCI_FRAME	V05
PCI_GNT	R04
PCI_IDSEL	U02
PCI_IRDY	Y06
PCI_PAR	V07
PCI_PERR	Y07
PCI_REQ	U01
PCI_RESET	R02
PCI_SERR	W08
PCI_STOP	V06
PCI_TRDY	W05
PORRESET	A13
PSC1_0	B11
PSC1_1	A11
PSC1_2	C10
PSC1_3	B10
PSC1_4	A10
PSC2_0	C09
PSC2_1	B09
PSC2_2	A09
PSC2_3	B08

Signal Name	Ball/Pin
MEM_MDQS_0	N20
MEM_MDQS_1	H20
MEM_MDQS_2	D18
MEM_MDQS_3	L18
PSC3_5	A06
PSC3_6	C05
PSC3_7	B05
PSC3_8	A05
PSC3_9	C04
RTC_XTAL_IN	C02
RTC_XTAL_OUT	C01
SRESET	A14
SYS_PLL_AVDD	B14
SYS_PLL_AVSS	C14
SYS_PLL_TPA	B15
SYS_XTAL_IN	A15
SYS_XTAL_OUT	D14
TEST_MODE_0	B02
TEST_MODE_1	A01
TEST_SEL_0	B01
TEST_SEL_1	C03
TIMER_0	Y20
TIMER_1	V18
TIMER_2	D03
TIMER_3	D02
TIMER_4	D01
TIMER_5	E03
TIMER_6	E02
TIMER_7	E01
USB_0	H01
USB_1	H02
USB_2	H03
USB_3	G01
USB_4	G02
USB_5	G03

Table 2-2. Signals by Signal Name (continued)

Signal Name	Ball/Pin
PSC2_4	A08
PSC3_0	C07
PSC3_1	B07
PSC3_2	A07
PSC3_3	C06
PSC3_4	B06
VDD_CORE	D10
VDD_CORE	D11
VDD_CORE	K04
VDD_CORE	M04
VDD_CORE	P04
VDD_CORE	U07
VDD_CORE	U12
VDD_CORE	U15
VDD_IO	D06
VDD_IO	D09
VDD_IO	E04
VDD_IO	F04
VDD_IO	H4
VDD_IO	T4
VDD_IO	U05
VDD_IO	U06
VDD_IO	U09
VDD_IO	U10
VDD_IO	U13
VDD_IO	U16
VDD_MEM_IO	D12
VDD_MEM_IO	D13
VDD_MEM_IO	D15
VDD_MEM_IO	D17
VDD_MEM_IO	E17
VDD_MEM_IO	F17
VDD_MEM_IO	H17
VDD_MEM_IO	K17
VDD_MEM_IO	M17

Signal Name	Ball/Pin
USB_6	G04
USB_7	F01
USB_8	F02
USB_9	F03
VDD_CORE	D05
VSS_IO/CORE	J12
VSS_IO/CORE	K09
VSS_IO/CORE	K10
VSS_IO/CORE	K11
VSS_IO/CORE	K12
VSS_IO/CORE	L09
VSS_IO/CORE	L10
VSS_IO/CORE	L11
VSS_IO/CORE	L12
VSS_IO/CORE	M09
VSS_IO/CORE	M10
VSS_IO/CORE	M11
VSS_IO/CORE	M12
VSS_IO/CORE	U04
VSS_IO/CORE	U17
VDD_CORE	D07

Table 2-2. Signals by Signal Name (continued)

Signal Name	Ball/Pin
VDD_MEM_IO	P17
VDD_MEM_IO	T17
VSS_IO/CORE	D04
VSS_IO/CORE	D16
VSS_IO/CORE	J09
VSS_IO/CORE	J10
VSS_IO/CORE	J11

Signal Name	Ball/Pin

Table 2-3. LocalPlus Bus Address / Data Pin Assignments

MPC5200 LocaLPlus Bus Address / Data Pins	EXT_AD31	EXT_AD30	EXT_AD29	EXT_AD28	EXT_AD27	EXT_AD26	EXT_AD25	EXT_AD24	EXT_AD23	EXT_AD22	EXT_AD21	EXT_AD20	EXT_AD19	EXT_AD18	EXT_AD17	EXT_AD16	EXT_AD15	EXT_AD14	EXT_AD13	EXT_AD12	EXT_AD11	EXT_AD10	EXT_AD9	EXT_AD8	EXT_AD7	EXT_AD6	EXT_AD5	EXT_AD4	EXT_AD3	EXT_AD2	EXT_AD1	EXT_AD0
16 bit Adr, 16 bit Data	D15	D14	D13	D12	D11	D10	D09	D8	D7	D6	D5	D4	D3	D2	D1	D0	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
24 bit Adr, 8 bit Data	D7	D6	D5	D4	D3	D2	D1	D0	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Muxed modes																																
All Muxed mode Address tenures	0	TS1ZE2	TS1ZE1	TS1ZE0	0	BS1	BS0	A24	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
8 bit Data tenure	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
16 bit Data tenure	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D3	D2	D0																
32 bit Data tenure	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

Table 2-4. LocalPlus Pin Functions

Pin name	BALL	LocalPlus Non-mux		LocalPlus MULTIPLEXED BUS				PCI BUS				ATA	MOST	Large Flash	RESET
		Addr /Data 24/8	Addr /Data 16/16	Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase	PCI Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase				
EXT_AD_31	V01	D7	D15	0	D31	D15	D7	A31	D31	0	0		D31	D15	
EXT_AD_30	R03	D6	D14	TSIZ0	D30	D14	D6	A30	D30	0	0		D30	D14	
EXT_AD_29	W01	D5	D13	TSIZ1	D29	D13	D5	A29	D29	0	0		D29	D13	
EXT_AD_28	T03	D4	D12	TSIZ2	D28	D12	D4	A28	D28	0	0		D28	D12	
EXT_AD_27	Y01	D3	D11	0	D27	D11	D3	A27	D27	0	0		D27	D11	
EXT_AD_26	T02	D2	D10	BS1	D26	D10	D2	A26	D26	0	0		D26	D10	
EXT_AD_25	W02	D1	D9	BS0	D25	D9	D1	A25	D25	0	0		D25	D9	
EXT_AD_24	U03	D0	D8	A24	D24	D8	D0	A24	D24	0	0		D24	D8	
EXT_AD_23	W03	A23	D7	A23	D23	D7	0	A23	D23	0	0		D23	D7	
EXT_AD_22	V03	A22	D6	A22	D22	D6	0	A22	D22	0	0		D22	D6	
EXT_AD_21	Y03	A21	D5	A21	D21	D5	0	A21	D21	0	0		D21	D5	
EXT_AD_20	V02	A20	D4	A20	D20	D4	0	A20	D20	0	0		D20	D4	
EXT_AD_19	Y04	A19	D3	A19	D19	D3	0	A19	D19	0	0		D19	D3	
EXT_AD_18	V04	A18	D2	A18	D18	D2	0	A18	D18	0	0	SA_2	D18	D2	
EXT_AD_17	Y05	A17	D1	A17	D17	D1	0	A17	D17	0	0	SA_1	D17	D1	
EXT_AD_16	W04	A16	D0	A16	D16	D0	0	A16	D16	0	0	SA_0	D16	D0	
EXT_AD_15	U08	A15	A15	A15	D15	0	0	A15	D15	D15	0	D15	D15	A15	
EXT_AD_14	W09	A14	A14	A14	D14	0	0	A14	D14	D14	0	D14	D14	A14	
EXT_AD_13	V08	A13	A13	A13	D13	0	0	A13	D13	D13	0	D13	D13	A13	
EXT_AD_12	Y09	A12	A12	A12	D12	0	0	A12	D12	D12	0	D12	D12	A12	
EXT_AD_11	V09	A11	A11	A11	D11	0	0	A11	D11	D11	0	D11	D11	A11	
EXT_AD_10	Y10	A10	A10	A10	D10	0	0	A10	D10	D10	0	D10	D10	A10	
EXT_AD_9	V10	A9	A9	A9	D9	0	0	A9	D9	D9	0	D9	D9	A9	
EXT_AD_8	W11	A8	A8	A8	D8	0	0	A8	D8	D8	0	D8	D8	A8	
EXT_AD_7	Y11	A7	A7	A7	D7	0	0	A7	D7	D7	D7	D7	D7	A7	
EXT_AD_6	U11	A6	A6	A6	D6	0	0	A6	D6	D6	D6	D6	D6	A6	
EXT_AD_5	W12	A5	A5	A5	D5	0	0	A5	D5	D5	D5	D5	D5	A5	
EXT_AD_4	V11	A4	A4	A4	D4	0	0	A4	D4	D4	D4	D4	D4	A4	
EXT_AD_3	Y12	A3	A3	A3	D3	0	0	A3	D3	D3	D3	D3	D3	A3	
EXT_AD_2	V12	A2	A2	A2	D2	0	0	A2	D2	D2	D2	D2	D2	A2	
EXT_AD_1	W13	A1	A1	A1	D1	0	0	A1	D1	D1	D1	D1	D1	A1	
EXT_AD_0	V13	A0	A0	A0	D0	0	0	A0	D0	D0	D0	D0	D0	A0	
PCI Dedicated Signals															
PCI_PAR	V07							PCI_PAR					A0	A16	
PCI_CBE_0	W10							PCI_CBE_0					A1	A17	
PCI_CBE_1	Y08							PCI_CBE_1					A2	A18	

Table 2-4. LocalPlus Pin Functions (continued)

Pin name	BALL	LocalPlus Non-mux		LocalPlus MULTIPLEXED BUS				PCI BUS				ATA	MOST	Large Flash	RESET
		Addr /Data 24/8	Addr /Data 16/16	Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase	PCI Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase				
PCI_CBE_2	W06							PCI_CBE_2					A3	A19	
PCI_CBE_3	Y02							PCI_CBE_3					A4	A20	
PCI_TRDY	W05							PCI_TRDY					A5	A21	
PCI_IRDY	Y06							PCI_IRDY					A6	A22	
PCI_STOP	V06							PCI_STOP					A7	A23	
PCI_DEVSEL	W07							PCI_DEVSEL					A8	A24	
PCI_FRAME	V05							PCI_FRAME					A9	A25	
PCI_SERR	W08							PCI_SERR					A10	Note 1	
PCI_PERR	Y07							PCI_PERR					A11	Note 1	
PCI_IDSEL	U02							PCI_IDSEL					A12	Note 1	
PCI_REQ	U01							PCI_REQ					A13	Note 1	
PCI_GNT	R04							PCI_GNT					A14	Note 1	
PCI_CL0CK	T01	CLK OUT	CLK OUT	CLK OUT	CLK OUT	CLK OUT	CLK OUT	Same as PCI_CLOCK					CLK OUT	CLK OUT	
PCI_RESET	R02							PCI_RESET					A15	Note 1	
ATA Dedicated Signals															
ATA_DRQ	V17											ATA_DRQ	A16		
ATA_DACK	Y18											ATA_DACK	A17		RST_CFG0
ATA_IOR	Y17											ATA_IOR	A18		RST_CFG1
ATA_IOW	W17											ATA_IOW	A19		RST_CFG2
ATA_IOCHRDY	W18											ATA_IOCHRDY	A20		
ATA_INTRQ	Y19											ATA_INTRQ	A21		
ATA_ISOLATION	Y16											ATA_ISOLATION	A22		
LocalPlus Dedicated Signals															
LP_RW	W16	LP_RW											LP_RW		RST_CFG3
LP_ALE	V14			LP_ALE									A23		RST_CFG4
LP_ACK	U14	LP_ACK											LP_ACK, Note 2		
LP_TS	Y13			LP_TS									LP_TS		RST_CFG5

Table 2-4. LocalPlus Pin Functions (continued)

Pin name	BALL	LocalPlus Non-mux		LocalPlus MULTIPLEXED BUS				PCI BUS				ATA	MOST	Large Flash	RESET
		Addr /Data 24/8	Addr /Data 16/16	Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase	PCI Address Phase	32-bit Data Phase	16-bit Data Phase	8-bit Data Phase				
LP_OE	D08	LP_OE											LP_OE		
LP_CS0	W14	CS_0 / CS_BOOT											CS_0 / CS_BOOT		
LP_CS1	Y14	CS_1											CS_1		
LP_CS2	V15	CS_2											CS_2		
LP_CS3	W15	CS_3											CS_3		
LP_CS4	Y15	CS_4										ATA_CS_0	CS_4		
LP_CS5	V16	CS_5										ATA_CS_1	CS_5		
PSC 3 Dedicated Signals															
PSC3_4	B06	CS_6											CS_6		
PSC3_5	A06	CS_7											CS_7		
GPIO_WKUP Dedicated Signals															
GPIO_WKUP_7	C12												TSIZ1		
JTAG Access Dedicated Signals															
TEST_SEL_1	C03												TSIZ2		

1. The PCI signals, which are not used as address in Large Flash mode, are drive low during a Large Flash access.
2. For a burst transaction LP_ACK signal indicates the burst

Table 2-5. LocalPlus Bus Address / Data Signals

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_31 Ball V01			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 logic 0 D7 D15 D31	hi - z	 logic 0 LocalPlus Data Bit 7 LocalPlus Data Bit 15 LocalPlus Data Bit 31
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D15 D7	hi - z	logic 0 LocalPlus Data Bit 15
LFLASH	D15	hi - z	Large Flash Data Bit D15
MOST Graphics	D31	hi - z	MOST Graphics Data Bit D31
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A31 logic 0 logic 0 D31	hi - z	 PCI Address Bit A31 logic 0 logic 0 PCI Data Bit 31

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_30 Ball R03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 TSIZE0 D6 D14 D30	hi - z	 LocalPlus TSIZE0 LocalPlus Data Bit 6 LocalPlus Data Bit 14 LocalPlus Data Bit 30
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D14 D6	hi - z	 LocalPlus Data Bit 14 LocalPlus Data Bit 6
LFLASH	D14	hi - z	Large Flash Data Bit D14
MOST Graphics	D30	hi - z	MOST Graphics Data Bit D30
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A30 logic 0 logic 0 D30	hi - z	 PCI Address Bit A30 logic 0 logic 0 PCI Data Bit 30
Pin EXT_AD_29 Ball W01			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 TSIZE1 D5 D13 D29	hi - z	 LocalPlus TSIZE1 LocalPlus Data Bit 5 LocalPlus Data Bit 13 LocalPlus Data Bit 29
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D13 D5	hi - z	 LocalPlus Data Bit 13 LocalPlus Data Bit 5
LFLASH	D13	hi - z	Large Flash Data Bit D13
MOST Graphics	D29	hi - z	MOST Graphics Data Bit D29
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A31 logic 0 logic 0 D29	hi - z	 PCI Address Bit A29 logic 0 logic 0 PCI Data Bit 29
Pin EXT_AD_28 Ball T03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 TSIZE2 D4 D12 D28	hi - z	 TSIZE2 LocalPlus Data Bit 4 LocalPlus Data Bit 12 LocalPlus Data Bit 28
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D12 D4	hi - z	 LocalPlus Data Bit 12 LocalPlus Data Bit 4
LFLASH	D12	hi - z	Large Flash Data Bit D12
MOST Graphics	D28	hi - z	MOST Graphics Data Bit D28
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A28 logic 0 logic 0 D28	hi - z	 PCI Address Bit A28 logic 0 logic 0 PCI Data Bit 28

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_27 Ball Y01			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 logic 0 D3 D11 D27	hi - z	 logic 0 LocalPlus Data Bit 3 LocalPlus Data Bit 11 LocalPlus Data Bit 27
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D11 D3	hi - z	 LocalPlus Data Bit 11 LocalPlus Data Bit 3
LFLASH	D11	hi - z	Large Flash Data Bit D11
MOST Graphics	D27	hi - z	MOST Graphics Data Bit D27
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A27 logic 0 logic 0 D27	hi - z	 PCI Address Bit A27 logic 0 logic 0 PCI Data Bit 27
Pin EXT_AD_26 Ball T02			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 BS1 D2 D10 D26	hi - z	 LocalPlus BS1 LocalPlus Data Bit 2 LocalPlus Data Bit 10 LocalPlus Data Bit 26
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D10 D2	hi - z	 LocalPlus Data Bit 10 LocalPlus Data Bit 2
LFLASH	D10	hi - z	Large Flash Data Bit D10
MOST Graphics	D26	hi - z	MOST Graphics Data Bit D26
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A26 logic 0 logic 0 D26	hi - z	 PCI Address Bit A26 logic 0 logic 0 PCI Data Bit 26
Pin EXT_AD_25 Ball W02			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 BS0 D1 D9 D25	hi - z	 BS0 LocalPlus Data Bit 1 LocalPlus Data Bit 9 LocalPlus Data Bit 25
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D9 D1	hi - z	 LocalPlus Data Bit 9 LocalPlus Data Bit 1
LFLASH	D9	hi - z	Large Flash Data Bit D9
MOST Graphics	D25	hi - z	MOST Graphics Data Bit D25
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A25 logic 0 logic 0 D25	hi - z	 PCI Address Bit A25 logic 0 logic 0 PCI Data Bit 25

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_24 Ball U03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A24 D0 D8 D24	hi - z	 LocalPlus Address Bit 24 LocalPlus Data Bit 0 LocalPlus Data Bit 8 LocalPlus Data Bit 24
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D8 D0	hi - z	 LocalPlus Data Bit 8 LocalPlus Data Bit 0
LFLASH	D8	hi - z	Large Flash Data Bit D8
MOST Graphics	D24	hi - z	MOST Graphics Data Bit D24
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A24 logic 0 logic 0 D24	hi - z	 PCI Address Bit A24 logic 0 logic 0 PCI Data Bit 24
Pin EXT_AD_23 Ball W03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A23 logic 0 D7 D23	hi - z	 Local Address Bit A23 logic 0 LocalPlus Data Bit 7 LocalPlus Data Bit 23
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D7 A23	hi - z	 LocalPlus Data Bit 7 LocalPlus Address Bit A23
LFLASH	D7	hi - z	Large Flash Data Bit D7
MOST Graphics	D23	hi - z	MOST Graphics Data Bit D23
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A23 logic 0 logic 0 D23	hi - z	 PCI Address Bit A23 logic 0 logic 0 PCI Data Bit D23
Pin EXT_AD_22 Ball V03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A22 logic 0 D6 D22	hi - z	 LocalPlus Address Bit A22 logic 0 LocalPlus Data Bit 6 LocalPlus Data Bit D22
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D6 A22	hi - z	 LocalPlus Data Bit D6 LocalPlus Address Bit A22
LFLASH	D6	hi - z	Large Flash Data Bit D6
MOST Graphics	D22	hi - z	MOST Graphics Data Bit D22
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A22 logic 0 logic 0 D22	hi - z	 PCI Address Bit A22 logic 0 logic 0 PCI Data Bit D22

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_21 Ball Y03			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A21 logic 0 D5 D21	hi - z	 LocalPlus Address Bit A21 logic 0 LocalPlus Data Bit 5 LocalPlus Data Bit D21
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D5 A21	hi - z	 LocalPlus Data Bit D5 LocalPlus Address Bit A21
LFLASH	D5	hi - z	Large Flash Data Bit D5
MOST Graphics	D21	hi - z	MOST Graphics Data Bit D21
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A21 logic 0 logic 0 D21	hi - z	 PCI Address Bit A21 logic 0 logic 0 PCI Data Bit D21
Pin EXT_AD_20 Ball V02			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A20 logic 0 D4 D20	hi - z	 LocalPlus Address Bit A20 logic 0 LocalPlus Data Bit 4 LocalPlus Data Bit D20
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D4 A20	hi - z	 LocalPlus Data Bit D4 LocalPlus Address Bit A20
LFLASH	D4	hi - z	Large Flash Data Bit D4
MOST Graphics	D20	hi - z	MOST Graphics Data Bit D20
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A20 logic 0 logic 0 D20	hi - z	 PCI Address Bit A20 logic 0 logic 0 PCI Data Bit D20
Pin EXT_AD_19 Ball Y04			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A19 logic 0 D3 D19	hi - z	 LocalPlus Address Bit A19 logic 0 LocalPlus Data Bit 3 LocalPlus Data Bit D19
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D3 A19	hi - z	 LocalPlus Data Bit D3 LocalPlus Address Bit A19
LFLASH	D3	hi - z	Large Flash Data Bit D3
MOST Graphics	D19	hi - z	MOST Graphics Data Bit D19
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A19 logic 0 logic 0 D19	hi - z	 PCI Address Bit A19 logic 0 logic 0 PCI Data Bit D19

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_18 Ball V04			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A18 logic 0 D2 D18	hi - z	 LocalPlus Address Bit A18 logic 0 LocalPlus Data Bit 2 LocalPlus Data Bit D18
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D2 A18	hi - z	 LocalPlus Data Bit D2 LocalPlus Address Bit A18
LFLASH	D2	hi - z	Large Flash Data Bit D2
MOST Graphics	D18	hi - z	MOST Graphics Data Bit D18
ATA	ATA_SA_2	hi - z	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A18 logic 0 logic 0 D18	hi - z	 PCI Address Bit A18 logic 0 logic 0 PCI Data Bit D18
Pin EXT_AD_17 Ball Y05			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A17 logic 0 D1 D17	hi - z	 LocalPlus Address Bit A17 logic 0 LocalPlus Data Bit 1 LocalPlus Data Bit D17
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D1 A17	hi - z	 LocalPlus Data Bit D1 LocalPlus Address Bit A17
LFLASH	D1	hi - z	Large Flash Data Bit D1
MOST Graphics	D17	hi - z	MOST Graphics Data Bit D17
ATA	-----	-----	-----
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A17 logic 0 logic 0 D17	hi - z	 PCI Address Bit A17 logic 0 logic 0 PCI Data Bit D17
Pin EXT_AD_16 Ball W04			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A16 logic 0 D0 D16	hi - z	 LocalPlus Address Bit A16 logic 0 LocalPlus Data Bit 0 LocalPlus Data Bit D16
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 D0 A16	hi - z	 LocalPlus Data Bit D0 LocalPlus Address Bit A16
LFLASH	D0	hi - z	Large Flash Data Bit D0
MOST Graphics	D16	hi - z	MOST Graphics Data Bit D16
ATA	ATA_SA_0	hi - z	ATA_SA_0
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A16 logic 0 logic 0 D16	hi - z	 PCI Address Bit A16 logic 0 logic 0 PCI Data Bit D16

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_15 Ball U08			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A15 logic 0 logic 0 D15	hi - z	 LocalPlus Address Bit A15 logic 0 logic 0 LocalPlus Data Bit D15
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A15 A15	hi - z	 LocalPlus Address Bit A15 LocalPlus Address Bit A15
LFLASH	A15	hi - z	Large Flash Address Bit A15
MOST Graphics	D15	hi - z	MOST Graphics Data Bit D15
ATA	ATA_DATA_15	hi - z	ATA Data Bit 15
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A15 logic 0 D15 D15	hi - z	 PCI Address Bit A15 logic 0 PCI Data Bit D15 PCI Data Bit D15
Pin EXT_AD_14 Ball W09			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A14 logic 0 logic 0 D14	hi - z	 LocalPlus Address Bit A14 logic 0 logic 0 LocalPlus Data Bit D14
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A14 A14	hi - z	 LocalPlus Address Bit A14 LocalPlus Address Bit A14
LFLASH	A14	hi - z	Large Flash Address Bit A14
MOST Graphics	D14	hi - z	MOST Graphics Data Bit D14
ATA	ATA_DATA_14	hi - z	ATA_DATA_14
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A14 logic 0 D14 D14	hi - z	 PCI Address Bit A14 logic 0 PCI Data Bit D14 PCI Data Bit D14
Pin EXT_AD_13 Ball V08			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A13 logic 0 logic 03 D13	hi - z	 LocalPlus Address Bit A13 logic 0 logic 0 LocalPlus Data Bit D13
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A13 A13	hi - z	 LocalPlus Address Bit A13 LocalPlus Address Bit A13
LFLASH	A13	hi - z	Large Flash Address Bit A13
MOST Graphics	D13	hi - z	MOST Graphics Data Bit D13
ATA	ATA_DATA_13	hi - z	ATA Data Bit D13
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A13 logic 0 D13 D13	hi - z	 PCI Address Bit A13 logic 0 PCI Data Bit D13 PCI Data Bit D13

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_12 Ball Y09			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A12 logic 0 logic 0 D12	hi - z	 LocalPlus Address Bit A12 logic 0 logic 0 LocalPlus Data Bit D12
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A12 A12	hi - z	 LocalPlus Address Bit A12 LocalPlus Address Bit A12
LFLASH	A12	hi - z	Large Flash Address Bit A12
MOST Graphics	D12	hi - z	MOST Graphics Data Bit D12
ATA	ATA_DATA_12	hi - z	ATA_DATA_12
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A12 logic 0 D12 D12	hi - z	 PCI Address Bit A12 logic 0 PCI Data Bit D12 PCI Data Bit D12
Pin EXT_AD_11 Ball V09			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A11 logic 0 logic 0 D11	hi - z	 LocalPlus Address Bit A11 logic 0 logic 0 LocalPlus Data Bit D11
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A11 A11	hi - z	 LocalPlus Address Bit A11 LocalPlus Address Bit A11
LFLASH	A11	hi - z	Large Flash Address Bit A11
MOST Graphics	D11	hi - z	MOST Graphics Data Bit D11
ATA	ATA_DATA_11	hi - z	ATA_DATA_11
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A11 logic 0 D11 D11	hi - z	 PCI Address Bit A11 logic 0 PCI Data Bit D11 PCI Data Bit D11
Pin EXT_AD_10 Ball Y10			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A10 logic 0 logic 0 D10	hi - z	 LocalPlus Address Bit A10 logic 0 logic 0 LocalPlus Data Bit D10
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A10 A10	hi - z	 LocalPlus Address Bit A10 LocalPlus Address Bit A10
LFLASH	A10	hi - z	Large Flash Address Bit A10
MOST Graphics	D10	hi - z	MOST Graphics Data Bit D10
ATA	ATA_DATA_10	hi - z	ATA_DATA_10
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A10 logic 0 D10 D10	hi - z	 PCI Address Bit A10 logic 0 PCI Data Bit D10 PCI Data Bit D10

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_9 Ball V10			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A9 logic 0 logic 0 D9	hi - z	 LocalPlus Address Bit A9 logic 0 logic 0 LocalPlus Data Bit D9
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A9 A9	hi - z	 LocalPlus Address Bit A9 LocalPlus Address Bit A9
LFLASH	A9	hi - z	Large Flash Address Bit A9
MOST Graphics	D9	hi - z	MOST Graphics Data Bit D22
ATA	ATA_DATA_9	hi - z	ATA_DATA_9
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A9 logic 0 D9 D9	hi - z	 PCI Address Bit A9 logic 0 PCI Data Bit D9 PCI Data Bit D9
Pin EXT_AD_8 Ball W11			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A8 logic 0 logic 0 D8	hi - z	 LocalPlus Address Bit A8 logic 0 logic 0 LocalPlus Data Bit D8
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A8 A8	hi - z	 LocalPlus Address Bit A8 LocalPlus Address Bit A8
LFLASH	A8	hi - z	Large Flash Address Bit A8
MOST Graphics	D8	hi - z	MOST Graphics Data Bit D8
ATA	ATA_DATA_8	hi - z	ATA_DATA_8
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A8 logic 0 D8 D8	hi - z	 PCI Address Bit A8 logic 0 PCI Data Bit D8 PCI Data Bit D8
Pin EXT_AD_7 Ball Y11			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A7 logic 0 logic 0 D7	hi - z	 LocalPlus Address Bit A7 logic 0 logic 0 LocalPlus Data Bit D7
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A7 A7	hi - z	 LocalPlus Address Bit A7 LocalPlus Address Bit A7
LFLASH	A7	hi - z	Large Flash Address Bit A7
MOST Graphics	D7	hi - z	MOST Graphics Data Bit D7
ATA	ATA_DATA_7	hi - z	ATA_DATA_7
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A7 D7 D7 D7	hi - z	 PCI Address Bit A7 PCI Data Bit D7 PCI Data Bit D7 PCI Data Bit D7

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_6 Ball U11			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A6 logic 0 logic 0 D6	hi - z	 LocalPlus Address Bit A6 logic 0 logic 0 LocalPlus Data Bit D6
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A6 A6	hi - z	 LocalPlus Address Bit A6 LocalPlus Address Bit A6
LFLASH	A6	hi - z	Large Flash Address Bit A6
MOST Graphics	D6	hi - z	MOST Graphics Data Bit D6
ATA	ATA_DATA_6	hi - z	ATA_DATA_6
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A6 D6 D6 D6	hi - z	 PCI Address Bit A6 PCI Data Bit D6 PCI Data Bit D6 PCI Data Bit D6
Pin EXT_AD_5 Ball W12			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A5 logic 0 logic 0 D5	hi - z	 LocalPlus Address Bit A5 logic 0 logic 0 LocalPlus Data Bit D5
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A5 A5	hi - z	 LocalPlus Address Bit A5 LocalPlus Address Bit A5
LFLASH	A5	hi - z	Large Flash Address Bit A5
MOST Graphics	D5	hi - z	MOST Graphics Data Bit D5
ATA	ATA_DATA_5	hi - z	ATA_DATA_5
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A5 D5 D5 D5	hi - z	 PCI Address Bit A5 PCI Data Bit D5 PCI Data Bit D5 PCI Data Bit D5
Pin EXT_AD_4 Ball V11			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A4 logic 0 logic 0 D4	hi - z	 LocalPlus Address Bit A4 logic 0 logic 0 LocalPlus Data Bit D4
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A4 A4	hi - z	 LocalPlus Address Bit A4 LocalPlus Address Bit A4
LFLASH	A4	hi - z	Large Flash Address Bit A4
MOST Graphics	D4	hi - z	MOST Graphics Data Bit D4
ATA	ATA_DATA_4	hi - z	ATA_DATA_4
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A4 D4 D4 D4	hi - z	 PCI Address Bit A4 PCI Data Bit D4 PCI Data Bit D4 PCI Data Bit D4

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_3 Ball Y12			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A3 logic 0 logic 0 D3	hi - z	 LocalPlus Address Bit A3 logic 0 logic 0 LocalPlus Data Bit D3
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A3 A3	hi - z	 LocalPlus Address Bit A3 LocalPlus Address Bit A3
LFLASH	A3	hi - z	Large Flash Address Bit A3
MOST Graphics	D3	hi - z	MOST Graphics Data Bit D3
ATA	ATA_DATA_3	hi - z	ATA_DATA_3
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A3 D3 D3 D3	hi - z	 PCI Address Bit A3 PCI Data Bit D3 PCI Data Bit D3 PCI Data Bit D3
Pin EXT_AD_2 Ball V12			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A2 logic 0 logic 0 D2	hi - z	 LocalPlus Address Bit A2 logic 0 logic 0 LocalPlus Data Bit D2
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A2 A2	hi - z	 LocalPlus Address Bit A2 LocalPlus Address Bit A2
LFLASH	A2	hi - z	Large Flash Address Bit A2
MOST Graphics	D2	hi - z	MOST Graphics Data Bit D2
ATA	ATA_DATA_2	hi - z	ATA_DATA_2
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A2 D2 D2 D2	hi - z	 PCI Address Bit A2 PCI Data Bit D2 PCI Data Bit D2 PCI Data Bit D2
Pin EXT_AD_1 Ball W13			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A1 logic 0 logic 0 D1	hi - z	 LocalPlus Address Bit A1 logic 0 logic 0 LocalPlus Data Bit D1
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A1 A1	hi - z	 LocalPlus Address Bit A1 LocalPlus Address Bit A1
LFLASH	A1	hi - z	Large Flash Address Bit A1
MOST Graphics	D1	hi - z	MOST Graphics Data Bit D1
ATA	ATA_DATA_1	hi - z	ATA_DATA_1
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A1 D1 D1 D1	hi - z	 PCI Address Bit A1 PCI Data Bit D1 PCI Data Bit D1 PCI Data Bit D1

Table 2-5. LocalPlus Bus Address / Data Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin EXT_AD_0 Ball V13			
LocalPlus Bus multiplexed mode Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A0 logic 0 logic 0 D0	hi - z	 LocalPlus Address Bit A0 logic 0 logic 0 LocalPlus Data Bit D0
LocalPlus non-mux 16-bit addr/16-bit data 24-bit addr/8-bit data	 A0 A0	hi - z	 LocalPlus Address Bit A0 LocalPlus Address Bit A0
LFLASH	A0	hi - z	Large Flash Address Bit A0
MOST Graphics	D0	hi - z	MOST Graphics Data Bit D0
ATA	ATA_DATA_0	hi - z	ATA_DATA_0
PCI Address Phase 8-Bit Data Phase 16-Bit Data Phase 32-Bit Data Phase	 A0 logic 0 logic 0 D0	hi - z	 PCI Address Bit A0 PCI Data Bit 0 PCI Data Bit 0 PCI Data Bit D0

Table 2-6. PCI Dedicated Signals

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PCI_PAR Ball V07			
PCI	PCI_PAR	logic 1	PCI Bus Parity
LFLASH	A16	logic 1	Large Flash Address Bit A16
MOST Graphics	A0	logic 1	MOST Graphics Address Bit A0
Pin $\overline{\text{PCI\_CBE\_0}}$ Ball W10			
PCI	PCI_CBE_0	logic 1	PCI Command Byte Enable 0
LFLASH	A17	logic 1	Large Flash Address Bit A17
MOST Graphics	A1	logic 1	MOST Graphics Address Bit A1
Pin $\overline{\text{PCI\_CBE\_1}}$ Ball Y08			
PCI	PCI_CBE_1	logic 1	PCI Command Byte Enable 1
LFLASH	A18	logic 1	Large Flash Address Bit A17
MOST Graphics	A2	logic 1	MOST Graphics Address Bit A1
Pin $\overline{\text{PCI\_CBE\_2}}$ Ball W06			
PCI	PCI_CBE_2	logic 1	PCI Command Byte Enable 2
LFLASH	A19	logic 1	Large Flash Address Bit A19
MOST Graphics	A3	logic 1	MOST Graphics Address Bit A3
Pin $\overline{\text{PCI\_CBE\_3}}$ Ball Y02			
PCI	PCI_CBE_3	logic 1	PCI Command Byte Enable 3
LFLASH	A20	logic 1	Large Flash Address Bit A20
MOST Graphics	A4	logic 1	MOST Graphics Address Bit A4

Table 2-6. PCI Dedicated Signals (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PCI_TRDY Ball W05			
PCI	PCI_TRDY	logic 1	PCI_TRDY PCI Target Ready
LFLASH	A21	logic 1	Large Flash Address Bit A21
MOST Graphics	A5	logic 1	MOST Graphics Address Bit A5
Pin PCI_IRDY Ball Y06			
PCI	PCI_IRDY	logic 1	PCI Initiator (HOST) Ready
LFLASH	A22	logic 1	Large Flash Address Bit A22
MOST Graphics	A6	logic 1	MOST Graphics Address Bit A6
Pin PCI_STOP Ball V06			
PCI	PCI_STOP	logic 1	PCI Transition Stop
LFLASH	A23	logic 1	Large Flash Address Bit A23
MOST Graphics	A7	logic 1	MOST Graphics Address Bit A7
Pin PCI_DEVSEL Ball W07			
PCI	PCI_DEVSEL	logic 1	PCI Device Select
LFLASH	A24	logic 1	Large Flash Address Bit A24
MOST Graphics	A8	logic 1	MOST Graphics Address Bit A8
Pin PCI_FRAME Ball V05			
PCI	PCI_FRAME	logic 1	PCI Frame Start
LFLASH	A25	logic 1	Large Flash Address Bit A25
MOST Graphics	A9	logic 1	MOST Graphics Address Bit A9
Pin PCI_SERR Ball W08			
PCI	PCI_SERR	logic 1	PCI System Error (open drain)
MOST Graphics	A10	logic 1	MOST Graphics Address Bit A10
Pin PCI_PERR Ball Y07			
PCI	PCI_SERR	logic 1	PCI Parity Error
MOST Graphics	A11	logic 1	MOST Graphics Address Bit A11
Pin PCI_IDSEL Ball U02			
PCI	PCI_IDSEL	logic 1	PCI Initial Device Select
MOST Graphics	A12	logic 1	MOST Graphics Address Bit A12
Pin PCI_REQ Ball U01			
PCI	PCI_REQ	logic 1	PCI Bus Request
MOST Graphics	A13	logic 1	MOST Graphics Address Bit A13
Pin PCI_GNT Ball R04			
PCI	PCI_GNT	logic 1	PCI Bus Grant
MOST Graphics	A14	logic 1	MOST Graphics Address Bit A14
Pin PCI_CLOCK Ball T01			
PCI	PCI_CLOCK	clk	PCI Clock

Table 2-6. PCI Dedicated Signals (continued)

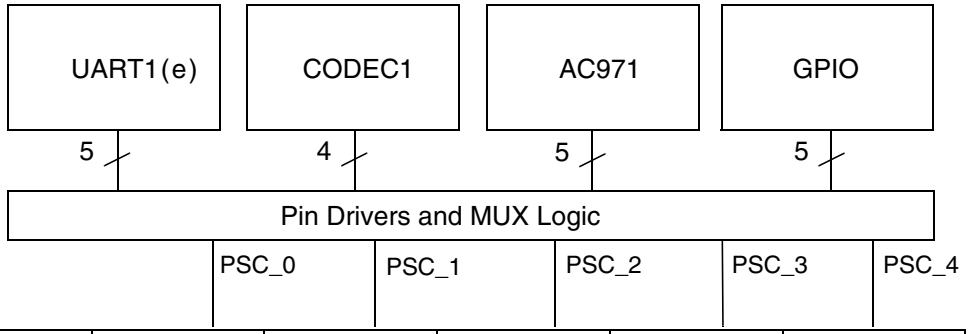
PIN / BALL NUMBER	Function	Reset Value	Description
Pin PCI_RESET Ball R02			
PCI	PCI_RESET	logic 0	PCI Reset Output (open drain)
MOST Graphics	A15	logic 0	MOST Graphics Address Bit A15

Table 2-7. ATA Dedicated Signals

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ATA_DRQ Ball V17			
ATA	ATA_DRQ	logic 0	ATA DMA Request
MOST Graphics	A16	logic 0	MOST Graphics Address Bit A16
Pin ATA_DACK Ball Y18			
ATA	ATA_DACK	logic 1	ATA DMA Request
MOST Graphics	A17	logic 1	MOST Graphics Address Bit A17
RESET Config.			bit 0 -- ppc_pll_cfg_4
Pin ATA_IOR Ball Y17			
ATA	ATA_IOR	logic 1	ATA read - 0, no read - 1
MOST Graphics	A18	logic 1	MOST Graphics Address Bit A18
RESET Config.	RST_CFG1		bit 1 -- ppc_pll_cfg_3
Pin ATA_IOW Ball W17			
ATA	ATA_IOW	logic 1	ATA write - 0, no write - 1
MOST Graphics	A19	logic 1	MOST Graphics Address Bit A19
RESET Config.	RST_CFG2		bit 2 -- ppc_pll_cfg_2
Pin ATA_IOCHDRY Ball W18			
ATA	ATA_IOCHDRY	logic 1	ATA negated to extend transfer
MOST Graphics	A20	logic 1	MOST Graphics Address Bit A20
Pin ATA_INTRQ Ball Y19			
ATA	ATA_INTRQ	logic 1	ATA Interrupt Request
MOST Graphics	A21	logic 1	MOST Graphics Address Bit A21
Pin ATA_ISOLATION Ball Y16			
ATA	ATA_ISOLATION	logic 1	ATA Levelshifter control signal
MOST Graphics	A22	logic 1	MOST Graphics Address Bit A22

Table 2-8. LocalPlus Dedicated Signals

PIN / BALL NUMBER	Function	Reset Value	Description
Pin $\overline{\text{LP\_RW}}$ Ball W16			
LocalPlus	Read/Write	logic 1	LocalPlus Read/Write LIne
Reset Configuration	RST_CFG3	logic 1	Bit 3 -- ppc_pll_cfg_1
Pin $\overline{\text{LP\_ALE}}$ Ball V14			
LocalPlus	Address Latch Enable	logic 1	LocalPlus Address Latch Enable for Multiplexed Transitions
MOST Graphics	A23	logic 1	MOST Graphics Address Bit A23
Reset Configuration	RST_CFG4	logic 1	Bit 4 ppc_pll_cfg_0
Pin LP_ACK Ball U14			
LocalPlus	LP Acknowledge	logic 1	Acknowledge signal for LP peripherals. Acknowledge signal for Large Flash or MOST Graphics, if bursts are not enabled.
LFLASH	BRST	logic 1	BURST indication for Large Flash, if bursts are enabled
MOST Graphics	BRST	logic 1	BURST indication for MOST Graphics, if bursts are enabled
Pin $\overline{\text{LP\_TS}}$ Ball Y13			
LocalPlus	LP Transfer Start	logic 1	LocalPlus Transfer Start
Reset Configuration 5	RST_CFG5	logic 1	Bit 5 -- xlb_clk_sel bit = 0: XLB_CLK = f_{system} / 4 bit = 1: XLB_CLK = f_{system} / 8
Pin $\overline{\text{LP\_OE}}$ Ball D08			
LocalPlus	LP Output Enable	logic 1	LocalPlus Output Enable



Function	Port_conf [29:31]	PSC_0	PSC_1	PSC_2	PSC_3	PSC_4
GPIO	00X	GPIO	GPIO	GPIO	GPIO	GPIO_W/WAKE_UP
AC97_1	01X	AC97_1_SDATA_OUT	AC97_1_SDATA_IN	AC97_1_SYNC	AC97_1_BITCLK	AC97_1_RES
UART1	100	UART1_TXD	UART1_RXD	UART1_RTS	UART2_CTS	GPIO_W/WAKE_UP
UART1e	101	UART1e_TXD	UART1e_RXD	UART1e_RTS	UART1e_CTS	UART1e_DCD
CODEC1	110	CODEC1_TXD	CODEC1_RXD	GPIO	CODEC1_CLK	CODEC1_FRAME
CODEC1 w/ MCLK	111	CODEC1_w/ MCLK_TXD	CODEC1_w/ MCLK_RXD	CODEC1_w/ MCLK_MCLK	CODEC1_w/ MCLK_CLK	CODEC1_w/ MCLK_FRAME

Note:

1. CODEC usage leaves pin 3 open for simple GPIO.
2. If port otherwise unused, all five pins are available as GPIO.
3. CODEC plus additional GPIO from elsewhere can implement Soft Modem or RS-232 functionality.
4. AC'97 usage is limited to PSC1 and PSC2.

Figure 2-4. PSC1 Port Map—5 Pins

Table 2-9. PSC1 Pin Functions

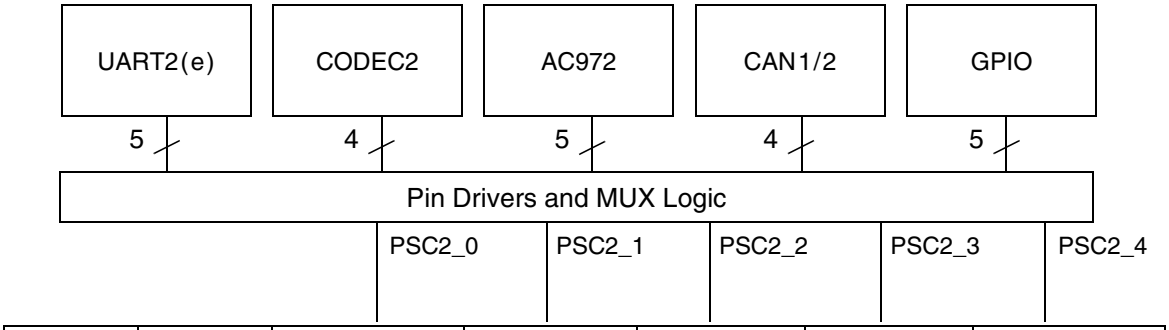
Pin Name	Dir.	GPIO	AC97_1	UART1	UART1e	CODEC1	CODEC1 w/ MCLK
PSC1_0	I/O	GPIO	AC97_1_SDATA_OUT	UART1_TXD	UART1e_TXD	CODEC1_TXD	CODEC1_w/ MCLK_TXD
PSC1_1	I/O	GPIO	AC97_1_SDATA_IN	UART1_RXD	UART1e_RXD	CODEC1_RXD	CODEC1_w/ MCLK_RXD
PSC1_2	I/O	GPIO	AC97_1_SYNC	UART1_RTS	UART1e_RTS	GPIO	CODEC1_w/ MCLK_MCLK
PSC1_3	I/O	GPIO	AC97_1_BITCLK	UART1_CTS	UART1e_CTS	CODEC1_CLK	CODEC1_w/ MCLK_CLK
PSC1_4	I/O	GPIO_W/WAKE_UP	AC97_1_RES	GPIO_W/WAKE_UP	UART1e_DCD	CODEC1_FRAME	CODEC1_w/ MCLK_FRAME

Table 2-10. PSC1 Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC1_0 Ball B11			
GPIO		hi - z	GPIO Simple General Purpose I/O
AC97_1		hi - z	AC97_1_SDATA_OUT AC97 Serial Data Out
UART1		hi - z	UART1_TXD Transmit Data
UART1e		hi - z	UART1e_TXD Transmit Data
CODEC1		hi - z	CODEC1_TXD Transmit Data
CODEC1_w/MCLK		hi - z	CODEC1_w/MCLK_TXD Transmit Data
Pin PSC1_1 Ball A11			
GPIO		hi - z	GPIO Simple General Purpose I/O
AC97_1		hi - z	AC97_1_SDATA_IN AC97 Serial Data In
UART1		hi - z	UART1_RXD Receive Data
UART1e		hi - z	UART1e_RXD Receive Data
CODEC1		hi - z	CODEC1_RXD Receive Data
CODEC1_w/MCLK		hi - z	CODEC1_w/MCLK_RXD Receive Data
Pin PSC1_2 Ball C10			
GPIO		hi - z	GPIO Simple General Purpose I/O
AC97_1		hi - z	AC97_1_SYNC AC97 Frame Sync
UART1		hi - z	UART1_RTS Ready To Send
UART1e		hi - z	UART1e_RTS Ready To Send
CODEC1		hi - z	GPIO Simple General Purpose I/O
CODEC1_w/MCLK		hi - z	CODEC1_w/MCLK _MCLK

Table 2-10. PSC1 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC1_3 Ball B10			
GPIO		hi - z	GPIO Simple General Purpose I/O
AC97_1		hi - z	AC97_1_BITCLK AC97 Bit Clock
UART1		hi - z	UART1_CTS UART Clear To Send
UART1e		hi - z	UART1e_CTS UARTe Clear To Send
CODEC1		hi - z	CODEC1_CLK CODEC Bit Clock
CODEC1_w/MCLK		hi - z	CODEC1_w/MCLK_CLK CODEC Bit Clock
Pin PSC1_4 Ball A10			
GPIO		hi - z	GPIO Simple General Purpose I/O with WAKE UP
AC97_1		hi - z	AC97_1_RES AC97 Reset
UART1		hi - z	GPIO Simple General Purpose I/O with WAKE UP
UART1e		hi - z	UART1e_DCD UARTe Carrier Detect
CODEC1		hi - z	CODEC1_FRAME CODEC Frame Sync
CODEC1_w/MCLK		hi - z	CODEC1_w/MCLK_FRAME CODEC Frame Sync



Function	Port_conf [25:27]	PSC_0	PSC_1	PSC_2	PSC_3	PSC_4
GPIO	000	GPIO	GPIO	GPIO	GPIO	GPIO_W/WAKE_UP
CAN1/2	001	CAN1_TX	CAN1_RX	CAN2_TX	CAN2_RX	GPIO_W/WAKE_UP
AC97_2	01X	AC97_2_SDATA_OUT	AC97_2_SDATA_IN	AC97_2_SYNC	AC97_2_BITCLK	AC97_2_RES
UART2	100	UART2_TXD	UART2_RXD	UART2_RTS	UART2_CTS	GPIO_W/WAKE_UP
UART2e	101	UART2e_TXD	UART2e_RXD	UART2e_RTS	UART2e_CTS	UART2e_DCD
CODEC2	110	CODEC2_TXD	CODEC2_RXD	GPIO	CODEC2_CLK	CODEC2_FRAME
CODEC2 w/ MCLK	111	CODEC2_w/ MCLK_TXD	CODEC2_w/ MCLK_RXD	CODEC2_w/ MCLK_MCLK	CODEC2_w/ MCLK_CLK	CODEC2_w/ MCLK_FRAME

Note:

1. CODEC usage leaves pin 3 open for simple GPIO.
2. CAN usage leaves pin 5 open for WakeUp GPIO.
3. CODEC plus additional GPIO from elsewhere can implement Soft Modem or RS-232 functionality.
4. AC97 usage is limited to PSC1 or PSC2.
5. MSCAN ports 1 and 2 can be configured here or on timer/I^2C ports. They cannot be split. (i.e., put CAN1 on PSC2 and CAN2 on the timer port).
6. CAN RX input supports WakeUp functionality.

Figure 2-5. PSC2 Port Map—5 Pins

Table 2-11. PSC2 Pin Functions

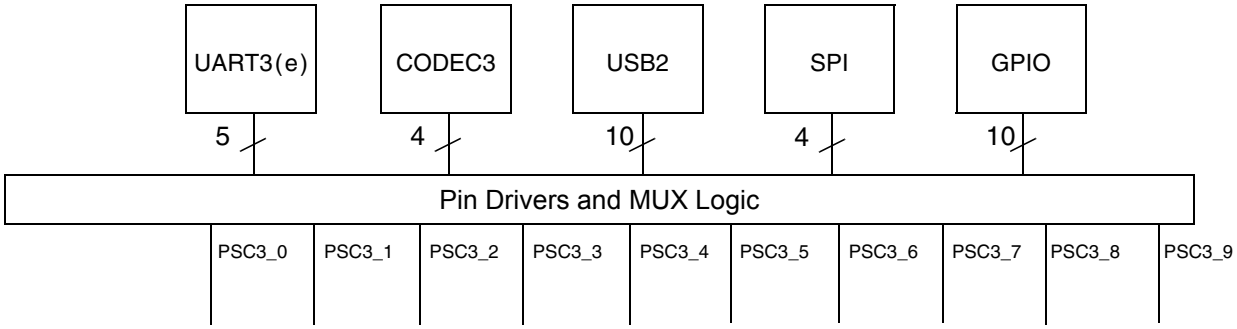
Pin Name	Dir.	GPIO	CAN1/2	AC97_2	UART2	UART2e	CODEC2	CODEC2 w/ MCLK
PSC2_0	I/O	GPIO	CAN1_TX	AC97_2_SDATA_OUT	UART2_TXD	UART2e_TXD	CODEC2_TXD	CODEC2_w/ MCLK_TXD
PSC2_1	I/O	GPIO	CAN1_RX	AC97_2_SDATA_IN	UART2_RXD	UART2e_RXD	CODEC2_RXD	CODEC2_w/ MCLK_RXD
PSC2_2	I/O	GPIO	CAN2_TX	AC97_2_SYNC	UART2_RTS	UART2e_RTS	GPIO	CODEC2_w/ MCLK_MCLK
PSC2_3	I/O	GPIO	CAN2_RX	AC97_2_BITCLK	UART2_CTS	UART2e_CTS	CODEC2_CLK	CODEC2_w/ MCLK_CLK
PSC2_4	I/O	GPIO_w/ WAKE_UP	GPIO_w/ WAKE_UP	AC97_2_RES	GPIO_w/ WAKE_UP	UART2e_DCD	CODEC2_FRAME	CODEC2_w/ MCLK_FRAME

Table 2-12. PSC2 Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC2_0 Ball C09			
GPIO		hi - z	GPIO Simple General Purpose I/O
CAN1, CAN2		hi - z	CAN1_TX CAN Transmit
AC97_2		hi - z	AC97_2_SDATA_OUT AC97 Serial Data Out
UART2		hi - z	UART2_TXD Transmit Data
UART2e		hi - z	UART2e_TXD Transmit Data
CODEC2		hi - z	CODEC2_TXD Transmit Data
CODEC2_w/MCLK		hi - z	CODEC2_w/MCLK_TXD Transmit Data
Pin PSC2_1 Ball B09			
GPIO		hi - z	GPIO Simple General Purpose I/O
CAN_1, CAN_2		hi - z	CAN1_RX CAN Receive
AC97_2		hi - z	AC97_2_SDATA_IN AC97 Serial Data In
UART2		hi - z	UART2_RXD Receive Data
UART2e		hi - z	UART2e_RXD Receive Data
CODEC2		hi - z	CODEC2_RXD Receive Data
Pin PSC2_2 Ball A09			
GPIO		hi - z	GPIO Simple General Purpose I/O
CAN1, CAN2		hi - z	CAN2_TX CAN Transmit
AC97_2		hi - z	AC97_2_SYNC AC97 Frame Sync
UART2		hi - z	UART2_RTS Ready To Send
UART2e		hi - z	UART2e_RTS Ready To Send
CODEC2		hi - z	GPIO Simple General Purpose I/O

Table 2-12. PSC2 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC2_3 Ball B08			
GPIO		hi - z	GPIO Simple General Purpose I/O
CAN1, CAN2		hi - z	CAN2_RX CAN Receive Data
AC97_2		hi - z	AC97_2_BITCLK AC97 Bit Clock
UART2		hi - z	UART2_CTS UART Clear To Send
UART2e		hi - z	UART2e_CTS UARTe Clear To Send
CODEC2		hi - z	CODEC2_CLK CODEC Bit Clock
Pin PSC2_4 Ball A08			
GPIO		hi - z	GPIO Simple General Purpose I/O with WAKE UP
CAN1, CAN2		hi - z	GPIO Simple General Purpose I/O with WAKE UP
AC97_2		hi - z	AC97_2_RES AC97 Reset
UART2		hi - z	GPIO Simple General Purpose I/O with WAKE UP
UART2e		hi - z	UART2e_DCD UARTe Carrier Detect
CODEC2		hi - z	CODEC2_FRAME CODEC Frame



Function	Port_conf [20:23]	PSC3_0	PSC3_1	PSC3_2	PSC3_3	PSC3_4	PSC3_5	PSC3_6	PSC3_7	PSC3_8	PSC3_9
GPIO	0000	GPIO	GPIO	GPIO	GPIO	LP_CS_6 or INTERRUPT	LP_CS_7 or INTERRUPT	GPIO	GPIO	INTERRUPT	GPIO_W/WAKE-UP
USB2	0001	USB2_OE	USB2_TXN	USB2_TXP	USB2_RXD	USB2_RXP	USB2_RXN	USB2_PRTPWR	USB2_SPEED	USB2_SUSPEND	USB2_OVRCNT
UART3	0100	UART3_TXD	UART3_RXD	UART3_RTS	UART3_CTS	LP_CS_6	LP_CS_7	GPIO	GPIO	INTERRUPT	GPIO_W/WAKE_UP
UART3e	0101	UART3e_TXD	UART3e_RXD	UART3e_RTS	UART3e_CTS	UART3e_DCD	LP_CS_7	GPIO	GPIO	INTERRUPT	GPIO_W/WAKE_UP
CODEC3	0110	CODEC3_TXD	CODEC3_RXD	CODEC3_CLK	CODEC3_FRAME	LP_CS_6	LP_CS_7	GPIO	GPIO	INTERRUPT	GPIO_W/WAKE_UP
CODEC3 w/ MCLK	0111	CODEC3_w/ MCLK_TXD	CODEC3_w/ MCLK_RXD	CODEC3_w/MCLK_CLK	CODEC3_w/MCLK_FRAME	LP_CS_6	LP_CS_7	CODEC3_w/MCLK_MCLK	GPIO	INTERRUPT	GPIO_W/WAKE-UP
SPI	100X	GPIO	GPIO	GPIO	GPIO	LP_CS_6	LP_CS_7	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK
UART3 / SPI	1100	UART3_TXD	UART3_RXD	UART3_RTS	UART3_CTS	LP_CS_6	LP_CS_7	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK
UART3e / SPI	1101	UART3e_TXD	UART3e_RXD	UART3e_RTS	UART3e_CTS	UART3e_DCD	LP_CS_7	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK
CODEC3 / SPI	111X	CODEC3_TXD	CODEC3_RXD	CODEC3_CLK	CODEC3_FRAME	LP_CS_6	LP_CS_7	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK

NOTES:
1. If Soft Modem or RS-232 functionality is desired, use UARTe/CODEC function and use available GPIO from this or any other port.
2. Second USB port (USB2) can be configured on PSC3 or on the Ethernet port, but not both locations.
3. PSC3_4 can be configured to be LP_CS6 or an interrupt GPIO, except when PS3 is in USB2 or UART3e modes In these modes, CS6 is not available.
4. PSC3_5 can be configured to be LP_CS7 or an interrupt GPIO, except when PS3 is in USB2 mode. In this mode, LP_CS7 is not available.

Figure 2-6. PSC3 Port Map—10 Pins

Table 2-13. PSC3 Pin Functions

Pin name	Dir.	GPIO	USB2	UART3	UART3e	CODEC3
PSC3_0	I/O (O)	GPIO	USB2_OE	UART3_TXD	UART3e_TXD	CODEC3_TXD
PSC3_1	I/O(I)	GPIO	USB2_TXN	UART3_RXD	UART3e_RXD	CODEC3_RXD
PSC3_2	I/O(I)	GPIO	USB2_TXP	UART3_RTS	UART3e_RTS	CODEC3_CLK
PSC3_3	I/O(I)	GPIO	USB2_RXD	UART3_CTS	UART3e_CTS	CODEC3_FRAME
PSC3_4	I/O(I)	LP_CS_6	USB2_RXP	LP_CS_6	UART3e_DCD	LP_CS_6
PSC3_5	I/O	LP_CS_7	USB2_RXN	LP_CS_7	LP_CS_7	LP_CS_7
PSC3_6	I/O	GPIO	USB2_PRTPWR	GPIO	GPIO	GPIO
PSC3_7	I/O	GPIO	USB2_SPEED	GPIO	GPIO	GPIO
PSC3_8	I/O	INTERRUPT_8	USB2_SUSPEND	INTERRUPT	INTERRUPT	INTERRUPT
PSC3_9	I/O	GPIO_W/WAKE-UP	USB2_OVRCNT	GPIO_W/WAKE_UP	GPIO_W/WAKE_UP	GPIO_W/WAKE_UP

Table 2-14. PSC3 Pin Functions (cont.)

Pin name	Dir.	CODEC3 w/ M	SPI	UART3 / SPI	UART3e / SPI	CODEC3 / SPI
PSC3_0	I/O	CODEC3_w/MCLK_TXD	GPIO	UART3_TXD	UART3e_TXD	CODEC3_TXD
PSC3_1	I/O	CODEC3_w/MCLK_RXD	GPIO	UART3_RXD	UART3e_RXD	CODEC3_RXD
PSC3_2	I/O	CODEC3_w/MCLK_CLK	GPIO	UART3_RTS	UART3e_RTS	CODEC3_CLK
PSC3_3	I/O	CODEC3_w/MCLK_FRAME	GPIO	UART3_CTS	UART3e_CTS	CODEC3_FRAME
PSC3_4	I/O	LP_CS_6	LP_CS_6	LP_CS_6	UART3e_DCD	LP_CS_6
PSC3_5	I/O	LP_CS_7	LP_CS_7	LP_CS_7	LP_CS_7	LP_CS_7
PSC3_6	I/O	CODEC3_w/MCLK_MCLK	SPI_MOSI	SPI_MOSI	SPI_MOSI	SPI_MOSI
PSC3_7	I/O	GPIO	SPI_MISO	SPI_MISO	SPI_MISO	SPI_MISO
PSC3_8	I/O	INTERRUPT	SPI_SS	SPI_SS	SPI_SS	SPI_SS
PSC3_9	I/O	GPIO_W/WAKE-UP	SPI_CLK	SPI_CLK	SPI_CLK	SPI_CLK

Table 2-15. PSC3 Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_0 Ball C07			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_OE USB Output Enable
UART3		hi - z	UART3_TXD Uart Transmit Data
UART3e		hi - z	UART3e_TXD Uart Transmit Data
CODEC3		hi - z	CODEC3_TXD CODEC Transmit Data
CODEC3_w/MCLK		hi - z	CODEC3_w/MCLK_TXD CODEC Transmit Data
SPI		hi - z	GPIO Simple General Purpose I/O
UART3, SPI		hi - z	UART3_TXD Uart Transmit Data
UART3e,SPI		hi - z	UART3e_TXD Uart Transmit Data
CODEC3, SPI		hi - z	CODEC3_TXD CODEC Transmit Data

Table 2-15. PSC3 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_1 Ball B07			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_TXN USB Transmit Negative
UART3		hi - z	UART3_RXD Uart Receive Data
UART3e		hi - z	UART3e_RXD Uart Receive Data
CODEC3		hi - z	CODEC3_RXD CODEC Receive Data
CODEC3_w/MCLK		hi - z	CODEC3_w/MCLK_RXD CODEC Receive Data
SPI		hi - z	GPIO Simple General Purpose I/O
UART3, SPI		hi - z	UART3_RXD Uart Receive Data
UART3e,SPI		hi - z	UART3e_RXD Uart Receive Data
CODEC3, SPI		hi - z	CODEC3_RXD CODEC Receive Data
Pin PSC3_2 Ball A07			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_TXP USB Transmit Positive
UART3		hi - z	UART3_RTS Uart Ready To Send
UART3e		hi - z	UART3e_RTS Uart Ready To Send
CODEC3		hi - z	CODEC3_CLK CODEC Bit Clock
CODEC3_w/MCLK		hi - z	CODEC3_w/MCLK_CLK CODEC Bit Clock
SPI		hi - z	GPIO Simple General Purpose I/O
UART3, SPI		hi - z	UART3_RTS Uart Ready to Send
UART3e, SPI		hi - z	UART3_RTS Uart Ready To Send
CODEC3, SPI		hi - z	CODEC3_CLK CODEC Clock

Table 2-15. PSC3 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_3 Ball C06			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_RXD USB Receive Data
UART3		hi - z	UART3_CTS Uart Clear To Send
UART3e		hi - z	UART3e_CTS Uart Clear To Send
CODEC3		hi - z	CODEC3_FRAME CODEC Frame Sync
CODEC3_w/MCLK		hi - z	CODEC3_w/MCLK_FRAME CODEC Frame Sync
SPI		hi - z	GPIO Simple General Purpose I/O
UART3, SPI		hi - z	UART3_CTS Uart Clear to Send
UART3e, SPI		hi - z	UART3e_CTS Uart Clear To Send
CODEC3, SPI		hi - z	CODEC3_FRAME CODEC Frame Sync
Pin PSC3_4 Ball B06			
GPIO		hi - z	$\overline{\text{LP\_CS\_6}}$
USB2		hi - z	USB2_RXP USB Receive Positive
UART3		hi - z	$\overline{\text{LP\_CS\_6}}$
UART3e		hi - z	UART3e_DCD UART3e Carrier Detect
CODEC3		hi - z	LP_CS_6
CODEC3_w/MCLK		hi - z	LP_CS_6
SPI		hi - z	LP_CS_6
UART3, SPI		hi - z	LP_CS_6
UART3e,SPI		hi - z	UART3e_DCD UART3e Carrier Detect
CODEC3, SPI		hi - z	LP_CS_6

Table 2-15. PSC3 Functions by Pin (continued)

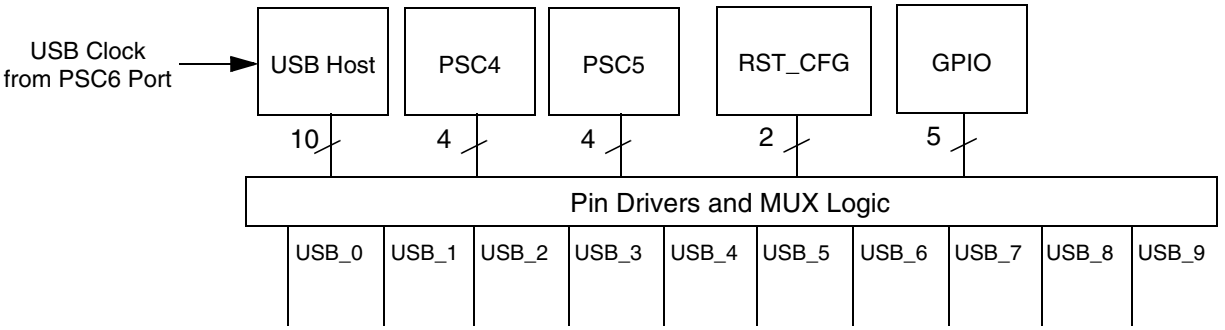
PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_5 Ball A06			
GPIO		hi - z	LP_CS_7
USB2		hi - z	USB2_RXN USB Receive Positive
UART3		hi - z	LP_CS_7
UART3e		hi - z	LP_CS_7
CODEC3		hi - z	LP_CS_7
CODEC3_w/MCLK		hi - z	CODEC3_w/MCLK_MCLK CODEC Clock
SPI		hi - z	LP_CS_7
UART3, SPI		hi - z	LP_CS_7
UART3e,SPI		hi - z	LP_CS_7
CODEC3, SPI		hi - z	LP_CS_7
Pin PSC3_6 Ball C05			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_PRTPWR USB Port Power
UART3		hi - z	GPIO Simple General Purpose I/O
UART3e		hi - z	GPIO_ Simple General Purpose I/O
CODEC3		hi - z	GPIO Simple General Purpose I/O
CODEC3_w/MCLK		hi - z	LP_CS_7
SPI		hi - z	SPI_MOSI SPI_Master Out Slave In
UART3, SPI		hi - z	SPI_MOSI SPI_Master Out Slave In
UART3e, SPI		hi - z	SPI_MOSI SPI_Master Out Slave In
CODEC3, SPI		hi - z	SPI_MOSI SPI_Master Out Slave In

Table 2-15. PSC3 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_7 Ball B05			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB2		hi - z	USB2_SPEED USB Speed
UART3		hi - z	GPIO Simple General Purpose I/O
UART3e		hi - z	GPIO Simple General Purpose I/O
CODEC3		hi - z	GPIO Simple General Purpose I/O
CODEC3_w/MCLK		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	SPI_MISO SPI Master In Slave Out
UART3, SPI		hi - z	SPI_MISO SPI Master In Slave Out
UART3e, SPI		hi - z	SPI_MISO SPI Master In Slave Out
CODEC3, SPI		hi - z	SPI_MISO SPI Master In Slave Out
Pin PSC3_8 Ball A05			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB_2		hi - z	USB2_SUSPEND USB Susupend
UART3		hi - z	INTERRUPT
UART3e		hi - z	INTERRUPT
CODEC3		hi - z	INTERRUPT
CODEC3_w/MCLK		hi - z	INTERRUPT
SPI		hi - z	SPI_SS SPI Slave Select
UART_3, SPI		hi - z	SPI_SS SPI Slave Select
UART3e, SPI		hi - z	SPI_SS SPI Slave Select
CODEC3, SPI		hi - z	SPI_SS SPI Slave Select

Table 2-15. PSC3 Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC3_9 Ball C04			
GPIO		hi - z	GPIO_W/WAKE_UP Simple General Purpose I/O with WAKE UP
USB2		hi - z	USB2_OVRCRNT USB Over Current
UART3		hi - z	GPIO_W/WAKE_UP Simple General Purpose I/O with WAKE UP
UART3e		hi - z	GPIO_W/WAKE_UP Simple General Purpose I/O with WAKE UP
CODEC3		hi - z	GPIO_W/WAKE_UP Simple General Purpose I/O with WAKE UP
CODEC3_w/MCLK		hi - z	GPIO_W/WAKE_UP Simple General Purpose I/O with WAKE UP
SPI		hi - z	SPI_CLK SPI Clock
UART3, SPI		hi - z	SPI_CLK SPI Clock
UART3e, SPI		hi - z	SPI_CLK SPI Clock
CODEC3, SPI		hi - z	SPI_CLK SPI Clock



Function	Port_conf [18:19]	USB_0	USB_1	USB_2	USB_3	USB_4	USB_5	USB_6	USB_7	USB_8	USB_9
RST_CFG	---		RST_CFG6	RST_CFG7							
GPIO	00	GPIO						GPIO	GPIO	GPIO	INTERRUPT
USB	01	USB1_OE	USB1_TXN	USB1_TXP	USB1_RXD	USB1_RXP	USB1_RXN	USB1_PORTPWR	USB1_SPEED	USB1_SUSPEND	USB1_OVERCNT
2x UART4/5	10	GPIO	UART4_RTS	UART4_TXD	UART4_RXD	UART4_CTS	UART5_RXD	UART5_TXD	UART5_RTS	UART5_CTS	INTERRUPT

NOTE:

1. If not used for USB, this port is available as a GPIO resource.
2. USB clock source can be generated internally or sourced fromUSB_CLK input.
3. Pins 3–5 are not mapped to any function other than USB.
4. RST_config bits are sampled only during Reset.
5. PSC4/5 can be used here or on the Ethernet port, but not in both places.

Figure 2-7. USB Port Map—10 Pins

Table 2-16. USB Pin Functions

Pin Name	Dir.	Reset Configuration	GPIO	USB	2x UART4/5
USB_0	I/O		GPIO	USB1_OE	GPIO
USB_1	I/O	RST_CFG6		USB1_TXN	UART4_RTS
USB_2	I/O	RST_CFG7		USB1_TXP	UART4_TXD
USB_3	I			USB1_RXD	UART4_RXD
USB_4	I			USB1_RXP	UART4_CTS
USB_5	I			USB1_RXN	UART5_RXD
USB_6	I/O		GPIO	USB1_PORTPWR	UART5_TXD
USB_7	I/O		GPIO	USB1_SPEED	UART5_RTS
USB_8	I/O		GPIO	USB1_SUSPEND	UART5_CTS
USB_9	I/O		INTERRUPT	USB1_OVERCNT	INTERRUPT

Table 2-17. USB Pin Functions by Pin

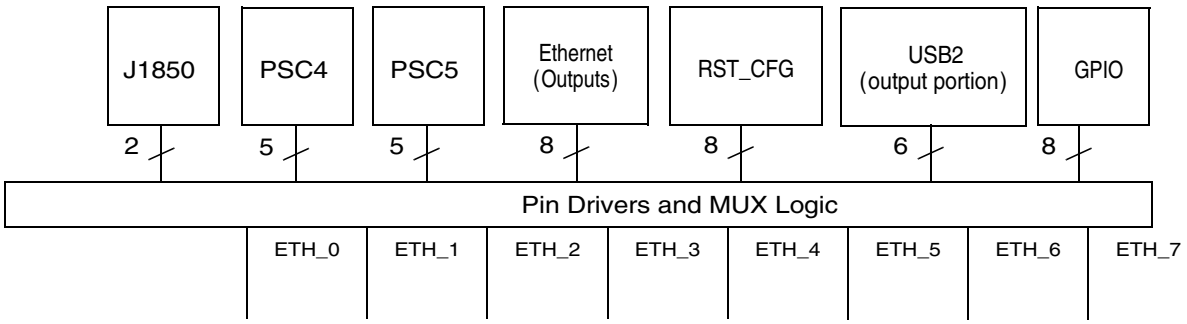
PIN / BALL NUMBER	Function	Reset Value	Description
Pin USB_0 Ball H01			
GPIO		hi - z	GPIO
USB1		hi - z	USB1_OE
RESET Config.		hi - z	----
UART4, UART5		hi - z	GPIO
Pin USB_1 Ball H02			
GPIO		hi - z	----
USB1		hi - z	USB1_TXN USB1 Transmit Negative
RESET Config.		hi - z	RST_CFG6 -- sys_pll_cfg_0 bit =0 : f_{system} = 16x SYS_XTAL_IN bit =1 : f_{system} = 12x SYS_XTAL_IN
UART4, UART5		hi - z	UART4_RTS
Pin USB_2 Ball H03			
GPIO		hi - z	----
USB1		hi - z	USB1_TXP USB1 Transmit Positive
RESET Config.		hi - z	RST_CFG7 (Pull bit low)
UART4, UART5		hi - z	UART4_TXD Uart Transmit Data

Table 2-17. USB Pin Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin USB_3 Ball G01			
GPIO		hi - z	----
USB1		hi - z	USB1_RXD USB1 Receive Data
RESET Config.		hi - z	----
UART4, UART5		hi - z	UART4_RXD Uart Receive Data
Pin USB_4 Ball G02			
GPIO		hi - z	----
USB1		hi - z	USB1_RXP USB1 Receive Positive
RESET Config.		hi - z	----
UART_, UART5		hi - z	UART4_CTS Uart Clear To Send
Pin USB_5 Ball G03			
GPIO		hi - z	----
USB1		hi - z	USB1_RXN USB1 Receive Negative
RESET Config.		hi - z	----
UART4, UART5		hi - z	UART5_RXD Uart Recieve Data
Pin USB_6 Ball G04			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB1		hi - z	USB1_PRTPWR USB Receive Negative
RESET Config.		hi - z	----
UART4, UART5		hi - z	UART5_TXD Uart Transmit Data
Pin USB_7 Ball F01			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB1		hi - z	USB1_SPEED USB Speed
RESET Config.		hi - z	----
UART4, UART5		hi - z	UART5_RTS Uart Ready To Send
Pin USB_8 Ball F02			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB1		hi - z	USB1_SUSPEND USB Suspend
RESET Config.		hi - z	----
UART4, UART5		hi - z	UART5_CTS Uart Clear To Send

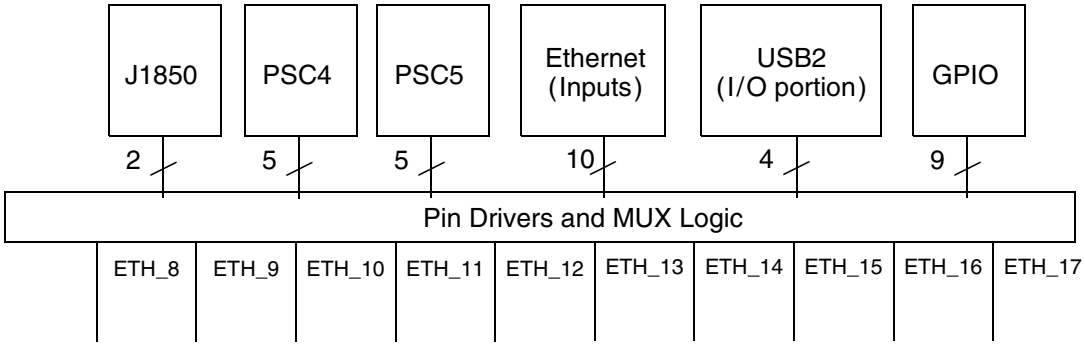
Table 2-17. USB Pin Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin USB_9 Ball F03			
GPIO		hi - z	GPIO Simple General Purpose I/O
USB1		hi - z	USB1_OVRCRNT USB1 Over Current
RESET Config.		hi - z	----
UART4, UART5		hi - z	INTERRUPT



Function	Port_conf [12:15]	ETH_0	ETH_1	ETH_2	ETH_3	ETH_4	ETH_5	ETH_6	ETH_7
RST_CFG	-----	RST_CFG8	RST_CFG15	RST_CFG10	RST_CFG11	RST_CFG12	RST_CFG13	RST_CFG14	-----
GPIO	0000	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT
USB2	0001	OUTPUT	OUTPUT	USB2_TXP	USB2_PRTPWR	USB2_SPEED	USB2_SUSPEND	USB2_OE	USB2_TXN
ETH7	0010	ETH7_TXEN	ETH7_TXD_0	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT
ETH7 / USB2	0011	ETH7_TXEN	ETH7_TXD_0	USB2_TXP	USB2_PRTPWR	USB2_SPEED	USB2_SUSPEND	USB2_OE	USB2_TXN
ETH_18 no MD	0100	ETH18_TXEN	ETH18_TXD_0	ETH18_TXD_1	ETH18_TXD_2	ETH18_TXD_3	ETH18_TXERR	OUTPUT	OUTPUT
ETH_18 w/ MD	0101	ETH18_w/MD_TXEN	ETH18_w/MD_TXD_0	ETH18_w/MD_TXD_1	ETH18_w/MD_TXD_2	ETH18_w/MD_TXD_3	ETH18_w/MD_TXERR	ETH18_w/MD_MDC	ETH18_w/MD_MDIO
ETH7 / UART4e/J1850	1000	ETH7_TXEN	ETH7_TXD_0	OUTPUT	UART4e_TXD	J1850_TX	UART4e_RTS	OUTPUT	OUTPUT
ETH7 /J1850	1001	ETH7_TXEN	ETH7_TXD_0	OUTPUT	OUTPUT	J1850_TX	OUTPUT	OUTPUT	OUTPUT
UART4/5e/J1850	1010	OUTPUT	UART5e_TXD	UART5e_RTS	UART4_TXD	J1850_TX	UART4_RTS	OUTPUT	OUTPUT
UART5e/J1850	1011	OUTPUT	UART5e__TXD	UART5e_RTS	OUTPUT	J1850_TX	OUTPUT	OUTPUT	OUTPUT
J1850	1100	OUTPUT	OUTPUT	OUTPUT	OUTPUT	J1850_TX	OUTPUT	OUTPUT	OUTPUT

Figure 2-8. Ethernet Output Port Map—8 Pins



Function	Port_ conf [12:15]	ETH_8	ETH_9	ETH_10	ETH_11	ETH_12	ETH_13	ETH_14	ETH_15	ETH_16	ETH_17
RST_CFG											
GPIO	0000	OUTPUT	OUTPUT	OUTPUT	OUTPUT		INTERRUPT	INTERRUPT	INTERRUPT	INTERRUPT	GPIO_W/WAKE_UP
USB2	0001	OUTPUT	OUTPUT	OUTPUT	OUTPUT		USB2_RXD	USB2_RXP	USB2_RXN	USB2_OVRCNT	GPIO_W/WAKE_UP
ETH7	0010	ETH7_CD	ETH7_RXCLK	ETH7_COL	ETH7_TXCLK	ETH7_RXD_0	INTERRUPT	INTERRUPT	INTERRUPT	INTERRUPT	GPIO_W/WAKE_UP
ETH7 / USB2	0011	ETH7_CD	ETH7_RXCLK	ETH7_COL	ETH7_TXCLK	ETH7_RXD_0	USB2_RXD	USB2_RXP	USB2_RXN	USB2_OVRCNT	GPIO_W/WAKE_UP
ETH_18 no MD	0100	ETH18_RXDV	ETH18_RXCLK	ETH18_COL	ETH18_TXCLK	ETH18_RXD_0	ETH18_RXD_1	ETH18_RXD_2	ETH18_RXD_3	ETH18_RXERR	ETH18_CRS
ETH_18 w/ MD	0101	ETH18_w/MD_RXDV	ETH18_w/MD_RXCLK	ETH18_w/MD_COL	ETH18_TXD	ETH18_w/MD_RXD_0	ETH18_w/MD_RXD_1	ETH18_w/MD_RXD_2	ETH18_w/MD_RXD_3	ETH18_w/MD_RXERR	ETH18_W/MD_CRS
ETH7 / UART4e/J1850	1000	ETH7_CD	ETH7_RXCLK	ETH7_COL	ETH7_TXCLK	ETH7_RXD_0	J1850_RX	UART4e_RXD	UART4e_CTS	UART4_DCD	GPIO_W/WAKE_UP
ETH7 /J1850	1001	ETH7_CD	ETH7_RXCLK	ETH7_COL	ETH7_TXCLK	ETH7_RXD_0	J1850_RX	INTERRUPT	INTERRUPT	INTERRUPT	GPIO_W/WAKE_UP
UART4/5e/J1850	1010	UART5e_CD	UART5e_CTS	OUTPUT	OUTPUT	UART5e_RXD	J1850_RX	UART4_RXD	UART4_CTS	UART4_CD	GPIO_W/WAKE_UP
UART5e/J1850	1011	UART5e_DCD	UART5e_CTS	OUTPUT	OUTPUT	UART5e_RXD	J1850_RX	INTERRUPT	INTERRUPT	INTERRUPT	GPIO_W/WAKE_UP
J1850	1100	GPIO	OUTPUT	OUTPUT	OUTPUT		J1850_RX	INTERRUPT	INTERRUPT	INTERRUPT	GPIO_W/WAKE_UP

Figure 2-9. Ethernet Input / Control Port Map—10 Pins

Table 2-18. Ethernet Pin Functions

Pin name	Dir.	Reset Configuration	GPIO	USB2	ETH7	ETH7 / USB2
ETH_0	I/O	RST_CFG8	OUTPUT	OUTPUT	ETH7_TXEN	ETH7_TXEN
ETH_1	I/O	RST_CFG15	OUTPUT	OUTPUT	ETH7_TXD_0	ETH7_TXD_0
ETH_2	I/O	RST_CFG10	OUTPUT	USB2_TXP	OUTPUT	USB2_TXP
ETH_3	I/O	RST_CFG11	OUTPUT	USB2_PRTPWR	OUTPUT	USB2_PRTPWR
ETH_4	I/O	RST_CFG12	OUTPUT	USB2_SPEED	OUTPUT	USB2_SPEED
ETH_5	I/O	RST_CFG13	OUTPUT	USB2_SUSPEND	OUTPUT	USB2_SUSPEND
ETH_6	I/O	RST_CFG14	OUTPUT	USB2_OE	OUTPUT	USB2_OE
ETH_7	I/O		OUTPUT	USB2_TXN	OUTPUT	USB2_TXN
ETH_8	I/O		GPIO	GPIO	ETH7__CD	ETH7__CD

Table 2-18. Ethernet Pin Functions (continued)

Pin name	Dir.	Reset Configuration	GPIO	USB2	ETH7	ETH7 / USB2
ETH_9	I/O		GPIO	GPIO	ETH7_RXCLK	ETH7_RXCLK
ETH_10	I/O		GPIO	GPIO	ETH7_COL	ETH7_COL
ETH_11	I/O		GPIO	GPIO	ETH7_TXCLK	ETH7_TXCLK
ETH_12	I				ETH7_RXD_0	ETH7_RXD_0
ETH_13	I/O		INTERRUPT	USB2_RXD	INTERRUPT	USB2_RXD
ETH_14	I/O		INTERRUPT	USB2_RXP	INTERRUPT	USB2_RXP
ETH_15	I/O		INTERRUPT	USB2_RXN	INTERRUPT	USB2_RXN
ETH_16	I/O		INTERRUPT	USB2_OVRCNT	INTERRUPT	USB2_OVRCNT
ETH_17	I/O		GPIO_W/WAKE-UP	GPIO_W/WAKE-UP	GPIO_W/WAKE-UP	GPIO_W/ WAKE-UP

Table 2-19. Ethernet Pin Functions (cont.)

Pin name	Dir.	ETH_18 no MD	ETH_18 w/ MD	ETH7 / UART4e/J1850	ETH7 /J1850	2UART4/5e/J1850	UART5e/J1850	J1850
ETH_0	I/O	ETH18_TXEN	ETH18_w/IMD_ TXEN	ETH7_TXEN	ETH7_TXEN	OUTPUT	OUTPUT	OUTPUT
ETH_1	I/O	ETH18_TXD_0	ETH18_w/ MD_TXD_0	ETH7_TXD_0	ETH7_TXD_0	UART5e__TXD	UART5e__TXD	OUTPUT
ETH_2	I/O	ETH18_TXD_1	ETH18_w/ MD_TXD_1	OUTPUT	OUTPUT	UART5e__RTS	UART5e__RTS	OUTPUT
ETH_3	I/O	ETH18_TXD_2	ETH18_w/ MD_TXD_2	UART4e_TXD	OUTPUT	P4_TXD	OUTPUT	OUTPUT
ETH_4	I/O	ETH18_TXD_3	ETH18_w/ MD_TXD_3	J1850_TX	J1850_TX	J1850_TX	J1850_TX	J1850_TX
ETH_5	I/O	ETH18_TXERR	ETH18_w/ MD_TXERR	UART4e__RTS	OUTPUT	UART4_RTS	OUTPUT	OUTPUT
ETH_6	I/O	OUTPUT	ETH18_w/ MD_MDC	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT
ETH_7	I/O	OUTPUT	ETH18_w/ MD_MDIO	OUTPUT	OUTPUT	OUTPUT	OUTPUT	OUTPUT
ETH_8	I/O	ETH18_RXDV	ETH18_w/ MD_RXDV	ETH7__CD	ETH7__CD	UART5e__DCD	UART5e__DCD	GPIO
ETH_9	I/O	ETH18_RXCLK	ETH18_w/ MD_RXCLK	ETH7_RXCLK	ETH7_RXCLK	UART5e__CTS	UART5e__CTS	GPIO
ETH_10	I/O	ETH18_COL	ETH18_w/ MD_COL	ETH7_COL	ETH7_COL	GPIO	GPIO	GPIO
ETH_11	I/O	ETH18_TXCLK	ETH18_w/ MD_TXCLK	ETH7_TXCLK	ETH7_TXCLK	GPIO	GPIO	GPIO
ETH_12	I	ETH18_RXD_0	ETH18_w/ MD_RXD_0	ETH7_RXD_O	ETH7_RXD_O	UART5e__RXD	UART5e__RXD	-
ETH_13	I/O	ETH18_RXD_1	ETH18_w/ MD_RXD_1	J1850_RX	J1850_RX	J1850_RX	J1850_RX	J1850_RX
ETH_14	I/O	ETH18_RXD_2	ETH18_w/ MD_RXD_2	UART4e__RXD	INTERRUPT	UART4_RXD	INTERRUPT	INTERRUPT
ETH_15	I/O	ETH18_RXD_3	ETH18_w/ MD_RXD_3	UART4e__CTS	INTERRUPT	UART4_CTS	INTERRUPT	INTERRUPT

Table 2-19. Ethernet Pin Functions (cont.)

Pin name	Dir.	ETH_18 no MD	ETH_18 w/ MD	ETH7 / UART4e/J1850	ETH7 /J1850	2UART4/5e/J1850	UART5e/J1850	J1850
ETH_16	I/O	ETH18_RXERR	ETH18_w/ MD_RXERR	UART4e__DCD	INTERRUPT	UART4_CD	INTERRUPT	INTERRUPT
ETH_17	I/O	ETH18_CRS	ETH18_w/ MD_CRS	GPIO_W/WAKE-UP	GPIO_W/WAKE-UP	GPIO_W/WAKE-UP	GPIO_W/WAKE-UP	GPIO_W/WAKE-UP

Table 2-20. Ethernet Output Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_0 Ball K01			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_TX_EN Ethernet Transmit Enable
ETH7 Wire / USB2		hi - z	ETH_TX_EN Ethernet Transmit Enable
ETH18 Wire w/o MD		hi - z	ETH_TX_EN Ethernet Transmit Enable
ETH18 Wire w/ MD		hi - z	ETH_TX_EN Ethernet Transmit Enable
ETH7 Wire, UART4e, J1850		hi - z	ETH_TX_EN Ethernet Transmit Enable
ETH7 Wire, J1850		hi - z	ETH_TX_EN Ethernet Transmit Enable
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config. 8		hi - z	bit 8 -- most_graphics_sel bit = 0: Most Graphics boot not enabled bit = 1: Most Graphics boot enabled.

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_1 Ball K02			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
ETH7 Wire / USB2		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
ETH18 Wire w/o MD		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
ETH18 Wire w/ MD		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
EHT7 Wire, UART4e, J1850		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
ETH7 Wire, J1850		hi - z	ETH_TXD_0 Ethernet Transmit Data Output
UART_4, UART5e, J1850		hi - z	UART5e_TXD Uart Transmit Data
UART5e, J1850		hi - z	UART5e_TXD Uart Transmit Data
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config.		hi - z	bit 15 -- large_flash_sel bit = 0: Large Flash boot not enabled bit = 1: Large Flash boot enabled. Note 3.

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_2 Ball K03			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_TXP USB Transmit Positive
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2_TXP USB Transmit Positive
ETH18 Wire w/o MD		hi - z	ETH_TXD_1 Ethernet Transmit Data Output
ETH18 Wire w/ MD		hi - z	ETH_TXD_1 Ethernet Transmit Data Output
EHT7 Wire, UART4e, J1850		hi - z	GPIO Simple General Purpose Output
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output
UART_4, UART5e, J1850		hi - z	UART5e_RTS Uart Transmit Data
UART5e, J1850		hi - z	UART5e_RTS Uart Transmit Data
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config.		hi - z	bit 10 -- ppc_msrip PPC Boot Address / Exception Table Loc. bit = 0: 0000 0100 (hex) bit = 1: fff0 0100 (hex)

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_3 Ball J01			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_PrtPWR USB Port Power
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2_PrtPWR USB Port Power
ETH18 Wire w/o MD		hi - z	ETH_TXD_2 Ethernet Transmit Data Output
ETH18 Wire w/ MD		hi - z	ETH_TXD_2 Ethernet Transmit Data Output
EHT7 Wire, UART4e, J1850		hi - z	UART_4_TXD Uart Transmit Data
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output
UART_4, UART5e, J1850		hi - z	UART_4_TXD Uart Transmit Data
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config.		hi - z	bit 11 -- boot_rom_wait bit = 0: 4 IPbus clocks of waitstate* bit = 1: 48 IPbus clocks of waitstate*

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_4 Ball J02			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_Speed USB Speed
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2_Speed USB Speed
ETH18 Wire w/o MD		hi - z	ETH_TXD_3 Ethernet Transmit Data Output
ETH18 Wire w/ MD		hi - z	ETH_TXD_3 Ethernet Transmit Data Output
EHT7 Wire, UART4e, J1850		hi - z	J1850_TX J1850 Transmit Data
ETH7 Wire, J1850		hi - z	J1850_TX J1850 Transmit Datat
UART_4, UART5e, J1850		hi - z	J1850_TX J1850 Transmit Data
UART5e, J1850		hi - z	J1850_TX J1850 Transmit Data
J1850		hi - z	J1850_TX J1850 Transmit Data
RESET Config.		hi - z	bit 12 -- boot_rom_swap bit = 0: no byte lane swap - same endian ROM image bit = 1: byte lane swap - different endian ROM image

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_5 Ball L03			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_Suspend USB Suspend
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2_Suspend USB Suspend
ETH18 Wire w/o MD		hi - z	ETH_TXERR Ethernet Transmit Error Output
ETH18 Wire w/ MD		hi - z	ETH_TXERR Ethernet Transmit Error Output
EHT7 Wire, UART4e, J1850		hi - z	UART_4_RTS Uart Ready To Send
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output
UART_4, UART5e, J1850		hi - z	UART_4_RTS Uart Ready To Send
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config.		hi - z	bit 13 -- boot_rom_size For “non-muxed” boot ROMs bit = 0: 8-bit boot ROM data bus / 24-bit boot ROM address bit = 1: 16-bit boot ROM data bus / 16-bit boot ROM address For "muxed" boot ROMs boot ROM addr is max 25 significant bits during address tenure. bit = 0: 16-bit ROM data bus bit = 1: 32-bit ROM data bus For "large flash" boot case boot Flash addr is 25 bits. bit = 0: 8-bit Flash data bus bit = 1: 16-bit Flash data bus

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_6 Ball N02			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_OE USB Output Enable
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2__OE USB Output Enable
ETH18 Wire w/o MD		hi - z	GPIO Simple General Purpose Output
ETH18 Wire w/ MD		hi - z	ETH_MDC Ethernet Transmit Error Output
EHT7 Wire, UART4e, J1850		hi - z	GPIO Simple General Purpose Output
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output
RESET Config.		hi - z	bit 14 -- boot_rom_type bit = 0: non-muxed boot ROM bus, single tenure transfer. bit = 1: muxed boot ROM bus, PPC like with address & data tenures, ALE_b & TS_b active. Note 3.

Table 2-20. Ethernet Output Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_7 Ball N01			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	USB2_TXN USB Transmit Negative
ETH7 Wire		hi - z	GPIO Simple General Purpose Output
ETH7 Wire / USB2		hi - z	USB2_TXN USB Transmit Negative
ETH18 Wire w/o MD		hi - z	GPIO Simple General Purpose Output
ETH18 Wire w/ MD		hi - z	ETH_MDIO Ethernet Management Data I/O
EHT7 Wire, UART4e, J1850		hi - z	GPIO Simple General Purpose Output
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output

Notes:

1. The external bus clock (pci_clk) will be 1/2 the frequency of the internal bus clock (ipb_clk) at powerup. Therefore, 4 IPbus wait states will translate to as little as 1 external wait state (i.e. peripheral must respond within 2 external clocks). The "slow" setting represents 48 IPbus clocks of wait, or 23 external clocks of wait External waits are "minus-1" because Chip Select may assert on falling edge of external bus clock (dependant on internal timing).
2. For muxed boot ROM types, the width of ALE_b & TS_b will be 2 IPbus clocks (i.e. 1 external clock). This represents the "wide ALE" setting in the LocalPlus Controller (LPC). Care must be taken if these clock relationships are to be changed during the boot process. For the 1-to-1 internal-to-external clock setting (which must be programmed by software into the CDM), be sure to change the ALE width setting (in LPC) *after* adjusting the clock relationship. Any fetches to the boot device between these two settings will result in ALE and TS being 2 external clocks wide.
3. Only one boot mode can be enabled at a time. Large Flash and Most Graphics cannot be enabled at the same time. If neither Large Flash or Most Graphics is enabled, boot will occur from the normal LocalPlus mode, either muxed or nonmuxed (depending on the "boot_rom_type" configuration input).

Table 2-21. Ethernet Input / Control Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_8 Ball M03			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_CD Ethernet Carrier Detect
ETH7 Wire / USB2		hi - z	ETH_CD Ethernet Carrier Detect
ETH18 Wire w/o MD		hi - z	ETH_RXDV Ethernet Receive Data Valid
ETH18 Wire w/ MD		hi - z	ETH_RXDV Ethernet Receive Data Valid
EHT7 Wire, UART4e, J1850		hi - z	ETH_CD Ethernet Carrier Detect
ETH7 Wire, J1850		hi - z	ETH_CD Ethernet Carrier Detect
UART_4, UART5e, J1850		hi - z	UART5e_DCD Uart Carrier Detect
UART5e, J1850		hi - z	UART5e_DCD Uart Carrier Detect
J1850		hi - z	GPIO Simple General Purpose Output
Pin ETH_9 Ball L01			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_RXCLK Ethernet Receive Clock
ETH7 Wire / USB2		hi - z	ETH_RXCLK Ethernet Receive Clock
ETH18 Wire w/o MD		hi - z	ETH_RXCLK Ethernet Receive Clock
ETH18 Wire w/ MD		hi - z	ETH_RXCLK Ethernet Receive Clock
EHT7 Wire, UART4e, J1850		hi - z	ETH_RXCLK Ethernet Receive Clock
ETH7 Wire, J1850		hi - z	ETH_RXCLK Ethernet Receive Clock
UART_4, UART5e, J1850		hi - z	ETH_RXCLK Ethernet Receive Clock
UART5e, J1850		hi - z	UART5e_CTS Uart Clear To Send
J1850		hi - z	UART5e_CTS Uart Clear To Send

Table 2-21. Ethernet Input / Control Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_10 Ball J03			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_COL Ethernet Collision Detect Input
ETH7 Wire / USB2		hi - z	ETH_COL Ethernet Collision Detect Input
ETH18 Wire w/o MD		hi - z	ETH_COL Ethernet Collision Detect Input
ETH18 Wire w/ MD		hi - z	ETH_COL Ethernet Collision Detect Input
EHT7 Wire, UART4e, J1850		hi - z	ETH_COL Ethernet Collision Detect Input
ETH7 Wire, J1850		hi - z	ETH_COL Ethernet Collision Detect Input
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output
Pin ETH_11 Ball L04			
GPIO		hi - z	GPIO Simple General Purpose Output
USB2		hi - z	GPIO Simple General Purpose Output
ETH7 Wire		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
ETH7 Wire / USB2		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
ETH18 Wire w/o MD		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
ETH18 Wire w/ MD		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
EHT7 Wire, UART4e, J1850		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
ETH7 Wire, J1850		hi - z	ETH_TXCLK Ethernet Transmit Clock Input
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output
UART5e, J1850		hi - z	GPIO Simple General Purpose Output
J1850		hi - z	GPIO Simple General Purpose Output

Table 2-21. Ethernet Input / Control Functions by Pin (continued)

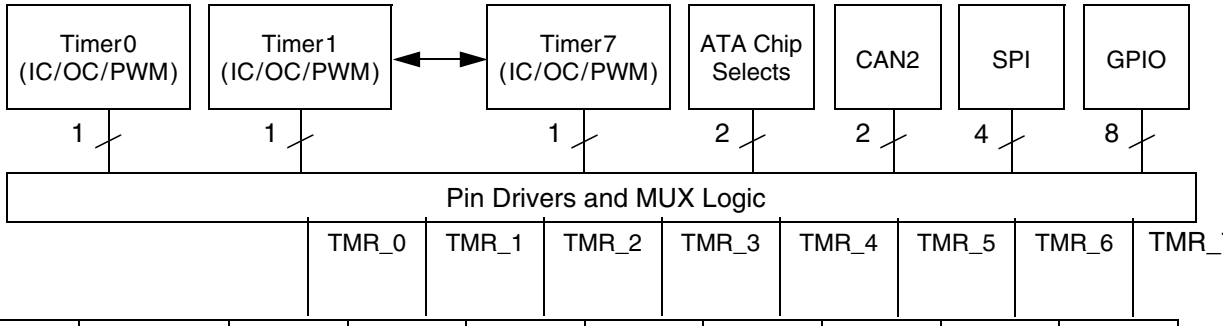
PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_12 Ball M02			
GPIO		hi - z	
USB2		hi - z	
ETH7 Wire		hi - z	ETH_RXD0 Ethernet Receive Data Input
ETH7 Wire / USB2		hi - z	ETH_RXD0 Ethernet Receive Data Input
ETH18 Wire w/o MD		hi - z	ETH_RXD0 Ethernet Receive Data Input
ETH18 Wire w/ MD		hi - z	ETH_RXD0 Ethernet Receive Data Input
EHT7 Wire, UART4e, J1850		hi - z	ETH_RXD0 Ethernet Receive Data Input
ETH7 Wire, J1850		hi - z	ETH_RXD0 Ethernet Receive Data Input
UART_4, UART5e, J1850		hi - z	UART5e_RXD Uart Receive Data
UART5e, J1850		hi - z	UART5e_RXD Uart Receive Data
J1850		hi - z	----
Pin ETH_13 Ball M01			
GPIO		hi - z	INTERRUPT
USB2		hi - z	USB_2_RECEIVE DIFFERENTIAL
ETH7 Wire		hi - z	INTERRUPT
ETH7 Wire / USB2		hi - z	USB_2_RECEIVE DIFFERENTIAL
ETH18 Wire w/o MD		hi - z	ETH_RXD1 Ethernet Receive Data Input
ETH18 Wire w/ MD		hi - z	ETH_RXD1 Ethernet Receive Data Input
EHT7 Wire, UART4e, J1850		hi - z	J1850_RX J1850 Receive Data
ETH7 Wire, J1850		hi - z	J1850_RX J1850 Receive Data
UART_4, UART5e, J1850		hi - z	J1850_RX J1850 Receive Data
UART5e, J1850		hi - z	J1850_RX J1850 Receive Data
J1850		hi - z	J1850_RX J1850 Receive Data

Table 2-21. Ethernet Input / Control Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_14 Ball N04			
GPIO		hi - z	INTERRUPT
USB2		hi - z	USB_2_RXP USB Receive Positive
ETH7 Wire		hi - z	INTERRUPT
ETH7 Wire / USB2		hi - z	USB_2_RXP USB Receive Positive
ETH18 Wire w/o MD		hi - z	ETH_RXD2 Ethernet Receive Data Input
ETH18 Wire w/ MD		hi - z	ETH_RXD2 Ethernet Receive Data Input
EHT7 Wire, UART4e, J1850		hi - z	UART4e_RXD Uart Receive Data
ETH7 Wire, J1850		hi - z	INTERRUPT
UART_4, UART5e, J1850		hi - z	UART4e_RXD Uart Receive Data
UART5e, J1850		hi - z	INTERRUPT
J1850		hi - z	INTERRUPT
Pin ETH_15 Ball N03			
GPIO		hi - z	INTERRUPT
USB2		hi - z	USB_2_RXN USB Receive Negative
ETH7 Wire		hi - z	INTERRUPT
ETH7 Wire / USB2		hi - z	USB_2_RXN USB Receive Negative
ETH18 Wire w/o MD		hi - z	ETH_RXD3 Ethernet Receive Data Input
ETH18 Wire w/ MD		hi - z	ETH_RXD3 Ethernet Receive Data Input
EHT7 Wire, UART4e, J1850		hi - z	UART4e_CTS Uart Clear To Send
ETH7 Wire, J1850		hi - z	INTERRUPT
UART_4, UART5e, J1850		hi - z	UART4e_CTS Uart Clear To Send
UART5e, J1850		hi - z	INTERRUPT
J1850		hi - z	INTERRUPT

Table 2-21. Ethernet Input / Control Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin ETH_16 Ball L02			
GPIO		hi - z	INTERRUPT
USB2		hi - z	USB_2_OVRCNT USB Over Current
ETH7 Wire		hi - z	INTERRUPT
ETH7 Wire / USB2		hi - z	USB_2_OVRCNT USB Over Current
ETH18 Wire w/o MD		hi - z	ETH_RXERR Ethernet Receive Error Input
ETH18 Wire w/ MD		hi - z	ETH_RXERR Ethernet Receive Error Input
EHT7 Wire, UART4e, J1850		hi - z	UART4e_DCD Uart Carrier Detect
ETH7 Wire, J1850		hi - z	INTERRUPT
UART_4, UART5e, J1850		hi - z	INTERRUPT
UART5e, J1850		hi - z	INTERRUPT
J1850		hi - z	INTERRUPT
Pin ETH_17 Ball J04			
GPIO		hi - z	GPIO Simple General Purpose Output with WAKE UP
USB2		hi - z	GPIO Simple General Purpose Output with WAKE UP
ETH7 Wire		hi - z	GPIO Simple General Purpose Output with WAKE UP
ETH7 Wire / USB2		hi - z	GPIO Simple General Purpose Output with WAKE UP
ETH18 Wire w/o MD		hi - z	ETH_CRS Ethernet Carrier Sense Input
ETH18 Wire w/ MD		hi - z	ETH_CRS Ethernet Carrier Sense Input
EHT7 Wire, UART4e, J1850		hi - z	GPIO Simple General Purpose Output with WAKE UP
ETH7 Wire, J1850		hi - z	GPIO Simple General Purpose Output with WAKE UP
UART_4, UART5e, J1850		hi - z	GPIO Simple General Purpose Output with WAKE UP
UART5e, J1850		hi - z	GPIO Simple General Purpose Output with WAKE UP
J1850		hi - z	GPIO Simple General Purpose Output with WAKE UP



Function	Port_config [2:3_6:7]	TIMER0	TIMER1	TIMER2	TIMER3	TIMER4	TIMER5	TIMER6	TIMER7
GPIO TIMER	00_0X 00_10	GPIO TIMER_0	GPIO TIMER_1	GPIO TIMER_2	GPIO TIMER_3	GPIO TIMER_4	GPIO TIMER_5	GPIO TIMER_6	GPIO TIMER_7
ATA_CS	00_11	ATA_CS_0	ATA_CS_1	GPIO TIMER_2	GPIO TIMER_3	GPIO TIMER_4	GPIO TIMER_5	GPIO TIMER_6	GPIO TIMER_7
CAN2	01_00	CAN2_TX	CAN2_RX	GPIO TIMER_2	GPIO TIMER_3	GPIO TIMER_4	GPIO TIMER_5	GPIO TIMER_6	GPIO TIMER_7
SPI	10_00	GPIO TIMER_0	GPIO TIMER_1	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK	GPIO TIMER_6	GPIO TIMER_7
SPI/ATA_CS	10_11	ATA_CS_0	ATA_CS_1	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK	GPIO TIMER_6	GPIO TIMER_7
CAN2/SPI	11_00	CAN2_TX	CAN2_RX	SPI_MOSI	SPI_MISO	SPI_SS	SPI_CLK	GPIO TIMER_6	GPIO TIMER_7

NOTES:
1. Each pin is individually selectable as a Timer or GPIO. Each Timer can be individually configured as Input Capture (IC), Output Compare (OC), or Pulse Width Modulator (PWM) (GPT X Enable and Mode Select Register).
   If a timer pin is configured as a GPIO or some other function (SPI, chip select or CAN), the timer module can still be used internally by software.
2. Timers 6 and 7, when configured as input capture, contain WakeUp functionality.
3. All Timer and GPIO function controls are within the Timer module register set.
4. CAN RX input supports WakeUp functionality.

Figure 2-10. Timer Port Map—8 Pins

Table 2-22. Timer Pin Functions

Pin Name	Dir.	GPIO	TIMER	ATA CHIP SEL	CAN2	SPI	CAN2 / SPI
TIMER 0	I/O	SIMPLE GPIO	TIMER 0	ATA_CS_0	CAN2_TX	SIMPLE GPIO	CAN2_TX
TIMER 1	I/O	SIMPLE GPIO	TIMER 1	ATA_CS_1	CAN2_RX	SIMPLE GPIO	CAN2_RX
TIMER 2	I/O	SIMPLE GPIO	TIMER 2	SIMPLE GPIO	SIMPLE GPIO	SPI_MOSI	SPI_MOSI
TIMER 3	I/O	SIMPLE GPIO	TIMER 3	SIMPLE GPIO	SIMPLE GPIO	SPI_MISO	SPI_MISO
TIMER 4	I/O	SIMPLE GPIO	TIMER 4	SIMPLE GPIO	SIMPLE GPIO	SPI_SS	SPI_SS
TIMER 5	I/O	SIMPLE GPIO	TIMER 5	SIMPLE GPIO	SIMPLE GPIO	SPI_CLK	SPI_CLK
TIMER 6	I/O	SIMPLE GPIO	TIMER 6	SIMPLE GPIO	SIMPLE GPIO	SIMPLE GPIO	SIMPLE GPIO
TIMER 7	I/O	SIMPLE GPIO	TIMER 7	SIMPLE GPIO	SIMPLE GPIO	SIMPLE GPIO	SIMPLE GPIO

Table 2-23. Timer Functions by Pin

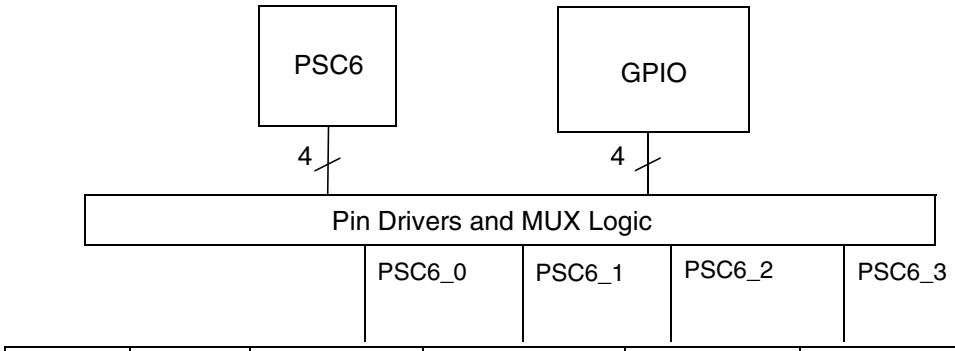
PIN / BALL NUMBER	Function	Reset Value	Description
Pin TIMER_0 Ball Y20			
TIMER		hi - z	TIMER_0
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	ATA_CS0 ATA Chip Select 0
CAN2		hi - z	CAN2_TX CAN 2 Transmit Data
SPI		hi - z	GPIO Simple General Purpose I/O
CAN2 / SPI		hi - z	CAN2_TX CAN 2 Transmit Data
Pin TIMER_1 Ball V18			
TIMER		hi - z	TIMER_1
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	ATA_CS0 ATA Chip Select 1
CAN2		hi - z	CAN2_RX CAN 2 Receive Data
SPI		hi - z	GPIO Simple General Purpose I/O
CAN2 / SPI		hi - z	CAN2_RX CAN 2 Receive Data
Pin TIMER_2 Ball D03			
TIMER		hi - z	TIMER_2
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	SPI _MOSI SPI Master Out Slave In
CAN2 / SPI		hi - z	SPI MOSI SPI Master Out Slave In

Table 2-23. Timer Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin TIMER_3 Ball D02			
TIMER		hi - z	TIMER_3
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	SPI _MISO SPI Master In Slave Out
CAN2 / SPI		hi - z	SPI MISO SPI Master In Slave Out
Pin TIMER_4 Ball D01			
TIMER		hi - z	TIMER_4
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	SPI _SS SPI Slave Select
CAN2 / SPI		hi - z	SPI SS SPI Slave Select
Pin TIMER_5 Ball E03			
TIMER		hi - z	TIMER_5
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	SPI _CLK SPI Clock
CAN2 / SPI		hi - z	SPI CLK SPI Clock

Table 2-23. Timer Functions by Pin (continued)

PIN / BALL NUMBER	Function	Reset Value	Description
Pin TIMER_6 Ball E02			
TIMER		hi - z	TIMER_6
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	GPIO Simple General Purpose I/O
CAN2 / SPI		hi - z	GPIO Simple General Purpose I/O
Pin TIMER_7 Ball E01			
TIMER		hi - z	TIMER_7
GPIO		hi - z	GPIO Simple General Purpose I/O
ATA CHIP SELECTS		hi - z	GPIO Simple General Purpose I/O
CAN2		hi - z	GPIO Simple General Purpose I/O
SPI		hi - z	GPIO Simple General Purpose I/O
CAN2 / SPI		hi - z	GPIO Simple General Purpose I/O



Function	Port_conf [9:11]	PSC6_0	PSC6_1	PSC6_2	PSC6_3
GPIO	000	GPIO_W/WAKE_UP	GPIO_W/WAKE_UP	GPIO	GPIO
UART6/ IrDA	101	UART6_RXD/ IrDA_RX	UART6_CTS	UART6_TXD/ IrDA_TX	UART6_RTS
CODEC6/ IrDA	111	CODEC6_RXD/ IrDA_RX	CODEC6_FRAME	CODEC6_TXD/ IrDA_TX	CODEC6_CLK/ IR_USB_CLK

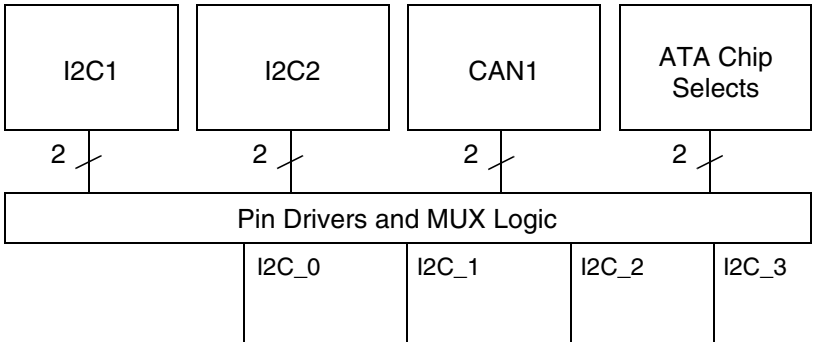
Figure 2-11. PSC6 Port Map—4 Pins

Table 2-24. PSC6 Pin Functions

Pin name	Dir.	GPIO	UART6/ IrDA	CODEC6 / IrDA
PSC6_0	I/O	WAKE_UP	UART6_RXD IrDA_RX	CODEC6_RXD Irda_RX
PSC6_1	I/O	WAKE_UP	UART6_CTS	CODEC6_FRAME
PSC6_2	I/O	SIMPLE GPIO	UART6_TXD IrDA_TX	CODEC6_TXD IrDA_TX
PSC6_3	I/O	SIMPLE GPIO	UART6_RTS	CODEC6_CLK/ IR_USB_CLK

Table 2-25. PSC6 Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin PSC6_0 Ball B12			
GPIO		hi - z	GPIO Simple General Purpose I/O with WAKE UP
UART6 / IrDA		hi - z	UART6_RXD Uart Receive Data IrDA_RX IrDA Receive Data
CODEC6 / IrDA		hi - z	CODEC6_RXD CODEC Receive Data IrDA_RX IrDA Receive Data
Pin PSC6_1 Ball C11			
GPIO		hi - z	GPIO Simple General Purpose I/O with WAKE UP
UART6		hi - z	UART6_CTS Uart Clear To Send
CODEC6		hi - z	CODEC6_FRAME CODEC Frame Sync
Pin PSC6_2 Ball A12			
GPIO		hi - z	GPIO Simple General Purpose I/O
UART6 / IrDA		hi - z	UART6_TXD Uart Transmit Data IrDA_TX Irda Transmit Data
CODEC6 / IrDA		hi - z	CODEC6_TXD CODEC Transmit Data IrDA_TX Irda Transmit Data
Pin PSC6_3 Ball C13			
GPIO		hi - z	GPIO Simple General Purpose I/O
UART6		hi - z	UART6_RTS Uart Clear To Send
CODEC6 / IrDA		hi - z	CODEC6_CLK IR_USB_CLK



Function	Port_conf	I2C_0	I2C_1	I2C_2	I2C_3
I2C1 / I2C2	default	I2C1_CLK	I2C1_IO	I2C2_CLK	I2C2_IO
CAN1 / I2C2	Port_conf[2:3]=01	CAN1_TX	CAN1_RX	I2C2_CLK	I2C2_IO
I2C1 / ATA CHIP SELECTS	Port_conf[6:7]=10	I2C1_CLK	I2C1_IO	ATA_CS_0	ATA_CS_1

NOTE:
1. CAN RX input supports WakeUp functionality.

Figure 2-12. I²C Port Map—4 Pins (two pins each, for two I²Cs)

Table 2-26. I²C Functions by Pin

PIN / BALL NUMBER	Function	Reset Value	Description
Pin I2C_0 Ball V19			
I2C_1 / I2C_2			I2C_1_CLK I2C Clock
CAN_1/I2C_2			CAN1_TX CAN Transmit Data
I2C_1/ATA_CS			I2C_1_CLK I2C Clock
Pin I2C_1 Ball W19			
I2C_1 / I2C_2			I2C_1_I/O
CAN1/CAN2			CAN1_RX
I2C_1/ATA_CS			I2C_1_I/O
Pin I2C_2 Ball V20			
I2C_1 / I2C_2			I2C_2_CLK I2C Clock
CAN1/CAN2			I2C_2_CLK I2C Clock
I2C_1/ATA_CS			ATA_CS0 ATA Chip Select 0
Pin I2C_3 Ball W20			
I2C_1 / I2C_2			I2C_2_I/O I2C I/O line
CAN1/CAN2			I2C_2_I/O I2C I/O line
I2C_1/ATA_CS			ATA_CS1 ATA Chip Select 1

Table 2-27. SDRAM Bus Pin Functions

PIN BALL NUMBER	Function	Reset Value	Description
Pin MEM_RAS Ball A18		logic 0	SDRAM Bus Row Address Strobe
Pin MEM_CAS Ball B19		logic 0	SDRAM Bus Column Address Strobe
Pin $\overline{\text{MEM\_WE}}$ Ball A19		logic 0	SDRAM Bus Write enable
Pin $\overline{\text{MEM\_CS\_0}}$ Ball B18		logic 1	SDRAM Bus Chip Select 0
Pin $\overline{\text{MEM\_CS\_1}}$ Ball C15 (GPIO_WKUP_6)		logic 1	SDRAM Bus Chip Select 1 (shared with GPIO_WKUP_6)
Pin MEM_CLK_EN Ball F20		logic 0	SDRAM Bus Clock Enable
Pin MEM_CLK Ball G19		logic 0	SDRAM Bus Memory Clock
Pin $\overline{\text{MEM\_CLK}}$ Ball G20		logic 1	SDRAM Bus Inverted Memory Clock
Pin MEM_MBA_1 Ball A17		logic 0	SDRAM Bus Memory Bank Address 1
Pin MEM_MBA_0 Ball C18		logic 0	SDRAM Bus Memory Bank Address 0
Pin MEM_MDQS_3 Ball L18		hi - z	SDRAM Bus Bidirectional Data Bus Strobe 3
Pin MEM_MDQS_2 Ball D18		hi - z	SDRAM Bus Bidirectional Data Bus Strobe 2
Pin MEM_MDQS_1 Ball H20		hi - z	SDRAM Bus Bidirectional Data Bus Strobe 1
Pin MEM_MDQS_0 Ball N20		hi - z	SDRAM Bus Bidirectional Data Bus Strobe 0
Pin MEM_DQM_3 Ball L17			SDRAM Bus Data Mask 3
Pin MEM_DQM_2 Ball A20			SDRAM Bus Data Mask 2
Pin MEM_DQM_1 Ball H19			SDRAM Bus Data Mask 1
Pin MEM_DQM_0 Ball N19			SDRAM Bus Data Mask 0
Pin MEM_MA_12 Ball F19		logic 0	SDRAM Bus Memory Address 12

Table 2-27. SDRAM Bus Pin Functions (continued)

PIN BALL NUMBER	Function	Reset Value	Description
Pin MEM_MA_11 Ball E20		logic 0	SDRAM Bus Memory Address 11
Pin MEM_MA_10 Ball B17		logic 0	SDRAM Bus Memory Address 10
Pin MEM_MA_9 Ball E19		logic 0	SDRAM Bus Memory Address 9
Pin MEM_MA_8 Ball D20		logic 0	SDRAM Bus Memory Address 8
Pin MEM_MA_7 Ball D19		logic 0	SDRAM Bus Memory Address 7
Pin MEM_MA_6 Ball C20		logic 0	SDRAM Bus Memory Address 6
Pin MEM_MA_5 Ball C19		logic 0	SDRAM Bus Memory Address 5
Pin MEM_MA_4 Ball B20		logic 0	SDRAM Bus Memory Address 4
Pin MEM_MA_3 Ball C16		logic 0	SDRAM Bus Memory Address 3
Pin MEM_MA_2 Ball B16		logic 0	SDRAM Bus Memory Address 2
Pin MEM_MA_1 Ball A16		logic 0	SDRAM Bus Memory Address 1
Pin MEM_MA_0 Ball C17		logic 0	SDRAM Bus Memory Address 0
Pin MEM_MDQ_31 Ball U18		hi - z	SDRAM Bus Data 31
Pin MEM_MDQ_30 Ball T18		hi - z	SDRAM Bus Data 30
Pin MEM_MDQ_29 Ball R18		hi - z	SDRAM Bus Data 29
Pin MEM_MDQ_28 Ball R17		hi - z	SDRAM Bus Data 28
Pin MEM_MDQ_27 Ball P18		hi - z	SDRAM Bus Data 27
Pin MEM_MDQ_26 Ball N18		hi - z	SDRAM Bus Data 26
Pin MEM_MDQ_25 Ball N17		hi - z	SDRAM Bus Data 25

Table 2-27. SDRAM Bus Pin Functions (continued)

PIN BALL NUMBER	Function	Reset Value	Description
Pin MEM_MDQ_24 Ball M18		hi - z	SDRAM Bus Data 24
Pin MEM_MDQ_23 Ball K18		hi - z	SDRAM Bus Data 23
Pin MEM_MDQ_22 Ball J17		hi - z	SDRAM Bus Data 22
Pin MEM_MDQ_21 Ball J18		hi - z	SDRAM Bus Data 21
Pin MEM_MDQ_20 Ball H18		hi - z	SDRAM Bus Data 20
Pin MEM_MDQ_19 Ball G18		hi - z	SDRAM Bus Data 19
Pin MEM_MDQ_18 Ball G17		hi - z	SDRAM Bus Data 18
Pin MEM_MDQ_17 Ball F18		hi - z	SDRAM Bus Data 17
Pin MEM_MDQ_16 Ball E18		hi - z	SDRAM Bus Data 16
Pin MEM_MDQ_15 Ball M20		hi - z	SDRAM Bus Data 15
Pin MEM_MDQ_14 Ball M19		hi - z	SDRAM Bus Data 14
Pin MEM_MDQ_13 Ball L20		hi - z	SDRAM Bus Data 13
Pin MEM_MDQ_12 Ball L19		hi - z	SDRAM Bus Data 12
Pin MEM_MDQ_11 Ball K20		hi - z	SDRAM Bus Data 11
Pin MEM_MDQ_10 Ball K19		hi - z	SDRAM Bus Data 10
Pin MEM_MDQ_9 Ball J20		hi - z	SDRAM Bus Data 9
Pin MEM_MDQ_8 Ball J19		hi - z	SDRAM Bus Data 8
Pin MEM_MDQ_7 Ball P19		hi - z	SDRAM Bus Data 7
Pin MEM_MDQ_6 Ball **P20**		hi - z	SDRAM Bus Data 6

Table 2-27. SDRAM Bus Pin Functions (continued)

PIN BALL NUMBER	Function	Reset Value	Description
Pin MEM_MDQ_5 Ball R19		hi - z	SDRAM Bus Data 5
Pin MEM_MDQ_4 Ball R20		hi - z	SDRAM Bus Data 4
Pin MEM_MDQ_3 Ball T19		hi - z	SDRAM Bus Data 3
Pin MEM_MDQ_2 Ball T20		hi - z	SDRAM Bus Data 2
Pin MEM_MDQ_1 Ball U19		hi - z	SDRAM Bus Data 1
Pin MEM_MDQ_0 Ball U20		hi - z	SDRAM Bus Data 0
Pin MEM_RDCLK Ball not pinned out		clk	SDRAM Bus Memory Read Clock (not pinned out)

Table 2-28. JTAG and Test Pin Functions

PIN BALL NUMBER	Function	Reset Value	Description
Pin JTAG_TCK Ball B04			JTAG Test Clock
Pin JTAG_TMS Ball A04			JTAG Test Mode Select
Pin JTAG_TDI Ball A03			JTAG Test Data In
Pin $\overline{\text{JTAG\_TRST}}$ Ball B03			JTAG Reset
Pin JTAG_TDO Ball A02			JTAG Test Data Out
Pin TEST_MODE_0 Ball B02			Test Mode Select 0 (for production test) NOTE: This pin requires a pull-down resistor.
Pin TEST_MODE_1 Ball A01			Test Mode Select 1 (for production test) NOTE: This pin requires a pull-down resistor.
Pin TEST_SEL_0 Ball B01			Scan Enable (for production test), PLL_BYPASS - input, CK_STOP - output
Pin TEST_SEL_1 Ball C03			ENID Input in Test Mode (for production test) NOTE: This pin requires a pull-down resistor.

Table 2-29. CLOCK / RESET Pin Functions

CLOCK / RESET	Functions	Reset Value	Description
Pin $\overline{\text{PORRESET}}$ Ball A13		logic 1	Power On Reset
Pin $\overline{\text{HRESET}}$ Ball B13		logic 1	Hard Reset
Pin $\overline{\text{SRESET}}$ Ball A14		logic 1	Soft Reset
Pin SYS_XTAL_IN Ball A15			APLL Chip clock crystal / external clock input
Pin SYS_XTAL_OUT Ball D14		clk	APLL Chip Clock Crystal
Pin SYS_PLL_TPA Ball B15			MPC5200 System Test Pll Output (analog output)

Table 2-30. Dedicated GPIO Pin Function

DEDICATED GPIO	Functions	Reset Value	Description
Pin GPIO_WKUP_6 Ball C15			
GPIO Wake_Up		logic 0	Asynchronous GPIO with Wake_Up Capability GPIO_WKUP_6
Memory Chip Select		logic 0	SDRAM Chip Select 1
Pin GPIO_WKUP_7 Ball C12			
GPIO Wake_Up		hi - z	Asynchronous GPIO with Wake_Up Capability GPIO_WKUP_7
LocalPlus MOST/Graphics TSIZ		hi - z	TSIZ1 for LocalPlus MOST/GRAPHICS mode

Table 2-31. Systems Integration Unit Pin Functions

SYSTEMS INTEGRATION UNIT	Functions	Reset Value	Descriptions
Pin $\overline{\text{LP\_CS0}}$ Ball W14		logic 1	LocalPlus Bus Chip Select 0
Pin $\overline{\text{LP\_CS1}}$ Ball Y14		logic 1	LocalPlus Bus Chip Select 1
Pin $\overline{\text{LP\_CS2}}$ Ball V15		logic 1	LocalPlus Bus Chip Select 2
Pin $\overline{\text{LP\_CS3}}$ Ball W15		logic 1	LocalPlus Bus Chip Select 3
Pin $\overline{\text{LP\_CS4}}$ Ball Y15		logic 1	LocalPlus Bus Chip Select 4
Pin $\overline{\text{LP\_CS5}}$ Ball V16		logic 1	LocalPlus Bus Chip Select 5

Table 2-31. Systems Integration Unit Pin Functions (continued)

SYSTEMS INTEGRATION UNIT	Functions	Reset Value	Descriptions
Pin $\overline{\text{LP\_OE}}$ Ball D08		logic 1	LocalPlus Bus Output Enable
Pin $\overline{\text{IRQ0}}$ Ball P03			External Interrupt 0
Pin $\overline{\text{IRQ1}}$ Ball P01			External Interrupt 1
Pin $\overline{\text{IRQ2}}$ Ball P02			External Interrupt 2
Pin $\overline{\text{IRQ3}}$ Ball R01			External Interrupt 3
Pin RTC_XTAL_IN Ball C02			Real Time Clock Crystal Input / External Clock Input
Pin RTC_XTAL_OUT Ball C01			Real Time Clock Crystal Ouput

Notes

Chapter 3
Memory Map

3.1 Overview

The following sections are contained in this document:

- MPC5200 Internal Register Memory Map
- MPC5200 Memory Map
- SDRAM Bus
- LocalPlus Bus
  - Memory Cycles
    - Boot Chip Select
    - Chip Selects
  - ATA Cycles
  - PCI Cycles
- MPC5200 Register Summaries
  - Memory Map Registers -- MBAR + 0x0000
  - SDRAM Registers -- MBAR + 0x0100
  - Clock Distribution Module Registers -- MBAR + 0x0200
  - Chip Select Configuration Registers -- MBAR + 0x0300
  - Interrupt Controller Registers -- MBAR + 0x0500
  - General Purpose Timer Registers -- MBAR + 0x0600
  - Slice Timer Control Registers -- MBAR + 0x0700
  - Real Time Clock Registers -- MBAR + 0x0800
  - MSCAN Registers -- MBAR + 0x0900
  - Simple GPIO Registers -- MBAR + 0x0B00
  - Wake-up GPIO Registers -- MBAR + 0x0C00
  - PCI Registers -- MBAR + 0x0D00
  - Serial Peripheral Interface Registers -- MBAR + 0x0F00
  - USB Host Registers -- MBAR + 0x1000
  - BestComm Registers -- MBAR + 0x1200
  - J1850 (BDLC Controller) Registers -- MBAR + 0x1300
  - XL BUS ARbitration Registers -- MBAR + 0x1F00
  - PSC1 Registers -- MBAR + 0x2000
  - PSC2 Registers -- MBAR + 0x2200
  - PSC3 Registers -- MBAR + 0x2400
  - PSC4 Registers -- MBAR + 0x2600
  - PSC5 Registers -- MBAR + 0x2800
  - PSC6 Registers -- MBAR + 0x2C00
  - Ethernet Registers -- MBAR + 0x3000
  - BestComm / PCI Interface Registers -- MBAR + 0x3800
  - ATA Bus Configuration Registers -- MBAR + 0x3A00
  - BestComm / LocalPlus Interface Registers -- MBAR + 0x3C00
  - I2C Configuration Registers -- MBAR + 0x3D00
  - SRAM Module -- MBAR + 0x8000

3.2 Internal Register Memory Map

Table 3-1. Internal Register Memory Map

Address	Name	Description	Reference
MBAR + 0x0000	MM	Memory Map Registers	Section 3.3.3
MBAR + 0x0100	SDRAM	SDRAM Memory Controller registers.	Section 8.7
MBAR + 0x0200	CDM	Clock Distribution Module registers.	Section 5.5
MBAR + 0x0300	CSC	Chip Select Controller registers.	Section 9.7.2
MBAR + 0x0500	ICTL	Interrupt Controller registers.	Section 7.2.3
MBAR + 0x0600	GPT	General Purpose Timer registers.	Section 7.4.4
MBAR + 0x0700	SLT	Slice Time registers.	Section 7.5.1
MBAR + 0x0800	RTC	Real-Time Clock registers.	Section 7.6.3
MBAR + 0x0900	CAN	MSCAN registers.	Section 19.5.2
MBAR + 0x0B00	GPS	GPIO Standard registers	Section 7.3.2.1
MBAR + 0x0C00	GPW	GPIO Wake up registers.	Section 7.3.2.2
MBAR + 0x0D00	PCI	PCI XLB Configuration registers	Section 10.3
MBAR + 0x0F00	SPI	Serial Peripheral Interface registers.	Section 17.3
MBAR + 0x1000	USB	Universal Serial Bus registers.	Section 12.4
MBAR + 0x1200	BDMA	BestComm DMA registers.	Section 13.12
MBAR + 0x1300	BDLC	J1850 (BDLC) registers	Section 19.7
MBAR + 0x1F00	XLARB	XL BUS ARBITRATION Registers	Section 16.2
MBAR + 0x2000	PSC1	Programmable Serial Controller 1 registers.	Section 15.2
MBAR + 0x2200	PSC2	Programmable Serial Controller 2 registers.	Section 15.2
MBAR + 0x2400	PSC3	Programmable Serial Controller 3 registers.	Section 15.2
MBAR + 0x2600	PSC4	Programmable Serial Controller 4 registers.	Section 15.2
MBAR + 0x2800	PSC5	Programmable Serial Controller 5 registers.	Section 15.2
MBAR + 0x2C00	PSC6	Programmable Serial Controller 6 / Infra-Red Data Association registers.	Section 15.2
MBAR + 0x3000	ETH	Ethernet registers.	Section 14.5
MBAR + 0x3800	BPCI	BestComm DMA PCI registers.	Section 10.3
MBAR + 0x3A00	ATA	Advanced Technology Attachment registers.	Section 11.3.1 Section 11.3.2 Section 11.3.3
MBAR + 0x3C00	BLPC	BestComm DMA LocalPlus registers	Section 9.7.2
MBAR + 0x3D00	I^2C	Inter-Integrated Circuit registers.	Section 18.3
MBAR + 0x8000	SRAM	On-chip Static RAM memory locations.	Section 13.13

3.3 MPC5200 Memory Map

The MPC5200 memory map has the following main regions:

- MPC5200 Internal Register Space
- External Busses
  - SDRAM Bus
  - LocalPlus Bus
    - External Chip Selects 0 - 7
    - Memory Space
    - Boot Space
    - Program Space
    - Data Space
- ATA Space

3.3.1 MPC5200 Internal Register Space

The internal registers of the MPC5200 are memory mapped, just like external RAM or any other peripheral devices. The addresses of the internal registers are expressed as offsets to the contents of the MBAR Register (Memory Base Address Register).

The Memory Base Address Register contains the upper 16 bits of the register address space. This sixteen bit value is contained in the lower 16 bits (bit 16 - bit 31) of the Memory Base Address Register. The default value at the release of RESET contained in the MBAR Register is 0x0000 8000. To form a register address, the lower sixteen bits of MBAR are left-justified, forming address bits A31 - A16. Then the 16-bit register offset address for a particular register is concatenated with this value to form a 32-bit address.

NOTE

On the LocalPlus Bus, A31 is the Most Significant Bit and A0 is the Least Significant Bit. It is most important to note that the internal registers of the MPC5200 use bit 0 as the Most Significant Bit and bit 31 as the Least Significant Bit.

The Memory Base Address Register is memory mapped, itself, and it is also the first register in the Internal Register Space. Because the default value in MBAR from the release of RESET is 0x0000 8000 and the MBAR register has an offset address of 0x0000 0000, the absolute address of MBAR becomes 0x8000 0000.

For an additional example, the offset addresses of the Clock Distribution Module Registers start at 0x0200. Using the default value in MBAR, the address of the first register in the Clock Distribution Module is 0x8000 0200.

NOTE

While the MBAR register can be read or written at anytime, the system software must remember the location to which MBAR gets reassigned. Once the contents of the MBAR register are changed, thus reassigning the entire register set to other memory locations, there is no mechanism for finding out the current contents of MBAR except by reading the register at its new address. Thus, it is good practice to store the current contents of MBAR in a predefined, user accessible location.

3.3.2 External Busses

There are two external data / address bus structures on the MPC5200. These are the LocalPlus Bus and the SDRAM Bus. The MPC5200 always begins execution from the release of RESET on the LocalPlus Bus and from the memory device connected to $\overline{\text{LP\_CS0}}$.

3.3.2.1 SDRAM Bus

The SDRAM BUS is designed to accommodate Synchronous Single Data Rate DRAM and Synchronous Double Data Rate DRAM. Program execution generally occurs from programs stored in the memory located on the SDRAM Bus. The SDRAM bus has burst read capability which greatly enhances the bandwidth of the SDRAM Bus. The Memory Clock that drives the SDRAM bus is equal to the XL Bus clock frequency.

From Power On Reset the SDRAM Bus is inactive, that is, the chip select line for the SDRAMs is inactive. The appropriate registers must first be programmed to configure the SDRAM Bus chip select line and make it active before program execution can begin on the SDRAM bus. In general, when a system begins operation from a Power On Reset, “programs stored as data” in memory devices on the LocalPlus Bus are transferred to the SDRAM bus memory by a program stored in the Boot Device on the LocalPlus Bus. Once the “programs stored as data” are transferred to the SDRAM bus memory, the Boot program then causes the CPU to jump to the start address of the program which is now located in SDRAM Bus memory and execution continues from the SDRAM Bus memory.

3.3.2.2 LocalPlus Bus

The LocalPlus Bus is designed to connect to ROM, FLASH, static RAM and other peripheral devices. It is not designed to accommodate DRAM's. Program execution begins from the LocalPlus Bus memory device connected to $\overline{\text{LP\_CS0}}$. In actual practice, the only programs that are usually executed from LocalPlus Bus memory are those used to initialize the MPC5200 and to transfer data from LocalPlus Bus memory to SDRAM bus memory. In general, programs are stored as data in non-volatile memory on the LocalPlus Bus and then transferred to the SDRAM Bus. Once the transfer occurs, program execution is transferred to a program residing in memory on the SDRAM Bus.

The LocalPlus Bus can be accessed by the CPU to perform direct reads and writes of external memory or the LocalPlus Bus can be a BestComm Peripheral. In this case, the CPU programs the BestComm Controller to automatically transfer data from a particular source address to the LocalPlus memory or from the LocalPlus memory to a particular destination address. Almost all peripheral modules, such as the PSC modules, and both the SDRAM Bus and LocalPlus Bus can be BestComm data sources or destinations.

There are 8 chip select lines, CS0 - CS7, associated with the LocalPlus Bus. Also, there are three basic memory access types that can be run on the LocalPlus bus. These are normal memory accesses, PCI cycles and ATA cycles.

The LocalPlus $\overline{\text{LP\_CS0}}$ pin can have two configurations. It can be the BOOT Chip Select line, which is its default condition from the release of RESET, and it can be configured after RESET to be $\overline{\text{LP\_CS0}}$. When configured as the BOOT Chip Select, this chip select line can select Program Space. Thus, program execution can occur from the memory device selected by $\overline{\text{LP\_CS0}}$. If the $\overline{\text{LP\_CS0}}$ pin is configured for data space by user software, then only Data Space Memory can be read or written.

Associated with each Chip Select line is a Start Address Register and a Stop Address Register. There are two Chip Select Start/Stop Address Register pairs associated with the $\overline{\text{LP\_CS0}}$ pin. One Chip Select Start/Stop Register pair is used to configure the $\overline{\text{LP\_CS0}}$ pin as the BOOT Chip Select and the other register pair configures the $\overline{\text{LP\_CS0}}$ pin to run normal memory access cycles in data space, only. Only one of the $\overline{\text{LP\_CS0}}$ Chip Select Start/Stop Address Register pairs should be active at any given time.

When enabled as the Boot Chip Select, only reads are possible. Reads of 64-bits are supported for instruction fetches. Burst reads are also supported. When enabled as a data space memory chip select, only Data Space reads and writes are supported. Code cannot be executed from a memory device connected to $\overline{\text{LP\_CS0}}$ when it is configured as a data space chip select. Bursting is not supported and reads are limited to 32-bits.

There are two additional Start/Stop Address Register pairs used for PCI cycles. These registers are not associated with any chip select line. Chip Select 4 and Chip Select 5 can be configured to run normal memory cycles or ATA cycles. Chip Select 1 - 3 and Chip Select 6 - 7 can only run normal memory cycles.

All the address related registers in this module are in the form of Start/Stop pairs. An address appearing on XL Bus is compared as equal-to-or-greater than the Start value and less-than-or-equal-to the Stop value. If both tests pass then a valid address "hit" occurs for the associated space. For Start values the unused bits are assumed to be zero, for Stop values the unused bits are assumed to be high.

Address registers (and the MBAR itself) have only 16 significant bits. Although these bits are right-justified in the registers they are actually interpreted as the most significant 17 bits of the address for comparison tests. For this reason, software must right shift an absolute address by 16 before writing it as a value into the desired START or STOP Address register. The same is true when reading values from these registers.

Start/Stop comparisons are enabled only if the corresponding enable bit in the MM Address Space Enable Register is high. The proper method for updating Start/Stop registers is to first write the enable bit to zero, update both the Start and Stop registers, and then re-enable the corresponding enable bit by writing it high.

NOTE

Failure to follow the above procedure could result in bus hanging and machine check errors.

3.3.3 Memory Map Space Register Description

These registers exist in the Memory Map register space relative to Memory Base Address Register (MBAR).

3.3.3.1 Memory Address Base Register —MBAR + 0x0000

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Base Address Register															
W																
RESET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	Reserved	These bits are reserved.
16:31	Base Address Register	Provides the offset to which all register space for MPC5200 is accessed. The reset value of this register is 0x8000, which provides for a MBAR of 0x8000 0000. All of MPC5200 registers are then accessible at MBAR+offset, where offset refers to the given value in Table 3-1 for the respective module.

3.3.3.2 Boot and Chip Select Addresses

MBAR offset	Name	Description
0x0004	CS0 Start Address	Chip Select 0 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0008	CS0 Stop Address	
0x000C	CS1 Start Address	Chip Select 1 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0010	CS1 Stop Address	
0x0014	CS2 Start Address	Chip Select 2 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0018	CS2 Stop Address	
0x001C	CS3 Start Address	Chip Select 3 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0020	CS3 Stop Address	
0x0024	CS4 Start Address	Chip Select 4 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0028	CS4 Stop Address	

MBAR offset	Name	Description
0x002C	CS5 Start Address	Chip Select 5 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0030	CS5 Stop Address	
0x004C	Boot Start Address	Boot Addressing through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses accesses the boot space. By default, the address space accessed starts at 0x0000 0000 or 0xFFF0 0000 depends on the reset configuration. The size of the boot address space after reset is 512Kbytes.
0x0050	Boot Stop Address	
0x0058	CS6 Start Address	Chip Select 6 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x005C	CS6 Stop Address	
0x0060	CS7 Start Address	Chip Select 7 through the LocalPlus Bus. Any access on an address between the Start and Stop Addresses enables this chip select.
0x0064	CS7 Stop Address	

All of these Base Address Registers work the same

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Base Address															
W																
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:15	Reserved	These bits are reserved.
16:31	Base Address	The 16 most significant bits of the Base Address. A value of 0x4000 would translate into a base address of 0x4000 0000.

3.3.3.3 SDRAM Chip Select Configuration Registers

MBAR offset	Name	Description
0x0034	SDRAM Chip Select 0	Contains the Base Addresses and configurations for SDRAM's connected to the SDRAM controller.
0x0038	SDRAM Chip Select 1	

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Base XLB Address												Reserved			
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 lsb
R	Reserved											SDRAM Size				
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:11	Base XLB Address	Start address for memory
12:26	Reserved	These bits are reserved.
27:31	SDRAM size	Should be set to size of SDRAM at corresponding SDRAM chip select. Settings are included in the following table.
Note: The Base XLB Address has to be SDRAM size aligned.		

SDRAM Memory Size	SDRAM size bit setting
11111	4GB
11110	2GB
11101	1GB
11100	512MB
11011	256MB
11010	128MB
11001	64MB
11000	32MB
10111	16MB
10110	8MB
10101	4MB
10100	2MB
10011	1MB
00001-10010	Reserved
0000	Disable

3.3.3.4 IPBI Control Register and Wait State Enable —MBAR+0x0054

The IPBI Control Register consists of the Enables for the Base Addresses set in Memory Map Space

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				CS7 Ena	CS6 Ena	Boot Ena	Reserved			CS5 Ena	CS4 Ena	CS3 Ena	CS2 Ena	CS1 Ena	CS0 Ena
W																
RESET	0	0	0	0	0	0	1	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 lsb
R	Reserved															WSE
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Name	Description
0:3	Reserved	These bits are reserved.
4	CS7 Ena	Chip Select 7 Enable
5	CS6 Ena	Chip Select 6 Enable
6	Boot Ena	Boot Enable
7:9	Reserved	These bits are reserved.
10	CS5 Ena	Chip Select 5 Enable
11	CS4 Ena	Chip Select 4 Enable
12	CS3 Ena	Chip Select 3 Enable
13	CS2 Ena	Chip Select 2 Enable
14	CS1 Ena	Chip Select 1 Enable
15	CS0 Ena	Chip Select 0 Enable
16:30	Reserved	These bits are reserved.
31	WSE	Wait State Enable bit. This bit should always be enabled when running an IP bus frequency of >66MHz.

Chapter 4
Resets and Reset Configuration

4.1 Overview

The following sections are contained in this document:

- Hard and Soft Reset Pins
- Reset Sequence
- Reset Operation
- Other Resets
- Reset Configuration

4.2 Hard and Soft Reset Pins

MPC5200 has three primary reset pins, which are implemented as open drain I/Os[1]:

- Power-On Reset—$\overline{\text{PORRESET}}$
- Hard Reset—$\overline{\text{HRESET}}$
- Soft Reset—$\overline{\text{SRESET}}$

$\overline{\text{PORRESET}}$ is a power-on reset input. It is asserted by an external source and must be held active for a specified period of time until power is stable to the MPC5200.

$\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ can be asserted by an external source or they can be asserted by reset generation logic internal to MPC5200.

Internal reset logic analyzes all internal and external reset sources and asserts internal and external reset signals appropriately.

When a hard reset ($\overline{\text{HRESET}}$) is detected, reset logic counters hold internal and external PORRESETor a minimum of 4096 reference clock cycles, or until the external reset source releases the reset, whichever is longer.

4.2.1 Power-On Reset—$\overline{\text{PORRESET}}$

$\overline{\text{PORRESET}}$ must be asserted externally when power is applied to the system for a required period of time (see Section 4.4, *Reset Operation*). When $\overline{\text{PORRESET}}$ is asserted, internal logic forces $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ active. The MPC5200 system oscillator must begin oscillation during $\overline{\text{PORRESET}}$ assertion, and the system APLL establishes a locked condition.

During $\overline{\text{PORRESET}}$ or $\overline{\text{HRESET}}$ the reset configuration word is sampled to establish the initial state of various vital internal MPC5200 functions. The reset configuration word is latched internally when $\overline{\text{PORRESET}}$ or $\overline{\text{HRESET}}$ is released.

When initiated by $\overline{\text{PORRESET}}$, the 4096 reference clock period minimum $\overline{\text{HRESET}}$ begins counting when $\overline{\text{PORRESET}}$ is released.

Source of power-on reset is an external, board level reset source like a push button, reset control logic, etc.

4.2.2 Hard Reset—$\overline{\text{HRESET}}$

$\overline{\text{HRESET}}$ is a bidirectional signal with a Schmitt-trigger input and an open drain output. $\overline{\text{HRESET}}$ requires an external pull-up. Assertion of external $\overline{\text{HRESET}}$ causes external $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$, and internal hard and soft resets, to be asserted for at least 4096 reference clock cycles.

During $\overline{\text{PORRESET}}$ or $\overline{\text{HRESET}}$ the reset configuration word is sampled to establish the initial state of various vital internal MPC5200 functions. The reset configuration word is latched internally when $\overline{\text{PORRESET}}$ or $\overline{\text{HRESET}}$ is released.

$\overline{\text{HRESET}}$ can also be asserted by internal sources. When $\overline{\text{HRESET}}$ is asserted internally, external $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ are also asserted.

Sources of hard reset are:

- $\overline{\text{PORRESET}}$ or $\overline{\text{HRESET}}$ pins asserted
- Hard reset asserted by debug module
- Reset signal asserted by watchdog timer or checkstop reset

1. All “open drain” outputs of MPC5200 are actually regular 3-state output drivers with the output data tied low, and the output enable controlled. Thus, unlike a true open drain, there is a current path from the external system to the MPC5200 I/O power rail if the external signal is driven above the MPC5200 I/O power rail voltage.

4.2.3 Soft Reset—SRESET

External SRESET is an open drain signal. SRESET requires an external pull-up. Assertion of SRESET causes assertion of the internal soft reset. Internal soft reset is actually an *interrupt* that takes the same exception vector as HRESET. In particular, this means that SRESET cannot abort a hung XLB operation, and no device should use SRESET in a way that interferes with any bus operation in progress.

SRESET can also be asserted by internal sources. When SRESET is asserted internally, external SRESET is also asserted.

Sources of soft reset:

- PORRESET, HRESET, or SRESET external pins asserted
- Soft reset bit in Clock Distribution Module (CDM) register asserted by processor
- Soft reset asserted by debug module

4.3 Reset Sequence

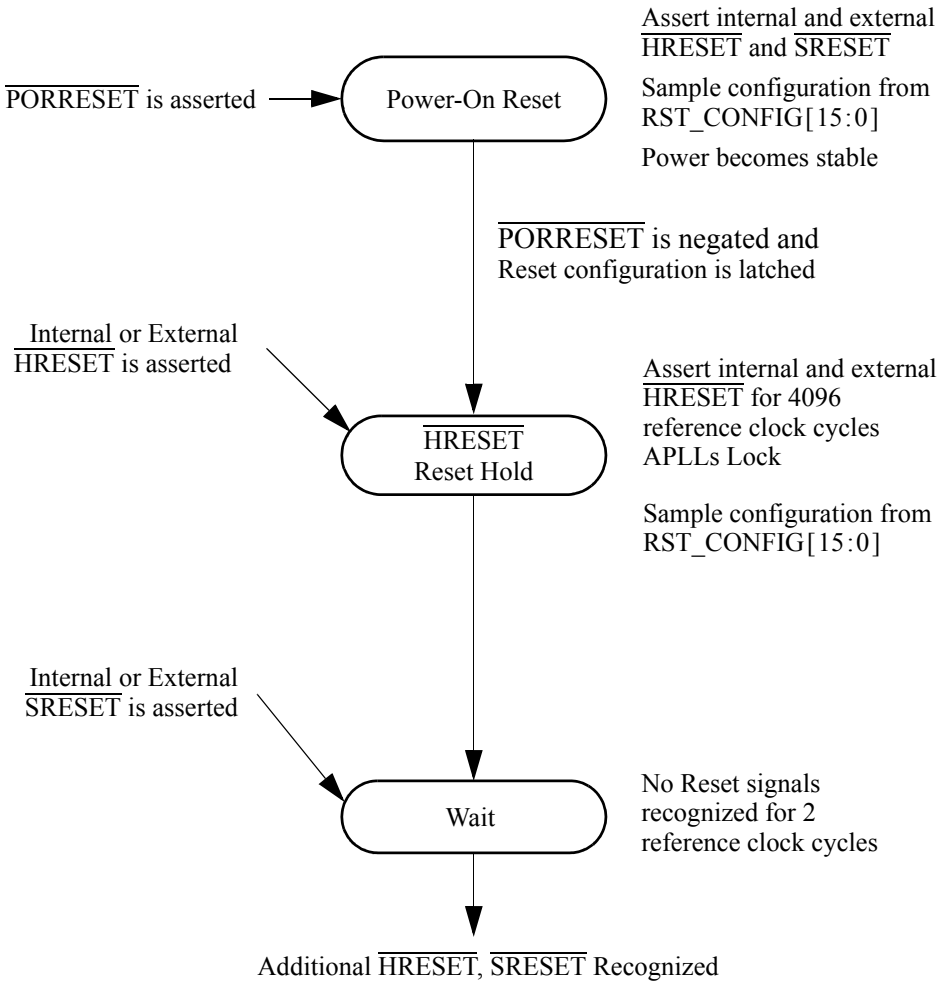


Figure 4-1. Reset sequence

4.4 Reset Operation

PORRESET must remain asserted for at least 100µs after all power supplies and the system oscillator input are stable and operating within specs. Following deassertion of power-on reset, HRESET and SRESET remain low for 4096 reference clock cycles.

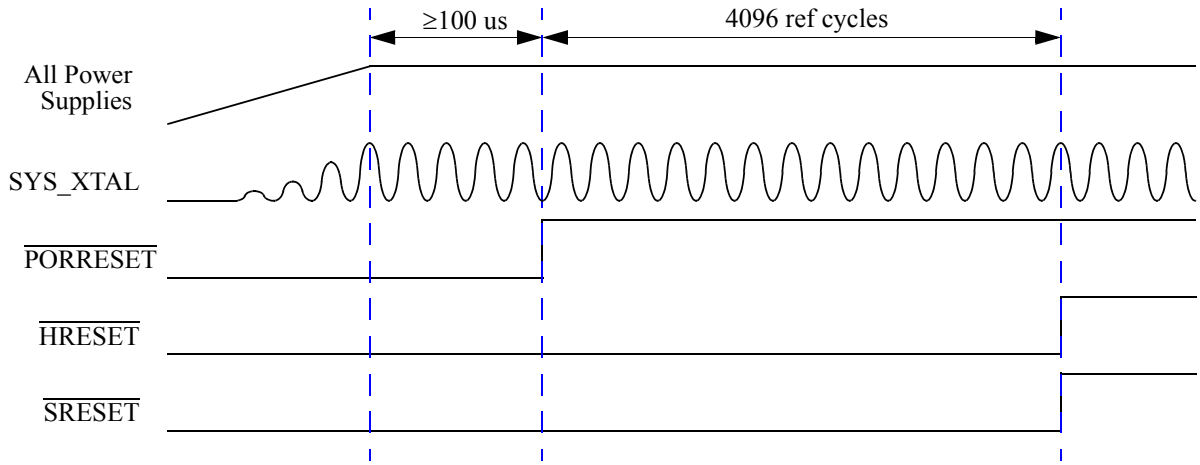


Figure 4-2. $\overline{\text{PORRESET}}$ Assertion

When external $\overline{\text{HRESET}}$ is asserted, internal reset logic catches the reset signal held low and asserts internal hard and soft resets for 4096 reference clock cycles. The external reset signal must be held low for at least 4 reference clock cycles (must catch 4 rising edges of reference clock) to be recognized and assert the internal reset signals.

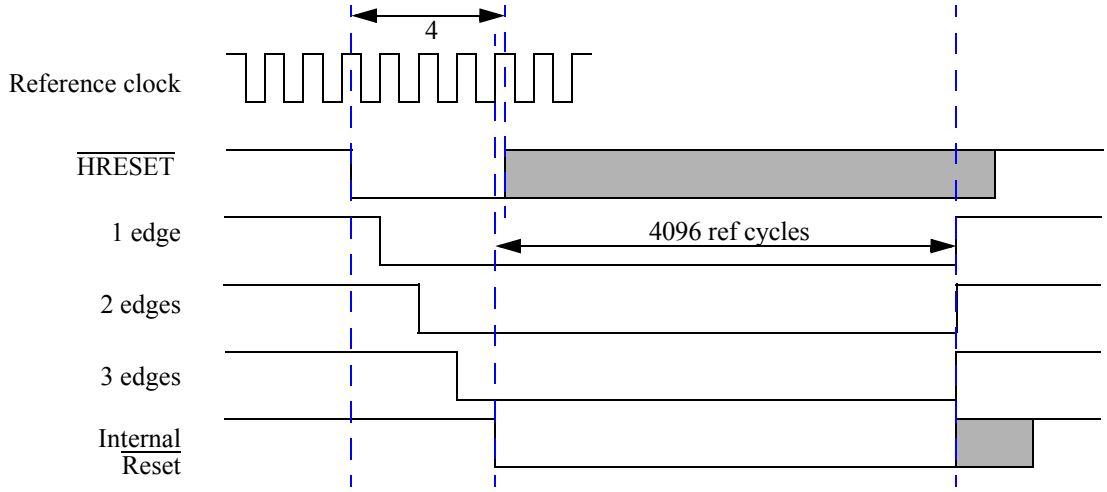


Figure 4-3. Internal Hard Reset vs External $\overline{\text{HRESET}}$ Assertion

The Clock Distribution Module contains a register that can be written by the microprocessor to assert soft reset. Writing the $\overline{\text{SRESET}}$ bit in this register to zero causes external $\overline{\text{SRESET}}$ and internal soft reset to be asserted.

4.5 Other Resets

MPC5200 has four other reset signals. These signals are specific to certain peripheral modules and are controlled in the context of that module, not globally.

Table 4-1. Module Specific Reset Signals

	Definition
PCI_RESET	PCI bus reset output. Generated by processor write to a PCI register.
AC97_1_RES	AC97 reset output. Generated from the AC97 PSC1 module.
AC97_2_RES	AC97 reset output. Generated from the AC97 PSC2 module.

Table 4-1. Module Specific Reset Signals (continued)

	Definition
JTAG_TRST	JTAG reset input. Generated externally from JTAG or debug control logic. This input only resets the JTAG logic. Other system resets ($\overline{\text{PORRESET}}$, $\overline{\text{HRESET}}$, and $\overline{\text{SRESET}}$) do not reset the JTAG logic. **Note:** For information on the reset signal JTAG_TRST and the relationship to other reset signals refer to the MPC5200 Hardware Specifications.
ATA Reset	This is NOT a reset pin on MPC5200. The ATA reset for the external drive must be supplied by the board level reset source, or if software control is required, generated via a GPIO.

4.6 Reset Configuration

The MPC5200 is initialized by sampling values found on specific device pins during power-on reset ($\overline{\text{PORRESET}}$) or hard-reset ($\overline{\text{HRESET}}$). These pins are outputs in normal operation, but are sampled as inputs during power-on reset or hard-reset. External pull-up or pull-down resistors on the board are used to force a value on these pins during power-on reset or hard-reset. These values are latched into the CDM Reset Configuration register at the end of power-on reset or hard-reset, then distributed to various peripherals. After power-on reset or hard-reset, these outputs overdrive the external pull-up or pull-down resistors and behave as functional outputs. Only during power-on reset or hard-reset these pins are inputs.

Table 4-2 gives the power-on reset or hard-reset configuration inputs.

Table 4-2. Reset Configuration Word Source Pins

Pkg Ball	Reset Config Pin	I/O Signal Name	CDM Reset Config Register Bit	Config Signal from CDM	Description
Y18	RST_CFG0	ATA_DACK	PORCFG[31]	ppc_pll_cfg_4	MPC5200 G2_LE PPC Core PLL Configuration
Y17	RST_CFG1	ATA_IOR	PORCFG[30]	ppc_pll_cfg_3	
W17	RST_CFG2	ATA_IOW	PORCFG[29]	ppc_pll_cfg_2	
W16	RST_CFG3	LP_RWB	PORCFG[28]	ppc_pll_cfg_1	
V14	RST_CFG4	LP_ALE	PORCFG[27]	ppc_pll_cfg_0	
Y13	RST_CFG5	LP_TS	PORCFG[26]	xlb_clk_sel	bit=0:XLB_CLK=f_{system} / 4 bit=1:XLB_CLK=f_{system} / 8
H02	RST_CFG6	USB1_1	PORCFG[25]	sys_pll_cfg_0	bit=0:f_{system}=16 x SYS_XTAL_IN bit=1:f_{system}=12 x SYS_XTAL_IN
H03	RST_CFG7	USB1_2	PORCFG[24]	sys_pll_cfg1	bit=0:f_{vcosys} = f_{system} bit=1:f_{vcosys} = 2 x f_{system}
K01	RST_CFG8	ETH0	PORCFG[23]	boot_rom_mg	bit=0:No Boot in Most Graphics Mode [1] bit=1:Boot in Most Graphics Mode [1,2,4]
K03	RST_CFG10	ETH2	PORCFG[21]	ppc_msrip	Microprocessor Boot Address/Exception table location. bit=0:0000_0100 (hex) bit=1:FFF0_0100 (hex)
J01	RST_CFG11	ETH3	PORCFG[20]	boot_rom_wait	bit=0:4 PCI bus clocks of wait state[1] bit=1:48 PCI bus clocks of wait state[1]
J02	RST_CFG12	ETH4	PORCFG[19]	boot_rom_swap	bit=0:no byte lane swap, same endian ROM image bit=1:byte lane swap, different endian ROM image

Table 4-2. Reset Configuration Word Source Pins (continued)

Pkg Ball	Reset Config Pin	I/O Signal Name	CDM Reset Config Register Bit	Config Signal from CDM	Description
L03	RST_CFG13	ETH5	PORCFG[18]	boot_rom_size	For non-muxed boot ROMs: [2,3] bit=0: 8bit boot ROM data bus, 24bit max boot ROM address bus bit=1: 16bit boot ROM data bus, 16bit boot ROM address bus For muxed boot ROMs: boot ROM address is max 25 significant bits during address tenure. bit=0: 16bit ROM data bus bit=1: 32bit ROM data bus
N02	RST_CFG14	ETH6	PORCFG[17]	boot_rom_type	bit=0: non-muxed boot ROM bus, single tenure transfer. [1] bit=1: muxed boot ROM bus, with address and data tenures, $\overline{\text{ALE}}$ and $\overline{\text{TS}}$ active. [1]
K02	RST_CFG15	ETH1	PORCFG[16]	large_flash_sel	bit=0:No Boot in Large Flash Mode [1] bit=1:Boot in Large Flash Mode [1,3,4]

Note:

1. If multipe settings are choosen the following priorities are valid:
   1. large_flash_sel
   2. boot_rom_mg
   3. boot_rom_type
2. The boot_rom_size configuration signal doesn't influence the address and data bus width of the MOST Graphics boot mode configuration. The maximum address bus width is fixed to 24 bit and the data bus width is fixed to 32 bit.
3. The boot_rom_size configurationsignal doesn't influence the address bus width of the Large Flash boot mode configuration. The maximum address bus width is fixed to 26 bit.
4. The PCI controller is disabled, if booting in Large Flash or MOST Graphics mode is selected.

Notes

Chapter 5
Clocks and Power Management

5.1 Overview

The following sections are contained in this document:

- Clock Distribution Module (CDM)
- MPC5200 Clock Domains
- Power Management
- CDM Registers

5.2 Clock Distribution Module (CDM)

The CDM is the source of all internally generated clocks and reset signals. The MPC5200 clock generation uses two analog phase locked loop (APLL) blocks. The system APLL takes an external reference frequency (nominal 27–33MHz) and generates the following internal clocks. See Table 5-1.

Table 5-1. Clock Distribution Module

Clock Name	Description
XLB CLOCK (xlb_clk)	Microprocessor on-chip 64-bit XLB clock. This is the fundamental MPC5200 frequency.
MEM_CLOCK (mem_clock)	SDRAM Controller memory clock supplied to external SDRAM devices. Max frequency is 132MHz. The memory clock frequency is always equal to the XLB frequency.
IPB CLock (ipb_clk)	Intellectual Property Bus (IPB) clock.
PCI CLOCK (pci_clk)	PCI Controller clock.
CORE CLOCK	Clock for the 603e G2_LE Core. The core APLL takes the XLB clock and generates the G2_LE CORE clock.
48MHz CLOCK USB CLOCK	48MHz clock for USB and IrDA (PSC6). This clock can be sourced internally from the CDM or from an external source via the IrDA_USB_CLK pin.

5.3 MPC5200 Clock Domains

The MPC5200 has 5 major clock domains, which are listed below. Details are given in the sections that follow.

- 603e G2_LE Core Clock Domain—internal processor core frequency
- Processor Bus (XLB) Clock Domain —internal 603e G2_LE Core processor bus
- SDRAM Memory Controller Clock Domain
- IPB Clock Domain—programming register and peripheral interface frequency
- PCI Clock Domain

The following smaller peripheral clock domains can be asynchronous to the fundamental clock frequencies on MPC5200:

Ethernet—The Ethernet Controller requires a 10MHz or 25MHz Tx/Rx clock. Both clocks are inputs to MPC5200, supplied from the Ethernet physical device (ETH_RXCLK, ETH_TXCLK pins). The Ethernet Controller Tx/Rx portion of MPC5200 is asynchronous to the rest of MPC5200.

USB—The Universal Serial Bus module Tx/Rx portion can be clocked by an external clock source (IR_USB_CLK pin) or can be clocked by an internally generated clock. Clock frequency must be 48MHz. When the clock source is externally supplied, the USB module Tx/Rx portion is asynchronous to the rest of MPC5200.

PSC—The PSC (Programmable Serial Controller) module is instantiated in the MPC5200 6 times (PSC1 to PSC6). The PSC has different modes of operation. In some cases the logic is clocked by internally generated clocks (i.e., UART mode), and in others the PSC is clocked by external clock sources (i.e., CODEC mode). If the PSC logic clocked from an external source then the logic is asynchronous to the rest of the chip.

When the PSC6 is configured as **IrDA**—The Infrared Data Association module Tx/Rx portion can be clocked by an external clock source (IR_USB_CLK pin) or can be clocked by an internally generated clock.

— When generated internally, the clock source can be a fix 48MHz clock generator or a programmable clock generator (Mclk).
— When generated externally, the frequency can be different

NOTE

Only one pin is allocated to supply the USB and PSC6/IrDA clock. If both modules require external clock generation, the frequency must be 48MHz.

SPI—The SPI (Serial Peripheral Interface) has a clock input pin, SPI_CLK, that can be supplied externally. The SPI module therefore has a small asynchronous clock domain.

I^{2}C—There are two I^{2}C (Inter-Integrated Circuit) modules on MPC5200. Both have input source clocks (I^{2}Cx_CLK) and therefore asynchronous clock domains.

RTC—The RTC (Real-Time Clock) has its own clock domain, which is clocked by an external 32.768KHz oscillator. The two oscillator pins are RTC_XTAL_IN and RTC_XTL_OUT. There is an asynchronous boundary between this clock domain and the IPB register interface.

JTAG—The JTAG (Joint Test Action Group) has its own clock domain clocked by the JTAG_TCK pin.

The following peripheral functions use clocks generated from CDM.

MSCAN—The MSCAN (Freescale [formerly Motorola] Scalable Controller Area Network) internal baud rate generator also uses the ipb_clk or can be derived from the oscillator clock sys_xtal_in. The resultant divided clock samples an incoming CAN data stream and generates an outgoing data stream.

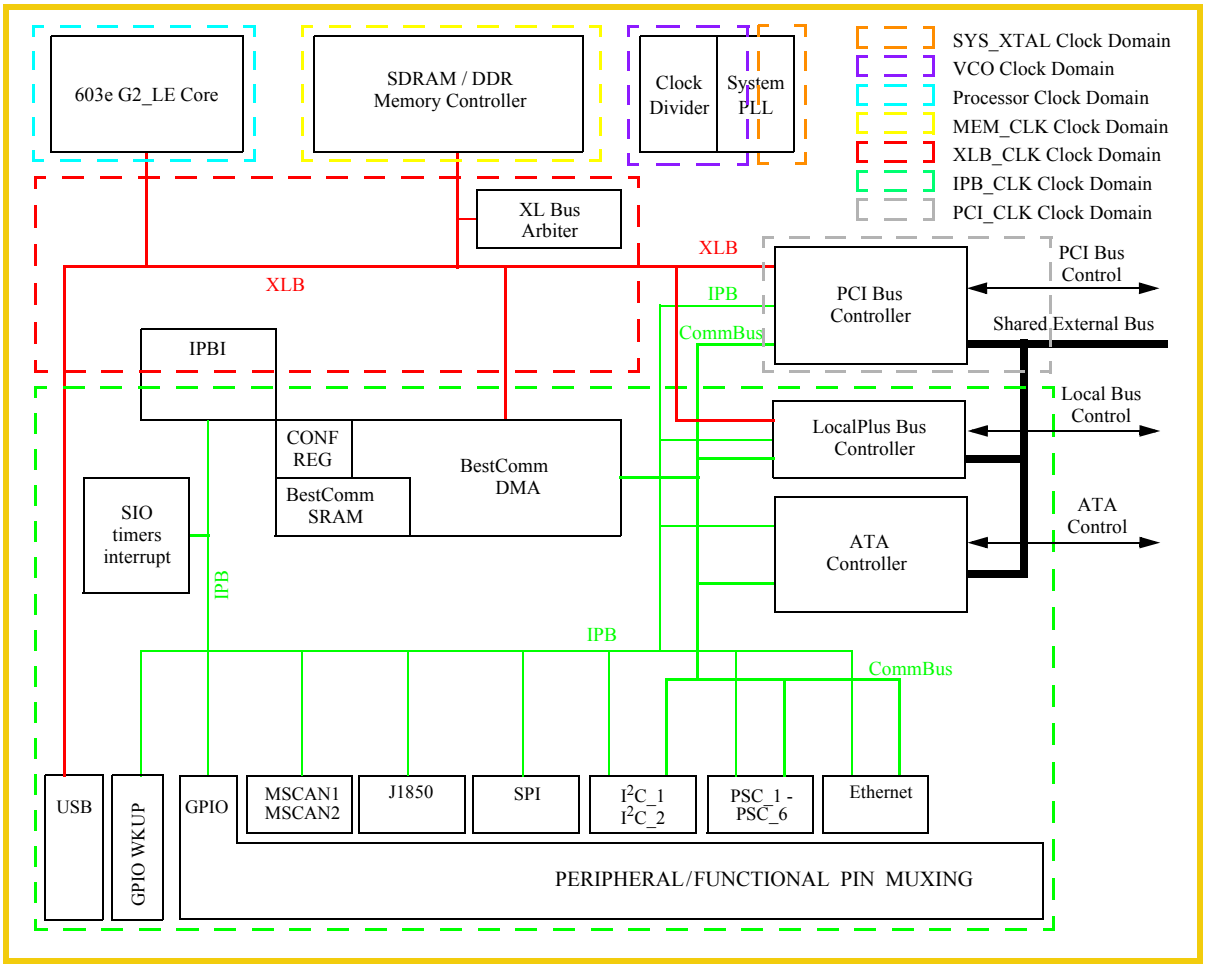


Figure 5-1. Primary Synchronous Clock Domains

5.3.1 MPC5200 Top Level Clock Relations

Figure 5-2 shows the CDM clock divide circuitry. This picture shows only the functional clocks. The clock network regarding the scan and bypass modes is not included.

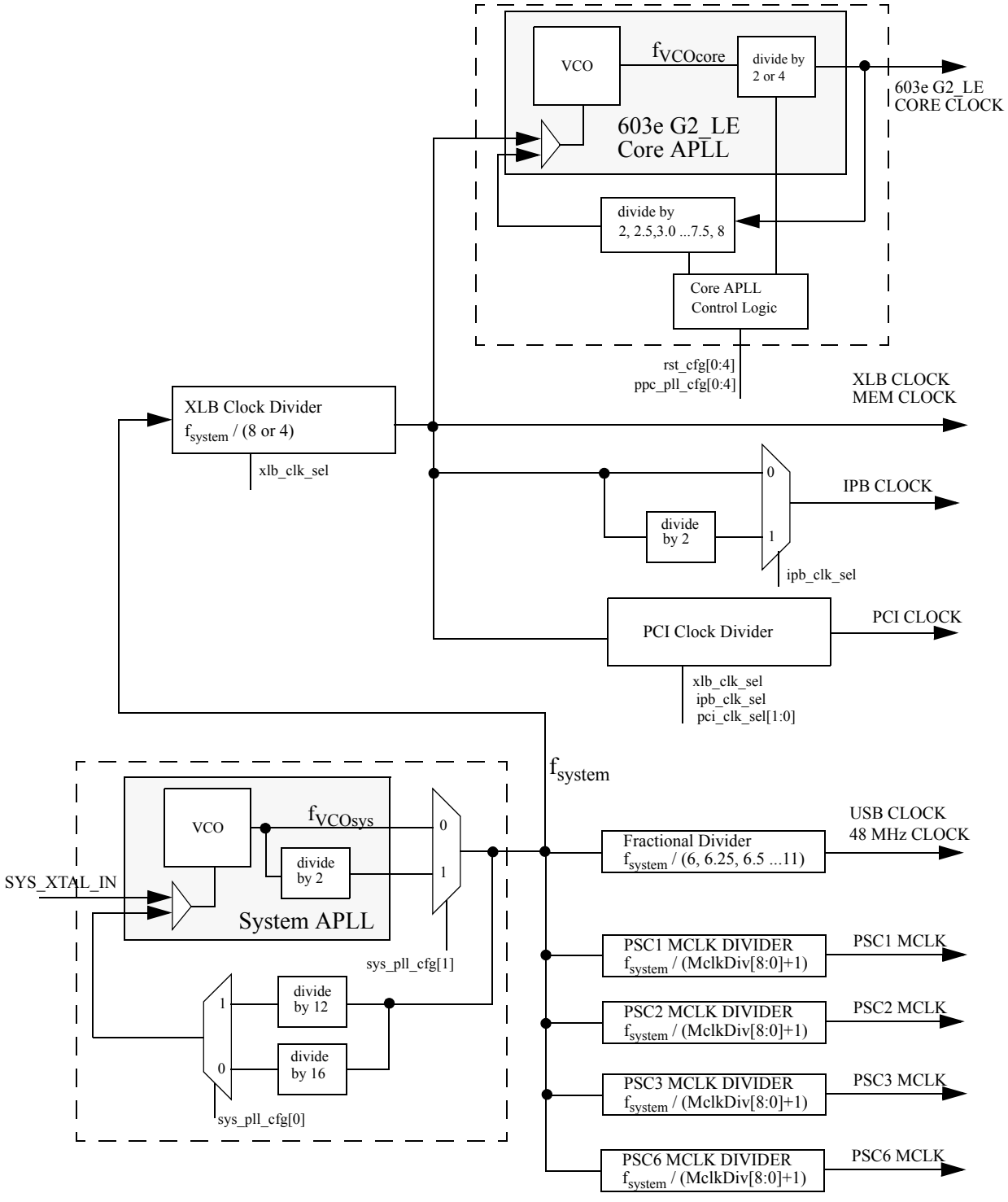


Figure 5-2. MPC5200 Clock Relations

Table 5-2 shows the System PLL configuration and the corresponding fsystem frequencies for a 27.0 MHz and 33.0 MHz input clock. Table 5-3 shows all possible clock ratios.

Table 5-2. System PLL Ratios

SYS_XTAL_IN	sys_pll_cfg[1]	sys_pll_cfg[0]	f_{VCOsys} [MHz]	f_{system} [MHz]
27.0	0	0	432.0	432.0
	0	1	324.0	324.0
	1[a]	0	864.0	432.0
	1	1	648.0	324.0
33.0	0	0	528.0	528.0
	0	1	396.0	396.0
	1[a]	0	1056.0	528.0
	1	1	792.0	396.0

[a] These are invalid configurations. The f_{VCOsys} frequencies exceed the maximum operation frequency. See MPC5200 Hardware Specification.

Table 5-3. MPC5200 Clock Ratios

xlb_clk_sel	XLB CLOCK	ipb_clk_sel	IPB CLOCK	pci_clk_sel[1:0]	PCI CLOCK	CLOCK Ratio XLB:IPB:PCI
0	f_{system} / 4	0	XLB	00	XLB	4:4:4
0	f_{system} / 4	0	XLB	01	XLB/2	4:4:2
0	f_{system} / 4	0	XLB	10	XLB/4	4:4:1
0	f_{system} / 4	0	XLB	11	XLB/4	4:4:1
0	f_{system} / 4	1	XLB /2	00	XLB/2	4:2:2
0	f_{system} / 4	1	XLB /2	01	XLB/4	4:2:1
0	f_{system} / 4	1	XLB /2	10	XLB/4	4:2:1
0	f_{system} / 4	1	XLB /2	11	XLB/4	4:2:1
1	f_{system} / 8	0	XLB	00	XLB	2:2:2
1	f_{system} / 8	0	XLB	01	XLB/2	2:2:1
1	f_{system} / 8	0	XLB	10	XLB/4	2:2:0.5
1	f_{system} / 8	0	XLB	11	XLB/4	2:2:0.5
1	f_{system} / 8	1	XLB /2	00	XLB/2	2:1:1
1	f_{system} / 8	1	XLB /2	01	XLB/4	2:1:0.5
1	f_{system} / 8	1	XLB /2	10	XLB/4	2:1:0.5
1	f_{system} / 8	1	XLB /2	11	XLB/4	2:1:0.5

Table 5-4. Typical System Clock Frequencies

f_{system} [MHz]	XLB Clock [MHz]	IPB CLock [MHz]	PCI CLOCK [MHz]	Clock Ratio XLB:IPB:PCI
528.0	132.0	132.0	66.0	4:4:2
			33.0	4:4:1
		66.0	66.0	4:2:2
			33.0	4:2:1
		33.0	33.0	4:1:1
	66.0	66.0	66.0	2:2:2
			33.0	2:2:1
		33.0	33.0	2:1:1

Table 5-4 shows the typical clock ratios with a 33.0 MHz clock input on the SYS_XTAL_IN pin and a System PLL divide value 16 (sys_pll_cfg[0] = 0).

NOTE

Frequency ranges in Table 5-3 and Table 5-4 represent possible ranges of operation. A variety of conditions may prevent the part from actually performing at these frequency ranges. For data relating to actual performance, see Section A.2, AC Timing.

5.3.2 603e G2_LE Core Clock Domain

The 603e G2_LE Core has its own APLL and clock domain, which is separate from, but synchronous with, the rest of the chip. The reference for the processor APLL is XLB clock. The 603e G2_LE Core can run at all integer and half-integer multiples of xlb_clk from 2x to 8x (i.e., 2x, 2.5x, 3x, 3.5x, 4x, 4.5x, 5x, 5.5x, 6x, 6.5x, 7x, 7.5x, 8x) to a maximum frequency of 396MHz. Table 5-5 shows the available core frequencies based on the xlb_clk frequency range.

NOTE

These frequencies are not guaranteed. Actual operation frequencies will depend on silicon characterization and operating conditions.

Table 5-5. 603e G2_LE Core Frequencies vs. XLB Frequencies

XLB Clock (MHz)		132	108	99	81	66	54	49.5	40.5	33	27
G2 Core PLL Bus to Core Multiplier[a]	x1	—	—	—	—	—	—	—	—	—	—
	x1.5	—	—	—	—	—	—	—	—	—	—
	x2	264	216	198	162	132	108	99	81	66	54
	x2.5	330	270	247.5	202.5	165	135	123.8	101.3	82.5	67.5
	x3	396	324	297	243	198	162	148.4	121.5	99	81
	x3.5		378	346.5	283.5	231	189	173.3	141.8	115.5	94.5
	x4			396	324	264	216	198	162	132	108
	x4.5				364.5	297	243	222.8	182.3	148.5	121.5
	x5					330	270	247.5	202.5	165	135
	x5.5					363	297	272.3	222.8	181.5	148.5
	x6					396	324	297	243	198	162
	x6.5						351	321.8	263.3	214.5	175.5
	x7						378	346.5	283.5	231	189
	x7.5							371.3	303.8	247.5	202.5
	x8							396	324	264	216

Note: 1x and 1.5x multiply ratios are not available in this version of the MPC5200.

[a] See Table 5-6, XLB to CORE clock ratio.

Table 5-6 gives the 603e G2_LE Core APLL and operating frequency options compared to the xlb_clk reference input (shown in Figure 5-2). The selection of a 603e G2_LE Core frequency is made at Power-On Reset (POR) via the reset configuration inputs. For more information see Section 4.6, *Reset Configuration*.

Frequency ranges indicated in Table 5-6 represent possible ranges for the processor APLL. A variety of conditions may prevent the part from actually performing at these frequency ranges. For data relating to actual performance, see the MPC5200 Hardware Specification.

Table 5-6. 603e G2_LE Core APLL Configuration Options

ppc_pll_cfg		Bus:Core Ratio (XLB : CORE CLOCK)	Core:VCO Ratio (CORE CLOCK: $f_{VCOcore}$)	Bus:VCO Ratio (XLB : $f_{VCOcore}$)
hex	[0:1:2:3:4]			
0x00	00000	—	—	—
0x01	00001	—	—	—
0x02	00010	—	—	—
0x04	00100	1:2	1:2	1:4
0x05	00101	1:2	1:4	1:8
0x06	00110	1:2.5	1:2	1:5
0x07	00111	1:4.5	1:2	1:9

Table 5-6. 603e G2_LE Core APLL Configuration Options (continued)

ppc_pll_cfg		Bus:Core Ratio (XLB : CORE CLOCK)	Core:VCO Ratio (CORE CLOCK: $f_{VCOcore}$)	Bus:VCO Ratio (XLB : $f_{VCOcore}$)
hex	[0:1:2:3:4]			
0x08	01000	1:3	1:2	1:6
0x09	01001	1:5.5	1:2	1:11
0x0A	01010	1:4	1:2	1:8
0x0B	01011	1:5	1:2	1:10
0x0C	01100	—	—	—
0x0D	01101	1:6	1:2	1:12
0x0E	01110	1:3.5	1:2	1:7
0x10	10000	1:3	1:4	1:12
0x11	10001	1:2.5	1:4	1:10
0x12	10010	1:6.5	1:2	1:13
0x14	10100	1:7	1:2	1:14
0x16	10110	1:7.5	1:2	1:15
0x18	11000	—	—	—
0x1C	11100	1:8	1:2	1:16
0x03 0x13	00011 10011	PLL off/bypassed xlb_clk clocks core directly, 1x bus-to-core		
0x0F 0x1F	01111 11111	PLL off, no core clocking occurs.		
0x15 0x17 0x19 0x1A 0x1B 0x1D 0x1E	10101 10111 11001 11010 11011 11101 11110	Reserved, should not be used.		
	Note: Shading implies same mode can be configured with ppc_pll_cfg[0]=0			

NOTE

The XLB CLOCK frequency and the ppc_pll_cfg[0:4] must be chosen such that resulting CORE CLOCK frequency and PLL ($f_{VCOcore}$) frequency do not exceed their respective maximium or minimum operating frequencies. Refer to Table 5-5 and MPC5200 Hareware Specification.

5.3.3 Processor Bus (XLB) Clock Domain

The XLB clock (xlb_clk) is the fundamental MPC5200 clock frequency. The following operate at this frequency:

- The internal processor address/data bus
- The internal SDRAM Controller
- External SDRAM

All functional blocks that interface to the XLB must operate at this frequency, or have a section of logic that operates at this frequency.

5.3.4 SDRAM Memory Controller Clock Domain

The Memory Controller uses the clocks shown in Table 5-7.

Table 5-7. SDRAM Memory Controller Clock Domain

Bits	Description
mem_clk	mem_clk is always the same frequency as xlb_clk.
mem_2x_clk, $\overline{\text{mem\_2x\_clk}}$	These internal clocks are twice the frequency of xlb_clk and are used to add more resolution to SDRAMC control signals
mem2x1x_clk (becomes mem_rd_clk)	This is the source of the internal memory read clock. It always operates at the memory data rate, 1x mem_clk for SDR, 2x mem_clk for DDR. The physical circuit path of mem2x1x_clk is matched as closely as possible to the on-chip portion of the memory clock output and the read data input; a tapped delay chain is used to match off-chip portions of the memory clock and read data path.

Figure 5-3 shows the clock relationships for the SDRAM Controller.

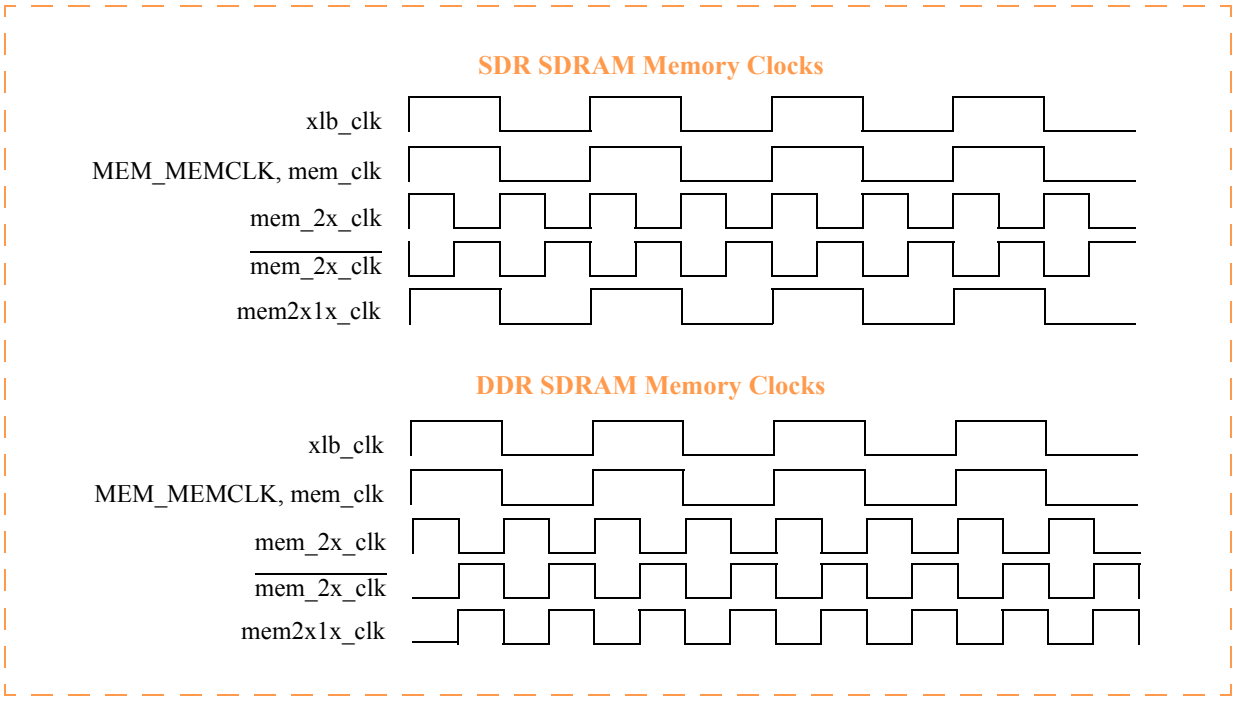


Figure 5-3. Timing Diagram—Clock Waveforms for SDRAM and DDR Memories

Since the XLB is 64bits and the SDRAM external bus is 32bits, when SDR (single data rate) SDRAM memory is used, the XLB bandwidth is only half utilized. When DDR (dual data rate) memory is used, the XLB bandwidth is fully used on SDRAM transactions.

MPC5200 supplies 2 external memory clocks as part of the SDRAM interface:

- MEM_MEMCLK
- MEM_MEMCLK

These 2 clocks are always the same frequency as XLB clock. SDR memory uses MEM_MEMCLK only; DDR memory uses both.

5.3.5 IPB Clock Domain

IPB clock can run at the same frequency as XLB clock, or 1/2 the frequency. BestComm runs at the IPB clock frequency as do all IPB control register access logic.

5.3.6 PCI Clock Domain

The PCI bus clock is the fundamental frequency of the PCI bus interface. The PCI clock can run at the XLB clock frequency, or 1/2 the XLB clock frequency, or 1/4 the XLB clock frequency, but PCI clock cannot be faster than IPB Clock.

5.4 Power Management

Power Management modes are listed below. Details are given in the sections that follow.

- Full-Power Mode
- Power Conservation Modes

The MPC5200 design is equipped with many power conservation features, which are supported in the peripherals and system logic. The 603e G2_LE Core has its own power-down modes:

- nap
- doze
- sleep

Individual peripheral functions can be disabled by stopping the module clock. In addition to clock control of individual peripheral functions, clock control sequencer (CCS) logic sequences the MPC5200 clock system to enter and exit a deep-sleep power mode. This limits power consumption to device leakage levels.

The MPC5200 system is driven by:

- a 27/33MHz system OSC, and
- a 32KHz real-time clock (RTC) OSC.

The 27/33MHz OSC drives the main clock system through a PLL that multiplies the frequency for the system buses and peripherals on the chip. The 603e G2_LE Core uses the XLB frequency as an input to the microprocessor PLL that generates the internal core frequencies.

The RTC clock domain is completely separate from the 27/33MHz clock domain, and all interactions between the RTC clock domain and any other are handled with synchronizers.

5.4.1 Full-Power Mode

In Full-Power mode both the system PLL and microprocessor PLL are locked and the main system clocks are supplied to the MPC5200 system. In this mode, the 603e G2_LE Core may use the Dynamic Power Mode (DPM). If this mode is enabled, logic not required for instruction execution, is not activated. This results in power reduction over a design that would be fully clocked during normal operation.

Performance in not decreased in Dynamic Power Mode, so it is recommended that it should never be disabled (although it is possible) when running the core at full speed.

MPC5200 peripherals can be individually enabled based on what functionality is required by the application running and the external stimulus presented to MPC5200. Peripherals not required can be powered-down through a write to an MPC5200 system control register which disables the peripheral and gates the peripheral clock.

5.4.2 Power Conservation Modes

Sleep modes in the MPC5200 design can be exercised through microprocessor sleep mode control and peripheral clock disables. In all modes except Deep-Sleep mode, the system crystal oscillator is enabled, and the system PLL and microprocessor PLL remain locked. Response time to WakeUP interrupts is faster than in the deep-sleep mode (see Section 5.4.4, *Deep-Sleep Mode*). Since clocks are still running in the MPC5200 chip, any interrupt normally present in the MPC5200 design can be used to wake up the power-down logic. See Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*, Clock Enable register.

5.4.3 603e G2_LE Core Power Modes

The 603e G2_LE Core power management modes are listed below. Details are given in the sections that follow.

- Dynamic Power Mode (default power state)
- Doze Mode
- Nap Mode
- Sleep Mode

These modes are controlled by writes to an internal 603e G2_LE Core control register. These modes only apply to the 603e G2_LE Core. Logic outside the 603e G2_LE Core remains active unless separately disabled. In any of these modes, peripherals can be enabled or disabled by writing to an MPC5200 system control register.

5.4.3.1 Dynamic Power Mode

This is the default power state mode. The core is fully powered and internal functional units are operating at the full processor clock speed. If Dynamic Mode is enabled, idle functional units automatically enter a low-power state. This does not effect:

- performance
- software execution
- external hardware

5.4.3.2 Doze Mode

All functional 603e G2_LE Core units are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in Doze Mode, any of the following actions returns the core to Full-Power Mode:

- an external asynchronous interrupt
- a system management interrupt
- a decrementer (DEC) exception
- a hard or soft reset
- a machine check input (MCP) signal

In Doze Mode, the core maintains the PLL in a fully powered state and locked to the system XLB clock input. Transition to Full-Power Mod takes only a few processor clock cycles.

5.4.3.3 Nap Mode

The Nap Mode further reduces 603e G2_LE Core power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. When in Nap Mode, any of the following actions returns the core to Full-Power Mod:

- an external asynchronous interrupt
- a system management interrupt
- a DEC exception
- a hard or soft reset
- an MCP signal

Transition to Full-Power Mode takes only a few processor clock cycles.

NOTE: It is not allowed to set the ccs_sleep_en bit of CDM Clock Control Sequencer Configuration Register before entering the nap mode. Otherwise all clocks will be disabled by entering the nap mode.

5.4.3.4 Sleep Mode

Sleep Mode reduces 603e G2_LE Core power consumption to a minimum. It does this by disabling all internal functional units.

Any of the following actions returns the core to Full-Power Mode:

- an external asynchronous interrupt
- a system management interrupt
- a hard or soft reset
- an MCP signal

In Sleep Mode it is possible to disable the 603e G2_LE Core PLL, further reducing power. this requires special sequencing logic external to the 603e G2_LE Core and is discussed in Section 5.4.4, *Deep-Sleep Mode*.

5.4.4 Deep-Sleep Mode

The MPC5200 system provides a very low power consumption mode where the 27/33MHz system oscillator, system PLL and 603e G2_LE Core PLL are shut down and disabled. Once MPC5200 is sequenced into this mode and clocks are static, the current draw of the device (except the RTC) is reduced to leakage levels. The internal state of the device is maintained in Deep Sleep as long as power is maintained.

The real-time clock (RTC) is not disabled in Deep Sleep. If the RTC is used, that portion of the chip still consumes power in Deep Sleep.

Exiting Deep Sleep mode is initiated in one of the following ways:

- An interrupt from the RTC logic
- An external asynchronous interrupt (wake up interrupt)
- An interrupt from one of the MSCAN modules (which occurs when a data transition occurs on the serial input).

The RTC clock is necessary to wake up MPC5200 using an RTC interrupt. However, no clock is required to trigger the wake up process in the case of an external interrupt or the MSCAN module interrupt. This means the RTC clock does not have to be present to use Deep Sleep

mode. The 603e G2_LE Core must enable the deep sleep process in the CDM module, then put itself into sleep mode before the 603e G2_LE Core PLL can be disabled.

Since MPC5200 clocks are stopped in Deep Sleep mode, the wake-up time is longer than in the 603e G2_LE Core-only power down modes. A power-on sequence must occur which re-locks both the MPC5200 system and processor PLLs.

The sequence of events to enter and exit Deep Sleep mode are initiated by the 603e G2_LE Core under software control and then sequenced in hardware by the Clock Control Sequencer (CCS) in CDM.

5.4.4.1 Entering Deep Sleep

When entering Deep Sleep mode, the following occurs:

- 603e G2_LE Core prepares the system for Deep Sleep power down.

  This could involve disabling peripheral interfaces, waiting for transmit/receive messages to complete, putting the SDRAM into self refresh mode, etc.
- 603e G2_LE Core finishes instructions in the execution pipeline.
- 603e G2_LE Core software enables the Deep Sleep mode with a write to a MPC5200 control register.
- 603e G2_LE Core Processor software writes sleep mode configuration to 603e G2_LE Core Processor control register.
- 603e G2_LE Core Processor asserts the QREQ signal indicating that it would like to enter sleep mode.
- CCS waits for 603e G2_LE Core Processor sleep (initiated by QREQ, since QACK is always asserted in MPC5200).
- CCS disables interrupts.
- CCS waits for the 603e G2_LE Core Processor to enter the sleep mode.
- CCS disables the OSC, system PLL, 603e G2_LE Core Processor PLL and gates the system clocks.

5.4.4.2 Exiting Deep Sleep

When exiting Deep Sleep mode, the following occurs:

- CCS receives an interrupt from a GPIO pin, RTC or a MSCAN peripheral.
- CCS enables the OSC and waits for the OSC to stabilize.
- CCS enables the system PLL and waits for the PLL to lock to the OSC clock.
- CCS enables system clocks.
- CCS enables the 603e G2_LE Core Processor PLL and waits for the PLL to lock to the system PLL clock.
- CCS enables interrupts, which triggers a wakeup interrupt to the 603e G2_LE Core Processor (from the WakeUp source).
- 603e G2_LE Core Processor wakes up and puts MPC5200 into full power mode and then services the wakeup interrupt

Waking up from Deep Sleep mode does not require the system to be reset or a boot sequence. The functional state of MPC5200 should remain the same as when it went into Deep Sleep. If the SDRAM was put into self refresh mode, its contents should also remain unchanged.

5.5 CDM Registers

The Clock Distribution Module (CDM) contains 14 32-bit registers. All registers are located at an offset from the value in the Module Base Address Register (MBAR). The CDM base offset is 0x0200.

Hyperlinks to the CDM registers are provided below:

- CDM JTAG ID Number Register—MBAR + 0x0200 (0x0200), read-only
- CDM Power On Reset Configuration Register (0x0204)
- CDM Bread Crumb Register—MBAR + 0x0208 (0x0208), never reset
- CDM Configuration Register (0x020C)
- CDM 48MHz Fractional Divider Configuration Register (0x0210)
- CDM Clock Enable Register (0x0214)
- CDM System Oscillator Configuration Register (0x0218)
- CDM Clock Control Sequencer Configuration Register (0x021C)
- CDM Soft Reset Register (0x0220)
- CDM System PLL Status Register (0x0224)
- PSC1 Mclock Config Register—MBAR + 0x0228 (0x0228)
- CDM PSC2 Mclock Config (0x022C)
- CDM PSC3 Mclock Config (0x0230)
- (0x0234)

5.5.1 CDM JTAG ID Number Register—MBAR + 0x0200

The CDM JTAG ID Number Register is a read-only register that contains the JTAG Identification number identifying MPC5200. The value is hard coded (0001101D hex) and cannot be modified.

Table 5-8. CDM JTAG ID Number Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JTAG Identification Number Register															
W	Unused															
RESET:	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 lsb
R	JTAG Identification Number Register															
W	Unused															
RESET:	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1

Device I.D. Register = 0001101D hex

Version	Device (MPC5200 – Initial Release)	Manufacturer (Freescale [formerly Motorola])	
0000	0000 0000 0001 0001	0000 0001 110	1

5.5.2 CDM Power On Reset Configuration Register—MBAR + 0x0204

This is a mostly read-only register containing the configuration value latched at POR, and the SDRAM Controller Read Clock delay tap select.

Table 5-9. CDM Power On Reset Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Write 0			tap_ select_0	tap_ select_1	tap_ select_2	tap_ select_3	tap_ select_4	Reserved, Read Only							sys_pll _bypass
W																
RESET:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	V

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	boot_ ram_lf	boot_ ram_type	boot_ ram_size	boot_ ram_swap	boot_ ram_wait	ppc_ msrip		boot_ ram_mg	sys_pll_ cfg_1	sys_pll_ cfg_0	xlb_ clk_sel	ppc_pll _cfg_0	ppc_pll _cfg_1	ppc_pll _cfg_2	ppc_pll _cfg_3	ppc_pll _cfg_4
W	Reserved, Read Only															
RESET:	V	V	V	V	V	V	—	V	V	V	V	V	V	V	V	V

Bit	Name	Description
0–2	—	Reserved for future use. Write 0.
3	tap_select_0	Indicates the delay of the internal sdram controller read clock with respect to the internal memory clock mem_clk (SDR) or $\overline{\text{mem\_2x\_clk}}$ (DDR).
4	tap_select_1	
5	tap_select_2	
6	tap_select_3	
7	tap_select_4	

Bit	Name	Description
8-14	—	Read Only. Do not write.
15	sys_pll_bypass	bit=0:Normal mode. The SYS OSC clock input is multiplied up by the system PLL, then the PLL VCO is divided down to produce internal clocks. bit=1:The SYS OSC clock input is used directly, bypassing the system PLL. No multiplication of the input frequency is performed, but the input frequency is divided to produce internal clocks just as the system PLL VCO frequency would be. sys_pll_cfg_1 and sys_pll_cfg_0 are ignored.
16	boot_rom_lf	Large Flash mode is selected
17	boot_rom_type	Latched pin value at reset. bit=0:non-muxed boot ROM bus, single tenure transfer. bit=1:muxed boot ROM bus, with address and data tenures, $\overline{\text{ALE}}$ and $\overline{\text{TS}}$ active.
18	boot_rom_size	Latched pin value at reset. For non-muxed boot ROMs: bit=0:8bit boot ROM data bus, 24bit max boot ROM address bus bit=1:16bit boot ROM data bus, 16bit boot ROM address bus For muxed boot ROMs: boot ROM address is max 25 significant bits during address tenure. bit=0:16bit ROM data bus bit=1:32bit ROM data bus
19	boot_rom_swap	Latched pin value at reset. bit=0:no byte lane swap, same endian ROM image bit=1:byte lane swap, different endian ROM image
20	boot_rom_wait	Latched pin value at reset. bit=0:4 PCI clocks of wait state bit=1:48 PCI clocks of wait state
21	ppc_msrip	Latched pin value at reset. microprocessor Boot Address/Exception table location. bit=0:0000_0100 (hex) bit=1:FFF0_0100 (hex)
22	—	Read Only. Do not write.
23	boot_rom_mg	Most/Graphic Mode is selected as BOOT mode
24	sys_pll_cfg_1	Latched pin value at reset. bit=0:No operation. bit=1:Internal System PLL frequency multiplication ratio specified by sys_pll_cfg_0 is doubled (24x, 32x). No net effect on any internal clocks, except that PLL VCO runs twice as fast. Useful in low frequency applications to keep VCO frequency (f_{vcosys}) above min, see MPC5200 Hardware Specification.
25	sys_pll_cfg_0	Latched pin value at reset. bit=0: f_{system}=16x SYS_XTAL_IN Frequency bit=1: f_{system}=12x SYS_XTAL_IN Frequency
26	xlb_clk_sel	Latched pin value at reset. bit=0:XLB_CLK= f_{system} / 4 bit=1:XLB_CLK= f_{system} / 8

Bit	Name	Description
27	ppc_pll_cfg_0	603e G2_LE Core core pll config pins. See also Table 5-6
28	ppc_pll_cfg_1	
29	ppc_pll_cfg_2	
30	ppc_pll_cfg_3	
31	ppc_pll_cfg_4	

5.5.3 CDM Bread Crumb Register—MBAR + 0x0208

The CDM Bread Crumb Register is a 32-bit register that is not reset. Its purpose is to let firmware designers leave some status code before entering a reset condition. Since this register is never reset, the value written is available after the reset condition has ended. There is no additional functionality to this register.

Table 5-10. CDM Bread Crumb Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CDM Bread Crumb Register (Never Reset)															
W																
RESET:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	CDM Bread Crumb Register (Never Reset)															
W																
RESET:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

5.5.4 CDM Configuration Register—MBAR + 0x020C

The CDM Configuration Register contains 3 bits that set IPB_CLK and PCI_CLK ratios.

Table 5-11. CDM Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Write 0							ddr_ mode	Reserved Write 0							xlb_ clk_sel
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	V

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved Write 0							ipb_ clk_sel	Reserved Write 0						pci_ clk_sel	
W																
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Bit	Name	Description
0–6	—	Reserved for future use. Write 0.
7	ddr_mode	SDRAM Controller DDR memory mode, read-only. bit=0:SDRAM Controller configured for SDR SDRAM (single data rate) bit=1:SDRAM Controller configured for DDR SDRAM (double data rate) This register location is a read-only status bit; write 0. The controlling register is in the SDRAM Controller register map. In the CDM this bit determines the frequency and phase of memory read clock.
8–14	—	Reserved for future use. Write 0.
15	xlb_clk_sel	XLB Clock Frequency bit=0:XLB CLK = f_{system} /4 bit=1:XLB CLK = f_{system} /8 This register location is a read-only status bit. The controlling register is the POR Configuration register - cdm_power_on_reset_configuration_register [26].
16–22	—	Reserved for future use. Write 0.
23	ipb_clk_sel	IPB Clock Select bit=0:IPB CLK = XLB_CLK bit=1:IPB CLK = XLB_CLK/2
24–29	—	Reserved for future use. Write 0.
30-31	pci_clk_sel	PCI Clock Select 00–PCI_CLK = IPB_CLK 01–PCI_CLK = IPB_CLK/2 10–PCI_CLK = XLB_CLK/4 See also Table 5-3 and Table 5-4.

NOTE

The clock ratio should only be changed if no module, which is clocked by the IPB and/or PCI clock, is currently running. Suggestion is to change the clock ratio during the boot time only.

5.5.5 CDM 48MHz Fractional Divider Configuration Register—MBAR + 0x0210

The CDM 48MHz Fractional Divider Configuration Register contains the control bits used in the 48MHz fractional divider.

Table 5-12. CDM 48MHz Fractional Divider Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved Write 0						ext_usb_48mhz_en	ext_irda_48mhz_en	Reserved Write 0							fd_en
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 lsb
R W	Rsrvd Write 0	cfgd_p3_cnt			Rsrvd Write 0	cfgd_p2_cntt			Rsrvd Write 0	cfgd_p1_cnt			Rsrvd Write 0	cfgd_p0_cnt		
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0–5	—	Reserved for future use. Write 0.
6	ext_usb_48MHz_en	USB External 48MHz Clock Select bit=1:USB 48MHz clock tree sourced from external clock from GPIO. bit=0:USB 48MHz clock tree sourced from CDM Fractional Divider.
7	ext_irda_48MHz_en	IrDA (PSC6) External 48MHz Clock Select bit=1:IRDA 48MHz clock tree sourced from external clock from GPIO. bit=0:IRDA 48MHz clock tree sourced from CDM Fractional Divider.
8–14	—	Reserved for future use. Write 0.
15	fd_en	CDM 48MHz Fractional Divider Enable bit=1:enable CDM Fractional Divider. bit=0:disable CDM Fractional Divider.
16	—	These fields hold 4 phase divide ratios used by the fractional divider. The fields are incompletely decoded; $f_{system}/11$ is obtained with 3 values. 110–fractional counter divide ratio $f_{system}/6$ 111–fractional counter divide ratio $f_{system}/7$ 000–fractional counter divide ratio $f_{system}/8$ 001–fractional counter divide ratio $f_{system}/9$ 010–fractional counter divide ratio $f_{system}/10$ 011–fractional counter divide ratio $f_{system}/11$ 10X–fractional counter divide ratio $f_{system}/11$
17–19	cgfd_p3_cnt[2:0]	
20	—	
21–23	cgfd_p2_cnt[2:0]	
24	—	
25–27	cgfd_p1_cnt[2:0]	
28	—	
29–31	cgfd_p0_cnt[2:0]	

5.5.6 CDM Clock Enable Register—MBAR + 0x0214

The CDM Clock Enable Register, or power management register, contains control bits that enable/disable peripheral clocks. Unused peripherals can have their clock stopped, reducing power consumption.

Table 5-13. CDM Clock Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Write 0												mem_clk_en	pci_clk_en	lpc_clk_en	slt_clk_en
W																
RESET:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	scom_clk_en	ata_clk_en	eth_clk_en	usb_clk_en	spi_clk_en	bdlc_clk_en	irrx_clk_en	irtx_clk_en	psc345_clk_en	psc2_clk_en	psc1_clk_en	psc6 clk_en	mscan_clk_en	i2c_clk_en	timer_clk_en	gpio_clk_en
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0	—	Reserved for test. Write 0.
1–11	—	Reserved for future use. Write 0.
12	mem_clk_en	Memory Clock Enable—controls SDRAM Controller module clocks Memory Controller IPB_CLK is not controlled by mem_clk_en.

Bit	Name	Description
13	pci_clk_en	PCI Bus Clock Enable—controls PCI bus control module clocks **Note:** PCI Arbiter and external PCI Bus clocks are not controlled by pci_clk_en.
14	lpc_clk_en	Local Plus Bus Clock Enable—controls LP bus control module clocks
15	slt_clk_en	Slice Timer Clock Enable—controls slice timer module clocks
16	scom_clk_en	BestComm Clock Enable—controls BestComm module clocks
17	ata_clk_en	ATA Clock Enable—controls ATA disk drive control module clocks
18	eth_clk_en	Ethernet Clock Enable—controls Ethernet Controller module clocks
19	usb_clk_en	Universal Serial Bus Clock Enable—controls USB module clock
20	spi_clk_en	SPI Clock Enable—controls SPI module clocks
21	bdlc_clk_en	BDLC Clock Enable—controls BDLC module clocks
22	irrx_clk_en	Infrared Receive Clock Enable—controls IrRx module clocks
23	irtx_clk_en	Infrared Transmit Clock Enable—controls IrTx module clocks
24	psc345_clk_en	PSC345 Clock Enable—control clock to the PSC3, PSC4 and PSC5 module
25	psc2_clk_en	PSC2 Clock Enable—control clock to the PSC2 module
26	psc1_clk_en	PSC1 Clock Enable—control clock to the PSC1 module
27	psc6_clk_en	PSC6 Clock Enable—control clock to the PSC6 module
28	mscan_clk_en	MSCAN Clock Enable—controls MSCAN module clocks
29	i2c_clk_en	I2C Clock Enable—controls I^2C module clocks
30	timer_clk_en	Timer Clock Enable—controls timer module clocks **Note:** 2 timers for wake-up mode do not have gated clocks.
31	gpio_clk_en	GPIO Clock Enable—controls some GPIO module clocks **Note:** GPIO wake-up mode circuitry uses free running IPB_CLK
Note: Enable value 1, enables the corresponding clock. Enable value 0, disables corresponding clock.		

5.5.7 CDM System Oscillator Configuration Register—MBAR + 0x0218

This register contains the System Oscillator disable bit. The system oscillator is disabled if an external clock source (not a crystal) drives the oscillator in package pin. The crystal oscillator pad cell is disabled to reduce power consumption (~6mW for system oscillator).

Table 5-14. CDM System Oscillator Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Write 0							sys_osc_ disable	Reserved Write 0							
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 lsb
R	Reserved Write 0															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0–6	—	Reserved for future use. Write 0.
7	sys_osc_disable	CDM System Oscillator Disable bit=1:System Oscillator is disabled. External clock source is required. bit=0:System Oscillator is enabled. 27–33MHz crystal is being used.
8–31	—	Reserved for future use. Write 0.

5.5.8 CDM Clock Control Sequencer Configuration Register—MBAR + 0x021C

This register contains the configuration that controls the CCS module. The CCS module lets MPC5200 enter deep sleep power down mode (all clocks stopped).

Table 5-15. CDM Clock Control Sequencer Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved Write 0							ccs_ sleep_en	Reserved Write 0							ccs_osc_ sleep_en
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 lsb
R W	Reserved Write 0															ccs_qreq _test
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	Name	Description
0–6	—	Reserved for future use. Write 0.
7	ccs_sleep_en	CCS Module Enable bit=1:CCS enabled. 603e G2_LE Core QREQ signal triggers deep sleep cycle. bit=0:CCS disabled and inactive. No deep sleep mode possible. **Note:** This bit should only be set before the processor should go into deep sleep mode. And it should be reseted after wake up. **Note:** It is not allowed to set this bit if a JTAG debugger or the nap mode should be used.
8–14	—	Reserved for future use. Write 0.
15	ccs_osc_sleep_en	CCS System Oscillator Disable Control bit=1:CCS can disable System Oscillator in deep sleep mode. bit=0:CCS cannot disable System Oscillator in deep sleep mode. Oscillator remains active.
16–30	—	Reserved for future use. Write 0.
31	ccs_qreq_test	CCS Test bit—Used in CCS module functional simulation to simulate a QREQ signal. bit=0:QREQ input to CCS forced active. bit=1:QREQ input to CCS comes directly from 603e G2_LE Core.

5.5.9 CDM Soft Reset Register—MBAR + 0x0220

This register contains 2 reset control bits.

Table 5-16. CDM Soft Reset Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							cdm_soft_reset	Reserved							cdm_no_ckstp_reset
W	Write 0								Write 0							
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W	Write 0															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0–6	—	Reserved for future use. Write 0.
7	cdm_soft_reset	CDM Soft Reset bit. bit=0: requests CDM soft reset. bit=1: CDM soft reset request inactive.
8–14	—	Reserved for future use. Write 0.
15	cdm_no_ckstp_reset	CDM No reset on checkstop. bit=0: Checkstop-in assertion causes HRESET. bit=1: Checkstop-in does not cause HRESET.
16–31	—	Reserved for future use. Write 0.

5.5.10 CDM System PLL Status Register—MBAR + 0x0224

This register contains control and status bits of the CDM PLL lock detect module.

Table 5-17. CDM System PLL Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							pll_lock	Reserved							pll_lost_lock
W	Write 0								Write 0							
RESET:	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 lsb
R	Reserved							pll_small_lock_window	Reserved							
W	Write 0								Write 0							
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	Name	Description
0–6	—	Reserved for future use. Write 0.
7	pll_lock[1]	CDM System PLL Lock Detect—read-only status bit. bit=1: CDM has detected System PLL lock condition. bit=0: CDM has NOT detected System PLL lock condition.
8–14	—	Reserved for future use. Write 0.
15	pll_lost_lock	CDM System PLL Lock Lost—hardware can only set this bit, register write must clear bit. bit=1: CDM detected loss of PLL lock after PLL lock has been achieved. bit=0: CDM has not detected loss of PLL lock (state before PLL lock occurs).
16–22	—	Reserved for future use. Write 0.
23	pll_small_ lock_window	PLL Small Lock Window—pulse width used to detect rising edge of PLL FREF clock. bit=1: lock window pulse width 2 f_{VCOsys} clock periods. bit=0: lock window pulse width 4 f_{VCOsys} clock periods.
24–31	—	Reserved for future use. Write 0.

Note:

1. System PLL Lock Condition—256 System PLL FREF clock rising edges within PLL_Lock_Window (System PLL FFB rising edge). In PLL bypass mode, Lock is active after 256 System Oscillator clock rising edges.
2. In current MPC5200 CDM the PLL Lock Circuitry is for information only. CDM does not wait for PLL lock to start clocks or use PLL_LOST_LOCK as an interrupt source.

5.5.11 PSC1 Mclock Config Register—MBAR + 0x0228

This register controls the generation of the Mclk for PSC1. Before modify the register value the divider must be disabled.

Table 5-18. CDM PSC1 Mclock Config

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved Write 0															
RESET:	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 lsb
R W	Mclock Enable	Reserved Write 0						MclkDiv[8:0]								
RESET:	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	Name	Description
0–15	—	Reserved for future use. Write 0.
16	Mclock Enable	PSC1 Mclock enable. bit=0: Turns off internally generated Mclock. bit=1: Turns on internally generated Mclock.
17-22	—	Reserved for future use. Write 0.
23-31	MclkDiv[8:0]	The counter divide the f_{system} frequency by MclkDiv+1. A vallue of 0x00 in this register turns off internally generated Mclock. For example, a value of 7 in this register, where fsystem clock is at a frequency of 528MHz would result in a Mclock rate of 528/8, or 66 MHz. **Note:** f_{system} clock is always 12 or 16 times the reference clock, sys_xtal_in, depending on sys_pll_cfg_0 at reset.

5.5.12 PSC2 Mclock Config Register—MBAR + 0x022C

This register controls the generation of the Mclock for PSC2. Before modify the register value the divider must be disabled.

Table 5-19. CDM PSC2 Mclock Config

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved Write 0															
RESET:	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 lsb
R W	Mclock Enable	Reserved Write 0						MclkDiv[8:0]								
RESET:	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	Name	Description
0–15	—	Reserved for future use. Write 0.
16	Mclock Enable	PSC2 Mclock enable. bit=0:Turns off internally generated Mclock. bit=1:Turns on internally generated Mclock.
17-22	—	Reserved for future use. Write 0.
23-31	MclkDiv[8:0]	The counter divide the f_{system} frequency by MclkDiv+1. A vallue of 0x00 in this register turns off internally generated Mclock. For example, a value of 7 in this register, where fsystem clock is at a frequency of 528MHz would result in a Mclock rate of 528/8, or 66 MHz. **Note:** f_{system} clock is always 12 or 16 times the reference clock, sys_xtal_in, depending on sys_pll_cfg_0 at reset.

5.5.13 PSC3 Mclock Config Register—MBAR + 0x0230

This register controls the generation of the Mclock for PSC3. Before modify the register value the divider must be disabled.

Table 5-20. CDM PSC3 Mclock Config

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R W	Mclock Enable	Reserved Write 0						MclkDiv[8:0]								
RESET:	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	Name	Description
0–15	—	Reserved for future use. Write 0.
16	Mclock Enable	PSC3 Mclock enable. bit=0:Turns off internally generated Mclock. bit=1:Turns on internally generated Mclock.
17-22	—	Reserved for future use. Write 0.
23-31	MclkDiv[8:0]	The counter divide the f_{system} frequency by MclkDiv+1. A vallue of 0x00 in this register turns off internally generated Mclock. For example, a value of 7 in this register, where fsystem clock is at a frequency of 528MHz would result in a Mclock rate of 528/8, or 66 MHz. **Note:** f_{system} clock is always 12 or 16 times the reference clock, sys_xtal_in, depending on sys_pll_cfg_0 at reset.

5.5.14 PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234

This register controls the generation of the Mclock for PSC6. Before modify the register value the divider must be disabled.

Table 5-21. CDM PSC6 Mclock Config

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Write 0															
W																
RESET:	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 lsb
R	Mclock Enable	Reserved Write 0						MclkDiv[8:0]								
W																
RESET:	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	Name	Description
0–15	—	Reserved for future use. Write 0.
16	Mclock Enable	PSC6 Mclock enable. bit=0:Turns off internally generated Mclock. bit=1:Turns on internally generated Mclock.
17-22	—	Reserved for future use. Write 0.
23-31	MclkDiv[8:0]	The counter divide the f_{system} frequency by MclkDiv+1. A vallue of 0x00 in this register turns off internally generated Mclock. For example, a value of 7 in this register, where fsystem clock is at a frequency of 528MHz would result in a Mclock rate of 528/8, or 66 MHz. **Note:** f_{system} clock is always 12 or 16 times the reference clock, sys_xtal_in, depending on sys_pll_cfg_0 at reset.

Chapter 6
G2_LE Processor Core

6.1 Overview

The following sections are contained in this document:

- MPC5200 G2_LE Processor Core Functional Overview
- G2_LE Core Reference Manual

6.2 MPC5200 G2_LE Processor Core Functional Overview

The MPC5200 integrates a G2_LE processor core based on, and compatible with, the 603e which is a PowerPC compliant microprocessor. The G2_LE core is completely embedded, as its address, data, and control signals are not visible external to MPC5200. The G2_LE core has the following features:

- 603e series PowerPC compliant processor core
- Dual Issue, superscalar architecture
- 16K instruction cache, 16K data cache
- Double precision FPU
- Instruction and data MMU
- Power management modes:
  - Nap
  - Doze
  - Sleep
  - Deep Sleep
- Standard & critical interrupt capability

For additional information on the capabilities and features of the G2_LE core, refer to 603e user documentation.

The G2_LE processor has a 32-bit address/64-bit data bus refered to as the 60X Local Bus (XLB). This bus is the main system connecting all internal mastering and slave modules. In addition to the G2_LE core, the USB host controller, PCI controller (as target) and BestComm controller can master the XLB.

The G2_LE core fetches 32-bit instructions (one word), two words at a time. After power-on reset, initial boot instructions are fetched from the LocalPlus bus, with CS0 active. The processor can execute code from the local bus or from the SDRAM controller. To facilitate high speed execution, boot code is typically copied from a Flash or ROM device attached to the LocalPlus bus, to SDRAM. The G2_LE core can execute code from the on-chip SRAM.

The G2_LE core has memory mapped access to all MPC5200 resources including:

- all on-chip programming registers
- all on-chip FIFOs and memories
- external SDRAM
- internal SRAM
- PCI-controlled address space
- external disk drive control register space (via PIO mode), etc.

When a master device wants access to the XLB, a request is made to the XLB Arbiter. When access is granted, the mastering device controls the XLB during the subsequent address tenure and data tenure.

Bursting is supported on the XLB. Critical Word First protocol is employed when the G2_LE core attempts to fill its address and data caches. Pipelining and cache coherency support (XLB address snooping) has been added to the MPC5200 to improve performance.

MPC5200 use the version 1.1 of the G2_LE core. The Processor version register (PVR) is 0x80822011. The G2_LE core has a System version register (SVR). The SVR numbers of MPC5200 are:

Table 6-1. SVR Values

Revision	SVR
1.0	80110010
1.1	80110011
1.2	80110012

6.3 G2_LE Core Reference Manual

A complete specification for the G2_LE core implementation used on the MPC5200 is obtained through a collection of documentation.

- PowerPC MicroprocessorFamily: The Programming Environments for 32-bit Microprocessors, Rev. 2: MPCFPE32B/AD
- G2 PowerPC Core Reference Manual, Rev. 1: G2CORERM/D

The programming environments manual provides information about resources defined by the PowerPC architecture that are common to PowerPC processors. Implementation variances relative to Rev. 2 of the Programming Environments Manual are available in the G2 Core Reference Manual.

The G2 Core Reference Manual can be obtained from the Freescale (formerly Motorola) Literature Distribution center at http://e-www.freescale.com. Click on the Dovumentation link to proceed to the Semiconductor Documentation Library. In the documentation form window, select “Reference Manual” and set the matching pages option button to “All”. An alphabetical list of refernce manuals will appear and the G2 core document ID is ‘G2CORERM/D’. From this line entry, you may order hard copies of the G2 Core Reference Manual or download a PDF copy of the manual.

6.4 Not supported G2_LE Core Feature

6.4.1 Not supported instruction

The G2_LE core supports two instructions that are not supported by the MPC5200. This two instructions are **eciowx** and **ecowx**. The execution of this two instruction will generate a $\overline{\text{TEA}}$ signal on XLB. This will cause a machine check exception or a checkstop.

6.4.2 Not supported XLB parity feature

The G2_LE core supports a address and data parity error detection for the XL bus. This feature is not supported by the MPC5200. The core input signals core_ap_in [0:3] are pulled-down to 0 and the core input signals core_dp_in [0:7] are pulled-up to 1. Enabling of the address or data parity error check by the HID0 [EBA, EBD] bits will generate a machine check exception or a checkstop depending on the HID0 [EMCP] bit.

Chapter 7
System Integration Unit (SIU)

7.1 Overview

The following sections are contained in this document:

- Interrupt Controller, includes:
  - Interrupt Controller Registers
- General Purpose I/O (GPIO), includes:
  - GPIO Standard Registers—MBAR+0x0B00
  - WakeUp GPIO Registers—MBAR+0x0C00
- General Purpose Timers (GPT), includes:
  - GPT Registers—MBAR + 0x0600
- Slice Timers, includes:
  - SLT Registers—MBAR + 0x0700
- Real-Time Clock, includes:
  - RTC Interface Registers—MBAR + 0x0800

NOTE

Watchdog timer functions are included in the GPT section.

The System Integration Unit (SIU) controls and support the functions listed above.

7.2 Interrupt Controller

A highly configurable Interrupt Controller directs all interrupt sources to the following 3 603e core interrupt pins:

- core_cint -- critical interrupt
- core_smi -- system management interrupt
- core_int -- standard interrupt

7.2.1 Block Description

The Interrupt Controller MUXes a variety of interrupt sources to the limited interrupt pins on the core. The interrupt sources and their descriptions are summarized in Table 7-1.

Table 7-1. Interrupt Sources

Source	No.	Description
External IRQ Interrupts	4	Can be programmed as level or edge sensitive. Provides interrupt requests to Interrupt Controller for external devices.
Slice Timers	2	“Tick” generators. Suitable for operating system update tick.
General Timers	8	Generates interrupt in Input Capture mode or Internal Timer mode. Timers 6 and 7 can interrupt from NAP/DOZE power-down.
BestComm and Peripherals	19	Various peripherals are priority programmed and encoded into HI or LO interrupt to the Interrupt Controller. BestComm Controller interrupt is connected to HI interrupt.
RTC	2	Stopwatch and periodic
WakeUp	8	These are special GPIO pins with WakeUP capability. There are 8 such pins funneled into one interrupt. The source module is gpio_wkup.
GPIO	8	GPIO pins with simple interrupt capability (not available in power down mode). The source module is gpio_std.
WatchDog Timer	0	No vector handler, generates $\overline{\text{SRESET}}$ output indication.
Total	51	

Table 7-1 does not include machine-check bus errors or transaction handshaking. Core interrupt pins given in Section 7.2.1.1, *Machine Check Pin—core_mcp* through Section 7.2.1.3, *Standard Interrupt—core_int* show core interrupt priority.

7.2.1.1 Machine Check Pin—core_mcp

NOTE

The core_mcp pin is not used. Bus errors occur on the XL bus, thus generating an internal machine-check exception, or are reflected as a normal interrupt from the offending source module.

Internally, bus errors (TEA, APE, DPE, etc.) cause a machine check exception to a single exception vector. This pin allows additional, external to the core, interrupts of the same type, but is not connected in this device.

7.2.1.2 System Management Interrupt—core_smi

The core_smi is a core pin for high priority interrupts. Table 7-2 defines the interrupts.

Table 7-2. System Management Interrupt Pin Interrupts

Interrupt	Description
Enables	The MSR[ee] bit must be set to enable interrupts at this core pin. The MSR[ee] bit is automatically cleared when an interrupt occurs. Therefore, the exception handler must re-set this bit when interrupt is cleared.
Recovery/Status	Recovery is highly dependant on system and software design. Where multiple sources are tied to the same interrupt, a status register is provided to distinguish the interrupting source.
Timing	Assertion of this interrupt is persistent (i.e., interrupt remains until cleared). If other interrupts are pending when first interrupt is cleared, the core_smi pin should remain asserted for handling once the current exception handler re-sets the MSR[ee] bit.
Connections	Standard external and internal interrupts can be connected to this high priority interrupt. Slice timer 1 is a dedicated connection.

7.2.1.3 Standard Interrupt—core_int

Identical to core_smi, but of lower priority. This interrupt is shared by a variety of internal low priority interrupts such as GPIO and RTC functions. Some programmable connection are provided. Table 7-3 gives a summary of the interrupt pins. Figure 7-2 shows the interrupt sources and core pins.

Table 7-3. Core Interrupt Pins Summary

Pin	Description	Sources	To Enable	Timing
core_mcp	Machine Check Pin	Tied inactive	—	—
core_cint	Critical Interrupt	BestComm HI, IRQ0, Slice Timer 0, CCS WakeUp	MSR[ce]	Persistent (remains until cleared)
core_smi	System Management Interrupt	Slice Timer 1, Programmable interrupts	MSR[ee]	Persistent
core_int	Standard Interrupt	Programmable interrupts	MSR[ee]	Persistent

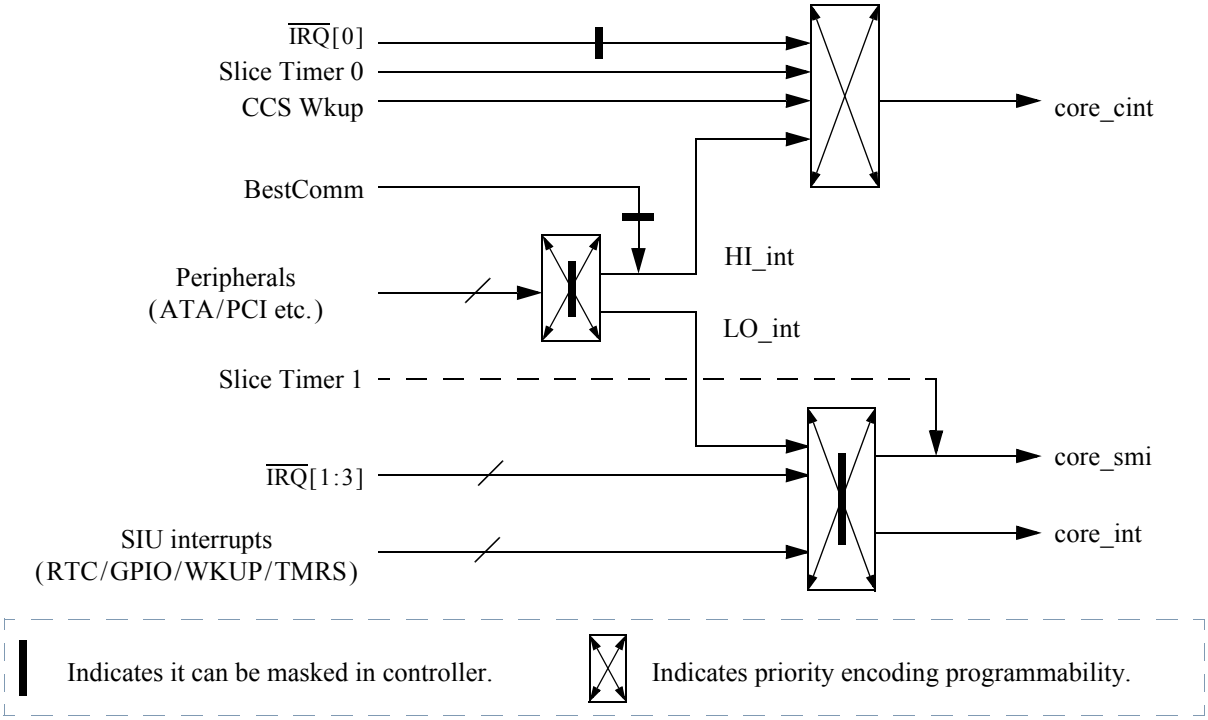


Figure 7-1. Interrupt Sources and Core Interrupt Pins

$\overline{\text{IRQ}}$[0:3] Interrupt Requests

$\overline{\text{IRQ}}$[0:3] provides interrupt requests to Interrupt Controllers for external devices such as:

- graphics controllers
- PCI interrupt controller
- ATAs
- transport de-multiplexers
- external I/O devices, etc.

These interrupts are programmable as edge or level sensitive. See Figure 7-1.

7.2.2 Interface Description

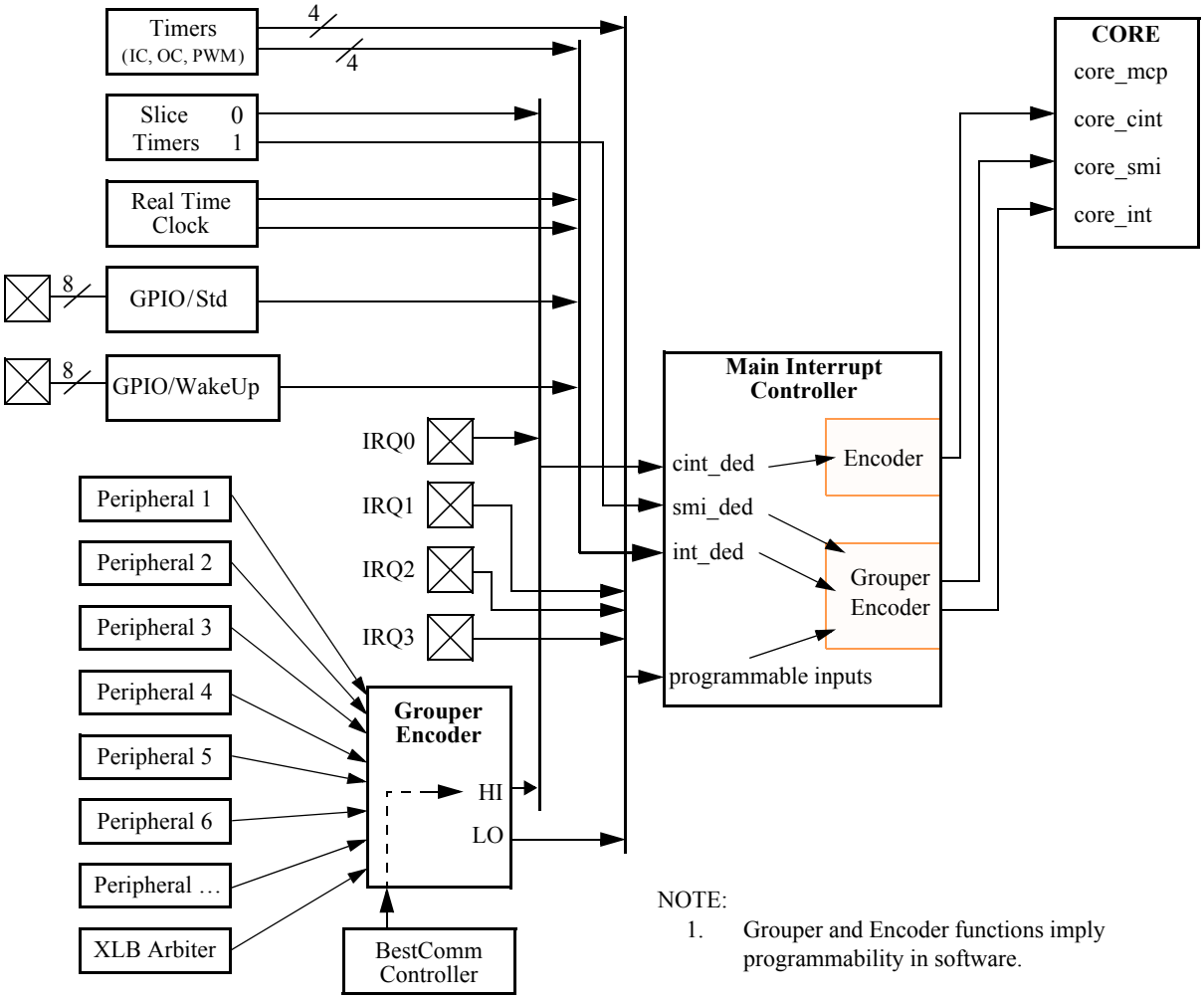


Figure 7-2. Interrupt Controller Routing Scheme

7.2.3 Programming Note

Under specific conditions, the Interrupt Controller may not support nested interrupts. The Interrupt Controller may prevent the assertion of a core_cint interrupt if a core_int or a core_smi is pending. Similarly, the Interrupt Controller may block a core_smi if a core_int is pending. If the 603e core received the core_cint assertion during an core_int or core_smi assertion, it would preempt the current interrupt service routine and process the Critical Interrupt Service routine immediately. Since the MPC5200 Interrupt Controller postpones the core_cint assertion until after a current core_int or core_smi is finished, there can be a delay before the 603e receives and services Critical Interrupt Sources.

The interrupt Controller always supports nested interrupt if the Critical Interrupt sources come from IRQ0, Slice TImer 0 or the wakeup logic. There is a difference when the critical source comes from HI_int (Peripheral Interrupt Group). As shown in Figure 7-2, each Peripheral Interrupt can assert a HI_int or LO_int condition. But only one Peripheral Interrupt can be active at the time, so the Interrupt Controller has not the ability to simultaneously assert both HI_int and LO_int. Therefore, the peripheral 2 which generated by default a core_int (LO_int) interrupt will prevent a BestComm Interrupt to generate a core_cint (HI_int) interrupt.

In addition, a Peripheral Interrupt directed to a core_cint can be prevented by a pending core_smi interrupt. Each Peripheral Interrupt (LO_int) can be programmed to cause a core_smi by setting the Main4_pri msb. Once again, the Interrupt Controller does not has the ability to simultaneously generate the HI_int and LO_int. Then, a Peripheral Interrupt, which generates a core_smi, prevents any Peripheral Interrupts to assert a core_cint.

Similarly, the Interrupt Controller can activate only one Main Interrupt source at the time. Main4 source is the collection of all LO_int Peripheral Interrupts. The Main4_pri can be programmed in order to generate a core_smi or a core_int. As result, a Peripheral Interrupt that causes a core_int will prevent all other Main Interrupt sources to generate a core_smi. Although the Interrupt Controller does not exhibit the

correct behavior, the 603e core always completes the core_int before treating the core_smi. In this case, the CPU does not authorize nested interrupt at the exception if the ISR set the 603e's MSR[EE] to support nested interrupt (core_smi and core_int).

In order to guaranty the assertion of the core_cint when a core_int is pending, the ISR needs to force the re-evaluation of the Peripheral Interrupt condition by writing "1" to the Peripheral Status Encoded Pse msb. The ISR has to repeatedly set this bit since the interrupt events are indeterministic. Moreover, the Peripheral Interrupt sources directed to core_cint needs to have their priorities to be higher than the LO_int Peripheral Interrupt sources. The Interrupt Controller always activates first the pending interrupt having the highest priority. Like for the Peripheral Interrupt Group, the ISR needs to set the Main Status Encoded MSe msb to force re-evaluation of the Main Interrupt Condition and each Main Interrupt Priority needs to be properly programmed.

7.2.4 Interrupt Controller Registers

The Interrupt Controller uses 13 32-bit registers. These registers are located at an offset from MBAR of 0x0500. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0500 + register address**

Hyperlinks to the Interrupt Controller registers are provided below:

- ICTL Peripheral Interrupt Mask Register (0x0500)
- ICTL Peripheral Priority and HI/LO Select 1 Register (0x0504)
- ICTL Peripheral Priority and HI/LO Select 2 Register (0x0508)
- ICTL Peripheral Priority and HI/LO Select 3 Register (0x050C)
- ICTL External Enable and External Types Register (0x0510)
- ICTL Critical Priority and Main Interrupt Mask Register) (0x0514)
- ICTL Main Interrupt Priority and INT/SMI Select 1 Register (0x0518)
- ICTL Main Interrupt Priority and INT/SMI Select 2 Register (0x051C)
- ICTL PerStat, MainStat, CritStat Encoded Register (0x0524)
- ICTL Critical Interrupt Status All Register (0x0528)
- ICTL Main Interrupt Status All Register (0x052C)
- ICTL Peripheral Interrupt Status All Register (0x0530)
- ICTL Bus Error Status Register (0x0538)
- ICTL Main Interrupt Emulation All Register (0x0540)
- ICTL Peripheral Interrupt Emulation All Register (0x0544)
- ICTL IRQ Interrupt Emulation All Register (0x0544)

7.2.4.1 ICTL Peripheral Interrupt Mask Register—MBAR + 0x0500

Table 7-4. ICTL Peripheral Interrupt Mask Register

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Per_mask								Reserved							
W																
RESET:	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bits	Name	Description
—	Per_mask	Bits 0:23—To mask/accept individual peripheral interrupt sources. This masking is in addition to interrupt enables, which may exist in each source module. 0=Accept interrupt from source module. 1=Ignore interrupt from source module. Important—See Note 1.
0	Per_mask	BestComm interrupt source
1	Per_mask	Peripheral 1 (PSC1)
2	Per_mask	Peripheral 2 (PSC2)
3	Per_mask	Peripheral 3 (PSC3)
4	Per_mask	Peripheral 4 (PSC6)
5	Per_mask	Peripheral 5 (Ethernet)
6	Per_mask	Peripheral 6 (USB)
7	Per_mask	Peripheral 7 (ATA)
8	Per_mask	Peripheral 8 (PCI Control module)
9	Per_mask	Peripheral 9 (PCI SC Initiator RX)
10	Per_mask	Peripheral 10 (PCI SC Initiator TX)
11	Per_mask	Peripheral 11 (PSC4)
12	Per_mask	Peripheral 12 (PSC5)
13	Per_mask	Peripheral 13 (SPI modf)
14	Per_mask	Peripheral 14 (SPI spif)
15	Per_mask	Peripheral 15 (I2C1)
16	Per_mask	Peripheral 16 (I2C2)
17	Per_mask	Peripheral 17 (CAN1)
18	Per_mask	Peripheral 18 (CAN2)
19	Per_mask	Reserved
20	Per_mask	Reserved
21	Per_mask	Peripheral 21 (XLB Arbiter)
22	Per_mask	Peripheral 22 (BDLC)
23	Per_mask	Peripheral 23 (BestComm LocalPlus)
24:31	—	Reserved

Note:

1. Setting these bits prevents an interrupt being presented to the core pins for the masked sources. Encoded status indications in the ICTL Perstat, MainStat, CritiStat Encoded Register are suppressed, but the binary "all" status bits (PSa in ICTL Peripheral Interrupt Status All Register) are active as long as the source module is presenting an active input to the Interrupt Controller.

7.2.4.2 ICTL Peripheral Priority and HI/LO Select 1 Register —MBAR + 0x0504

Table 7-5. ICTL Peripheral Priority and HI/LO Select 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Per0_pri				Per1_pri				Per2_pri				Per3_pri			
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 lsb
R	Per4_pri				Per5_pri				Per6_pri				Per7_pri			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
—	Per[x]_pri	Priority encoding is done using 4 configuration bits per input source. Each group of 4 bits controls the source priority in relation to other peripheral sources. The most significant bit (msb) of each config nibble is called the HI/LO or "bank" bit. If this bit is high it implies not only a high priority, but causes this interrupt source to assert a HI interrupt condition. Under most circumstances this creates a Critical Interrupt assertion to the core. See Note 1. Peripherals with identical priority settings (either zero or non-zero) are default prioritized with "lower peripheral has higher priority". In other words, Per1 has a default priority higher than Per2.
0:3	Per0_pri	Peripheral 0 = BestComm interrupt (fixed as highest peripheral)
4:7	Per1_pri	Peripheral 1 = PSC1 interrupt source
8:11	Per2_pri	Peripheral 2 = PSC2
12:15	Per3_pri	Peripheral 3 = PSC3
16:19	Per4_pri	Peripheral 4 = PSC6
20:23	Per5_pri	Peripheral 5 = Ethernet
24:27	Per6_pri	Peripheral 6 = USB
28:31	Per7_pri	Peripheral 7 = ATA

Note:

1. Per0_pri, associated with the BestComm interrupt source, is not programmable and always has the highest peripheral priority and always results in a HI interrupt condition to the Interrupt Controller. These bits are writable and readable, but have no effect on controller operation.

7.2.4.3 ICTL Peripheral Priority and HI/LO Select 2 Register —MBAR + 0x0508

Table 7-6. ICTL Peripheral Priority and HI/LO Select 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Per8_pri				Per9_pri				Per10_pri				Per11_pri			
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 lsb
R	Per12_pri				Per13_pri				Per14_pri				Per15_pri			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
—	Per[x]_pri	Identical to Peripheral_Priority 1 Register, but related to peripheral interrupt sources 8 through 15. All bits are programmable and significant.
0:3	Per8_pri	Peripheral 8 = PCI Control module
4:7	Per9_pri	Peripheral 9 = PCI SC Initiator RX
8:11	Per10_pri	Peripheral 10 = PCI SC Initiator TX
12:15	Per11_pri	Peripheral 11 = PSC4
16:19	Per12_pri	Peripheral 12 = PSC5
20:23	Per13_pri	Peripheral 13 = SPI modf
24:27	Per14_pri	Peripheral 14 = SPI spif
28:31	Per15_pri	Peripheral 15 = I2C1

7.2.4.4 ICTL Peripheral Priority and HI/LO Select 3 Register —MBAR + 0x050C

Table 7-7. ICTL Peripheral Priority and HI/LO Select 3 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Per16_pri				Per17_pri				Per18_pri				Per19_pri			
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 lsb
R	Per20_pri				Per21_pri				Per22_pri				Per23_pri			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
—	Per[x]_pri	Identical to Peripheral_Priority 2 register, but related to peripheral interrupt sources 16–21. All bits are programmable and significant.
0:3	Per16_pri	Peripheral 16 = I2C2
4:7	Per17_pri	Peripheral 17 = CAN1

Bits	Name	Description
8:11	Per18_pri	Peripheral 18 = CAN2
12:15	Per19_pri	Reserved
16:19	Per20_pri	Reserved
20:23	Per21_pri	Peripheral 21 = XLB Arbiter
24 :27	Per22_pri	Peripheral 22 = BDLC
28 :31	Per23_pri	Peripheral 23 = BestComm LocalPlus

7.2.4.5 ICTL External Enable and External Types Register —MBAR + 0x0510

Table 7-8. ICTL External Enable and External Types Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				ECLR(4)				Etype0		Etype1		Etype2		Etype3	
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 lsb
R	Reserved			MEE	EENA(4)				Reserved							CEb
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	—	Reserved
—	ECLR[x]	These bits clear external $\overline{\text{IRQ}}$ interrupt indications. When an $\overline{\text{IRQ}}$ input is configured as an edge-sensitive input, the Interrupt Controller must be notified that the specific interrupt has been serviced. Software must write 1 to the appropriate bit position to clear the interrupt indication. ECLR bits are always read as 0 (i.e., they do not contain status).
4	ECLR0	$\overline{\text{IRQ}}$[0], write 1 to clear
5	ECLR1	$\overline{\text{IRQ}}$[1], write 1 to clear
6	ECLR2	$\overline{\text{IRQ}}$[2], write 1 to clear
7	ECLR3	$\overline{\text{IRQ}}$[3], write 1 to clear
8:9	Etype0	These bits control how the Interrupt Controller interprets the $\overline{\text{IRQ}}$[0] input pin. 00 = Input is level sensitive and active hi 01 = Input is edge sensitive, rising edge active” 10 = Input is edge sensitive, falling edge active” 11 = Input is level sensitive, and active low”
10:11	Etype1	Same as above, but for the $\overline{\text{IRQ}}$[1] input pin.
12:13	Etype2	Same as above, but for the $\overline{\text{IRQ}}$[2] input pin.
14:15	Etype3	Same as above, but for the $\overline{\text{IRQ}}$[3] input pin.
16:18	—	Reserved—unused bits, writing has no effect, always read as 0.
19	MEE	Master External Enable—clearing this bit masks all $\overline{\text{IRQ}}$ input transitions (including status indications).

Bits	Name	Description
—	EENA[x]	Individual enable bits for each $\overline{IRQ}$ input pin. Setting the associated bit lets the related $\overline{IRQ}$ pin generate interrupts. In either case, status indications in PSa and CSa (ICTL Peripheral Interrupt Status All Register) are active.
20	EENA0	$\overline{IRQ}$[0]
21	EENA1	$\overline{IRQ}$[1]
22	EENA2	$\overline{IRQ}$[2]
23	EENA3	$\overline{IRQ}$[3]
24:30	—	Reserved
31	CEb	Critical Enable—a special control bit, which if set, directs critical interrupt sources to the normal core Interrupt pin. This is for system programmer who prefers to handle all interrupts in a single ISR. The status operation remains unchanged, it is necessary to parse Critical Status information prior to Normal Status information to detect critical interrupt sources routed to the normal interrupt pin.

7.2.4.6 ICTL Critical Priority and Main Interrupt Mask Register—MBAR + 0x0514

Table 7-9. ICTL Critical Priority and Main Interrupt Mask Register)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Crit0_pri		Crit1_pri		Crit2_pri		Crit3_pri		Reserved							Main_ Mask
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 lsb
R	Main_Mask															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	Crit0_pri	Priority encoding value for Critical Interrupt 0, $\overline{IRQ}$[0] input pin. There are four Critical Interrupt sources that can be uniquely prioritized (a higher Priority value creates a higher priority, i.e. a value of 3 is the highest priority value). In the case of identical priority value, the lower numbered interrupt source has priority. This makes $\overline{IRQ}$[0] the highest default priority (being the lowest numbered source).
2:3	Crit1_pri	Priority encoding value for Slice Timer 0 interrupt source. Hard-wired as critical interrupt source number 1, it has the second highest default priority.
4:5	Crit2_pri	Priority encoding value for HI_int interrupt source. Hard-wired as critical interrupt source number 2. It is programmable such that any peripheral source can be directed to it, and thus get maximum priority service.
6:7	Crit3_pri	Priority encoding value for CCS WakeUp source. Hard-wired as critical interrupt source number 3.
8:14	—	Reserved

Bits	Name	Description
—	Main_Mask[x]	To mask/accept individual main interrupt sources (as opposed to peripheral or critical interrupt sources). This masking is in addition to interrupt enables, which may exist in each source module. 0=Default. Accept interrupt from source module. 1=Ignore interrupt from source module. Take care if masking LO_int, which is a collection of multiple Peripheral sources in a single presentation. Masking LO_int essentially prevents any LO Peripheral from generating an interrupt, even when those interrupts are enabled (i.e., unmasked) in Per_Mask, Reg0. **Important**—See Note 1.
15	Main_Mask0	Slice Timer 1, which is hardwired to SMI interrupt output. See Note 2.
—	—	Interrupt sources below are bank/priority programmable (in Reg6 and Reg7).
16	Main_Mask1	$\overline{IRQ}$[1] ($\overline{IRQ}$[1] input pin interrupt)
17	Main_Mask2	$\overline{IRQ}$[2] ($\overline{IRQ}$[2] input pin interrupt)
18	Main_Mask3	$\overline{IRQ}$[3] ($\overline{IRQ}$[3] input pin interrupt)
19	Main_Mask4	LO_int (source programmable from Peripheral ints)
20	Main_Mask5	RTC_pint (Real time clock, periodic interrupt)
21	Main_Mask6	RTC_sint (Real time clock, stopwatch and alarm interrupt)
22	Main_Mask7	GPIO_std (collected GPIO interrupts, non-WakeUp)
23	Main_Mask8	GPIO_wkup (collected WakeUp interrupts)
24	Main_Mask9	TMR0 (internal Timer resource)
25	Main_Mask10	TMR1 (internal Timer resource)
26	Main_Mask11	TMR2 (internal Timer resource)
27	Main_Mask12	TMR3 (internal Timer resource)
28	Main_Mask13	TMR4 (internal Timer resource)
29	Main_Mask14	TMR5 (internal Timer resource)
30	Main_Mask15	TMR6 (internal Timer resource)
31	Main_Mask16	TMR7 (internal Timer resource)

Note:

1. Setting these bits prevents an interrupt being presented to the masked sources core pins. Encoded status indications (MSe in Reg9) are therefore suppressed, but the binary all status bits (MSa in RegB) are active as long as the source module is presenting an active input to the Interrupt Controller. Masking $\overline{IRQ}$[1:3], is redundant with External ENA bits in Reg4, but both masks are applied.
2. Slice Timer 1 is hard-coded and neither bank nor priority adjustable.

7.2.4.7 ICTL Main Interrupt Priority and INT/SMI Select 1 Register —MBAR + 0x0518

Table 7-10. ICTL Main Interrupt Priority and INT/SMI Select 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Main1_pri				Main2_pri				Main3_pri				Main4_pri			
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 lsb
R	Main5_pri				Main6_pri				Main7_pri				Main8_pri			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	Main1_pri	Main interrupt source 1 ($\overline{IRQ}$[1]) priority encoding value. All four bits are used to set a priority value (higher value equals higher priority). MSbit is also used as a bank bit to direct this interrupt source to SMI interrupt output (if bank = 1), or to normal INT interrupt output (if bank = 0). For interrupt sources set at the same priority value, default priority is the lower numbered interrupt has higher priority. This means main source 1 has a higher default priority than main source 2. See Note 1.
4:7	Main2_pri	Main interrupt source 2 ($\overline{IRQ}$[2] input pin) priority encoding value.
8:11	Main3_pri	Main interrupt source 3 ($\overline{IRQ}$[3] input pin) priority encoding value.
12:15	Main4_pri	Main interrupt source 4 (LO_int) priority encoding value. LO_int is a collection of any Peripheral Interrupts directed to this interrupt source. Peripheral interrupts sources are directed to either LO_int, or to the critical interrupt source HI_int.
16:19	Main5_pri	Main interrupt source 5 (RTC_periodic) priority encoding value.
20:23	Main6_pri	Main interrupt source 6 (RTC_stopwatch and RTC_alarm) priority encoding value.
24:27	Main7_pri	Main interrupt source 7 (GPIO_std) priority encoding value. GPIO_std is a collection of all simple interrupt GPIO pins enabled for Interrupt operation.
28:31	Main8_pri	Main Interrupt source 8 (GPIO_wkup) priority encoding value. GPIO_wkup is a collection of all enabled WakeUp capable GPIO sources. WakeUp interrupt sources also operate in normal powered-up modes so all GPIO interrupt sources are represented by main interrupt sources 7 and 8 (also see Timer GPIOs in Reg7).

Note:

1. Main source 0 (Slice Timer 1) is not listed, it is fixed as both the highest priority main interrupt and to generate an SMI interrupt output only.

7.2.4.8 ICTL Main Interrupt Priority and INT/SMI Select 2 Register—MBAR + 0x051C

Table 7-11. ICTL Main Interrupt Priority and INT/SMI Select 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Main9_pri				Main10_pri				Main11_pri				Main12_pri			
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 lsb
R	Main13_pri				Main14_pri				Main15_pri				Main16_pri			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	Main9_pri	Main interrupt source 9 (TMR0) priority encoding value. All 4 bits are used to set a priority value (higher value equals higher priority). The msb is also used as a bank bit to direct this interrupt source to SMI interrupt output (if bank = 1), or to normal INT interrupt output (if bank = 0). For interrupt sources set at the same priority value, default priority is the lower numbered interrupt has higher priority. This means main source 9 has a higher default priority than main source 10. Timer 0 is one of eight internal timer resources that can be configured as input capture, output compare, or PWM output. As such, there is an I/O pin associated with each timer. The timer can use this pin as GPIO, in which case the internal timer function becomes available. These eight timers complete the MPC5200 GPIO structure. All potential GPIO interrupt sources are represented by main sources 7, 8, and 9–16.
4:7	Main10_pri	Main interrupt source 10 (TMR1) priority encoding value.
8:11	Main11_pri	Main interrupt source 11 (TMR2) priority encoding value.
12:15	Main12_pri	Main interrupt source 12 (TMR3) priority encoding value.
16:19	Main13_pri	Main interrupt source 13 (TMR4) priority encoding value.
20:23	Main14_pri	Main interrupt source 14 (TMR5) priority encoding value.
24:27	Main15_pri	Main interrupt source 15 (TMR6) priority encoding value. See Note 1.
28:31	Main16_pri	Main interrupt source 16 (TMR7) priority encoding value. See Note 1.

Note:

1. This timer has WakeUp functionality and therefore can provide a WakeUp interrupt source.

7.2.4.9 ICTL Perstat, MainStat, MainStat, CritStat Encoded Register—MBAR + 0x0524

Table 7-12. ICTL PerStat, MainStat, CritStat Encoded Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		PSe						Reserved		MSe					
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 lsb
R	Reserved					CSe			Reserved							CEbSh
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	—	Reserved
2:7	PSe	Peripheral Status Encoded—makes a singular indication of the current peripheral interrupt (6bits indicating 1 of 24 possible peripheral interrupts). The msb operates as a flag bit and is set if any peripheral interrupt is currently being presented by the Interrupt Controller (e.g., if peripheral interrupt source 0 is current, then this register reads as 0x20). Normally it would not be necessary to clear this status register since all peripheral interrupt sources are level sensitive. Once an interrupt source negates at the input of the controller, the new input condition is re-evaluated without software intervention. However, if ISR does not clear the interrupt source (at the source module), then the controller is locked on the current interrupt and cannot re-evaluate the input condition (possibly to detect the presence of a higher priority interrupt). Therefore, ISR can force a re-evaluation of the input condition by writing 1 to the msb of PSe. This sticky-bit clear operation is optional and can be used at the discretion of the ISR writer. The encoded value cross-reference to a specific source is described in ICTL Peripheral Interrupt Mask Register and re-stated in ICTL Peripheral Interrupt Status All Register. In all cases, the peripheral status encoded value converts to a single source module (i.e., no additional status parsing is required at the Interrupt Controller).
8:9	—	Reserved
10:15	MSe	Main Status Encoded—makes a singular indication of the current main interrupt (6 bits indicating 1 of 17 possible main interrupts). The msb operates as a flag bit, as described above. The msb can also be written to 1 to force a re-evaluation of the main interrupt sources. The cross-reference of the encoded value to a particular source is described in Reg5 (main mask) and re-stated in ICTL Main Status All Register. All MSe values convert to a single source module, EXCEPT Main source 4 (LO_int), which indicates a peripheral source is active. In this case it is necessary to parse the PSe to determine which peripheral source is active. See Note 1.
16:20	—	Reserved

Bits	Name	Description
21:23	CSe	Critical Status Encoded—makes a singular indication of the current critical interrupt (3bits indicating 1 of 4 possible interrupts). The msb operates as a Flag bit, as described above. This msb can also be written to 1 to force a re-evaluation of the critical interrupt sources. 00 = $\overline{IRQ}$ input pin is the source. See Note 2. 01 = Slice Timer 0 is the source. 10 = HI_int is the source. See Note 3. 11 = CCS module is the source. WakeUp from deep-sleep. See Note 4.
24:30	—	Reserved
31	CEbSh	Critical Enable bar Shadow bit—this is a special bit that shadows the setting programmed into ICTL External Enable and External Types Register. This bit indicates whether Critical interrupt sources have or have not been directed to the normal INT core pin. If Critical interrupts are directed to INT (CEbSh = 1), to detect higher priority interrupt sources, INT ISR must always parse the CSe prior to MSe or PSe. All other processing remains the same. This shadow bit is provided here so a single read to this register can obtain all necessary information to make the interrupt source determination.

Note:

1. For Main sources 1, 2, and 3 that represent $\overline{IRQ}$[1:3] respectively, if the $\overline{IRQ}$ pin is set as edge sensitive, it is REQUIRED that the MSe flag bit be cleared (i.e., written to 1) or the appropriate ECLR bit in ICTL External Enable and External Types Register be set to clear this interrupt indication. Only one method should be used, not both (this limit is only true for multiple edge-sensitive $\overline{IRQ}$ inputs).
2. For $\overline{IRQ}$[0] set as edge sensitive, it is REQUIRED that either the CSe flag bit be cleared (i.e., written to 1) or the ECLR[0] bit in ICTL External Enable and External Types Register be set to clear this interrupt indication. You can do both if desired, and you can do it regardless of the $\overline{IRQ}$[0] interrupt type.
3. This indicates a peripheral source programmed for HI bank priority is the source. It is necessary to parse the PSe value to determine the peripheral source module.
4. For recovery from deep-sleep mode, it is necessary to acknowledge this WakeUp interrupt by writing 1 to the msb of this field (CSe). Only then does the CCS module release it's power-down internal signal and let MPC5200 operate normally.

7.2.4.10 ICTL Critical Interrupt Status All Register—MBAR + 0x0528

Table 7-13. ICTL Critical Interrupt Status All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				CSa				Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	—	Reserved
—	CSa[x]	Critical Interrupt Status All—Indicates all pending interrupts, including the currently active interrupt (if any). CSa is binary, showing each active interrupt input in its corresponding bit position. See Note 1. Number in parenthesis indicates equivalent encoded value in CSe, ICTL PerStat, MainStat, CritStat Encoded Register.
4	CSa0	indicates $\overline{\text{IRQ}}$[0] interrupt
5	CSa1	Slice Timer 0 interrupt
6	CSa2	HI_int interrupt
7	CSa3	WakeUp from deep-sleep mode (CCS) interrupt
8:31	—	Reserved

Note:

1. No direct mask register is defined for critical interrupts. However, $\overline{\text{IRQ}}$[0] can be masked by the MEE bit in Reg4, in which case CSa status does not occur. If only the EENA[0] bit in ICTL External Enable and External Types Register is cleared, then CSa status occurs, but controller does not assert a core interrupt.

7.2.4.11 ICTL Main Interrupt Status All Register—MBAR + 0x052C

Table 7-14. ICTL Main Interrupt Status All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															MSa
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 lsb
R	MSa															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:14	—	Reserved
—	MSa[x]	Main Interrupt Status All. Indicates all pending interrupts. Is binary, showing each active interrupt in its corresponding bit position. See Note 1. Number in parenthesis indicates equivalent encoded value in MSe, Reg9.
15	MSa0	Slice_Timer 1 (SMI interrupt only)
16	MSa1	$\overline{\text{IRQ}}$[1] input pin
17	MSa2	$\overline{\text{IRQ}}$[2] input pin
18	MSa3	$\overline{\text{IRQ}}$[3] input pin
19	MSa4	LO_int (some Peripheral source)
20	MSa5	RTC_periodic interrupt
21	MSa6	RTC_stopwatch interrupt
22	MSa7	GPIO std interrupt

Bits	Name	Description
23	MSa8	GPIO WakeUp interrupt
24	MSa9	TMR0 interrupt
25	MSa10	TMR1 interrupt
26	MSa11	TMR2 interrupt
27	MSa12	TMR3 interrupt
28	MSa13	TMR4 interrupt
29	MSa14	TMR5 interrupt
30	MSa15	TMR6 interrupt
31	MSa16	TMR7 interrupt

Note:

1. All main interrupt sources are directly maskable in Main_Mask, ICTL Critical Priority and Main Interrupt Mask Register. If masked in Main_Mask, status information still shows in MSa. However, if interrupt is not enabled at the source module (i.e., in source module registers) the Interrupt Controller cannot observe or record status information for that interrupt.

7.2.4.12 ICTL Peripheral Interrupt Status All Register—MBAR + 0x0530

Table 7-15. ICTL Peripheral Interrupt Status All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								PSa							
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 lsb
R	PSa													Reserved		PSa21
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	—	Reserved
—	PSa[x]	Peripheral Interrupt Status All. Indicates all pending interrupts. Is binary, showing each active interrupt in its corresponding bit position. See Note 1. Number in parenthesis indicates equivalent encoded value in PSe, ICTL PerStat, MainStat, CritStat Encoded Register.
8	PSa23	BestComm LocalPlus
9	PSa22	BDLC
10	PSa0	BestComm interrupt source
11	PSa1	PSC1
12	PSa2	PSC2
13	PSa3	PSC3
14	PSa4	PSC6
15	PSa5	Ethernet

Bits	Name	Description
16	PSa6	USB
17	PSa7	ATA
18	PSa8	PCI Control module
19	PSa9	PCI SC Initiator Rx
20	PSa10	PCI SC Initiator Tx
21	PSa11	PSC4
22	PSa12	PSC5
23	PSa13	SPI modf
24	PSa14	SPI spif
25	PSa15	I^2C1
26	PSa16	I^2C2
27	PSa17	CAN1
28	PSa18	CAN2
29:30	—	Reserved
31	PSa21	XLB Arbiter

Note:

1. These interrupts are directly maskable by ICTL Peripheral Interrupt Mask Register. However, PSa status occurs regardless of Per_Mask setting, as long as the source module interrupt is enabled in the source module registers.

7.2.4.13 ICTL Peripheral Interrupt Status All Register—MBAR + 0x0538

Table 7-16. ICTL Bus Error Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						BE1	BE0	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:5	—	Reserved
6	BE1	Bus Error 1—Indicates write attempt to read-only register, clear with a write to 1.
7	BE2	Bus Error 0—Indicates access to unimplemented register, clear with a write to 1.
8:31	—	Reserved

7.2.4.14 ICTL Main Interrupt Emulation All Register—MBAR + 0x0540

Table 7-17. ICTL Main Interrupt Emulation All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															MEa
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 lsb
R	MEa															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:14	—	Reserved
—	MEa[x]	This register provides a way for software to emulate the assertion of a particular Main/SIU interrupt. The actual interrupt is the OR or the normal interrupt source and each of these test register bits. The order is exactly the same as the MSa in ICTL Main Interrupt Status All Register. The MEa[x] bits ARE masked by the Main_Mask setting, so they operate as much as possible as the real interrupt source. Even the IRQ sources, which may be programmed as edge sensitive, will react just like the pin when emulated here with test bit assertion/negation. One exception is LO-int, which if asserted here, will NOT create a corresponding Peripheral Status indication. If relying on MEa[x] assertion/negation to emulate and test an ISR routine it is important to disable all source modules so that real source interrupts will not disturb the test generated interrupt.
15	MEa0	Slice_Timer 1 (SMI interrupt only)
16	MEa1	$\overline{\text{IRQ}}$[1] input pin
17	MEa2	$\overline{\text{IRQ}}$[2] input pin
18	MEa3	$\overline{\text{IRQ}}$[3] input pin
19	MEa4	LO_int (some Peripheral source)
20	MEa5	RTC_periodic interrupt
21	MEa6	RTC_stopwatch interrupt
22	MEa7	GPIO std interrupt
23	MEa8	GPIO WakeUp interrupt
24	MEa9	TMR0 interrupt
25	MEa10	TMR1 interrupt
26	MEa11	TMR2 interrupt
27	MEa12	TMR3 interrupt
28	MEa13	TMR4 interrupt
29	MEa14	TMR5 interrupt
30	MEa15	TMR6 interrupt
31	MEa16	TMR7 interrupt

7.2.4.15 ICTL Peripheral Interrupt Emulation All Register—MBAR + 0x0544

Table 7-18. ICTL Peripheral Interrupt Emulation All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								PEa							
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 lsb
R	PEa													Reserved		PEa21
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	—	Reserved
—	PEa[x]	This register provides a way for software to emulate the assertion of a particular Peripheral interrupt. The actual interrupt is the OR or the normal interrupt source and each of these test register bits. The order is exactly the same as the PSa in ICTL Peripheral Interrupt Status All Register. The PEa[x] bits ARE masked by the Per_Mask setting, so they operate as much as possible as the real interrupt source. Test assertion of a Periperhal source will cause HI-int or LO-int indications which will be reflected in the Main or Critical status registers. If relying on PEa[x] assertion/negation to emulate and test an ISR routine it is important to disable all source modules so that real source interrupts will not disturb the test generated interrupt.
8	PEa23	BestComm LocalPlus
9	PEa22	BDLC
10	PEa0	BestComm interrupt source
11	PEa1	PSC1
12	PEa2	PSC2
13	PEa3	PSC3
14	PEa4	PSC6
15	PEa5	Ethernet
16	PEa6	USB
17	PEa7	ATA
18	PEa8	PCI Control module
19	PEa9	PCI SC Initiator Rx
20	PEa10	PCI SC Initiator Tx
21	PEa11	PSC4
22	PEa12	PSC5
23	PEa13	SPI modf
24	PEa14	SPI spif
25	PEa15	I^2C1
26	PEa16	I^2C2

Bits	Name	Description
27	PEa17	CAN1
28	PEa18	CAN2
29:30	—	Reserved
31	PEa21	XLB Arbiter

7.2.4.16 ICTL IRQ Interrupt Emulation All Register—MBAR + 0x0548

Table 7-19. ICTL IRQ Interrupt Emulation All Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				IRQEa				Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	—	Reserved
—	IRQEa[x]	This register provides a way for software to emulate the assertion of a particular external interrupt pin. The actual interrupt is the OR of the normal interrupt source and each of these IRQEa[x] bits. This register represents the four IRQ inputs. This register is redundant with IICTL Main Interrupt Emulation All Register for IRQ1-3 but is the only source to emulate IRQ0. It provides a single register with which to test and develop an ISR for the external interrupt sources. Each bit operates as if it were the pin itself, i.e. edge sensitive operation would require multiple test writes to create the emulation of a pulsing input. See Note 1
4	IRQEa0	$\overline{\text{IRQ}}$[0] input pin emulation
5	IRQEa1	$\overline{\text{IRQ}}$[1] input pin emulation
6	IRQEa2	$\overline{\text{IRQ}}$[2] input pin emulation
7	IRQEa3	$\overline{\text{IRQ}}$[3] input pin emulation
8:31	—	Reserved

Note:

1. The emulation is only possible if the IRQ pins are externally pulled down. Otherwise the OR between the external pin values and the IRQEa[x] bits is whole the time one.

7.3 General Purpose I/O (GPIO)

There are a total of 56 possible GPIO pins on the MPC5200. Virtually all of these pins are shared with alternate hardware functions. Therefore, GPIO availability is entirely dependant on the peripheral set a particular application requires.

There are 5 basic types of GPIO pins, controlled by separate register groupings, and in some cases, different register modules:

- 24 “Simple” GPIO, controlled in the standard GPIO register module.
- 8 “Output Only” GPIO, controlled in the standard GPIO register module.
- 8 “Interrupt” GPIO, controlled in the standard GPIO register module.
- 8 “Wakeup” GPIO, controlled in the WakeUp GPIO register module.
- 8 “Timer” GPIO, controlled in the General Purpose Timer register module.

There is a hierarchy of GPIO functionality. Higher function GPIO can be programmed to operate at any lower functional level. The hierarchy, from lowest to highest, is as follows:

- **Output Only**—As the name suggests, these GPIO cannot be programmed as Inputs. As outputs, they can be programmed to emulate an Open-Drain output.
- **Simple**—Same as Output Only, but with additional capability to be programmed as inputs, with a corresponding Input Value register that can be read by software.
- **Interrupt**—Same as Simple, but with additional capability of generating an Interrupt to the CPU during normal powered-up mode. The Interrupt Type can be programmed as edge (any/rising/falling/2nd edge) sensitive. These GPIO are sometimes referred to as “Simple Interrupt”.
- **Wakeup**—Same as Interrupt, but with additional capability of generating an Interrupt during Deep Sleep mode. Includes Interrupt Type registers and has an extra enable bit to distinguish between Simple Interrupt or WakeUp Interrupt operation.
- **Timer GPIO**—Operates with Simple GPIO capability, but can generate CPU Interrupts if configured as Input Capture timer mode. These Timer GPIO have special capabilities and limitations, which are described in Section 7.4, General Purpose Timers (GPT). Timer GPIO does not fit cleanly into the GPIO functional hierarchy concept, and should therefore be considered as a unique GPIO function.

GPIO functionality is available on an I/O pin only if the pin is enabled for GPIO usage in the Section 7.3.2.1.1, *GPS Port Configuration Register—MBAR + 0x0B00*. The GPIOPCR register controls the top level pin-muxing, which sets an I/O pin’s usage between some hardware function(s) and GPIO. If the pin is available for GPIO, the associated GPIO registers must be enabled and configured by software to complete the GPIO operation for that specific pin. If a Timer GPIO is consumed by an alternate hardware function, it is still available to work as an internal General Purpose Timer (GPT).

Simple GPIO are controlled by a group of registers in the Standard GPIO module. They are organized in relation to the multi-function hardware port groupings. For example, you will see a GPIO field named PSC1 (4 bits) that corresponds to the 4 Simple GPIO available on the PSC1 port group. There is also a WakeUp GPIO on the PSC1 port. However, this pin, as GPIO, would be controlled by a separate register in the Wakeup GPIO module. Even though the pins are physically scattered throughout the multi-function port groups, register control groupings exist for the:

- 8 Wakeup GPIO pins
- 8 Interrupt GPIO pins, and
- 8 Output-Only GPIO pins.

Only Simple GPIO register groupings correspond to the physical pin groupings.

Table 7-20 lists all 56 GPIO pins.

Table 7-20. GPIO Pin List

GPIO PIN	Alternate Functionality	Interrupt	WakeUp
TIMER_0	Timer_GPIO/ATA/CAN2	Only as Timer	No
TIMER_1	Timer_GPIO/ATA/CAN2	Only as Timer	No
TIMER_2	Timer_GPIO/SPI	Only as Timer	No
TIMER_3	Timer_GPIO/SPI	Only as Timer	No
TIMER_4	Timer_GPIO/SPI	Only as Timer	No
TIMER_5	Timer_GPIO/SPI	Only as Timer	No
TIMER_6	Timer_GPIO	Only as Timer	Yes (Timer IC)
TIMER_7	Timer_GPIO	Only as Timer	Yes (Timer IC)

Table 7-20. GPIO Pin List (continued)

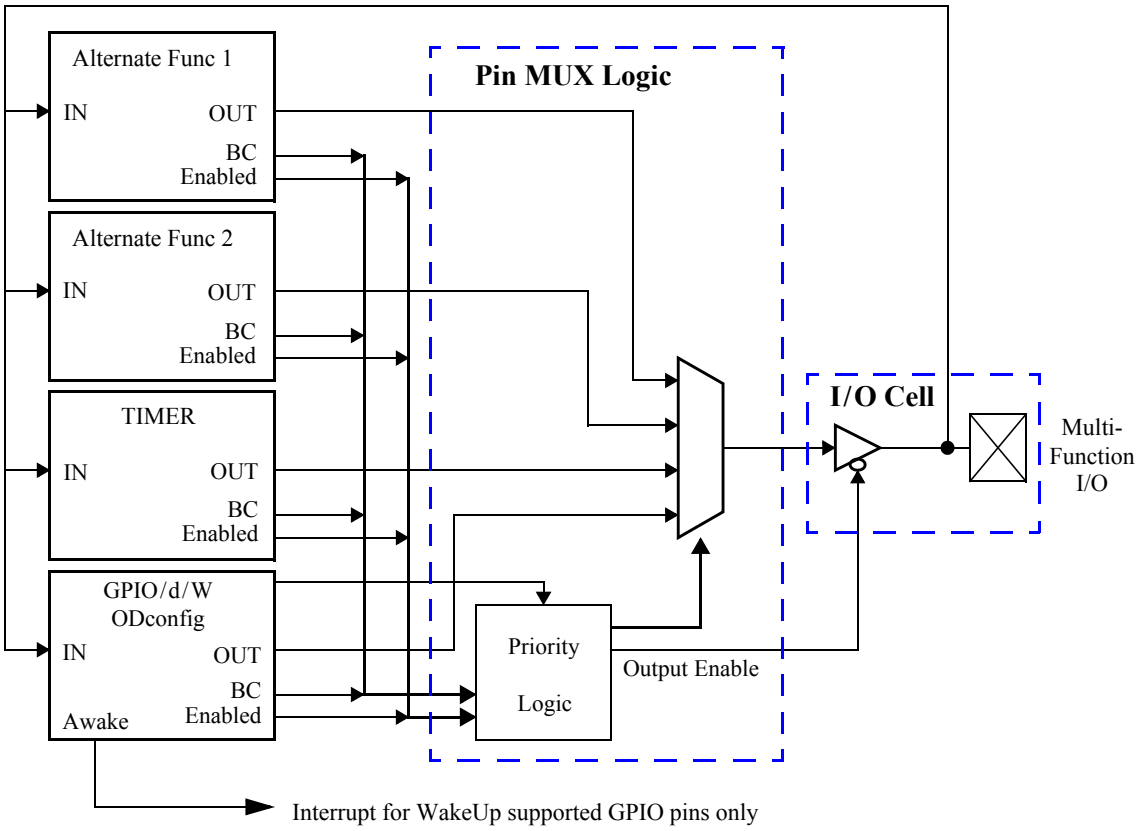
GPIO PIN	Alternate Functionality	Interrupt	WakeUp
PSC1_0	UART1/AC971/CODEC1	No	No
PSC1_1	UART1/AC971/CODEC1	No	No
PSC1_2	UART1/AC971	No	No
PSC1_3	UART1/AC971/CODEC1	No	No
PSC1_4	UART1/AC971/CODEC1	Yes	Yes
PSC2_0	UART2/AC972/CODEC2/CAN1	No	No
PSC2_1	UART2/AC972/CODEC2/CAN1	No	No
PSC2_2	UART2/AC972/CAN2	No	No
PSC2_3	UART2/AC972/CODEC2/CAN2	No	No
GPIO_WKUP_1(PSC2_4)	UART2/AC972/CODEC2	Yes	Yes
GPIO_PSC3_0	USB2/CODEC3/UART3	No	No
GPIO_PSC3_1	USB2/CODEC3/UART3	No	No
GPIO_PSC3_2	USB2/CODEC3/UART3	No	No
GPIO_PSC3_3	USB2/CODEC3/UART3	No	No
GPIO_SINT_0(PSC3_4)	USB2/UART3	Yes	No
GPIO_SINT_1(PSC3_5)	USB2	Yes	No
GPIO_PSC3_6	USB2/SPI	No	No
GPIO_PSC3_7	USB2/SPI	No	No
GPIO_SINT_2(PSC3_8)	USB2/SPI	Yes	No
GPIO_WKUP_2(PSC3_9)	USB2/SPI	Yes	Yes
GPIO_USB_0	USB1 (OE)	No	No
GPIO_USB_1	USB1 (PORTPWR)/UART5 (TXD)	No	No
GPIO_USB_2	USB1 (SPEED)/UART5 (RTS)	No	No
GPIO_USB_3	USB1 (SUSPEND)/UART5 (CTS)	No	No
GPIO_SINT_3(USB)	USB1 (OvrCrnt)	Yes	No
GPIO_ETHO_0(out only)	Ethernet	No	No
GPIO_ETHO_1(out only)	Ethernet/UART5	No	No
GPIO_ETHO_2(out only)	Ethernet/USB2/UART5	No	No
GPIO_ETHO_3(out only)	Ethernet/USB2/UART4	No	No
GPIO_ETHO_4(out only)	Ethernet/USB2/J1850	No	No
GPIO_ETHO_5(out only)	Ethernet/USB2/UART4	No	No
GPIO_ETHO_6(out only)	Ethernet/USB2	No	No
GPIO_ETHO_7(out only)	Ethernet/USB2	No	No
GPIO_ETHI_0	Ethernet/UART5	No	No
GPIO_ETHI_1	Ethernet/UART5	No	No

Table 7-20. GPIO Pin List (continued)

GPIO PIN	Alternate Functionality	Interrupt	WakeUp
GPIO_ETHI_2	Ethernet	No	No
GPIO_ETHI_3	Ethernet	No	No
GPIO_SINT_4(ETH)	Ethernet/USB2/J1850	Yes	No
GPIO_SINT_5(ETH)	Ethernet/USB2/UART4	Yes	No
GPIO_SINT_6(ETH)	Ethernet/USB2/UART4	Yes	No
GPIO_SINT_7(ETH)	Ethernet/USB2/UART4	Yes	No
GPIO_WKUP_3(ETH)	Ethernet	Yes	Yes
GPIO_IRDA_0	IRDA/UART6/Codec6	No	No
GPIO_IRDA_1	IRDA(and/or USB)/UART6/Codec6	No	No
GPIO_WKUP_4(IRDA)	IRDA/UART6/Codec6	Yes	Yes
GPIO_WKUP_5(IRDA)	IRDA/UART6/Codec6	Yes	Yes
GPIO_WKUP_6	Dedicated GPIO Pin/SDRAM CS1	Yes	Yes
GPIO_WKUP_7	Dedicated GPIO Pin/LocalPlus Most/Graphics mode TSIZ1	Yes	Yes

7.3.1 GPIO Pin Multiplexing

Figure 7-3 shows the GPIO/Generic MUX cell.



Note:

1. Open-Drain Emulation is supported on the GPIO function.
2. Pin MUX Logic is controlled by the Port Configuration Register and supersedes any individual GPIO register programming.

Figure 7-3. GPIO/Generic MUX Cell

7.3.1.1 PSC1 (UART1/AC97/CODEC1)

The PSC1 port has 5 pins with hardware support for:

- CODEC
- UART (4 pins consumed)
- UARTe (expanded with carrier detect input–5 pins consumed)
- AC97

Unused pins can serve as simple GPIOs, with one available as a WakeUp input. For use as AC97, this WakeUp GPIO becomes available. A special mode is available in which the CD input for UART use can be unused. This makes a WakeUp GPIO available on this port. CODEC usage makes one simple GPIO available. Use of this port for AC97 consumes all 5 pins and leaves no GPIO available.

Refer to the port-mapping illustrations Figure 2-4.

7.3.1.2 PSC2 (CAN1/2/UART2/AC97/CODEC2)

The PSC2 port has 5 pins with hardware support for:

- CAN
- CODEC
- UART (4 pins consumed)
- UARTe (expanded with carrier detect input–5 pins consumed)
- AC97

Unused pins can serve as simple GPIOs, with one available as a WakeUp input. For use as AC97, this WakeUp GPIO becomes available. A special mode is available in which the CD input for UART use can be unused. This makes a WakeUp GPIO available on this port. CODEC usage makes one simple GPIO available. Use of this port for AC97 consumes all 5 pins and leaves no GPIO available.

Refer to the port-mapping illustrations Figure 2-5.

7.3.1.3 PSC3 (USB2/CODEC3/SPI/UART3)

The PSC3 port has 10pins with hardware support for:

- CODEC
- Expanded UART (5 pins consumed)
- SPI (4 pins consumed)
- USB secondary port (10 pins consumed)

SPI can simultaneously exist, with no pins leftover for GPIO. Similarly, CODEC or UART can exist with SPI leaving no leftover pins. Unless, CD input on UART is designated unused, in which case a WakeUp GPIO becomes available. Any unused pins are available for related RS232 GPIO functionality.

Refer to the port-mapping illustrations Figure 2-6.

7.3.1.4 USB1/RST_CONFIG

This is a 10-bit port dedicated to primary USB. GPIO becomes available **only** if the USB function is not used. When this occurs, the following GPIO becomes available:

- 4 Simple GPIO
- 1 Interrupt GPIO

Other pins on this port serve as Reset Configuration inputs.

7.3.1.5 Ethernet/USB2/UART4/5/J1850/RST_CONFIG

This port consists of 8 output data pins and 10 control pins (in ethernet mode). For GPIO grouping these are the EthO and EthI ports, respectively. The output-only pins (EthO) are also used for input reset configuration data, therefore these pins must act as output only in all other cases. No peripheral is allowed to overdrive the reset configuration pull-up/pull-down settings. The 8 GPIOs on the EthO port are therefore output-only, and only available if the pin is otherwise unused (beyond reset config).

NOTE

The ethernet pin, MDIO, is actually an I/O. However, there should be no danger of an external chip driving this pin during power-up.

This port is configured such that 7-wire Ethernet and a secondary USB port can exist simultanaeouly. This configuration makes available 1 GPIO WakeUp pin.

Full Ethernet consumes all 18 pins, unless the optional MDIO and MDC pins are specified as unused. In this case, 2 Output Only GPIO are available.

Meanwhile, there are other cases becasue many pins can be used for UART, J1850. Please Refer to the port-mapping illustrations for details.

USB stand-alone usage leaves available:

- 2 Output Only GPIO
- 4 Simple GPIO
- 1 WakeUp GPIO

7-wire Ethernet stand-alone leaves available:

- 6 Output Only GPIO
- 4 Interrupt GPIO
- 1 WakeUp GPIO

1850 stand-alone leaves available:

- 7 Output Only GPIO
- 4 Simple GPIO
- 3 Interrupt GPIO
- 1 WakeUp GPIO

Total GPIO available on this port is:

- 8 Output Only GPIO
- 4 Simple GPIO
- 4 Interrupt GPIO
- 1 WakeUp GPIO

7.3.1.6 PSC6

The PSC6 port has 4 pins, which includes:

- 2 Simple GPIO
- 2 WakeUp GPIO

Hardware functions available are:

- IRDA
  - 3 pins with clock input
  - 2 pins with internal clock
- UART (4 pins)
- Codec (4 pins)

The IRDA clock pin can be used as a Input USB clock and is separately programmable for this use.

- If unused, the IRDA Receive pins are available as WakeUp GPIO.
- If unused, the IRDA Transmit pin and the Clock pin are available as Simple GPIO.

7.3.1.7 I^2C

There are 2 I^2C ports consisting of 2 pins each. Although no GPIO is available on these pins, they can be alternately programmed as CAN1 pins (on I^2C1) and/or as the ATA Chip Selects (on I^2C2). If the alternate function is specified, the associated I^2C port is consumed and unavailable.

7.3.1.8 GPIO Timer Pins

The GPIO Timer port consists of 8 pins. Each pin is driven by a internal timer module, which can do either of the following:

- drive the pin in Output Compare mode and Pulse Width Modulation mode, or
- monitor the pin as input in Input Capture mode.

Additionally, the timer module can operate the pin as a Simple GPIO. This GPIO control is handled in the Timer Module register, see Section 7.4.4, GPT Registers—MBAR + 0x0600. If the pin is controlled as a GPIO, then the Timer Module timer can be used as an internal CPU timer.

The Timer pins can be reconfigured for alternate functionality in the Port Configuration Register, as follows:

- Timer pins 0 and 1 can operate as CAN2 Tx/Rx or ATA Chip Selects.
- Timer pins 2–5 can operate as the SPI port.

- Timer pins 6 and 7 are dedicated as Timer GPIO and have no alternate function.

Although the Timer as GPIO only operates to the Simple GPIO level, Interrupt capability can be achieved by configuring the Timer for Input Capture mode.

7.3.1.9 Dedicated GPIO Port

There is a dedicated GPIO port group that consists of 2 pins. Both pins operate at the WakeUp GPIO level. They are designated:

- GPIO_WKUP_6
- GPIO_WKUP_7

However, GPIO_WKUP_6 is not dedicated and can be programmed to operate as a second SDRAM memory chip select. As such, this pin is connected to the Memory Vdd supply. For Dual Data Rate memory, the GPIO_WKUP_6 pin is driven at the reduced 2.5 V level.

If not used as a memory chip select, the GPIO_WKUP_6 pin serves as a memory voltage compatible GPIO.

7.3.2 GPIO Programmer's Model

The GPIO programmer's model contains 3 separate register sets (or modules), each at different offsets from MBAR. These register sets are:

1. GPIO Standard Registers—MBAR+0x0B00. Output Only, Simple, and Interrupt GPIO are controlled by registers within this module. There are 3 register groupings for individual control of each of the named GPIO types.
2. WakeUp GPIO Registers—MBAR+0x0C00. WakeUp GPIO are controlled by this register set
3. GPT Registers—MBAR + 0x0600. Timer functions and Timer GPIO are controlled by this module.

All GPIO functionality is dependent on the Port Configuration Register (PCR) setting. The PCR is the first register in the GPIO Standard Module. This register controls the Pin MUX Logic. Therefore, the PCR also controls the physical routing of MPC5200 I/O pins to and from internal logic. The PCR is expected to be configured early in the boot process and set to a static value that supports the given peripheral set of a specific application.

NOTE

The PCR is **not** accessible during Deep Sleep mode.

7.3.2.1 GPIO Standard Registers—MBAR+0x0B00

The GPIO Standard Register set has separate registers for each GPIO type.

- Simple
- Output Only
- Interrupt

These registers are at an offset of MBAR + 0x0B00.

The GPIO Standard Register set uses 16 32-bit registers. These registers are located at an offset from MBAR of 0x0B00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0B00 + register address**

Hyperlinks to the GPIO pin type registers are provided below:

- GPS Port Configuration Register (0x0B00)
- GPS Simple GPIO Enables Register (0x0B04)
- GPS Simple GPIO Open Drain Type Register (0x0B08)
- GPS Simple GPIO Data Direction Register (0x0B0C)
- GPS Simple GPIO Data Output Values Register (0x0B10)
- GPS Simple GPIO Data Input Values Register (0x0B14)
- GPS GPIO Output-Only Enables Register (0x0B18)
- GPS GPIO Output-Only Data Value Out Register (0x0B1C)
- GPS GPIO Simple Interrupt Enables Register (0x0B20)
- GPS GPIO Simple Interrupt Open-Drain Emulation Register (0x0B24)
- GPS GPIO Simple Interrupt Data Direction Register (0x0B28)
- GPS GPIO Simple Interrupt Data Value Out Register (0x0B2C)
- GPS GPIO Simple Interrupt Interrupt Enable Register (0x0B30)
- GPS GPIO Simple Interrupt Interrupt Types Register (0x0B34)
- GPS GPIO Simple Interrupt Master Enable Register (0x0B38)
- GPS GPIO Simple Interrupt Status Register (0x0B3C)

7.3.2.1.1 GPS Port Configuration Register—MBAR + 0x0B00

Table 7-21. GPS Port Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CS1	Rsvd	ALTs		CS7	CS6	ATA		IR_USB_CLK	IRDA			Ether			
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 lsb
R	PCI_DIS	USB_SE	USB		PSC3				Rsvd	PSC2			Rsvd	PSC1		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	CS1	Memory Chip Select bit 0 = gpio_wkup_6 1 = mem_cs1 (second SDRAMC chip select) on gpio_wkup_6 pin
1	—	Reserved
2:3	ALTs	Alternatives, see Note 2 00 = No Alternatives: CAN1/2 on PSC2 according to PSC2 setting. SPI on PSC3 according to PSC3 setting. 01 = ALT CAN position: CAN1 on I2C1, CAN2 on Tmr0/1 pins, see Note 1 10 = ALT SPI position: SPI on Tmr2/3/4/5 pins, see Note 2 11 = Both on ALT
4	CS7	0 = Interrupt GPIO on PSC3_5 (see note 6) 1 = CS7 on PSC3_5
5	CS6	0 = Interrupt GPIO on PSC3_4 (see note 6) 1 = CS6 on PSC3_4
6:7	ATA	Advanced Technology Attachment 00 = No ATA chip selects, csb_4/5 used as normal chip select 01 = ATA cs0/1 on csb_4/5 10 = ATA cs0/1 on i2c2 clk/io 11 = ATA cs0/1 on Tmr0/1, see Note 1
8	IR_USB_CLK	Infrared USB Clock 0 = IrDA/USB 48MHz clock generated internally, pin is GPIO 1 = IrDA/USB clock is sourced externally, input only

Bit	Name	Description
9:11	IRDA	Infrared Data Association 000 = All IrDA pins are GPIOs 001 = Reserved 010 = Reserved 011 = Reserved 100 = Reserved 101 = UART (without CD) / IrDA 110 = Reserved 111 = CODEC (without MCLK) / IrDA
12: 15	Ether	Ethernet 0000 = All 18 Ethernet pins are GPIOs 0001 = USB2 on Ethernet, see Note 3 0010 = Ethernet 10Mbit (7-wire) mode 0011 = Ethernet 7-wire and USB2, see Note 3 0100 = Ethernet 100Mbit without MD 0101 = Ethernet 100Mbit with MD 011X = Reserved 1000 = Ether 7-wire, UARTe, J1850 1001 = Ether 7-wire, J1850 1010 = Two UARTes, J1850 1011 = One UARTe, J1850 1100 = J1850 1101 = Reserved 111X = Reserved
16	PCI_DIS	0 = PCI controller enabled 1 = PCI controller disabled. When Large Flash or Most Graphics modes are enabled on the localPlus bus interface, the PCI interface can not be used (PCI control signals are used to support these modes). When these modes are enabled (see LocalPlus control registers), the PCI controller must be disabled to prevent interference. If these modes are enabled at boot, this bit will come out of reset set to 1. If these modes are not enabled at boot, this bit will come out of reset set to 0.
17	USB_SE	USB Single Ended mode. The USB interface is able to support both Differential and Single Ended modes. This bit allows the USB I/O interface to be programmed to Single Ended mode. Differential mode supplies TXP/TXN and RXP/TXN. Single ended mode supplies TXP/TX_SE0 and RXP/RX_SE0. This bit controls "all" USB ports (i.e. they are not individually programmable). Default is Differential mode. 0 = Differential mode (Default after reset) 1 = Single ended mode
18:19	USB	00 = 4 GPIOs and 1 **Interrupt GPIO** 01 = USB 10 = Two UARTs 11 = Reserved

Bit	Name	Description
20:23	PSC3	Programmable Serial Controller 3 0000 = All PSC3 pins are GPIOs 0001 = USB2 on PSC3, no GPIOs available, see Note 3 001X = Reserved 0100 = UART functionality without CD 0101 = UARTe functionality with CD 0110 = CODEC3 functionality 0111 = CODEC3 functionality (with MCLK) 100X = SPI 101X = Reserved 1100 = SPI with UART3 1101 = SPI with UART3e 111X = SPI with CODEC3
24	—	Reserved
25:27	PSC2	Programmable Serial Controller 2 000 = All PSC2 pins are GPIOs 001 = CAN1&2 on PSC2 pins, see Note 3 01X = AC97 functionality 100 = UART functionality without CD 101 = UARTe functionality with CD 110 = CODEC2 functionality(without MCLK) 111 = CODEC2 functionality (with MCLK)
28	—	Reserved
29:31	PSC1	Programmable Serial Controller 1 00X = All PSC1 pins are GPIOs 01X = AC97 functionality 100 = UART functionality without CD 101 = UARTe functionality with CD 110 = CODEC1 functionality (without MCLK) 111 = CODEC1 functionality (with MCLK)

Note:

1. ALT CAN cannot exist with ATA on Tmr0/1, not with CAN on PSC2.
2. ALT SPI cannot exist with any SPI on PCS3.
3. USB cannot exist on both Either and PSC3.
4. See Section 7.3.1, *GPIO Pin Multiplexing* or Table 2-1 or Table 2-2 to determine GPIO availability for the various PCR field settings.
5. If Large Flash or Most Graphics mode is enabled at boot, using a reset configuration bit, PCI disable will come out of reset set to 1. If these modes are not enabled at boot, this bit will come out of reset set to 0.
6. PSC3_4 and PSC3_5 default to zero (interrupt gpio) after reset. However, if the PSC3 is pro-grammed to USB2 mode RXP and RXN will be on these pins. If PSC is programmed to UARTe mode, CD will be on the PSC3_4 pin.

7.3.2.1.2 GPS Simple GPIO Enables Register—MBAR + 0x0B04

Table 7-22. GPS Simple GPIO Enables Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-22. GPS Simple GPIO Enables Register

R	Reserved		IRDA		ETHR				Reserved				USB			
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 lsb
R	Reserved		PSC3						PSC2				PSC1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:3	IRDA	Individual enable bits for the 2 Simple GPIO on IRDA port. bit 2 controls GPIO_IRDA_1 (IR_USB_CLK pin) bit 3 controls GPIO_IRDA_0 (IRDA_TX pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO
4:7	ETHR	Individual enable bits for the 4 Simple GPIO on ETHR port. bit 4 controls GPIO_ETHI_3 (ETH_11 pin) bit 5 controls GPIO_ETHI_2 (ETH_10 pin) bit 6 controls GPIO_ETHI_1 (ETH_9 pin) bit 7 controls GPIO_ETHI_0 (ETH_8 pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO
8:11	—	Reserved
12:15	USB	Individual enable bits for the 4 Simple GPIO on USB port. bit 12 controls GPIO_USB_3 (USB1_8 pin) bit 13 controls GPIO_USB_2 (USB1_7 pin) bit 14 controls GPIO_USB_1 (USB1_6 pin) bit 15 controls GPIO_USB_0 (USB1_0 pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO
16:17	—	Reserved
18:23	PSC3	Individual enable bits for the 6 Simple GPIO on PSC3 port. bit 18 controls GPIO_ PSC3_5 (PSC3_7 pin) bit 19 controls GPIO_ PSC3_4 (PSC3_6 pin) bit 20 controls GPIO_ PSC3_3 (PSC3_3 pin) bit 21 controls GPIO_ PSC3_2 (PSC3_2 pin) bit 22 controls GPIO_ PSC3_1 (PSC3_1 pin) bit 23 controls GPIO_ PSC3_0 (PSC3_0 pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO

Bit	Name	Description
24:27	PSC2	Individual enable bits for the 4 Simple GPIO on PSC2 port. bit 24 controls GPIO_PSC2_3 (PSC2_3 pin) bit 25 controls GPIO_PSC2_2 (PSC2_2 pin) bit 26 controls GPIO_PSC2_1 (PSC2_1 pin) bit 27 controls GPIO_PSC2_0 (PSC2_0 pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO
28:31	PSC1	Individual enable bits for the 4 Simple GPIO on PSC1 port. bit 28 controls GPIO_PSC1_3 (PSC1_3 pin) bit 29 controls GPIO_PSC1_2 (PSC1_2 pin) bit 30 controls GPIO_PSC1_1 (PSC1_1 pin) bit 31 controls GPIO_PSC1_0 (PSC1_0 pin) 0 = Disabled for GPIO (default) 1 = Enabled for GPIO

7.3.2.1.3 GPS Simple GPIO Open Drain Type Register —MBAR + 0x0B08

Table 7-23. GPS Simple GPIO Open Drain Type Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		IRDA		ETHR				Reserved				USB			
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 lsb
R	Reserved		PSC3						PSC2				PSC1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:3	IRDA	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 2 controls GPIO_IRDA_1 (IR_USB_CLK pin) bit 3 controls GPIO_IRDA_0 (IRDA_TX pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
4:7	ETHR	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 4 controls GPIO_ETHI_3 (ETH_11 pin) bit 5 controls GPIO_ETHI_2 (ETH_10 pin) bit 6 controls GPIO_ETHI_1 (ETH_9 pin) bit 7 controls GPIO_ETHI_0 (ETH_8 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
8:11	—	Reserved

Bit	Name	Description
12:15	USB	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 12 controls GPIO_USB_3 (USB1_8 pin) bit 13 controls GPIO_USB_2 (USB1_7 pin) bit 14 controls GPIO_USB_1 (USB1_6 pin) bit 15 controls GPIO_USB_0 (USB1_0 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
16:17	—	Reserved
18:23	PSC3	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 18 controls GPIO_ PSC3_5 (PSC3_7 pin) bit 19 controls GPIO_ PSC3_4 (PSC3_6 pin) bit 20 controls GPIO_ PSC3_3 (PSC3_3 pin) bit 21 controls GPIO_ PSC3_2 (PSC3_2 pin) bit 22 controls GPIO_ PSC3_1 (PSC3_1 pin) bit 23 controls GPIO_ PSC3_0 (PSC3_0 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
24:27	PSC2	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 24 controls GPIO_PSC2_3 (PSC2_3 pin) bit 25 controls GPIO_PSC2_2 (PSC2_2 pin) bit 26 controls GPIO_PSC2_1 (PSC2_1 pin) bit 27 controls GPIO_PSC2_0 (PSC2_0 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
28:31	PSC1	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 28 controls GPIO_PSC1_3 (PSC1_3 pin) bit 29 controls GPIO_PSC1_2 (PSC1_2 pin) bit 30 controls GPIO_PSC1_1 (PSC1_1 pin) bit 31 controls GPIO_PSC1_0 (PSC1_0 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)

7.3.2.1.4 GPS Simple GPIO Data Direction Register—MBAR + 0x0B0C

Table 7-24. GPS Simple GPIO Data Direction Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		IRDA		ETHR				Reserved				USB			
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 lsb
R	Reserved		PSC3						PSC2				PSC1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:3	IRDA	Individual bits to control directionality of the pin as GPIO. bit 2 controls GPIO_IRDA_1 (IR_USB_CLK pin) bit 3 controls GPIO_IRDA_0 (IRDA_TX pin) 0 = Pin is Input (default) 1 = Pin is Output
4:7	ETHR	Individual bits to control directionality of the pin as GPIO. bit 4 controls GPIO_ETHI_3 (ETH_11 pin) bit 5 controls GPIO_ETHI_2 (ETH_10 pin) bit 6 controls GPIO_ETHI_1 (ETH_9 pin) bit 7 controls GPIO_ETHI_0 (ETH_8 pin) 0 = Pin is Input (default) 1 = Pin is Output
8:11	—	Reserved
12:15	USB	Individual bits to control directionality of the pin as GPIO. bit 12 controls GPIO_USB_3 (USB1_8 pin) bit 13 controls GPIO_USB_2 (USB1_7 pin) bit 14 controls GPIO_USB_1 (USB1_6 pin) bit 15 controls GPIO_USB_0 (USB1_0 pin) 0 = Pin is Input (default) 1 = Pin is Output
16:17	—	Reserved
18:23	PSC3	Individual bits to control directionality of the pin as GPIO. bit 18 controls GPIO_ PSC3_5 (PSC3_7 pin) bit 19 controls GPIO_ PSC3_4 (PSC3_6 pin) bit 20 controls GPIO_ PSC3_3 (PSC3_3 pin) bit 21 controls GPIO_ PSC3_2 (PSC3_2 pin) bit 22 controls GPIO_ PSC3_1 (PSC3_1 pin) bit 23 controls GPIO_ PSC3_0 (PSC3_0 pin) 0 = Pin is Input (default) 1 = Pin is Output

Bit	Name	Description
24:27	PSC2	Individual bits to control directionality of the pin as GPIO. bit 24 controls GPIO_PSC2_3 (PSC2_3 pin) bit 25 controls GPIO_PSC2_2 (PSC2_2 pin) bit 26 controls GPIO_PSC2_1 (PSC2_1 pin) bit 27 controls GPIO_PSC2_0 (PSC2_0 pin) 0 = Pin is Input (default) 1 = Pin is Output
28:31	PSC1	Individual bits to control directionality of the pin as GPIO. bit 28 controls GPIO_PSC1_3 (PSC1_3 pin) bit 29 controls GPIO_PSC1_2 (PSC1_2 pin) bit 30 controls GPIO_PSC1_1 (PSC1_1 pin) bit 31 controls GPIO_PSC1_0 (PSC1_0 pin) 0 = Pin is Input (default) 1 = Pin is Output

7.3.2.1.5 GPS Simple GPIO Data Output Values Register —MBAR + 0x0B10

Table 7-25. GPS Simple GPIO Data Output Values Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		IRDA		ETHR				Reserved				USB			
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 lsb
R	Reserved		PSC3						PSC2				PSC1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:3	IRDA	Individual bits to control the state of pins configured as GPIO output. bit 2 controls GPIO_IRDA_1 (IR_USB_CLK pin) bit 3 controls GPIO_IRDA_0 (IRDA_TX pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
4:7	ETHR	Individual bits to control the state of pins configured as GPIO output. bit 4 controls GPIO_ETHI_3 (ETH_11 pin) bit 5 controls GPIO_ETHI_2 (ETH_10 pin) bit 6 controls GPIO_ETHI_1 (ETH_9 pin) bit 7 controls GPIO_ETHI_0 (ETH_8 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
8:11	—	Reserved
12:15	USB	Individual bits to control the state of pins configured as GPIO output. bit 12 controls GPIO_USB_3 (USB1_8 pin) bit 13 controls GPIO_USB_2 (USB1_7 pin) bit 14 controls GPIO_USB_1 (USB1_6 pin) bit 15 controls GPIO_USB_0 (USB1_0 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
16:17	—	Reserved
18:23	PSC3	Individual bits to control the state of pins configured as GPIO output. bit 18 controls GPIO_ PSC3_5 (PSC3_7 pin) bit 19 controls GPIO_ PSC3_4 (PSC3_6 pin) bit 20 controls GPIO_ PSC3_3 (PSC3_3 pin) bit 21 controls GPIO_ PSC3_2 (PSC3_2 pin) bit 22 controls GPIO_ PSC3_1 (PSC3_1 pin) bit 23 controls GPIO_ PSC3_0 (PSC3_0 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin

Bit	Name	Description
24:27	PSC2	Individual bits to control the state of pins configured as GPIO output. bit 24 controls GPIO_PSC2_3 (PSC2_3 pin) bit 25 controls GPIO_PSC2_2 (PSC2_2 pin) bit 26 controls GPIO_PSC2_1 (PSC2_1 pin) bit 27 controls GPIO_PSC2_0 (PSC2_0 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
28:31	PSC1	Individual bits to control the state of pins configured as GPIO output. bit 28 controls GPIO_PSC1_3 (PSC1_3 pin) bit 29 controls GPIO_PSC1_2 (PSC1_2 pin) bit 30 controls GPIO_PSC1_1 (PSC1_1 pin) bit 31 controls GPIO_PSC1_0 (PSC1_0 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin

7.3.2.1.6 GPS Simple GPIO Data Input Values Register —MBAR + 0x0B14

Table 7-26. GPS Simple GPIO Data Input Values Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		IRDA		ETHR				Reserved				USB			
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 lsb
R	Reserved		PSC3						PSC2				PSC1			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:3	IRDA	Individual status bits reflecting the state of corresponding GPIO pins. bit 2 reflects GPIO_IRDA_1 (IR_USB_CLK pin) bit 3 reflects GPIO_IRDA_0 (IRDA_TX pin)
4:7	ETHR	Individual status bits reflecting the state of corresponding GPIO pins. bit 4 reflects GPIO_ETHI_3 (ETH_11 pin) bit 5 reflects GPIO_ETHI_2 (ETH_10 pin) bit 6 reflects GPIO_ETHI_1 (ETH_9 pin) bit 7 reflects GPIO_ETHI_0 (ETH_8 pin)
8:11	—	Reserved

Bit	Name	Description
12:15	USB	Individual status bits reflecting the state of corresponding GPIO pins. bit 12 reflects GPIO_USB_3 (USB1_8 pin) bit 13 reflects GPIO_USB_2 (USB1_7 pin) bit 14 reflects GPIO_USB_1 (USB1_6 pin) bit 15 reflects GPIO_USB_0 (USB1_0 pin)
16:17	—	Reserved
18:23	PSC3	Individual status bits reflecting the state of corresponding GPIO pins. bit 18 reflects GPIO_ PSC3_5 (PSC3_7 pin) bit 19 reflects GPIO_ PSC3_4 (PSC3_6 pin) bit 20 reflects GPIO_ PSC3_3 (PSC3_3 pin) bit 21 reflects GPIO_ PSC3_2 (PSC3_2 pin) bit 22 reflects GPIO_ PSC3_1 (PSC3_1 pin) bit 23 reflects GPIO_ PSC3_0 (PSC3_0 pin)
24:27	PSC2	Individual status bits reflecting the state of corresponding GPIO pins. bit 24 reflects GPIO_PSC2_3 (PSC2_3 pin) bit 25 reflects GPIO_PSC2_2 (PSC2_2 pin) bit 26 reflects GPIO_PSC2_1 (PSC2_1 pin) bit 27 reflects GPIO_PSC2_0 (PSC2_0 pin)
28:31	PSC1	Individual status bits reflecting the state of corresponding GPIO pins. bit 28 reflects GPIO_PSC1_3 (PSC1_3 pin) bit 29 reflects GPIO_PSC1_2 (PSC1_2 pin) bit 30 reflects GPIO_PSC1_1 (PSC1_1 pin) bit 31 reflects GPIO_PSC1_0 (PSC1_0 pin)
Note: These status bits operate regardless of the function on the pin.		

7.3.2.1.7 GPS GPIO Output-Only Enables Register —MBAR + 0x0B18

Table 7-27. GPS GPIO Output-Only Enables Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETHR								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ETHR	Individual bits to enable each Output Only GPIO pin—all reside on the Ethernet port. bit 0 controls GPIO_ETHO_7 (ETH_7 pin) bit 1 controls GPIO_ETHO_6 (ETH_6 pin) bit 2 controls GPIO_ETHO_5 (ETH_5 pin) bit 3 controls GPIO_ETHO_4 (ETH_4 pin) bit 4 controls GPIO_ETHO_3 (ETH_3 pin) bit 5 controls GPIO_ETHO_2 (ETH_2 pin) bit 6 controls GPIO_ETHO_1 (ETH_1 pin) bit 7 controls GPIO_ETHO_0 (ETH_0 pin) 0 = Disabled for GPIO use (default) 1 = Enabled for GPIO use
8:31	—	Reserved

7.3.2.1.8 GPS GPIO Output-Only Data Value Out Register —MBAR + 0x0B1C

Table 7-28. GPS GPIO Output-Only Data Value Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETHR								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ETHR	Individual bits to control the state of enabled Output Only GPIO pins. bit 0 controls GPIO_ETHO_7 (ETH_7 pin) bit 1 controls GPIO_ETHO_6 (ETH_6 pin) bit 2 controls GPIO_ETHO_5 (ETH_5 pin) bit 3 controls GPIO_ETHO_4 (ETH_4 pin) bit 4 controls GPIO_ETHO_3 (ETH_3 pin) bit 5 controls GPIO_ETHO_2 (ETH_2 pin) bit 6 controls GPIO_ETHO_1 (ETH_1 pin) bit 7 controls GPIO_ETHO_0 (ETH_0 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
8:31	—	Reserved

7.3.2.1.9 GPS GPIO Simple Interrupt Enable Register—MBAR + 0x0B20

Table 7-29. GPS GPIO Simple Interrupt Enables Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIGPIOe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	SIGPIOE	Individual bits to enable each Interrupt GPIO pin (pins are scattered). bit 0 controls GPIO_SINT_7 (ETH_16 pin) bit 1 controls GPIO_SINT_6 (ETH_15 pin) bit 2 controls GPIO_SINT_5 (ETH_14 pin) bit 3 controls GPIO_SINT_4 (ETH_13 pin) bit 4 controls GPIO_SINT_3 (USB1_9 pin) bit 5 controls GPIO_SINT_2 (PSC3_8 pin) bit 6 controls GPIO_SINT_1 (PSC3_5 pin) bit 7 controls GPIO_SINT_0 (PSC3_4 pin) 0 = disabled for GPIO use (default) 1 = enabled for GPIO use
8:31	—	Reserved

7.3.2.1.10 GPS GPIO Simple Interrupt Open-Drain Emulation Register —MBAR + 0x0B24

Table 7-30. GPS GPIO Simple Interrupt Open-Drain Emulation Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIODe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	SIODe	Individual bits to cause open drain emulation for pins configured as GPIO output. bit 0 controls GPIO_SINT_7 (ETH_16 pin) bit 1 controls GPIO_SINT_6 (ETH_15 pin) bit 2 controls GPIO_SINT_5 (ETH_14 pin) bit 3 controls GPIO_SINT_4 (ETH_13 pin) bit 4 controls GPIO_SINT_3 (USB1_9 pin) bit 5 controls GPIO_SINT_2 (PSC3_8 pin) bit 6 controls GPIO_SINT_1 (PSC3_5 pin) bit 7 controls GPIO_SINT_0 (PSC3_4 pin) 0 = Normal CMOS output (default) 1 = Open Drain emulation (a drive to high creates Hi-Z)
8:31	—	Reserved

7.3.2.1.11 GPS GPIO Simple Interrupt Data Direction Register —MBAR + 0x0B28

Table 7-31. GPS GPIO Simple Interrupt Data Direction Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIDDR								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	SIDDR	Individual bits to control direction of the pin as GPIO. bit 0 controls GPIO_SINT_7 (ETH_16 pin) bit 1 controls GPIO_SINT_6 (ETH_15 pin) bit 2 controls GPIO_SINT_5 (ETH_14 pin) bit 3 controls GPIO_SINT_4 (ETH_13 pin) bit 4 controls GPIO_SINT_3 (USB1_9 pin) bit 5 controls GPIO_SINT_2 (PSC3_8 pin) bit 6 controls GPIO_SINT_1 (PSC3_5 pin) bit 7 controls GPIO_SINT_0 (PSC3_4 pin) 0 = Pin is Input (default) 1 = Pin is Output
8:31	—	Reserved

7.3.2.1.12 GPS GPIO Simple Interrupt Data Value Out Register —MBAR + 0x0B2C

Table 7-32. GPS GPIO Simple Interrupt Data Value Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIDVO								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	SIDVO	Individual bits to control the state of pins configured as GPIO output. bit 0 controls GPIO_SINT_7 (ETH_16 pin) bit 1 controls GPIO_SINT_6 (ETH_15 pin) bit 2 controls GPIO_SINT_5 (ETH_14 pin) bit 3 controls GPIO_SINT_4 (ETH_13 pin) bit 4 controls GPIO_SINT_3 (USB1_9 pin) bit 5 controls GPIO_SINT_2 (PSC3_8 pin) bit 6 controls GPIO_SINT_1 (PSC3_5 pin) bit 7 controls GPIO_SINT_0 (PSC3_4 pin) 0 = Drive 0 on the pin (default) 1 = Drive 1 on the pin
8:31	—	Reserved

7.3.2.1.13 GPS GPIO Simple Interrupt Interrupt Enable Register —MBAR + 0x0B30

Table 7-33. GPS GPIO Simple Interrupt Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIINTEN								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	SIINTEN	Individual bits to enable Interrupt generation for each GPIO pin configured as an Input. bit 0 controls GPIO_SINT_7 (ETH_16 pin) bit 1 controls GPIO_SINT_6 (ETH_15 pin) bit 2 controls GPIO_SINT_5 (ETH_14 pin) bit 3 controls GPIO_SINT_4 (ETH_13 pin) bit 4 controls GPIO_SINT_3 (USB1_9 pin) bit 5 controls GPIO_SINT_2 (PSC3_8 pin) bit 6 controls GPIO_SINT_1 (PSC3_5 pin) bit 7 controls GPIO_SINT_0 (PSC3_4 pin) 0 = Pin cannot generate an Interrupt (default) 1 = Pin can generate an Interrupt if configured as an Input GPIO
8:31	—	Reserved
Note: See Interrupt Type data in GPS GPIO Simple Interrupt Interrupt Types Register —MBAR + 0x0B34 Register. Also, the Master Interrupt Enable bit must be set in the GPS GPIO Simple Interrupt Master Enable Register —MBAR + 0x0B38 Register, before any Simple Interrupt pin can generate an Interrupt.		

7.3.2.1.14 GPS GPIO Simple Interrupt Interrupt Types Register —MBAR + 0x0B34

Table 7-34. GPS GPIO Simple Interrupt Interrupt Types Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ITYP7		ITYP6		ITYP5		ITYP4		ITYP3		ITYP2		ITYP1		ITYP0	
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	ITYP[0:7]	GPIO Interrupt Type bits for Simple-Interrupt GPIO pin 7. ITYP7—bits 0:1 controls GPIO_SINT_7 (ETH_16 pin) ITYP6—bits 2:3 controls GPIO_SINT_6 (ETH_15 pin) ITYP5—bits 4:5 controls GPIO_SINT_5 (ETH_14 pin) ITYP4—bits 6:7 controls GPIO_SINT_4 (ETH_13 pin) ITYP3—bits 8:9 controls GPIO_SINT_3 (USB1_9 pin) ITYP2—bits 10:11 controls GPIO_SINT_2 (PSC3_8 pin) ITYP1—bits 12:13 controls GPIO_SINT_1 (PSC3_5 pin) ITYP0—bits 14:15 controls GPIO_SINT_0 (PSC3_4 pin) 00 = Interrupt on any transition 01 = Interrupt on rising edge 10 = Interrupt on falling edge 11 = Interrupt on pulse (any two transitions)
16:31	—	Reserved

7.3.2.1.15 GPS GPIO Simple Interrupt Master Enable Register —MBAR + 0x0B38

Table 7-35. GPS GPIO Simple Interrupt Master Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			ME	Reserved											
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:2	—	Reserved
3	ME	GPIO Simple Interrupt Master Enable pin—This pin must be high before any Simple Interrupt pin can generate an interrupt. This bit should remain clear while programming individual interrupts, then set high as a final step. This prevents any spurious interrupt occurring during programming.
4:31	—	Reserved

7.3.2.1.16 GPS GPIO Simple Interrupt Status Register—MBAR + 0x0B3C

Table 7-36. GPS GPIO Simple Interrupt Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ISTAT								IVAL							
W	rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc								
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ISTAT	Interrupt Status—status bit for GPIO Simple interrupt pins 7 to 0, where 1 indicates an interrupt has occurred. Clear bit with a Sticky bit write to 1. Bit 0 reflects GPIO_SINT_7 (ETH_16 pin) Bit 1 reflects GPIO_SINT_6 (ETH_15 pin) Bit 2 reflects GPIO_SINT_5 (ETH_14 pin) Bit 3 reflects GPIO_SINT_4 (ETH_13 pin) Bit 4 reflects GPIO_SINT_3 (USB1_9 pin) Bit 5 reflects GPIO_SINT_2 (PSC3_8 pin) Bit 6 reflects GPIO_SINT_1 (PSC3_5 pin) Bit 7 reflects GPIO_SINT_0 (PSC3_4 pin)
8:15	IVAL	Input Value—status bit for GPIO Simple Interrupt pins 7 to 0. This is the raw state of the input pin at the time this register is read. It is not latched to the state that caused the Interrupt (if any). Bit 8 reflects GPIO_SINT_7 (ETH_16 pin) Bit 9 reflects GPIO_SINT_6 (ETH_15 pin) Bit 10 reflects GPIO_SINT_5 (ETH_14 pin) Bit 11 reflects GPIO_SINT_4 (ETH_13 pin) Bit 12 reflects GPIO_SINT_3 (USB1_9 pin) Bit 13 reflects GPIO_SINT_2 (PSC3_8 pin) Bit 14 reflects GPIO_SINT_1 (PSC3_5 pin) Bit 15 reflects GPIO_SINT_0 (PSC3_4 pin) IVAL is always available regardless of enable or setting, even if not used as GPIO. Writing to this byte has no effect.
16:31	—	Reserved

7.3.2.2 WakeUp GPIO Registers—MBAR+0x0C00

The WakeUp GPIO Register Set provides GPIO control for the 8 WakeUp GPIO pins. These pins are scattered throughout the pin groups, but are all controlled in this module. It should be noted that WakeUp GPIO can operate as Simple Interrupt GPIO. Because of this, there are separate registers to enable these pins as Wakeup interupts and/or Simple Interrupts. The distiniction between these two types of interrupts is made according to the powered state of MPC5200.

- In Deep Sleep mode, the WakeUp Interrupt enables are used.
- In all other modes, the Simple Interrupt enables are used.

In either of the above types of interrupts, we are referring to the WakeUp GPIO and the registers in this module. These are not to be confused with the Simple Interrupt GPIO pins, which are controlled in the previous module, GPIO Standard.

This WakeUp GPIO register set uses 10 32-bit registers. These registers are located at an offset from MBAR of 0x0C00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0C00 + register address**

Hyperlinks to the WakeUp GPIO registers are provided below:

• GPW WakeUp GPIO Enables Register (0x0C00)	• GPW WakeUp GPIO Individual Interrupt Enable Register (0x0C14)
• GPW WakeUp GPIO Open Drain Emulation Register (0x0C04)	• GPW WakeUp GPIO Interrupt Types Register (0x0C18)
• GPW WakeUp GPIO Data Direction Register (0x0C08)	• GPW WakeUp GPIO Master Enables Register (0x0C1C)
• GPW WakeUp GPIO Data Value Out Register (0x0C0C)	• GPW WakeUp GPIO Data Input Values Register (0x0C20)
• GPW WakeUp GPIO Interrupt Enable Register (0x0C10)	• GPW WakeUp GPIO Status Register (0x0C24)

7.3.2.2.1 GPW WakeUp GPIO Enables Register—MBAR + 0x0C00

Table 7-37. GPW WakeUp GPIO Enables Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WGPIOe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WGPIOe	Bits to enable the operation of individual WaleUp GPIO pins. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Pin not enabled for any GPIO use (default). 1 = Pin enabled for use as GPIO.
8:31	—	Reserved

7.3.2.2.2 GPW WakeUp GPIO Open Drain Emulation Register —MBAR + 0x0C04

Table 7-38. GPW WakeUp GPIO Open Drain Emulation Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WODe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WODe	Bits to control open drain emulation for individual WakeUp GPIO configured as outputs. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Normal CMOS output (default). 1 = Open Drain emulation (a drive to high creates Hi-Z).
8:31	—	Reserved

7.3.2.2.3 GPW WakeUp GPIO Data Direction Register—MBAR + 0x0C08

Table 7-39. GPW WakeUp GPIO Data Direction Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDDR[7:0]								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WDDR[7:0]	Individual bits to control directionality of the pin as GPIO. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Pin is Input (default). 1 = Pin is Output.
8:31	—	Reserved

7.3.2.2.4 GPW WakeUp GPIO Data Value Out Register —MBAR + 0x0C0C

Table 7-40. GPW WakeUp GPIO Data Value Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDVO								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WDVO	Individual bits to control the state of pins configured as GPIO output. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Drive 0 on the pin (default). 1 = Drive 1 on the pin. **Note:** If pin is emulating open drain, this setting results in Hi-Z
8:31	—	Reserved

7.3.2.2.5 GPW WakeUp GPIO Interrupt Enable Register—MBAR + 0x0C10

Table 7-41. GPW WakeUp GPIO Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WUPe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WUPe	Individual bits to enable generation of WakeUp interrupt for WakeUp GPIO configured as input. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Pin cannot generate WakeUp Interrupt (default). 1 = Pin can generate WakeUp Interrupt while MPC5200 is in Deep Sleep mode. **Note:** These enable bits apply ONLY when MPC5200 is in Deep Sleep mode.
8:31	—	Reserved
Note: Only valid when Port Configuration indicates GPIO usage and pin is configured as input in the associated DDR bit in GPIOWDO. Also, Master Interrupt Enable bit in GPIOWME must be set.		

7.3.2.2.6 GPW WakeUp GPIO Individual Interrupt Enable Register —MBAR + 0x0C14

Table 7-42. GPW WakeUp GPIO Individual Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WINe								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WINe	Individual bits to enable generation of Simple interrupt for WakeUp GPIO configured as input. Bit 0 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 controls GPIO_WKUP_5 (PSC6_1 pin) Bit 3 controls GPIO_WKUP_4 (PSC6_0 pin) Bit 4 controls GPIO_WKUP_3 (ETH_17 pin) Bit 5 controls GPIO_WKUP_2 (PSC3_9 pin) Bit 6 controls GPIO_WKUP_1 (PSC2_4 pin) Bit 7 controls GPIO_WKUP_0 (PSC1_4 pin) 0 = Pin cannot generate Simple Interrupt (default). 1 = Pin can generate Simple Interrupt while MPC5200 is **not** in Deep Sleep mode. **Note:** These enable bits apply **only** when MPC5200 is **not** in Deep Sleep mode.
8:31	—	Reserved
Note: Only valid when Port Configuration indicates GPIO usage and pin is configured as input in the associated DDR bit in GPIOWDO. Also, Master Interrupt Enable bit in GPIOWME must be set.		

7.3.2.2.7 GPW WakeUp GPIO Interrupt Types Register—MBAR + 0x0C18

Table 7-43. GPW WakeUp GPIO Interrupt Types Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Ityp7		Ityp6		Ityp5		Ityp4		Ityp3		Ityp2		Ityp7		Ityp0	
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	Ityp7	GPIO Interrupt Type bits for WakeUp GPIO pins 7–0 00=Interrupt at any transition 01=Interrupt on rising edge 10=Interrupt on falling edge 11=Interrupt on pulse (any 2 transitions) The above interrupt types describe operation for interrupts occuring while MPC5200 is **not** in Deep Sleep mode (i.e., Simple Interrupt types). For operation while in Deep Sleep mode the interpretation of these bits is slightly different, because no clocking is present in this mode and it is therefore impossible to detect an edge on the input. For Deep Sleep mode the bits are interpretted as follows: 00 = Not Valid, no interrupt can be detected 01 = Level High, any high creates WakeUp from Deep Sleep 10 = Level Low, any low creates WakeUp from Deep Sleep 11 = Not Valid, no interrupt can be detected. ITYP7 controls GPIO_WKUP_7 (GPIO_WKUP_7 pin) ITYP6 controls GPIO_WKUP_6 (GPIO_WKUP_6 pin) ITYP5 controls GPIO_WKUP_5 (PSC6_1 pin) ITYP4 controls GPIO_WKUP_4 (PSC6_0 pin) ITYP3 controls GPIO_WKUP_3 (ETH_17 pin) ITYP2 controls GPIO_WKUP_2 (PSC3_9 pin) ITYP1 controls GPIO_WKUP_1 (PSC2_4 pin) ITYP0 controls GPIO_WKUP_0 (PSC1_4 pin) **Note:** Any GPIO WakeUp interrupt creates a Main Level 2 interrupt in the Interrupt Controller.
2:3	Ityp6	
4:5	Ityp5	
6:7	Ityp4	
8:9	Ityp3	
10:11	Ityp2	
12:13	Ityp1	
14:15	Ityp0	
16:31	—	Reserved

7.3.2.2.8 GPW WakeUp GPIO Master Enables Register —MBAR + 0x0C1C

Table 7-44. GPW WakeUp GPIO Master Enables Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							ME	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	—	Reserved
7	ME	WakeUp GPIO Master Enable pin. This pin must be high before **any** WakeUp GPIO pin can generate an interrupt. This bit should remain clear while programming individual interrupts and then set high as a final step. This prevents any spurious interrupt occuring during programming.
8:31	—	Reserved

7.3.2.2.9 GPW WakeUp GPIO Data Input Values Register —MBAR + 0x0C20

Table 7-45. GPW WakeUp GPIO Data Input Values Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WIVAL								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	WIVAL	Input Value bits for GPIO WakeUp pins 7–0. This is the raw state of the input pin at the time this register is read. It is not latched to the state that caused the interrupt (if any). This status bit is always available, regardless of any enable or setting. For example, even if the pin is not used as GPIO. Writing to this byte has no effect. Bit 0 reflects GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 reflects GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 reflects GPIO_WKUP_5 (PSC6_1 pin) Bit 3 reflects GPIO_WKUP_4 (PSC6_0 pin) Bit 4 reflects GPIO_WKUP_3 (ETH_17 pin) Bit 5 reflects GPIO_WKUP_2 (PSC3_9 pin) Bit 6 reflects GPIO_WKUP_1 (PSC2_4 pin) Bit 7 reflects GPIO_WKUP_0 (PSC1_4 pin)
8:31	—	Reserved

7.3.2.2.10 GPW WakeUp GPIO Status Register—MBAR + 0x0C24

Table 7-46. GPW WakeUp GPIO Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Istat								Reserved							
W	rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc								
RESET:	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	Istat	Interrupt status bits for GPIO WakeUp pins 7–0. 1 indicates an interrupt occurred. Cleared with a sticky-bit write to a 1 to clear the interrupt condition. Bit 0 reflects interrupt on GPIO_WKUP_7 (GPIO_WKUP_7 pin) Bit 1 reflects interrupt on GPIO_WKUP_6 (GPIO_WKUP_6 pin) Bit 2 reflects interrupt on GPIO_WKUP_5 (PSC6_1 pin) Bit 3 reflects interrupt on GPIO_WKUP_4 (PSC6_0 pin) Bit 4 reflects interrupt on GPIO_WKUP_3 (ETH_17 pin) Bit 5 reflects interrupt on GPIO_WKUP_2 (PSC3_9 pin) Bit 6 reflects interrupt on GPIO_WKUP_1 (PSC2_4 pin) Bit 7 reflects interrupt on GPIO_WKUP_0 (PSC1_4 pin)
8:31	—	Reserved

7.4 General Purpose Timers (GPT)

Eight (8) General-Purpose Timer (GPT) pins are configurable for:

- Input Capture
- Output Compare
- Pulse Width Modulation (PWM) Output
- Simple GPIO
- Internal CPU timer
- Watchdog Timer (on GPT0 only)

Timer modules run off the internal IP bus clock. Each Timer is associated to a single I/O pin. Each Timer has a 16-bit prescaler and 16-bit counter, thus achieving a 32-bit range (but only 16-bit resolution).

7.4.1 Timer Configuration Method

Use the following method to configure each timer:

1. Determine the Mode Select field (Timer_MS) value for the desired operation.
2. Program any other registers associated with this mode.
3. Program Interrupt enable as desired.
4. Enable the Timer by writing the Mode Select value into the Timer_MS field.

7.4.2 Mode Overview

The following gives a brief description of the available modes:

1. **Input Capture**—In this mode the I/O pin is an Input. Once enabled, the counters run until the specified "Capture Event" occurs (rise, fall, either, or pulse). At the Capture Event, the counter value is latched in the status register. If enabled, a CPU interrupt is generated. The GP Timers 6 & 7 are active during low power modes (except for deep sleep), and therefore have the ability to initiate a wake up the device from a low-power mode.
2. **Output Compare**—In this mode the I/O pin is an Output. When enabled the counters run until they reach the programmed Terminal Count value. At this point, the specified "Output Event" is generated (toggle, pulse hi, or pulse low). If enabled, a CPU interrupt is generated.
3. **PWM**—In this mode the I/O pin is an Output. The user can program "Period" and "Width" values to create an adjustable, repeating output waveform on the I/O pin. A CPU interrupt can be generated at the beginning of each PWM Period, at which time a new Width value can be loaded. The new Width value, which represents "ON time", is automatically applied at the beginning of the **next** period. Note that there is no interrupt at the beginning of the first PWM Period. This mode is suitable for PWM audio encoding.
4. **Simple GPIO**—In this mode the I/O pin operates as a GPIO pin. It can be specified as Input or Output, according to the programmable GPIO field. GPIO mode is mutually exclusive of modes 1 through 3 (listed above). In GPIO mode, modes 5 through 6 (listed below) remain available.
5. **CPU Timer**—The I/O pin is not used in this mode. Once enabled, the counters run until they reach a programmed Terminal Count. When this occurs, an interrupt can be generated to the CPU. This Timer mode can be used simultaneously with the Simple GPIO mode.
6. **Watchdog Timer**—This is a special CPU Timer mode, available only on Timer 0. The user must enable the Watchdog Timer mode, which is not active upon reset. The Terminal Count value is programmable. If the counter is allowed to expire, a full MPC5200 reset occurs. To prevent the Watchdog Timer from expiring, software must periodically write a specific value to a specific register (in Timer 0). This causes the counter to reset.

7.4.3 Programming Notes

Programmers should observe the following notes:

1. Intermediate values of the Timer internal counters are **not** readable by software.
2. The Stop_Cont bit operates differently for different modes. In general, this bit controls whether the Timer halts at the end of a current mode, or resets and continues with a repetition of the mode. See the Bit Description for precise operation.
3. The Timer_MS field operates somewhat as a Global Enable. If it is zero, then all Timer modes are disabled and internal counters are reset. See the Bit Descriptions for more detail.
4. There is a CE (Counter Enable) bit that operates somewhat independently of the Timer_MS field. This bit controls the Counter for CPU Timer or Watchdog Timer modes only. See the Bit Descriptions to understand the operation of these bits across the various modes.

7.4.4 GPT Registers—MBAR + 0x0600

Each GPT uses 4 32-bit registers. These registers are located at an offset from MBAR of 0x0600. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0600 + register address**

Hyperlinks to the Interrupt Controller registers are provided below:

- GPT 0 Enable and Mode Select Register (0x0600)
- GPT 1 Enable and Mode Select Register (0x0610)
- GPT 2 Enable and Mode Select Register (0x0620)
- GPT 3 Enable and Mode Select Register (0x0630)
- GPT 4 Enable and Mode Select Register (0x0640)
- GPT 5 Enable and Mode Select Register (0x0650)
- GPT 6 Enable and Mode Select Register (0x0660)
- GPT 7 Enable and Mode Select Register (0x0670)
- GPT 0 PWM Configuration Register (0x0608)
- GPT 1 PWM Configuration Register (0x0618)
- GPT 2 PWM Configuration Register (0x0628)
- GPT 3 PWM Configuration Register (0x0638)
- GPT 4 PWM Configuration Register (0x0648)
- GPT 5 PWM Configuration Register (0x0658)
- GPT 6 PWM Configuration Register (0x0668)
- GPT 7 PWM Configuration Register (0x0678)
- GPT 0 Counter Input Register (0x0604)
- GPT 1 Counter Input Register (0x0614)
- GPT 2 Counter Input Register (0x0624)
- GPT 3 Counter Input Register (0x0634)
- GPT 4 Counter Input Register (0x0644)
- GPT 5 Counter Input Register (0x0654)
- GPT 6 Counter Input Register (0x0664)
- GPT 7 Counter Input Register (0x0674)
- GPT 0 Status Register (0x060C)
- GPT 1 Status Register (0x061C)
- GPT 2 Status Register (0x062C)
- GPT 3 Status Register (0x063C)
- GPT 4 Status Register (0x064C)
- GPT 5 Status Register (0x065C)
- GPT 6 Status Register (0x066C)
- GPT 7 Status Register (0x067C)

7.4.4.1 GPT 0 Enable and Mode Select Register—MBAR + 0x0600
GPT 1 Enable and Mode Select Register—MBAR + 0x0610
GPT 2 Enable and Mode Select Register—MBAR + 0x0620
GPT 3 Enable and Mode Select Register—MBAR + 0x0630
GPT 4 Enable and Mode Select Register—MBAR + 0x0640
GPT 5 Enable and Mode Select Register—MBAR + 0x0650
GPT 6 Enable and Mode Select Register—MBAR + 0x0660
GPT 7 Enable and Mode Select Register—MBAR + 0x0670

Table 7-47. GPT 0 Enable and Mode Select Register
GPT 1 Enable and Mode Select Register
GPT 2 Enable and Mode Select Register
GPT 3 Enable and Mode Select Register
GPT 4 Enable and Mode Select Register
GPT 5 Enable and Mode Select Register
GPT 6 Enable and Mode Select Register
GPT 7 Enable and Mode Select Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OCPW								Reserved		OCT		Reserved		ICT	
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 lsb
R	WDen	Reserved		CE	Rsvd	Stop_Cont	Open_Drn	IntEn	Reserved		GPIO		Rsvd	Timer_MS		
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	OCPW	Output Compare Pulse Width—Applies to OC Pulse types only. This field specifies the number of IP bus clocks (non-prescaled) to create a short output pulse at each Output Event. This pulse is generated at the end of the OC period and overlays the next OC period (rather than adding to the period). **Note:** This field is alternately used as the Watchdog reset field if Watchdog Timer mode is enabled.
8:9	—	Reserved
10:11	OCT	Output Compare Type—describes action to occur at each output compare event, as follows: 00=Special case, output is immediately forced low without respect to each output compare event. 01=Output pulse highs, initial value is low (OCPW field applies). 10=Output pulses low, initial value is high (OCPW field applies). 11=Output toggles. GPIO modalities can be used to achieve an initial output state prior to enabling OC mode. It is important to move directly from GPIO output mode to OC mode and not to pass through the Timer_MS=000 state. To prevent the Internal Timer Mode from engaging during the GPIO state, CE bit should be held low during the configuration steps. GPIO initialization is needed when presetting the I/O to 1 in conjunction with a simple toggle OCT setting. **Note:** For Stop Mode operation (see Stop_Cont bit below) it is necessary to pass through the mode_sel = 0 state to restart the output compare counters with their programmed values. See prescale and count fields in GPT 0 Counter Input Register.
12:13	—	Reserved
14:15	ICT	Input Capture Type—describes the input transition type required to trigger an input capture event, as follows: 00=Any input transition causes an IC event. 01=IC event occurs at input rising edge. 10=IC event occurs at input falling edge. 11=IC event occurs at any input pulse (i.e., at 2nd input edge). **BE AWARE:** For ICT=11 (pulse capture), status register records only the pulse width.
16	WDen	Watchdog enable—bit enables watchdog operation. A timer expiration causes an internal MPC5200 reset. Watchdog operation requires the Timer_MS field be set for internal timer mode and the CE bit to be set high. In this mode the OCPW byte field operates as a watchdog reset field. Writing A5 to the OCPW field resets the watchdog timer, preventing it from expiring. As long as the timer is properly configured, the watchdog operation continues. This bit (and functionality) is implemented only for Timer 0. 1 = enabled
17:18	—	Reserved
19	CE	Counter Enable—bit enables or resets the internal counter during Internal timer modes only. CE must be high to enable these modes. If low, counter is held in reset. This bit is secondary to the timer mode select bits (Timer_MS). If Timer_MS is1XX, internal timer modes are enabled. CE can then enable or reset the internal counter without changing the Timer_MS field. GPIO operation is also available in this mode. 1 = enabled
20	—	Reserved

Bit	Name	Description
21	Stop_Cont	Stop Continuous—Applies to multiple modes, as follows: 0 = Stop 1 = Continuous • IC mode Stop operation—At each IC event, counter is reset. Continuous operation—counter is not reset at each IC event. Effect is to create Status count values that are cumulative between Capture events. If the special Pulse Mode Capture type is specified, the Stop_Cont bit is not used, operation fixed as if it were Stop. • OC mode Stop operation—Counter resets and stops at first OC event. Note: Software needs to pass through Timer_MS=000 state to restart timer. Continuous operation—counter resets and continues at each OC event. Effect to is create back-to-back periodic OC events. **BE AWARE**—In this mode the polarity of Stop_cont is reversed. Also, in Stop Mode, the output event falsely retriggers at the expiration of the prescale count. This means the software has to service and output event prior to the prescale expiring. Service is defined as programming mode_sel field to 0, which causes the programmed prescale and count values to be reset. • PWM mode Bit not used, operation is always Continuous. • CPU Timer mode Stop operation—On counter expiration, Timer waits until Status bit is cleared by passing through Timer_MS=000 state before beginning a new cycle. Continuous operation—On counter expiration, Timer resets and immediately begin a new cycle. Effect is to generate fixed periodic timeouts. • WatchDog Timer and GPIO modes Bit not used.
22	Open_Drn	Open Drain 0 = Normal I/O 1 = Open Drain emulation—affects all modes that drive the I/O pin (GPIO, OC, & PWM). Any output “1” is converted to a tri-state at the I/O pin.
23	IntEn	Enable interrupt—enables interrupt generation to the CPU for all modes (IC, OC, PWM, and Internal Timer). IntEn is not required for watchdog expiration to create a reset. 1 = enabled
24:25	—	Reserved
26:27	GPIO	GPIO mode type. Simple GPIO functionality that can be used simultaneously with the Internal Timer mode. It is not compatible with IC, OC, or PWM modes, since these modes dictate the usage of the I/O pin. 0x=Timer enabled as simple GPIO input 10=Timer enabled as simple GPIO output, value=0 11=Timer enabled as simple GPIO output, value=1 (tri-state if Open_Drn=1) While in GPIO modes, internal timer mode is also available. To prevent undesired timer expiration, keep the CE bit low.

Bit	Name	Description
28	—	Reserved
29:31	Timer_MS	Timer Mode Select (and module enable). 000=Timer module not enabled. Associated I/O pin is in input state. All Timer operation is completely disabled. Control and status registers are still accessible. This mode should be entered when timer is to be re-configured, except where the user does not want the I/O pin to become an input. 001=Timer enabled for input capture. 010=Timer enabled for output compare. 011=Timer enabled for PWM. 1xx=timer enabled for simple GPIO. Internal timer modes available. CE bit controls timer counter.

7.4.4.2 GPT 0 Counter Input Register—MBAR + 0x0604
GPT 1 Counter Input Register—MBAR + 0x0614
GPT 2 Counter Input Register—MBAR + 0x0624
GPT 3 Counter Input Register—MBAR + 0x0634
GPT 4 Counter Input Register—MBAR + 0x0644
GPT 5 Counter Input Register—MBAR + 0x0654
GPT 6 Counter Input Register—MBAR + 0x0664
GPT 7 Counter Input Register—MBAR + 0x0674

Table 7-48. GPT 0 Counter Input Register
GPT 1 Counter Input Register
GPT 2 Counter Input Register
GPT 3 Counter Input Register
GPT 4 Counter Input Register
GPT 5 Counter Input Register
GPT 6 Counter Input Register
GPT 7 Counter Input Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Prescale															
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 lsb
R	Count															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	Prescale	Prescale amount applied to internal counter (in IP bus clocks). **BE AWARE**—The prescale field should be written prior to enabling any timer mode. A prescale of 0x0001 means one IP bus clock per count increment. If prescale is 0 when any timer mode is started, it results in an effective prescale of 64K. The counter will immediately begin and an output event will occur with the 64K prescale, rather than the desired value.
16:31	COUNT	Sets number of prescaled counts applied to reference events, as follows: IC—Field has no effect, internal counter starts at 0. OC—Number of prescaled counts counted before creating output event. PWM—Number of prescaled counts defining the PWM output period. Internal Timer—Number of prescaled counts counted before timer (or watchdog) expires. **Note:** Reading this register only returns the programmed value, intermediate values of the internal counter are not available to software.

7.4.4.3 GPT 0 PWM Configuration Register—MBAR + 0x0608
GPT 1 PWM Configuration Register—MBAR + 0x0618
GPT 2 PWM Configuration Register—MBAR + 0x0628
GPT 3 PWM Configuration Register—MBAR + 0x0638
GPT 4 PWM Configuration Register—MBAR + 0x0648
GPT 5 PWM Configuration Register—MBAR + 0x0658
GPT 6 PWM Configuration Register—MBAR + 0x0668
GPT 7 PWM Configuration Register—MBAR + 0x0678

Table 7-49. GPT 0 PWM Configuration Register
GPT 1 PWM Configuration Register
GPT 2 PWM Configuration Register
GPT 3 PWM Configuration Register
GPT 4 PWM Configuration Register
GPT 5 PWM Configuration Register
GPT 6 PWM Configuration Register
GPT 7 PWM Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WIDTH															
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 lsb
R	Reserved							PWMOP	Reserved							LOAD
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	WIDTH	PWM only. Defines ON time for output in prescaled counts. Similar to count value, which defines the period. ON time overlays the period time. If WIDTH = 0, output is always OFF. If WIDTH exceeds count value, output is always ON. ON and OFF polarity is set by the PWMOP bit.
16:22	—	Reserved

Bit	Name	Description
23	PWMOP	Pulse Width Mode Output Polarity—Defines PWM output polarity for OFF time. Opposite state is ON time polarity. PWM cycles begin with ON time.
24:30	—	Reserved
31	LOAD	Bit forces immediate period update. Bit auto clears itself. A new period begins immediately with the current count and width settings. If LOAD = 0, new count or width settings are not updated until end of current period. **Note:** Prescale setting is not part of this process. Changing prescale value while PWM is active causes unpredictable results for the period in which it was changed. The same is true for PWMOP bit.

7.4.4.4 GPT 0 Status Register—MBAR + 0x060C
GPT 1 Status Register—MBAR + 0x061C
GPT 2 Status Register—MBAR + 0x062C
GPT 3 Status Register—MBAR + 0x063C
GPT 4 Status Register—MBAR + 0x064C
GPT 5 Status Register—MBAR + 0x065C
GPT 6 Status Register—MBAR + 0x066C
GPT 7 Status Register—MBAR + 0x067C

This is a read-only register.

Table 7-50. GPT 0 Status Register
GPT 1 Status Register
GPT 2 Status Register
GPT 3 Status Register
GPT 4 Status Register
GPT 5 Status Register
GPT 6 Status Register
GPT 7 Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPTURE															
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 lsb
R	Rsvd	OVF			Reserved			PIN	Reserved				TEXP	PWMP	COMP	CAFT
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	Capture	Read of internal counter, latch at reference event. This is pertinent only in IC mode, in which case it represents the count value at the time the Input Event occurred. Capture status does not shadow the internal counter while an event is pending, it is updated only at the time the Input Event occurs. **Note:** If ICT is set to 11, which is Pulse Capture Mode, the Capture value records the width of the pulse. Also, the Stop_Cont bit is irrelevant in Pulse Capture Mode, operation is as if Stop_Cont were 0.
16	—	Reserved

Bit	Name	Description
17:19	OVF	Represents how many times internal counter has rolled over. This is pertinent only during IC mode and would represent an extremely long period of time between Input Events. However, if Stop_Cont = 1 (indicating cumulative reporting of Input Events), this field could come into play. **Note:** This field is cleared by any “sticky bit” status write in the 4 bit fields below (28, 29, 30, 31).
20:22	—	Reserved
23	PIN	Registered state of the I/O PIN (all modes). The IP bus Clock registers the state of the I/O input. Valid, even if Timer is not enabled.
24:27	—	Reserved
28	TEXP	Timer Expired in Internal Timer mode. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note.
29	PWMP	PWM end of period occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note.
30	COMP	OC reference event occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note.
31	CAPT	IC reference event occurred. Cleared by writing 1 to this bit position. Also cleared if Timer_MS is 000 (i.e., Timer not enabled). See Note.
Note: To clear any of these bits, it is necessary to clear **all** of them. An F must be written to bits 28:31.		

7.5 Slice Timers

Two Slice Timers are included to provide shorter term periodic interrupts. Each timer consists of a 24-bit counter with **no** prescale. Running off the IP bus clock, each timer can generate interrupts from 7.75 uS to 508 mS in 30 nS steps (based on 33 MHz IP bus clock). The counters count up from zero and expire/interrupt when they reach the programmed terminal count. They can be configured to automatically reset to zero and resume counting or wait until the Status/Interrupt is serviced before beginning a new cycle.

The current count value can be read without disturbing the count operation. Each Slice Timer has a Status bit to indicate the Timer has expired. If enabled, a CPU interrupt is generated at count expiration. Each Timer has a separate Interrupt. Slice Timer 0 represents CPU interrupt Critical Level 2 and Slice Timer 1 represents Main Level 0 (which is hardwired to the core_smi pin). Clearing the Status and/or Interrupt is accomplished by writing 1 to the Status bit, or disabling the Timer entirely with the Timer Enable (TE) bit.

As a safety, the Timer does not count until a Terminal Count value of greater than 255 is programmed into it. Also, writing a Terminal Count value of 0 is converted to all 1s, resulting in a maximum duration timeout.

7.5.1 SLT Registers—MBAR + 0x0700

There are two SLT Timers. Each one uses four 32-bit registers. These registers are located at an offset from MBAR of 0x0700. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0700 + register address**

Hyperlinks to the Interrupt Controller registers are provided below:

- SLT 0 Terminal Count Register (0x0700)
- SLT 1 Terminal Count Register (0x0710)
- SLT 0 Control Register (0x0704)
- SLT 1 Control Register (0x0714)
- SLT 0 Count Value Register (0x0708) Read Only
- SLT 1 Count Value Register (0x0718) Read Only
- SLT 0 Timer Status Register (0x070C) Read Only
- SLT 1 Timer Status Register (0x071C) Read Only

7.5.1.1 SLT 0 Terminal Count Register—MBAR + 0x0700
SLT 1 Terminal Count Register—MBAR + 0x0710

Table 7-51. SLT 0 Terminal Count Register
SLT 1 Terminal Count Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								Terminal Count							
W																
RESET:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Terminal Count															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0:7	—	Reserved
8:31	Terminal Count	The user programs this register to set the Terminal Count value to be used by the Timer. This register can be updated even if the Timer is running, the new value takes effect immediately. The internal counter is compared to this register to determine if Terminal Count has been reached. **Note:** The Timer will not begin counting until a value greater than 255 is programmed into the Terminal Count Register. A value less than 255 will essentially suspend the Timer. Writing a value of zero to this register is considered invalid and will be converted to all ones, creating a maximum duration count period. Defaults at reset: TerminalCount will default to all ones, all other control bits willn default to zero.

7.5.1.2 SLT 0 Control Register—MBAR + 0x0704
SLT 1 Control Register—MBAR + 0x0714

Table 7-52. SLT 0 Control Register
SLT 1 Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					Run_Wait	Interrupt Enable	Timer Enable	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5	Run_ Wait	A high indicates the Timer should run continuously while enabled. When the Timer counter reaches terminal count it immediately resets to 0 and resumes counting. If the Run/Wait bit is set low, the Timer Counter expires, but then waits until the Timer is cleared (either by writing 1 to the status bit or by disabling and re-enabling the Timer), before resuming operation.
6	Interrupt Enable	CPU Interrupt is generated only if this bit is high. This bit does **not** affect operation of the Timer Counter or Status Bit registers.
7	Timer Enable	While this bit is high the Timer operates normally, while low the Timer is reset and remains idle.
8:32	—	Reserved

7.5.1.3 SLT 0 Count Value Register—MBAR + 0x0708
SLT 1 Count Value Register—MBAR + 0x0718

Table 7-53. SLT 0 Count Value Register
SLT 1 Count Value Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								TimerCount							
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 lsb
R	TimerCount															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	—	Reserved
8:31	Timer Count	Provides current state of the Timer counter. This register does not chodange while a read is in progress, but the actual Timer counter continues unaffected.

7.5.1.4 SLT 0 Timer Status Register—MBAR + 0x070C
SLT 1 Timer Status Register—MBAR + 0x071C

Table 7-54. SLT 0 Timer Status Register
SLT 1 Timer Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							ST	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	—	Reserved
7	ST	This status bit goes high whenever the Timer has reached Terminal Count. The bit is cleared by writing 1 to its bit position. If Interrupts are enabled, clearing this status bit also clears the interrupt.
8:31	—	Reserved

7.6 Real-Time Clock

The Real-Time Clock (RTC) uses an external 32KHz crystal to provide:

- alarm
- stop-watch
- periodic interrupts
  - minute
  - second
  - midnight rollover(day)

The clock runs as long as power is maintained and the crystal is running, regardless of MPC5200 power-down states.

The RTC module has the following features:

- full clock features
- minute countdown timer—provides 256-minute capability, slightly over 4 hours
- programmable alarm—operates on time of day only, not related to calendar
- periodic interrupts for:
  - 1 second
  - 1 minute
  - 1 day—operates only at midnight rollover
- calendar features:
  - day
  - date
  - year
- Crystal support (32.768KHz only)

RTC registers are writable, letting time and date be updated. If software enabled, RTC operates during all MPC5200 power-down modes. At a reset , control registers are put in a default state such that no interrupts generate until software enabled.

The RTC has two CPU interrupt signals connected to the Interrupt Controller, they are:

- RTC_Periodic, which is Main Level 5 fed by the Day, Minute, or Second sources.
- RTC_Stopwatch, which is Main Level 6 fed by the Alarm or Stopwatch sources.

Periodic interrupts are separately enabled by control bits, and a global enable must be asserted to allow any of the periodic sources to generate a CPU interrupt. Clearing Periodic interrupts is accomplished by writing 1 to the appropriate status bit.

Stopwatch and Alarm interrupts are enabled simply by initiating the function. In the Stopwatch case, this means starting the Stopwatch, in the Alarm case, this means enabling the Alarm. Clearing Stopwatch or Alarm interrupts is accomplished by writing 1 to the appropriate status bit.

Either of the RTC interrupts to the CPU can be used to awaken the MPC5200 from any power down mode.

7.6.1 Real-Time Clock Signals

Table 7-55. Real-Time Clock Signals

Signal	I/O	Definition
RTC_XTAL_IN	I	Real-time Clock External Crystal/External Clock Input
RTC_XTAL_OUT	O	Real-time Clock External Crystal

Figure 7-4 shows a suggested circuit using an Epson® MC-405 32.768 KHz quartz crystal oscillator.

NOTE

External component values are highly dependent on the crystal. These values will be different for different brands of crystals.

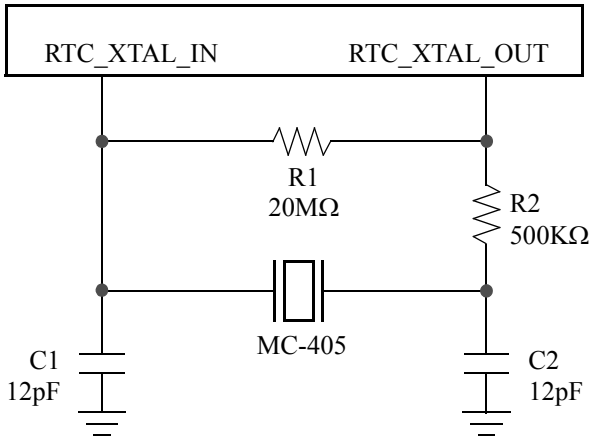


Figure 7-4. Diagram—Suggested Crystal Oscillator Circuit

7.6.2 Programming Note

Accesses to the RTC control registers are performed on the IP bus clock domain, but the RTC itself runs on the (much) slower 32 KHz crystal domain. When software initiates a setting of the Time and/or Date, it must be realized that many IP bus clocks may go by before the setting actually takes effect. If this is a system concern then it is recommended that software poll the Time and/or Date Status fields to confirm the setting has occurred. This requires some careful bit manipulation of the expected status versus the written control values, particularly if the output status is designated as 12-Hour format (input control format is always 24-Hour).

It should be noted that updates to the RTC control registers, such as time and date set, must be synchronized with the 32KHz clock domain. It can take four 32KHz clock cycles for this synchronizing hand shake to complete. Multiple time/date updates made within this four clock synchronizing period may not be properly accepted by the RTC logic.

7.6.3 RTC Interface Registers—MBAR + 0x0800

RTC uses 8 32-bit registers. These registers are located at an offset from MBAR of 0x0800. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0800 + register address**

Hyperlinks to the Interrupt Controller registers are provided below:

- RTC Time Set Register (0x0800)
- RTC Date Set Register (0x0804)
- RTC Current Date Register (0x0814), read-only
- RTC Alarm and Stopwatch Interrupt Register (0x0818), read-only

- RTC New Year and Stopwatch Register (0x0808)
- RTC Alarm and Interrupt Enable Register (0x080C)
- RTC Current Time Register (0x0810), read-only
- RTC Periodic Interrupt and Bus Error Register (0x081C), read-only
- RTC Test Register/Divides Register (0x0820)

7.6.3.1 RTC Time Set Register—MBAR + 0x0800

Table 7-56. RTC Time Set Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						set_time	pause_time	Reserved		SlctHour	C24Hour_set				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

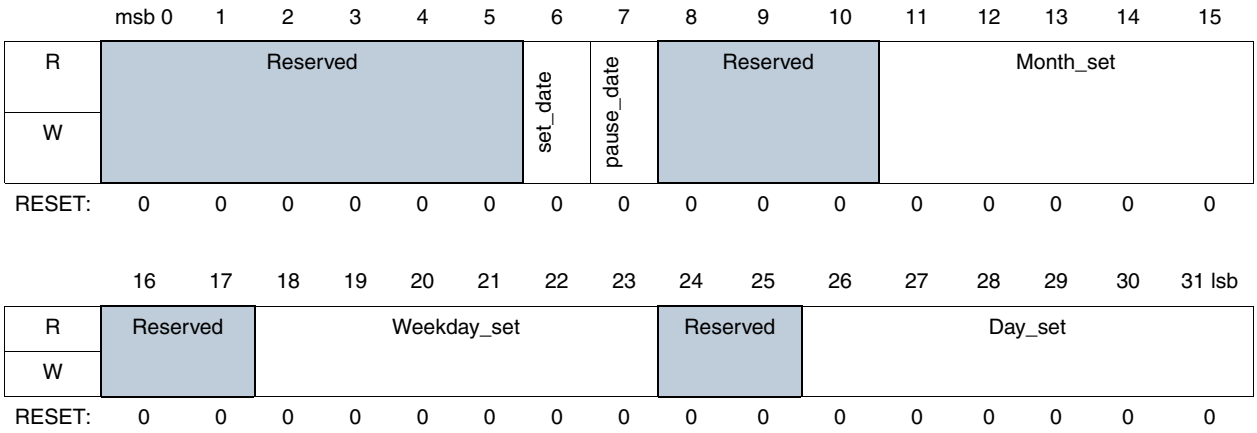
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved		Minute_set						Reserved		Second_set					
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:5	—	Reserved
6	set_time	A bit used in conjunction with pause_time bit (below) to cause a new time to be programmed into the RTC. After a proper software sequence, the values in the *_set fields below are loaded. The proper software sequence is: 1. Write register with pause_time 1 and set_time 0 2. Write register with pause_time 1 and set_time 1 3. Write register with pause_time 1 and set_time 0 4. Write register with pause_time 0 and set_time 0 At completion of Step 4, RTC is updated with the new time. The C24Hour_set, Minute_set, and the Second_set fields should remain consistent values throughout the four steps (i.e., at the desired new time values). **Note:** Read-modify-write operations may disrupt this procedure, it is advised that four simple writes occur. Byte writes to this byte are also acceptable.
7	pause_time	Used with set_time above to perform time update. Must be zero for normal operation.
8:9	—	Reserved
10	SlctHour	This bit determines the hour output format. • low bit = 24-hour format • high bit = 12-hour format with AM/PM **Note:** This bit does NOT affect time set procedure, it only affects how the Hour Status field is presented.
11:15	C24Hour_set	Hour in 24-hour format written in RTC after successful state machine transition by set_time and pause_time bits. **Note:** This field is always written with 24-Hour format, it is NOT affected by SlctHour bit above.
16:17	—	Reserved

Bits	Name	Description
18:23	Minute_set	Minute written in RTC after successful state machine transition by set_time and pause_time bits.
24:25	—	Reserved
26:31	Second_set	Second written in RTC after successful state machine transition by set_time and pause_time bits.

7.6.3.2 RTC Date Set Register—MBAR + 0x0804

Table 7-57. RTC Date Set Register



Bits	Name	Description
0:5	—	Reserved
6	set_date	Operation of pause_date and set_date is similar to pause_time and set_time described in the time set register.
7	pause_date	Used with set_date above to perform date update. Must be zero for normal operation.
8:10	—	Reserved
11:15	Month_set	New month written in RTC after successful state machine transition by set_date and pause_date bits. Actually the lower 4 bits is used
16:17	—	Reserved
18:23	Weekday_set	New weekday written in RTC after state machine transition by set_date and pause_date bits. 1 = Monday; 7 = Sunday. Actually the lower 3 bits is used.
24:25	—	Reserved
26:31	Date_set	New date written in RTC after state machine transition by set_date and pause_date bits. Actually the lower 5 bits is used. **Note:** Year_set in the following register is also part of the date set function.

7.6.3.3 RTC New Year and Stopwatch Register—MBAR + 0x0808

Table 7-58. RTC New Year and Stopwatch Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							write_SW	SW_set							
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 lsb
R	Reserved				Year_set											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	write_SW	Typical stopwatch operation is to write initial value into 8-bit wide SW_set and assert write_SW bit. The write_SW bit is immediately auto cleared, but it triggers the stopwatch minute countdown to begin.
8:15	SW_set	Number of minutes to be written into stopwatch. Max is 255, a little over 4 hours.
16:19	—	Reserved
20:31	Year_set	New year written in RTC after successful state machine transition by set_date and pause_date bits. **Note:** This is part of date set function in the previous register.

7.6.3.4 RTC Alarm and Interrupt Enable Register—MBAR + 0x080C

Table 7-59. RTC Alarm and Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							Alm_enable	Reserved			Alm_24H_set				
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 lsb
R	Reserved		Alm_Min_set						Reserved				MPEb	IntEn_day	IntEn_min	IntEn_sec
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	Alm_enable	Alarm Enable bit for once-a-day Alarm. If high, Alarm status/interrupt operation is enabled. If low, Alarm setting is not compared to time of day.

Bits	Name	Description
8:10	—	Reserved
11:15	Alm_24Hset	Hour setting (in 24 hour format) to be compared to time of day for the purpose of generating Alarm Status/Interrupt. Can be written at any time.
16:17	—	Reserved
18:23	Alm_Min_set	Minute setting to be compared to time of day for the purpose of generating Alarm Status/Interrupt. Can be written at any time.
24:27	—	Reserved
28	MPEb	Master Periodic Enable bar. Must be written low after reset to allow periodic interrupts.
29	IntEn_day	Enable bit of periodic interrupts at midnight rollover.
30	IntEn_min	Enable bit of periodic interrupts at minute rollover.
31	IntEn_sec	Enable bit of periodic interrupts at second rollover.
Note: The Interrupt enable bits (28, 29, 30, 31) control the Periodic Interrupt coming from the RTC. The separate Stopwatch/Alarm Interrupt signal does not have a specific interrupt enable bit. An Alarm interrupt is automatically generated if Alarm is enabled and the Alarm setting matches time of day. Similarly, a Stopwatch expiration, which shares the Alarm interrupt signal, automatically occurs once the Stopwatch is initiated and the Stopwatch counter expires.		

7.6.3.5 RTC Current Time Register—MBAR + 0x0810

This is a read-only register.

Table 7-60. RTC Current Time Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved											Hour				
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 lsb
R	Reserved		Minute						Reserved		Second					
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

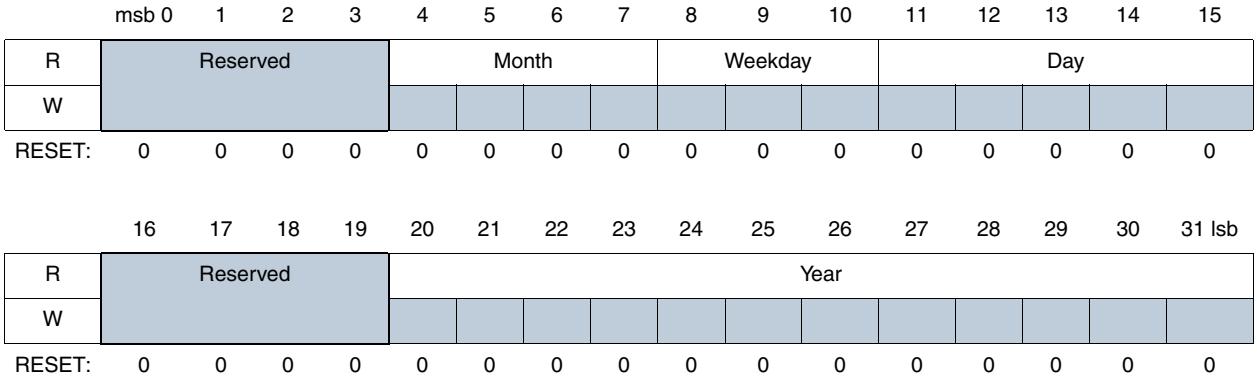
Bits	Name	Description
0:10	—	Reserved
11:15	Hour	Hour format can be either 24-hour or 12-hour with AM/PM. If 24-hour format is selected (SlctHour low in Reg 0), the whole 5-bit hour field designates current time in 24-hour format. If 12-hour format is selected (SlctHour high in Reg 0), the MSB of hour field indicates: • Hour[0]=0: AM, • Hour[0]=1: PM and • Hour[1:4] designates current time in 12-hour format.
16:17	—	Reserved
18:23	Minute	Shows minutes in current time.

Bits	Name	Description
24:25	—	Reserved
26:31	Second	Shows seconds in current time.

7.6.3.6 RTC Current Date Register—MBAR + 0x0814

This is a read-only register.

Table 7-61. RTC Current Date Register

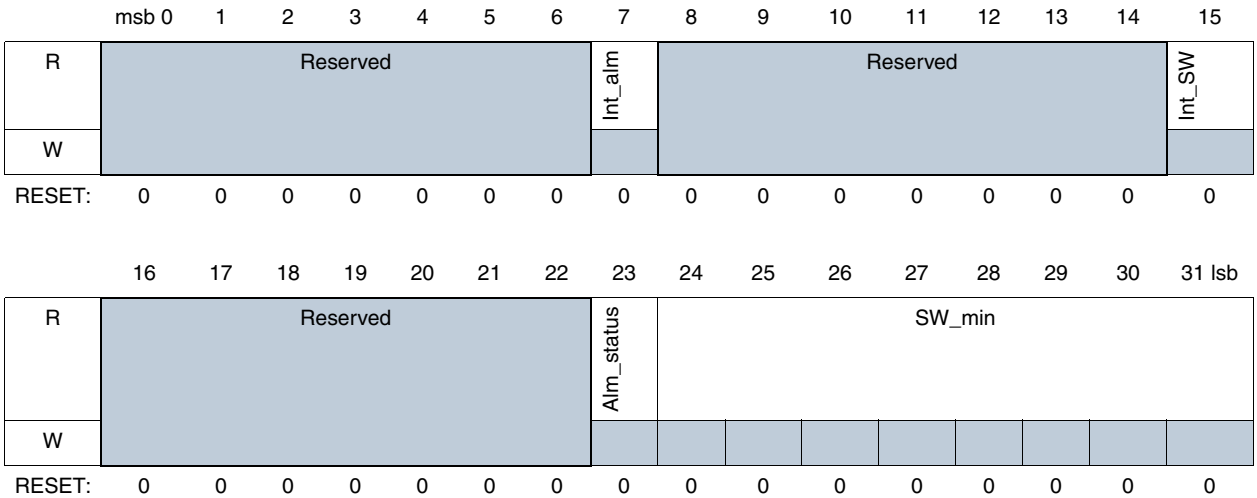


Bits	Name	Description
0:3	—	Reserved
4:7	Month	Shows current month. 1 = January; 12 = December
8:10	Weekday	Indicates day of week. (Monday = 1, Sunday = 7)
11:15	Date	Shows current date. Calendar feature is implemented, therefore, day rollover at the end of month including February (and Leap Years) is automatic.
16:19	—	Reserved
20:31	Year	Shows current year. Max is 4052.

7.6.3.7 RTC Alarm and Stopwatch Interrupt Register—MBAR + 0x0818

This is a read-only register.

Table 7-62. RTC Alarm and Stopwatch Interrupt Register



Bits	Name	Description
0:6	—	Reserved
7	Int_alm	Status bit indicating that enabled once-a-day Alarm has occurred (active high). Alarm interrupt has been activated. This bit and the Interrupt is cleared by writing 1 to this bit position. **Note:** A Stopwatch interrupt, if also active, must be cleared before the interrupt signal to the CPU is negated.
8:14	—	Reserved
15	Int_SW	Status bit indicating that Stopwatch expiration has occurred (active high). Stopwatch interrupt has been activated. This bit and the Interrupt are cleared by writing 1 to this bit position. **Note:** An Alarm interrupt, if also active, must be cleared before the interrupt signal to the CPU is negated.
16:22	—	Reserved
23	Alm_status	Status bit indicating that once-a-day Alarm has occurred. Same as Int_alm bit above except that clearing this bit does NOT clear the interrupt.
24:31	SW_min	Minutes remaining in stopwatch.

7.6.3.8 RTC Periodic Interrupt and Bus Error Register—MBAR + 0x081C

This is a read-only register.

Table 7-63. RTC Periodic Interrupt and Bus Error Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							Bus_error_1	Reserved							Int_day
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 lsb
R	Reserved							Int_min	Reserved							Int_sec
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	Bus_error_1	Internal status register—If high, indicates software has attempted a write access to a read-only register in this module. No actual register contents are corrupted if this happens. Cleared by writing 1 to this bit position.
8:14	—	Reserved
15	Int_day	Periodic interrupt at midnight. High indicates interrupt has occurred. OR'd function of Int_day, Int_min and Int_sec produces RTC periodic interrupt to CPU interface. Cleared by writing 1 to this bit position.

Bits	Name	Description
16:22	—	Reserved
23	Int_min	Periodic interrupt at each minute rollover. High indicates interrupt has occurred. Cleared by writing 1 to this bit position.
24:30	—	Reserved
31	Int_sec	Periodic interrupt at each second rollover. High indicates interrupt has occurred. Cleared by writing 1 to this bit position.

7.6.3.9 RTC Test Register/Divides Register—MBAR + 0x0820

This register is used during manufacturing test to expedite RTC testing and is not intended to be a user register. However, no protection from software access is provided.

Table 7-64. RTC Test Register/Divides Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	PTERM							ETERM							
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	—	Reserved
1:7	PTERM	Prescale Termination value, the number of 32KHz clocks per 7-bit prescale counter. Default at reset is the maximum (and proper) value of 128 decimal. Any value lower than this causes the RTC to run fast.
8:15	ETERM	External Termination value, the number of prescaled counts per 8-bit external counter. Default at reset is the maximum (and proper) value of 256 decimal. Any value lower than this causes the RTC to run fast.
16:31	—	Reserved
Note: The 32.768KHz crystal frequency is divided by PTERM, which is then divided by ETERM to produce a 1 second time interval. It is conceivable that a system might wish to adjust these values to produce a more locally accurate clock rate. However, be aware that these values are affected by reset. Therefore, any adjustment value must be stored and retrieved from non-volatile memory. Further, the adjustment could only increase the clock rate, not decrease it.		

Chapter 8
SDRAM Memory Controller

8.1 Overview

The following sections are contained in this document:

8.2 Terminology and Notation

Synchronous DRAM devices (SDR-SDRAM, DDR-SDRAM) are organized internally as columns by rows by “banks”. Older type asynchronous DRAMs (FP, EDO) had rows and columns, but no internal banks. Historically, the word “bank” was often used to refer to the set of memory devices all activated for the same address range (same $\overline{\text{RAS}}$). To avoid confusion between these two meanings of “bank”, this document uses the term “bank” for the internal banks of an SDRAM device, and the term “space” to indicate the memory device(s) activated for a common address range (same $\overline{\text{CS}}$).

8.2.1 “Endian”-ness

Endian-ness is a source of seemingly endless confusion, yet it need not be. The source of the confusion usually seems to be that bit number and/or byte address are improperly equated with significance. In fact, bit number and byte address ***neither govern, nor imply***, significance.

- Significance can ***only*** exist within an ***arithmetic*** context. An arithmetic context can be explicit or implicit.
- An ***explicit*** arithmetic context is the scope of an arithmetic ***operator***, that is, the operand(s) and result(s).
- An ***implicit*** arithmetic context exists within any collection of bits representing an atomic arithmetic object, that is, a ***number***. Significance does ***not*** extend beyond the boundaries of the bit range.
- With a single exception, an arithmetic context, and therefore significance, can ***only*** exist within an ***execution*** context, that is, an abstract process or the actual hardware to which it is mapped. The single exception is an implicit context: A ***byte*** is an implicit arithmetic context.
- Within an atomic object “obj[m:n]”, in an arithmetic context, the most significant bit(byte) is on the ***left*** (m), and the least significant bit(byte) is on the ***right*** (n), unless otherwise specified. The bit numbering and byte addressing order from left to right (ascending or descending) is ***strictly cosmetic***.

  Note that “:” in “m:n” is an arithmetic operator, and therefore m and n are arithmetic objects within the operator scope. (If they weren’t, then the terms “ascending” and “descending” would be meaningless.)

  Note furthermore that the scope of the “:” operator does not include “obj”: obj could be of a non-arithmetic type. One particular value (pattern of the bit range) of obj could represent “red”; another value might represent “cold”. The bit patterns are ***enumerations*** of the legal values of obj; sometimes arithmetic operations on enumeration values are valid for the concepts they represent, sometimes not.

  An enumeration value is not an arithmetic context. An enumeration value is a representation of an object value; the nature of the object itself need not be numeric.

The enumeration value of a collection of bits (e.g. process variable) is only meaningful ***in the context*** of a process which manipulates those bits in a manner consistent with the concept they represent. (A process itself may be just a concept represented by a collection of bits manipulated (by a processing unit) in a manner consistent with the concept they represent.) And ***in the context*** of the concept they represent, bits and bytes may have significance.

But while transporting bits from one location to another, the hardware transport media almost never have any knowledge of the concepts represented by the data, or the contexts in which they are valid (this does not include protocol bits of the media, which may be added and

stripped along the way). Nor is the transportation of data an execution context. Without knowledge of atom boundaries and significance (if any), the following convention is the *de facto* standard:

- "**Bit significance, byte address**": From every observation point in a system, the relative ***address order*** of bytes shall be maintained, and the relative ***significance*** of bits ***within*** each individual byte shall be maintained, ***as if*** they represented an 8 bit unsigned binary integer. This is the implicit arithmetic context of bytes. The "native" bit numbering and address significance order of different observers shall have ***no bearing*** on the byte address or bit significance order of visible data.

  Byte "swapping", the intentional transposition of bytes' relative addresses between a source and a destination to maintain inter-byte significance, is ***improper***.

  Bit "swizzling", the intentional renumbering of bit positions, is perfectly legal ***if necessary*** to maintain intra-byte bit significance or inter-byte address order. When necessary, it is required; when not necessary, it is prohibited.

To correctly join data path segments in accordance with this convention, the bit significance ***and byte addressing*** of each segment must be specified.

In this document, significance is always msb on the left, lsb on the right, if any significance relationship exists.

All multi-bit components of the internal XL bus are defined with bit numbers and byte addresses (if any) ***ascending*** from left to right: XLA[0:31], XLD[0:63]. The address of byte XLD[0:7] is a modulo 8 boundary, 8*n* (0x00, 0x08, 0x10, 0x18); the address of byte XLD[56:63] is a modulo 8 boundary plus offset 7, 8*n*+7 (0x07, 0x0F, 0x17, 0x1F).

All internal IP busses are defined with bit numbers ***descending*** from left to right: IPA[31:0], IPD[31:0]. The byte addresses of IPD[31:0] are defined ***acsending*** from left to right: IPD[31:24] is a modulo 4 address boundary, 4*n* (0x00, 0x04, 0x08, 0x0C); IPD[7:0] is a modulo 4 address boundary plus offset 3, 4*n*+3 (0x03, 0x07, 0x0B, 0x0F). IPA[31:0] correspond left-to-right with XLA[0:31]. IPD[31:0] correspond left-to-right with XLD[0:31] (XLA[29] == 0) or XLD[32:63] (XLA[29] == 1).

The Memory Controller registers are defined with byte addresses and ***bus*** bit numbers ***ascending*** from left to right; but object bit ***fields*** within the registers may have ***ascending*** or ***descending*** bit numbers. The numbering order of bits as a bus does not govern the numbering order of bits within a data object.

All external memory interface busses are defined with ***descending*** bit numbers: MEM_MA[12:0], MEM_MBA[1:0], MEM_MDQ[31:0], MEM_DQM[3:0], MEM_MDQS[3:0]. Byte addressing of MEM_MDQ[31:0], MEM_DQM[3:0], and MEM_MDQS[3:0] is ***ascending***: MEM_MDQ[31:24], MEM_DQM[***3***], and MEM_MDQS[***3***] are associated with address offset ***0*** modulo 4 (4*n*); MEM_MDQ[7:0], MEM_DQM[***0***], and MEM_MDQS[***0***] are associated with address offset ***3*** modulo 4 (4*n*+3).

8.3 Features

The MPC5200 SDRAM Memory Controller has the following features:

- Supports either:
  - SDR SDRAM—memory I/Os are powered at 3.3V
  - DDR SDRAM—memory I/Os are powered at 2.5V

  DDR SDRAM transfers data at twice the rate and uses MEM_CLK and $\overline{\text{MEM\_CLK}}$ as a differential pair.
- 32-bit memory data bus

NOTE

It is **not** possible to connect only a 16-bit device to one half of the data bus.

- Maximum address space 512MB; 256MB per $\overline{\text{CS}}$:
  - Up to 13 bits of row address (RA[12:0])
  - Up to 12 bits of column address (CA[11:0])
  - 2 bits of bank address (BA[1:0])
  - Cannot use all 13 bits of RA and all 12 bits of CA at the same time. Maximum total address bits (RA+CA+BA) ≤ 26; 26 address bits x 4Byte data bus = 256MB.

NOTE

In this document the Auto Precharge control signal (A10 usually), conveyed on the memory address bus along with column address, is ***never*** included in the stated CA width; it is always ***in addition*** to the CA width.

The Memory Controller does not support memory devices with >8 CA bits, but <12 RA bits.

RA[12:0] correspond directly with MEM_MA[12:0]. CA[7:0] correspond directly with MEM_MA[7:0]. CA[11:8] do ***not*** correspond directly with MEM_MA[12:8].

- Maximum of 2 pinned-out Chip Selects ($\overline{\text{CS}}$).
  - $\overline{\text{CS0}}$ is pinned out all the time (i.e., a dedicated pin).

— $\overline{CS1}$ is only available if the GPIO_WKUP6 pin is programmed to be an SDRAM chip select. The default function of the pin is GPIO_WKUP6.
— To configure the GPIO_WKUP6 pin as SDRAM chip select, write 1 to the Port Configuration register msb.Section 7.3.2.1.1, *GPS Port Configuration Register—MBAR + 0x0B00*
— The size of each $\overline{CS}$ space is independent. It is possible but not recommended to overlap the address space pointed to by the 2 independent chip select.

NOTE

Maximum 4 physical memory devices total, all $\overline{CS}$.

- Minimum allocatable address space 1MB:
  — 8 bits of row address;
  — 8 bits of column address;
  — 2 bits of bank address;
  — 1 chip select;

NOTE

Minimum allocatable address space is much smaller (8Mb) than the lowest density available (64Mb). Excess memory bits are not used or simply wasted.

- 32 Byte PowerPC G2_LE critical word first burst transfer;
- Supports PowerPC G2_LE bus, 2-stage address/data pipeline (one data tenure in progress, one pipelined address tenure);
- Supports SDRAM Power Down and Self Refresh modes;
- Supports page mode and bursting to maximize the data rate;

NOTE

The SDRAM Memory Controller (MC) does not support error detect or parity check.

8.3.1 Devices Supported

Supported SDRAM devices (SDR and DDR both) are:

- 64Mbit;
- 128Mbit;
- 256Mbit;
- 512Mbit;
- 1Gbit when available, assuming the same interface style;
- 2Gbit when available, assuming the same interface style;

The MPC5200 limits external memory to a maximum of 4 memory chips placed within 5 cm of the MPC5200 processor. Flight delay on the board should be no more than 0.5 ns each way, and ***all*** signals must be matched. The maximum load is 20pF/pin (for the address/control signals) and 6 pF for the data and data strobe signals for the DDR case. These restrictions allow the read data clock to be obtained from a simple programmable tapped delay line.

Table 8-1. Legal Memory Configurations

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
11	8	2	1	1 x 64Mb 512K x 4bank x 32bit	8MB
			2	2 x 64Mb 512K x 4bank x 32bit	16MB
12	8	2	1	2 x 64Mb 1M x 4bank x 16bit	16MB
				1 x 128Mb 1M x 4bank x 32bit	
			2	4 x 64Mb 1M x 4bank x 16bit	32MB
				2 x 128Mb 1M x 4bank x 32bit	
12 13	9 8	2 2	1	4 x 64Mb 2M x 4bank x 8bit	32MB
				2 x 128Mb 2M x 4bank x 16bit	
				1 x 256Mb 2M x 4bank x 32bit	
			2	4 x 128Mb 2M x 4bank x 16bit	64MB
				2 x 256Mb 2M x 4bank x 32bit	
12 13	10 9	2 2	1	4 x 128Mb 4M x 4bank x 8bit	64MB
				2 x 256Mb 4M x 4bank x 16bit	
				1 x 512Mb 4M x 4bank x 32bit	
			2	4 x 256Mb 4M x 4bank x 16bit	128MB
				2 x 512Mb, 2 $\overline{CS}$ 4M x 4bank x 32bit	
12 13	11 10	2 2	1	4 x 256Mb 8M x 4bank x 8bit	128MB
				2 x 512Mb 8M x 4bank x 16bit	
				1 x 1Gb 8M x 4bank x 32bit	
			2	4 x 512Mb 8M x 4bank x 16bit	256MB
				2 x 1Gb 8M x 4bank x 32bit	

Table 8-1. Legal Memory Configurations (continued)

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
12 13	12 11	2 2	1	4 x 512Mb 16M x 4bank x 8bit	256MB
				2 x 1Gb 16M x 4bank x 16bit	
				1 x 2Gb 16M x 4bank x 32bit	
			2	4 x 1Gb 16M x 4bank x 16bit	512MB
				2 x 2Gb 16M x 4bank x 32bit	
11 12	8 8	2 2	1 1	1 x 64Mb 512K x 4bank x 32bit + 1 x 128Mb 1M x 4bank x 32bit	24MB
11 12 13	8 9 8	2 2	1 1	1 x 64Mb 512K x 4bank x 32bit + 1 x 256Mb 2M x 4bank x 32bit	40MB
11 12 13	8 10 9	2 2	1 1	1 x 64Mb 512K x 4bank x 32bit + 1 x 512Mb 4M x 4bank x 32bit	72MB
11 12 13	8 11 10	2 2	1 1	1 x 64Mb 512K x 4bank x 32bit + 1 x 1Gb 8M x 4bank x 32bit	136MB
11 12 13	8 12 11	2 2	1 1	1 x 64Mb 512K x 4bank x 32bit + 1 x 2Gb 16M x 4bank x 32bit	264MB
12 12 13	8 9 8	2 2	1 1	2 x 64Mb 1M x 4bank x 16bit + 2 x 128Mb 2M x 4bank x 16bit	48MB
12 12 13	8 10 9	2 2	1 1	2 x 64Mb 1M x 4bank x 16bit + 2 x 256Mb 4M x 4bank x 16bit	80MB

Table 8-1. Legal Memory Configurations (continued)

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
12	8	2	1	2 x 64Mb 1M x 4bank x 16bit +	144MB
12 13	11 10	2	1	2 x 512Mb 8M x 4bank x 16bit	
12	8	2	1	2 x 64Mb 1M x 4bank x 16bit +	272MB
12 13	12 11	2	1	2 x 1Gb 16M x 4bank x 16bit	
12	8	2	1	1 x 128Mb 1M x 4bank x 32bit +	48MB
12 13	9 8	2	1	1 x 256Mb 2M x 4bank x 32bit	
12	8	2	1	1 x 128Mb 1M x 4bank x 32bit +	80MB
12 13	10 9	2	1	1 x 512Mb 4M x 4bank x 32bit	
12	8	2	1	1 x 128Mb 1M x 4bank x 32bit +	144MB
12 13	11 10	2	1	1 x 1Gb 8M x 4bank x 32bit	
12	8	2	1	1 x 128Mb 1M x 4bank x 32bit +	272MB
12 13	12 11	2	1	1 x 2Gb 16M x 4bank x 32bit	
12	9	2	1	2 x 128Mb 2M x 4bank x 16bit +	96MB
12	10	2	1	2 x 256Mb 4M x 4bank x 16bit	
13	8	2	1	2 x 128Mb 2M x 4bank x 16bit +	96MB
13	9	2	1	2 x 256Mb 4M x 4bank x 16bit	
12	9	2	1	2 x 128Mb 2M x 4bank x 16bit +	160MB
12	11	2	1	2 x 512Mb 8M x 4bank x 16bit	

Table 8-1. Legal Memory Configurations (continued)

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
13 13	8 10	2 2	1 1	2 x 128Mb 2M x 4bank x 16bit + 2 x 512Mb 8M x 4bank x 16bit	160MB
12 12	9 12	2 2	1 1	2 x 128Mb 2M x 4bank x 16bit + 2 x 1Gb 16M x 4bank x 16bit	288MB
13 13	8 11	2 2	1 1	2 x 128Mb 2M x 4bank x 16bit + 2 x 1Gb 16M x 4bank x 16bit	288MB
12 12	9 10	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 512Mb 4M x 4bank x 32bit	96MB
13 13	8 9	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 512Mb 4M x 4bank x 32bit	96MB
12 12	9 11	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 1Gb 8M x 4bank x 32bit	160MB
13 13	8 10	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 1Gb 8M x 4bank x 32bit	160MB
12 12	9 12	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 2Gb 16M x 4bank x 32bit	288MB
13 13	8 11	2 2	1 1	1 x 256Mb 2M x 4bank x 32bit + 1 x 2Gb 16M x 4bank x 32bit	288MB

Table 8-1. Legal Memory Configurations (continued)

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
12 12	10 11	2 2	1 1	2 x256Mb 4M x 4bank x 16bit + 2 x 512Mb 8M x 4bank x 16bit	192MB
13 13	9 10	2 2	1 1	2 x256Mb 4M x 4bank x 16bit + 2 x 512Mb 8M x 4bank x 16bit	192MB
12 12	10 12	2 2	1 1	2 x256Mb 4M x 4bank x 16bit + 2 x 1Gb 16M x 4bank x 16bit	320MB
13 13	9 11	2 2	1 1	2 x256Mb 4M x 4bank x 16bit + 2 x 1Gb 16M x 4bank x 16bit	320MB
12 12	10 11	2 2	1 1	1 x 512Mb 4M x 4bank x 32bit + 1 x 1Gb 8M x 4bank x 32bit	192MB
13 13	9 10	2 2	1 1	1 x 512Mb 4M x 4bank x 32bit + 1 x 1Gb 8M x 4bank x 32bit	192MB
12 12	10 12	2 2	1 1	1 x 512Mb 4M x 4bank x 32bit + 1 x 2Gb 16M x 4bank x 32bit	320MB
13 13	9 11	2 2	1 1	1 x 512Mb 4M x 4bank x 32bit + 1 x 2Gb 16M x 4bank x 32bit	320MB
12 12	10 12	2 2	1 1	2 x 512Mb 8M x 4bank x 32bit + 2 x 1Gb 16M x 4bank x 32bit	384MB

Table 8-1. Legal Memory Configurations (continued)

Row Bits	Column Bits	Bank Bits	Spaces ($\overline{CS}$)	Physical	Address Range
13	9	2	1	2 x 512Mb 8M x 4bank x 32bit +	384MB
13	11	2	1	2 x 1Gb 16M x 4bank x 32bit	
12	11	2	1	1 x 1Gb 8M x 4bank x 32bit +	384MB
12	12	2	1	1 x 2Gb 16M x 4bank x 32bit	
13	10	2	1	1 x 1Gb 8M x 4bank x 32bit +	384MB
13	11	2	1	1 x 2Gb 16M x 4bank x 32bit	

Figure 8-1 shows an example memory configuration of 1 space ($\overline{CS}$) of 4 devices of 128Mbit (4M x 4 banks x 8bit) DDR SDRAM, for a total memory size of 64MB.

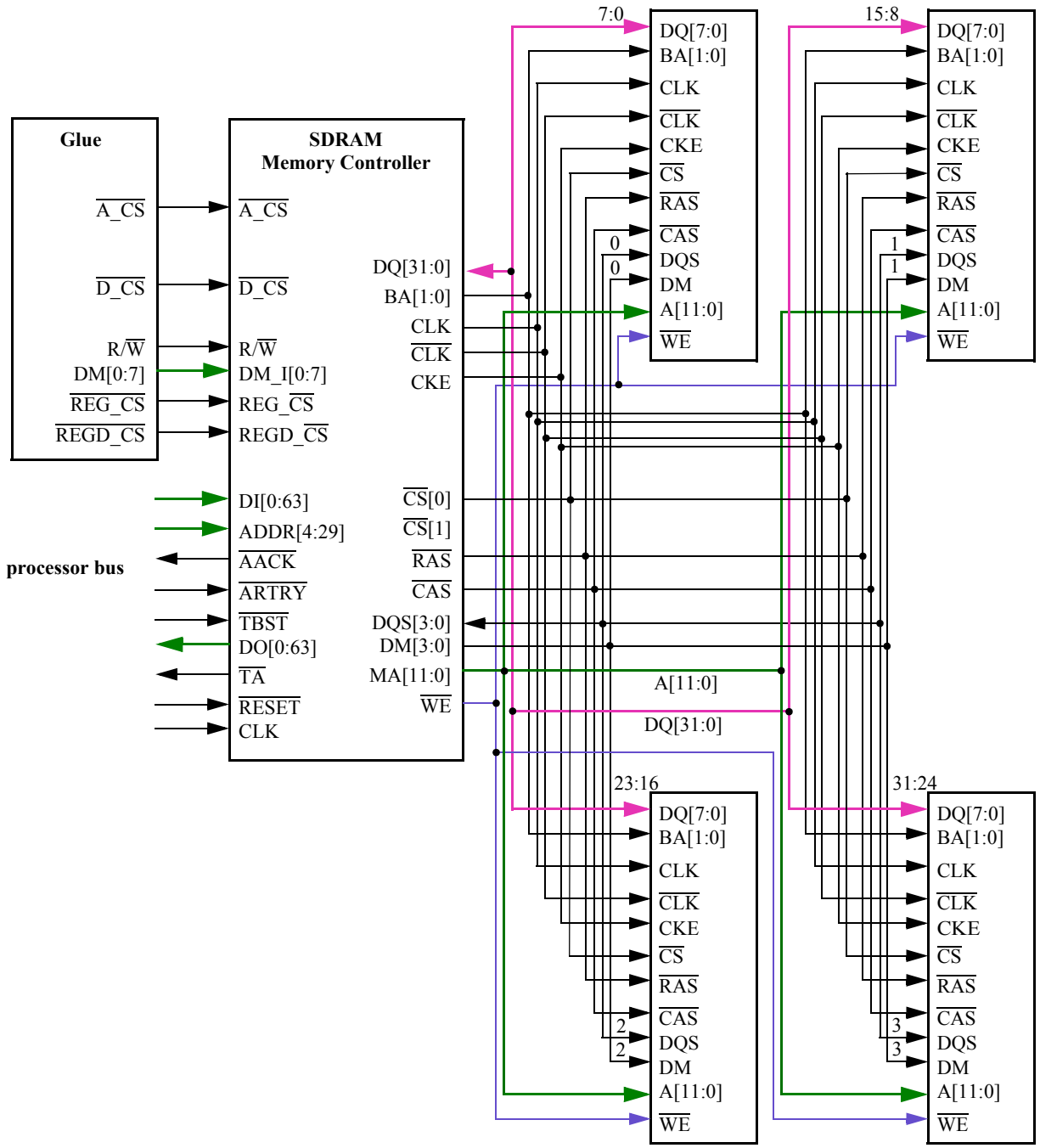


Figure 8-1. Block Diagram—SDRAM Subsystem Example

Both chip selects contribute together to access the whole memory. Each $\overline{CS}$ base address and size are programmed independently. Each $\overline{CS}$ base address must be size-aligned.

The MPC5200 does not support DIMM memory modules, however it can support a DIMM-compatible EEPROM using an on-chip I^2C chip interface (with appropriate configuration of pin functions).

8.4 Functional Description

8.4.1 External Signals (SDRAM Side)

Table 8-2. SDRAM External Signals

Signal Name	Description
Outputs	
MEM_CLK	Memory Clock (frequency is the same as the internal XL bus clock). Maximum allowed value is 132 MHz.
MEM_CLK	Inverted Memory Clock, used for DDR-SDRAM devices only.
MEM_CLK_EN	Memory Clock Enable (CKE). When low, the SDRAM is disabled. Used to switch memory into and out of self-refresh/power-down modes.
$\overline{\text{MEM\_CS}}$[0], $\overline{\text{MEM\_CS}}$[1]	Memory Command Select. Each space has a command select to enable commands
MEM_RAS	Memory Row Address Select
MEM_CAS	Memory Column Address Select
MEM_WE	Memory Write Enable
MEM_MA[12:0]	Memory Multiplexed Address. These are used as row address, column address, or Mode(Extended Mode) register data, depending on the command issued. Row address during Active command. Column address during Read and Write commands. MEM_MA10 is used as a control signal instead of an address line, to control Auto Precharge operation. The Auto Precharge control bit is not counted as a column address bit. The Memory Controller does not use Auto Precharge. Mode register data during Load Mode Register and Load Extended Mode Register (DDR only) commands.
MEM_MBA[1:0]	Memory Bank Address, or Mode register select, depending on the command issued. Bank address during Precharge Selected, Active, Read, and Write commands. The Memory Controller does not use the Precharge Selected command. Mode register select during Load Mode Register and Load Extended Mode Register (DDR only) commands. Although SDR memory only has a single internal Mode register, the Bank Address bits must still be valid.
MEM_DQM[3:0]	Memory Data Mask. Addressing = 0:3 0 Data byte read/write is enabled 1 Data byte read/write is inhibited SDR memories 3-state inhibited data during reads; DDR memories ignore Data Mask during reads. The memory controller never masks read data.
Bidirectional Signals	
MEM_MDQ[31:0]	Memory Data. Addressing = 0:3.
MEM_MDQS[3:0]	Memory Data Strobe, DDR only. Addressing = 0:3.
Note: Signals $\overline{\text{MEM\_RAS}}$, $\overline{\text{MEM\_CAS}}$, $\overline{\text{MEM\_WE}}$, and MEM_CLK_EN encode the SDRAM commands to control the different SDRAM operations.	

8.4.2 Block Diagram

Figure 8-2 shows the SDRAM MC block diagram. It is important to notice:

- the internal XL bus is 64 bits wide
- the external interface to the SDRAM is only 32 bits wide

The SDRAM row, column, and bank address bits are extracted from internal address XLA[4:29]; XLA[29:31], TSIZ[0:2], and $\overline{\text{TBST}}$ control the data path (MDQ, DQM).

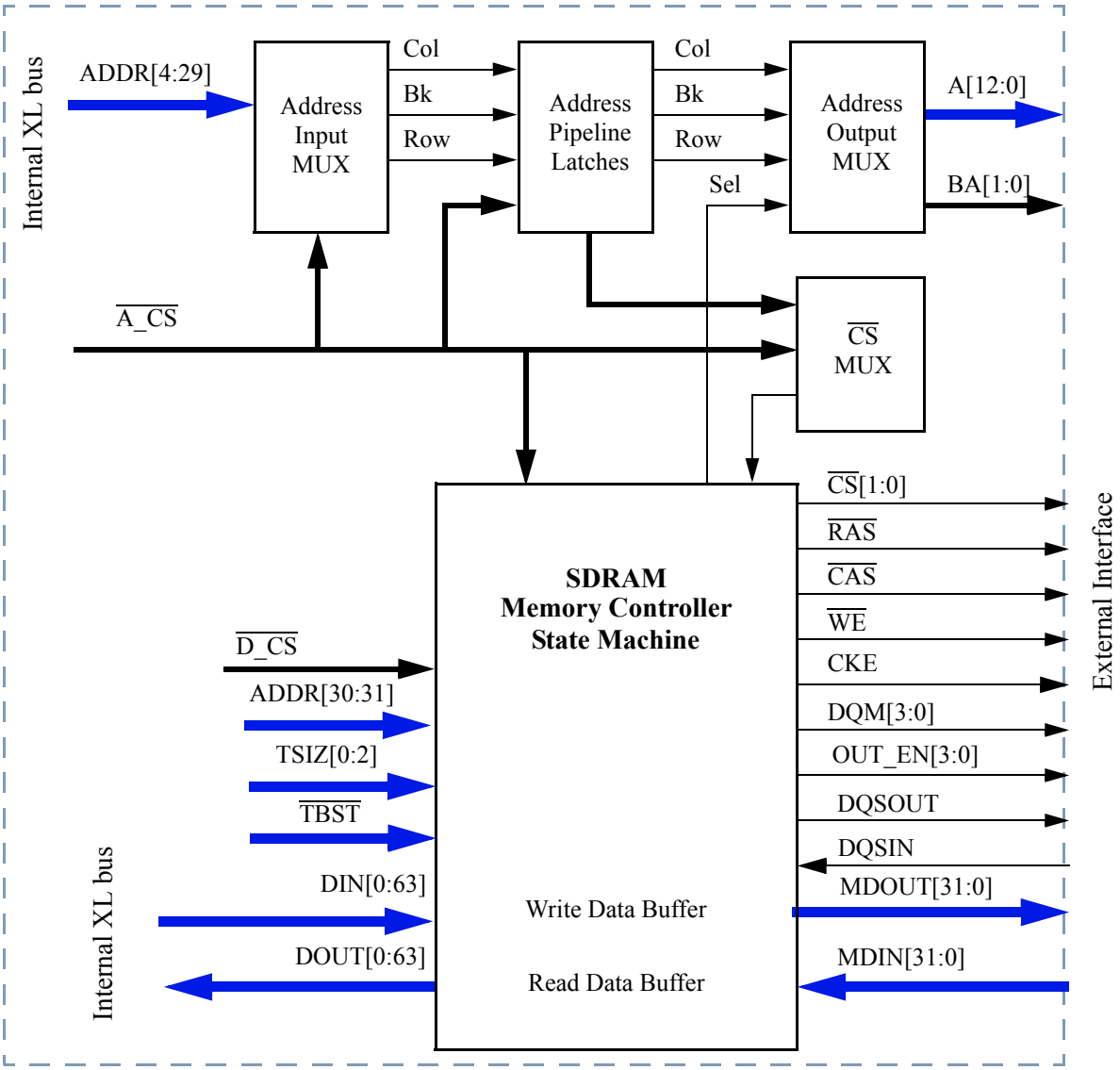


Figure 8-2. Block Diagram—SDRAM Memory Controller

8.4.3 Transfer Size

All SDRAMs are “burst oriented” for read and write operations. The memory will move a full burst of data for every Read and Write command unless the command is interrupted by a new command, explicitely terminated, or the data is masked. (Data mask does not shorten the command, it only inhibits data capture.) The Memory Controller can interrupt certain commands, but does not support the explicit Burst Terminate command.

The Memory Controller supports Burst and Non-Burst, or Single, transfers corresponding to the homonymous XL bus transfer types. A Burst transfer is a 32 Byte block, 4 XLB data beats (8 memory data beats), spanning a modulo 32 address range. The starting address can be any modulo 8 boundary within the modulo 32 range; the address “wraps” from the highest address to the lowest address of the range if the starting address is not aligned at the beginning of the range. No data is masked during a burst.

The beat address order of the XL bus is ***sequential***. Based on the start address issued by the internal master, the address order of the 4 XLD beats in a burst transfer is one of the following:

- 0x00, 0x08, 0x10, 0x18 (memory data address order 0x00, 0x04, 0x08, 0x0c, ...)
- 0x08, 0x10, 0x18, 0x00
- 0x10, 0x18, 0x00, 0x08
- 0x18, 0x00, 0x08, 0x10

To implement single-beat transfers, the Memory Controller uses DM[3:0] to mask unwanted bytes or words. The Memory Controller supports all single-beat transfer sizes from 1 to 8 contiguous bytes within a single modulo 8 address range.

A Single transfer is exactly 1 beat on the XLD bus. The relevant data for a Single transfer is always within the first 2 beats on the memory bus, allowing the command to be aborted (interrupt) as soon as possible.

8.4.4 Commands

When an internal bus master accesses SDRAM address space, the Memory Controller generates the corresponding SDRAM command. Table 8-3 lists SDRAM commands supported by the Memory Controller.

Table 8-3. SDRAM Commands

Function	Symbol	CKE	CS	RAS	CAS	WE	BA[1:0]	A10	Other A
Command Inhibit	INH	H	H	X	X	X	X	X	X
No Operation	NOP	H	L	H	H	H	X	X	X
Read	READ	H	L	H	L	H	V	L	V
Write	WRIT	H	L	H	L	L	V	L	V
Bank Active	ACT	H	L	L	H	H	V	V	V
Precharge All Banks	PALL	H	L	L	H	L	X	H	X
Load Mode Register	LMR	H	L	L	L	L	LL	V	V
Load Extended Mode Register	LEMR	H	L	L	L	L	LH	V	V
CBR Auto Refresh	AREF	H	L	L	L	H	X	X	X
Self Refresh	SREF	H→L	L	L	L	H	X	X	X
Power Down	PDWN	H→L	H	X	X	X	X	X	X

Note:
1. H = High
2. L = Low
3. V = Valid
4. X = Don't care

Many commands require a delay before the next command may be issued; sometimes the delay depends on the type of the next command. These delay requirements are managed by the values programmed in the Memory Controller Configuration registers.

8.4.4.1 Load Mode/Extended Mode Register Command

The Load Mode Register (LMR) and Load Extended Mode Register (LEMR) commands are used during SDRAM initialization only.

When a bus master writes to the Memory Controller Mode register, the Memory Controller generates the LMR or LEMR command to forward the data to the memory. In these two operations, data written to the Memory Controller is put on the SDRAM address and bank select busses. The bank select data selects the Mode or Extended Mode register.

The Memory Controller Mode register must be enabled before writing, and disabled after all memory Mode register operations are complete. This is done by setting or clearing the Control register mode_en bit.See Section 8.7.1, *Mode Register—MBAR + 0x0100*

Some of the configuration parameters required by the memory are also needed by the Memory Controller for command generation. The parameters are:

- burst length
- latency

These must be programmed in the Memory Controller Configuration registers separately from setting the memory Mode register.

8.4.4.2 Precharge All Banks Command

The Memory Controller issues the Precharge command only when necessary for one of the following conditions:

- Access to a new row
- Refresh interval elapsed
- Software commanded Precharge

NOTE

DRAMs also have a maximum bank open period, after which a precharge is required. The Memory Controller does not time the bank open period because the refresh interval is always less.

The Precharge command puts SDRAM into an idle state. In this state, the following commands can be issued:

- Refresh
- Bank Active
- Load Mode/Extended Mode Register

NOTE

The Memory Controller does not support the Precharge Selected Bank memory command.

8.4.4.3 Bank Active Command

SDRAM devices have 4 internal banks. A particular row and bank of memory must be activated to allow read and write accesses. For page mode support, the Memory Controller keeps the active row and bank(s) open as long as possible.

In an SDRAM device each internal bank can have one active row. The Bank Active command activates a row of one bank. The Memory Controller only supports the same active row in all banks of each $\overline{CS}$ space independently. The page size of a $\overline{CS}$ space is equal to the space size divided by the number of rows; but the page may not be contiguous in the XLB address space because the XLA bits for memory column address bits [11:8] and memory column address [7:0] are not consecutive. The size of a contiguous page segment is 4KB, corresponding to 8 CA bits plus 2 BA bits times 4Bytes of data.

Each $\overline{CS}$ space almost always has an active row. If no row is already active, any read or write access will activate one; and the only reasons that a row is deactivated are to activate a different one instead, or to perform a refresh.

8.4.4.4 Read Command

When the Memory Controller receives a read request via the XL bus, it first checks the row and bank of the new access. If the address falls within the active row of an active bank, it is a page hit, and the Read command is issued as soon as possible (pending any delays required by previous commands). If the address is within the active row, but the needed bank is inactive, or if there is no active row, the Memory Controller will issue a Bank Active command followed by the Read command. If the address is not within the active row, the Memory Controller will issue a Precharge command to close the active row, followed by a Bank Active command to activate the necessary bank and row for the new access, followed finally by the Read command.

The Precharge and Bank Active commands (if necessary) can sometimes be issued in parallel with an on-going data movement.

All Reads, whether Burst or Single, must be allowed to complete the entire burst length on the memory bus. With SDR memory, the Data Masks are negated throughout the entire Read burst length. With DDR memory, the Data Masks are ***asserted*** throughout the entire Read burst length; but DDR memory ignores the Data Masks during Reads.

8.4.4.5 Write Command

When the Memory Controller receives a write request via the XL bus, it first checks the row and bank of the new access. If the address falls within the active row of an active bank, it is a page hit, and the Write command is issued as soon as possible (pending any delays required by previous commands). If the address is within the active row but the needed bank is inactive, or if there is no active row, the Memory Controller will issue a Bank Active command followed by the Write command. If the address is not within the active row, the Memory Controller will issue a Precharge command to close the active row, followed by a Bank Active command to activate the necessary row and bank for the new access, followed finally by the Write command.

The Precharge and Bank Active commands (if necessary) can sometimes be issued in parallel with an on-going data movement.

With both SDR and DDR memory, a Read command can be issued overlapping the masked beats at the end of a previous Single Write of the same $\overline{CS}$; the Read command aborts the remaining (unnecessary) Write beats. With DDR memory, a Read of one $\overline{CS}$ can even overlap the masked beats at the end of a previous Single Write of the other $\overline{CS}$. The Write is not aborted, but the masks remain asserted. This is not possible with SDR memory, because SDR memory cannot be read with the masks asserted.

8.4.4.6 Auto Refresh Command

The Memory Controller issues Auto Refresh commands according to the ref_interval value specified in the Memory Controller Control register. Each time the programmed refresh interval elapses, the Memory Controller issues a Precharge All Banks command followed by an Auto Refresh command.

If a memory access is in progress at the time the refresh interval elapses, the Memory Controller schedules the refresh after the transfer is finished; but the interval timer continues counting so that the average refresh rate is constant.

After refresh, the SDRAM is in an idle state and waits for an Active command.

8.4.4.7 Self Refresh and Power Down Commands

The Memory Controller issues either a Power Down or a Self Refresh command if the Control register cke bit is changed from asserted to negated. If the ref_en bit of the same register is asserted when cke is negated, the controller issues a Self Refresh command; if the ref_en bit is negated, the controller issues a Power Down command. The ref_en bit may be changed in the same register write that changes the cke bit; the controller will act upon the new value of the ref_en bit.

Unlike an Auto Refresh, the controller does ***not*** automatically issue a Precharge command before the Self Refresh command. It is a software responsibility to command a Precharge, using the Control register soft_pre bit, by a separate write before negating the cke bit.

The memory is reactivated from Power Down or Self Refresh mode by reasserting the cke bit.

If a normal refresh interval elapses while the memory is in Self Refresh mode, a Precharge and Auto Refresh will be performed as soon as the memory is reactivated. If the memory is put into and brought out of Self Refresh all within a single refresh interval, the next automatic refresh will occur on schedule.

In Self Refresh mode, the memory does not require an external clock. The MEM_CLK can be stopped for maximum power savings by negating the Memory Controller Clock Enable bit of the CDM Clock Enable register. See Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*. If the Memory Controller clock is stopped, the refresh interval timer must be reset before the memory is reactivated (if periodic refresh is to be resumed). The refresh interval timer is reset by negating the Control register ref_en bit. This can be done at any time while the memory is in Self Refresh mode, before or after the Memory Controller clock is stopped/restarted, but ***not*** with the same Control register write that negates cke; this would put the memory in Power Down mode. To restart periodic refresh when the memory is reactivated, the ref_en bit must be reasserted; this can be done before the memory is reactivated, or in the same Control register write that cke is reasserted.

NOTE

As soon as the CKE signal is negated (set to a logical 0) a SDRAM memory device does ***NOT*** answer any longer to any command (all its input but the CKE are ignored) until the CKE is re-asserted and a minimum time has elapsed (as specified by the memory vendor).

8.5 Operation

8.5.1 Power-Up Initialization

The SDRAM and SDRAM MC must be initialized after power-up. SDRAM parameters may be read from an I^2C serial EEPROM, or compiled into the boot ROM. See Section 18, *Inter-Integrated Circuit (I^2C)* if using serial EEPROM.

The steps below should be followed to initialize the memory system.

NOTE

The sequence might change slightly from device to device. Refer to the device data sheet for the most up-to-date information. In any case of conflict between this document and the device data sheet, the data sheet shall prevail.

Step 1. After reset is deactivated, pause for the amount of time indicated in the SDRAM specification. Usually 100 µs or 200 µs.

Step 2. Determine the number of SDRAM $\overline{CS}$ spaces. If using both $\overline{CS}$ spaces, configure GPIO_WKUP6/$\overline{CS1}$ for $\overline{CS1}$ mode.

If all the memory and controller register values have been precalculated and stored in ROM, skip step 3 and go directly to step 4. Otherwise, continue with step 3.

Step 3. Read the SDRAM parameters (type, size, address muxing, timing), and determine the memory clock frequency. (The memory clock frequency is always equal to the XLB frequency.) Using the SDRAM parameters and the clock frequency, calculate all the memory and controller register values now. Certain register fields are mandatory:

- Memory Mode register Burst Mode = Sequential
- Memory Mode register Burst Length = 8
- Controller Configuration register 2 burst_length = 7
- Controller Control register cke = 1

Do not write any registers yet. Use these register values as default values for the following operations. An operation can override the default, but overrides do not carry forward to subsequent operations.

Step 4. Write the SDRAMCS Configuration registers and the controller Config registers 1 & 2.

Step 5. Write the controller Control register with these overrides:

— assert the mode_en bit (1).

— negate ref_en (0).

Step 6. (DDR only) Write the controller Control register to issue a Precharge All Banks command (soft_pre=1); maintain mode_en=1, ref_en=0, all other bits default.

Step 7. (DDR only) Write to the memory Extended Mode register to enable the DLL.

Step 8. (DDR only) Write to the memory Mode register to reset the DLL.

Step 9. (DDR only) Pause for the DLL lock time specified by the memory (roughly 100 µs. See memory datasheet for detailed time).

Step 10. Write to the controller Control register to issue a Precharge All Banks command (soft_pre=1); maintain mode_en=1, ref_en=0.

Step 11. Write to the controller Control register to issue 2 or more Auto Refresh commands (soft_ref=1); maintain mode_en=1, ref_en=0. Each command requires a separate write.

Step 12. Write to the memory Mode register to specify normal operation.

Step 13. Write to the controller Control register to specify normal operation.

8.5.2 Read Clock

The MPC5200 implements a simple, software adjustable, tapped delay circuit, which consists of a buffer string with 32 selectable tap points. This circuit delays an on-chip clock to match the off-chip round-trip path delay of the clock out plus read data return. This delayed clock is used to latch read data from the memories into the controller. The delay tap must be programmed after controller and memory device initialization, and before normal read accesses begin. It must also be rechecked periodically during normal operation, and adjusted if necessary. The maximum rechecking interval depends on voltage and especially temperature (Tj) stability of the MPC5200 device. The delay chain may require frequent adjustment during periods of rapid temperature change, such as boot or wake up.

For DDR memories, the MDQS signals from the memories are used as data capture qualifiers. The data is latched with the internally generated read clock, not derived from the MDQS signals from the memories. The MDQS signals are not used with SDR memories, and require pull-up or pull-down resistors.

8.5.2.1 Read Clock Programming Algorithm

```
Set 32 flag variables “valid[0:31]” all true;
// valid[0:31] is a single 32bit unit. Each bit is an independent
// flag.
For each sdram chip select {
  Write a known data pattern to a small address region; // Note 1.
  For (i = 0; i < 32; i++) {
    Write 1byte RstCfg reg (MBAR + 0x0204) = i; // tapSel=i.
    For each starting address within the region { // Note 2.
      Read burst from the starting address; // MUST be BURST read.
      If (read data != known pattern) {
        A: // Label “A”.
        valid[i] = false;
        next i;
      }
    }
  }
}
firstGoodTap = lastGoodTap = 32;
countGoodTap = 0;
If (valid[0]) {
  valid[0:31] = ~valid[0:31] // Invert all bits.
  invert = true;
}
For (i = 0; i < 32; i++) {
  If (valid[i]) {
    If (firstGoodTap == 32) {
      firstGoodTap = i;
    }
    lastGoodTap = i;
```

```
    countGoodTap++;
  }
}
If (firstGoodTap + countGoodTap - lastGoodTap - 1 != 0) {
  die; // "Holes" in the range. Note 3.
}
bestTap = (lastGoodTap + firstGoodTap)/2;
If (invert) {
  valid[0:31] = ~valid[0:31]; // Invert bits back to original.
  firstGoodTap ^= lastGoodTap; // Swap firstGood Tap ...
  lastGoodTap ^= firstGoodTap; // ... and lastGoodTap ...
  firstGoodTap ^= lastGoodTap; // ... in place.
  firstGoodTap = (firstGoodTap + 1) % 32;
  lastGoodTap = (lastGoodTap + 31) % 32;
  countGoodTap = 32 - countGoodTap;
  bestTap = (bestTap + 16) % 32;
}
If (countGoodTap < magicNumber) {
  die; // Note 4.
}
Write 1byte RstCfg reg (MBAR + 0x00000204) = bestTap;
xxx00200: // tea exception address, Note 5.
Restore SPRs; // No stacking or unstacking necessary.
Go to Label A;
```

Notes:

1. A region must be reserved in every active $\overline{\text{CS}}$ space. The minimum region size is the length of a burst. A suggested data pattern:

   Write (region_base + 0x0000:0x005C) =
   0xFFFFFFFF, 0x00000000, 0xFFFFFFFF, 0x00000000,
   0xFFFF0000, 0x0000FFFF, 0xFFFF0000, 0x0000FFFF,
   0xFF00FF00, 0x00FF00FF, 0xFF00FF00, 0x00FF00FF,
   0xF0F0F0F0, 0x0F0F0F0F, 0xF0F0F0F0, 0x0F0F0F0F,
   0xCCCCCCCC, 0x33333333, 0xCCCCCCCC, 0x33333333,
   0xAAAAAAAA, 0x55555555, 0xAAAAAAAA, 0x55555555
2. Multiple reads must be performed, and the correct data must be different for each. For the pattern suggested above, the read addresses could be:

   For (addr = 0x0000; addr < 0x0060, addr += 0x0020)
3. The definition of “die” may be different for different systems. One implementation could be “run the algorithm again”. Another implementation could be “rerun the algorithm separately on each $\overline{\text{CS}}$ space, and boot with the space with the most margin only”. Another implementation could be “boot a failsafe kernel in on-chip SRAM and send a service request”.
4. The larger the value of countGoodTap, the greater the tolerance for voltage and temperature drift, and the less often the delay tap needs to be rechecked. A small value of countGoodTap indicates little margin for drift tolerance, and the delay tap should be rechecked more often. magicNumber is a minimum acceptable margin to proceed with system boot. Different values may be appropriate for different systems. A value somewhere in the range 3 to 9 is suggested as a guideline.
5. In DDR mode, as the SDRAM read is attempted with every possible value of tap select, it is possible for the access to hang; therefore timeout must be enabled and a tea handler routine is necessary. Timeout cannot occur in SDR mode.

8.6 Programming the SDRAM Controller

The Memory Controller registers consist of:

- Section Table 8-4., *Memory Controller Mode Register* / SDRAM MC Extended Mode Register (MBAR+0x0100), write only
- Section Table 8-5., *Memory Controller Control Register* (MBAR+0x0104)
- Section Table 8-8., *Memory Controller Configuration Register 1* (MBAR+0x0108)
- Section Table 8-9., *Memory Controller Configuration Register 2* (MBAR+0x010C)

All registers are 32bit-aligned in memory (modulo 4 address boundary).

8.7 Memory Controller Registers (MBAR+0x0100:0x010C)

8.7.1 Mode Register—MBAR + 0x0100

Each time the 32-bit write-only Mode register (mode[0:31]) is written (and cmd is set to 1), the controller generates a Load Mode Register or Load Extended Mode Register command to memory.

The memory Mode/Extended Mode registers must be initialized during the system boot sequence; but before writing to the controller Mode register, the mode_en and cke bits in the Control register must be set to 1. After memory initialization is complete, the Control register mode_en bit should be cleared to prevent subsequent access to the controller Mode register.

Table 8-4. Memory Controller Mode Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R															Rsvd	
W	MEM_MBA [1:0]		MEM_MA[11:0]													cmd
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	MEM_MBA [1:0]	See SDRAM data sheet. Select either the memory device Mode register or the memory device Extended Mode register, if present.
2:13	MEM_MA[11:0]	See SDRAM data sheet. MPC5200 supports: Read CAS Latency, SDR: 2, 3 Read CAS Latency, DDR: 2, 2.5 Burst type: Sequential only Burst length: 8 only Other fields: As appropriate Specific bit allocation can vary from device to device. All devices in all $\overline{CS}$ spaces must have compatible format(s), because all are written at the same time with the same value.
14	—	Reserved
15	cmd	1 Generate a (Extended) Mode Register Set memory command. Applied to all $\overline{CS}$ at once. 0 Do not generate any memory command.
16:31	—	Reserved

8.7.2 Control Register—MBAR + 0x0104

The 32-bit read/write Control register controls specific operations and generates some SDRAM commands. This register is reset only by a power-up reset signal.

Table 8-5. Memory Controller Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	mode_en	cke	ddr	ref_en	Rsvd			hi_addr	Rsvd	drive_rule	ref_interval[0:5]					
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 lsb
R	Reserved				dqs_oe				Reserved							Rsvd
W														soft_ref	soft_pre	
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	mode_en	0 Mode register locked, cannot be written. 1 Mode register enabled, can be written.
1	cke	0 MEM_CLK_EN negated (low). 1 MEM_CLK_EN asserted (high). cke must be set to 1 to perform normal read and write operations. Set cke to 0 to put the memory in Self Refresh or Power Down mode.
2	ddr	0 SDR mode. 1 DDR mode.
3	ref_en	0 Automatic refresh disabled. 1 Automatic refresh enabled. In general, refresh must be enabled, unless the system is known to access memory in a pattern that is guaranteed to open every row in every bank within every refresh period t_{REF}. Some memory data sheets do not spec t_{REF}, but spec t_{REFI} instead. In this case, $t_{REF} = t_{REFI}$ x #rows. NOTE: The number of Refresh commands required in t_{REF} is #rows; if refresh is disabled, the number of Read/Write commands required in t_{REF} is #rows x 4banks.
4:6	—	Reserved
7	hi_addr	Control the use of internal address bits XLA[4:7] as row or column bits on the MEM_MA bus. See Table 8-6.
8	—	Reserved (must be written 0)
9	drive_rule	0 “Tri-state except to write” mode: MPC5200 drives the MDQ and MDQS lines only when necessary to perform write commands. 1 “Drive except to read” mode: MPC5200 tri-states the MDQ and MDQS lines only when necessary to perform read commands. “Drive except to read” mode prevents unterminated memory signals from floating for extended periods. However, terminated routing is always recommended over unterminated.

Bit	Name	Description
10:15	ref_interval[0:5]	The average periodic interval at which the controller generates refresh commands to memory; measured in increments of 64 x MEM_CLK period. 1) Multiply t_{REFI} by the MEM_CLK frequency. (If the memory data sheet does not define t_{REFI}, it can be calculated by $t_{REFI} = t_{REF} / \#rows$.) Example: Assume t_{REF} = 64ms, #rows = 4K, MEM_CLK = 133MHz. Then: t_{REFI} = 64ms / 4K = 15.625µs; 15.625µs x 133MHz = 2078.1 2) Divide the previous result by 64, rounding toward 0 2078.1 / 64 = 32.471; discard the fractional part. 3) Subtract 1 from the previous result. The new result is ref_interval. 32 - 1 = 31 = 0x1f
16:19	—	Reserved
20:23	dqs_oe[3:0]	Each bit individually controls one MEM_MDQS output. 0 The corresponding MEM_MDQS pin is ***never*** driven, regardless of memory operation and drive_rule. Always set to 0000 for SDR. 1 The corresponding MEM_MDQS pin ***can*** be driven, depending on memory operations and drive_rule. DDR only.
24:28	—	Reserved
29	soft_ref	0 No operation. 1 Generate a non-periodic Auto Refresh command as soon as possible. This is a write-only bit; always returns 0 on a read. A software requested refresh is completely independent of the periodic refresh interval counter. Software refresh is only possible when mode_en==1.
30	soft_pre	0 No operation. 1 Generate a Precharge All command as soon as possible. This is a write-only bit; always returns 0 on a read. Software precharge is only possible when mode_en==1.
31	—	Reserved

The Table 8-6 indicates how the internal address bits XLA[4:7] are multiplexed internally to support higher column or row address bits.

Table 8-6. High Address Usage

hi_addr	XL Bus Address Line Mapping to Column or Row Address			
	4	5	6	7
0	CA12	CA11	CA9	CA8
1	CA11	CA9	CA8	RA12

Table 8-7. SDRAM Address Multiplexing

Device	Structure	Row bits × Col bits × Bank bits	hi_addr	Internal XLA[4:29]							
				4	5	6	7	8	9:19	20:21	22:29
64Mbit	2Mx32bit	11x8x2	0	—[a]	—	—	—	—	RA [10:0]	BA [1:0]	CA [7:0]
	4M×16bit	12×8×2	0	—	—	—	—	RA[11:0]			
	8M×8bit	12×9×2	0	—	—	—	CA8				
		13×8×2	1	—	—	—	RA12				

Table 8-7. SDRAM Address Multiplexing

Device	Structure	Row bits × Col bits × Bank bits	hi_ addr	Internal XLA[4:29]							
				4	5	6	7	8	9:19	20:21	22:29
128Mbit	4M×32bit	12×8×2	0	—	—	—	—	RA[11:0]		BA [1:0]	CA [7:0]
	8M×16bit	12×9×2	0	—	—	—	CA8				
		13×8×2	1	—	—	—	RA12				
	16M×8bit	12×10×2	0	—	—	CA9	CA8				
		13×9×2	1	—	—	CA8	RA12				
256Mbit	8M×32bit	12×9×2	0	—	—	—	CA8	RA[11:0]		BA [1:0]	CA [7:0]
		13×8×2	1	—	—	—	RA12				
	16M×16bit	12×10×2	0	—	—	CA9	CA8				
		13×9×2	1	—	—	CA8	RA12				
	32M×8bit	12×11×2	0	—	CA11	CA9	CA8				
		13×10×2	1	—	CA9	CA8	RA12				
512Mbit	16M×32bit	12×10×2	0	—	—	CA9	CA8	RA[11:0]		BA [1:0]	CA [7:0]
		13×9×2	1	—	—	CA8	RA12				
	32M×16bit	12×11×2	0	—	CA11	CA9	CA8				
		13×10×2	1	—	CA9	CA8	RA12				
	64M×8bit	12×12×2	0	CA12	CA11	CA9	CA8				
		13×11×2	1	CA11	CA9	CA8	RA12				
1Gbit	32Mx32bit	12×11×2	0	—	CA11	CA9	CA8	RA[11:0]		BA [1:0]	CA [7:0]
		13×10×2	1	—	CA9	CA8	RA12				
	64Mx16bit	12×12×2	0	CA12	CA11	CA9	CA8				
		13×11×2	1	CA11	CA9	CA8	RA12				
2Gbit	64Mx32bit	12×12×2	0	CA12	CA11	CA9	CA8	RA[11:0]		BA [1:0]	CA [7:0]
		13×11×2	1	CA11	CA9	CA8	RA12				

[a] All MEM_MA pins are driven in all cases, but only the bits used by memory are listed.

8.7.3 Configuration Register 1—MBAR + 0x0108

The 32-bit read/write Configuration register 1 stores delay values necessary between specific SDRAM commands. During initialization, software loads values to the register according to the SDRAM information obtained from the data sheet. This register is reset only by a power-up reset signal.

The Read and Write Latency fields govern the relative timing of commands and data, and must be exact values. All other fields govern the relative timing from one command to another, they have minimum values but any larger value is also legal (but with decreased performance). The “suggested values” are based on the maximum routing delay of memory signals and the MPC5200 maximum memory frequency of 133MHz; they do not guarantee maximum performance for actual board routing delay or operating frequency.

The minimum values of certain fields can be different for SDR and DDR SDRAM, even if the data sheet timing is the same, because:

- In SDR mode, the Memory Controller counts the delay in MEM_CLK
- In DDR mode, the Memory Controller counts the delay in 2xMEM_CLK (also referred to as MEM_CLK2)

MEM_CLK—Memory Controller clock—is the speed of the SDRAM interface and is equal to the internal XL bus clock. MEM_CLK is fixed at boot time along with the XL bus clock, via the HW RESET WORD setting. It is an integer multiple of the external reference clock (e.g., 66MHz, 99MHz or 132MHz if a 33MHz reference is used).

MEM_CLK2—double frequency of MEM_CLK—DDR uses both edges of the bus-frequency clock (MEM_CLK) to read/write data.

Table 8-8. Memory Controller Configuration Register 1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	srd2rwp				Rsvd	swt2rwp			rd_latency				Rsvd	act2rw		
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 lsb
R	Rsvd	pre2act			ref2act				Rsvd	wr_latency			Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	srd2rwp	Single Read to Read/Write/Precharge delay. Limiting case is Read to Write: CL + burst + round trip delay + bus turnaround - t_{DQSS}(DDR only) - 1; round up. For DDR, t_{DQSS}=1clk, bus turnaround=t_{HZ} + 0.5clk: If CL==2: 2 + 4 + 1 + 0.75ns + 0.5 - 1 - 1 = 5.5clk + 0.75ns, round to 0x6. If CL==2.5: 2.5 + 4 + 1 + 0.75ns + 0.5 -1 - 1 = 6clk + 0.75ns, round to 0x7. **Note:** This controller does not support Burst Terminate, therefore a single read will take as long as a Burst read. For SDR, bus turnaround=t_{HZ} + 1clk: If CL==2: 2 + 8 + 1 + 5.4ns + 1 - 1 = 11clk + 5.4ns, round to 0xC. If CL==3: 3 + 8 + 1 + 5.4ns + 1 - 1 = 12clk + 5.4ns, round to 0xD.
4	—	Reserved
5:7	swt2rwp	Single Write to Read/Write/Precharge delay. Limiting case is Write to Precharge. For DDR, suggested value = 0x3 (t_{WR} + 1 clk) For SDR, suggested value = 0x2 (t_{WR})
8:11	rd_latency	Read CAS Latency. For DDR: If CL==2, write 0x6 If CL==2.5, write 0x7 For SDR: If CL==2, write 0x2 If CL==3, write 0x3 **Note:** NOTE: CL=2.5 is not supported for SDR.
12	—	Reserved

Bit	Name	Description
13:15	act2rw	Active to Read/Write delay. Suggested value at 132 MHz = 0x02 Rule: t_{RCD}/MEM_CLK-1. Round up to nearest integer. EXAMPLE: If t_{RCD} = 20ns and MEM_CLK = 99 MHz 20ns / 10.1 ns = 1.98; round to 2; write 0x1. If t_{RCD} = 20 ns and MEM_CLK = 132 MHz 20ns / 7.5 ns = 2.66; round to 3; write 0x2.
16	—	Reserved
17:19	pre2act	Precharge to Active or Refresh delay. Suggested value at 132 MHz = 0x02 Rule: t_{RP}/MEM_CLK-1. Round up to nearest integer. EXAMPLE: If t_{RP} = 20ns and MEM_CLK = 99 MHz 20ns / 10.1 ns = 1.98; round to 2; write 0x1. If t_{RP} = 20 ns and MEM_CLK = 132 MHz 20ns / 7.5 ns = 2.66; round to 3; write 0x2.
20:23	ref2act	Refresh to Active delay. Suggested value at 132 MHz = 0x9 Rule: t_{RFC}/MEM_CLK - 1. Round up to nearest integer. EXAMPLE: If t_{RFC} = 75ns and MEM_CLK = 99 MHz 75ns / 10.1ns = 7.425; round to 8; write 0x7. If t_{RFC} = 75ns and MEM_CLK = 132 MHz 75ns / 7.5ns = 10; round to 9; write 0x9.
24	—	Reserved
25:27	wr_latency	Write latency. For DDR, write 0x3 For SDR, write 0x0
28:31	—	Reserved

8.7.4 Configuration Register 2—MBAR + 0x010C

The 32-bit read/write Configuration register 2 stores delay values necessary between specific SDRAM commands. During initialization, software loads values to the register according to the SDRAM information obtained from the data sheet. This register is reset only by a power-up reset signal.

The Burst Length field must be exact. All other fields govern the relative timing from one command to another, they have minimum values but any larger value is also legal (but with decreased performance).

All delays in this register are expressed in MEM_CLK.

Table 8-9. Memory Controller Configuration Register 2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	brd2rp				bwt2rwp				brd2wt				burst_length			
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	brd2rp	Burst Read to Read/Precharge delay. Limiting case is Read to Read. For DDR, suggested value = 0x4 (BurstLength/2) For SDR, suggested value = 0x8 (BurstLength)
4:7	bwt2rwp	Burst Write to Read/Write/Precharge delay. Limiting case is Write to Precharge. For DDR, suggested value = 0x6 (BurstLength/2 + t_{WR}) For SDR, suggested value = 0x8 (BurstLength + t_{WR} - 2)
8:11	brd2wt	Burst Read to Read/Write/Precharge delay. Limiting case is Read to Write: CL + burst + round trip delay + bus turnaround - t_{DQSS}(DDR only) - 1; round up. For DDR, t_{DQSS}=1clk, bus turnaround=t_{HZ} + 0.5clk: If CL==2: 2 + 4 + 1 + 0.75ns + 0.5 - 1 - 1 = 5.5clk + 0.75ns, round to 0x6. If CL==2.5: 2.5 + 4 + 1 + 0.75ns + 0.5 -1 - 1 = 6clk + 0.75ns, round to 0x7. For SDR, bus turnaround=t_{HZ} + 1clk: If CL==2: 2 + 8 + 1 + 5.4ns + 1 - 1 = 11clk + 5.4ns, round to 0xC. If CL==3: 3 + 8 + 1 + 5.4ns + 1 - 1 = 12clk + 5.4ns, round to 0xD.
12:15	burst_length	Write 0x07 (Burst Length - 1)
16:31	—	Reserved

The Figure 8-3. Programmable Command Timings shows the timings which can be programmed by the two Controller Configuration Register. The timing diagram uses the suggested values for a DDR memory and a 132 MHz memory clock. The displayed Commands are the limiting cases.

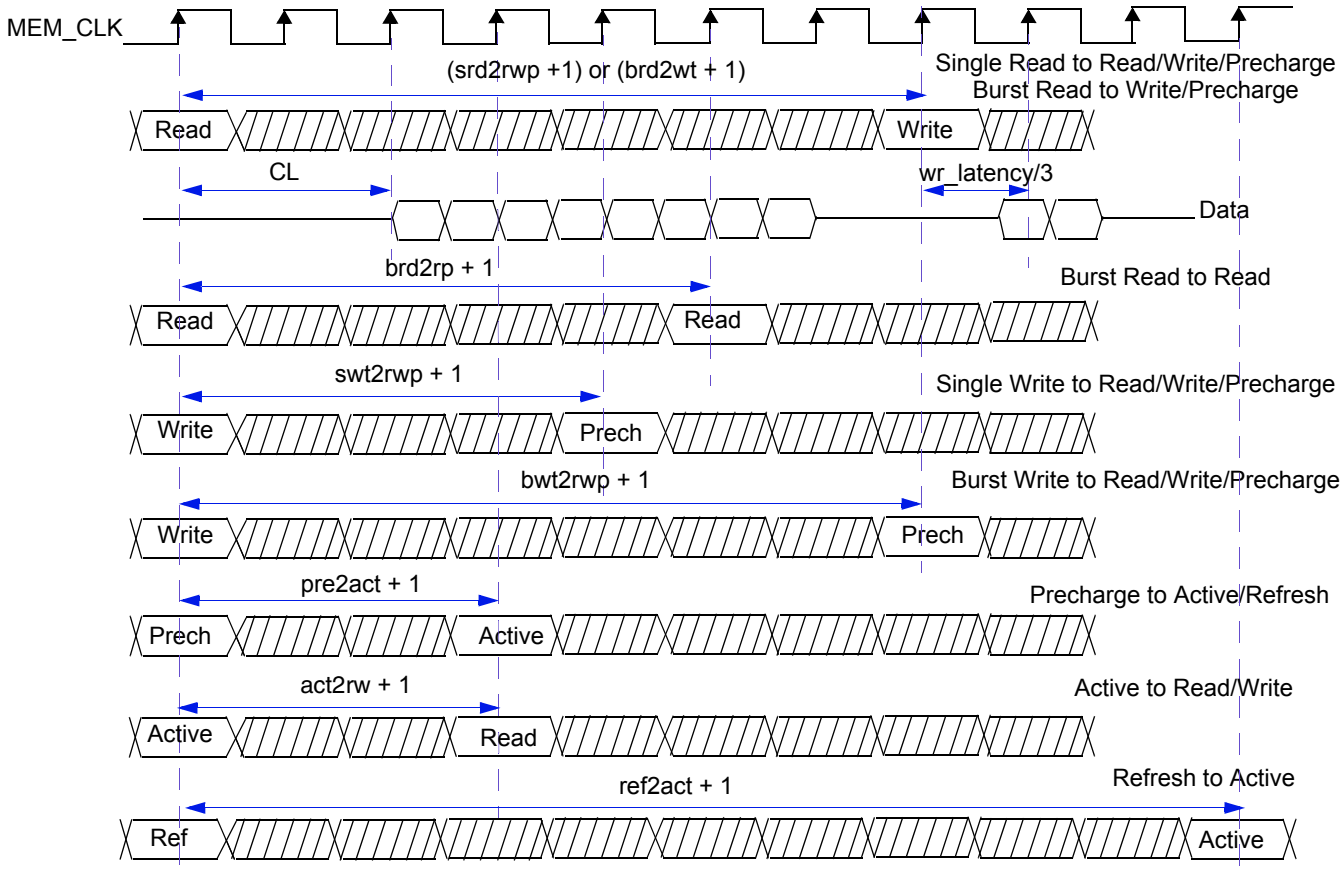


Figure 8-3. Programmable Command Timings

8.8 Address Bus Mapping

This is an illustration of how the XL bus address enters the Memory Controller and is broken down into Row, Column, and Bank Address fields. Shown below is the 32-bit XL bus address. The Memory Controller uses bits 4:31.

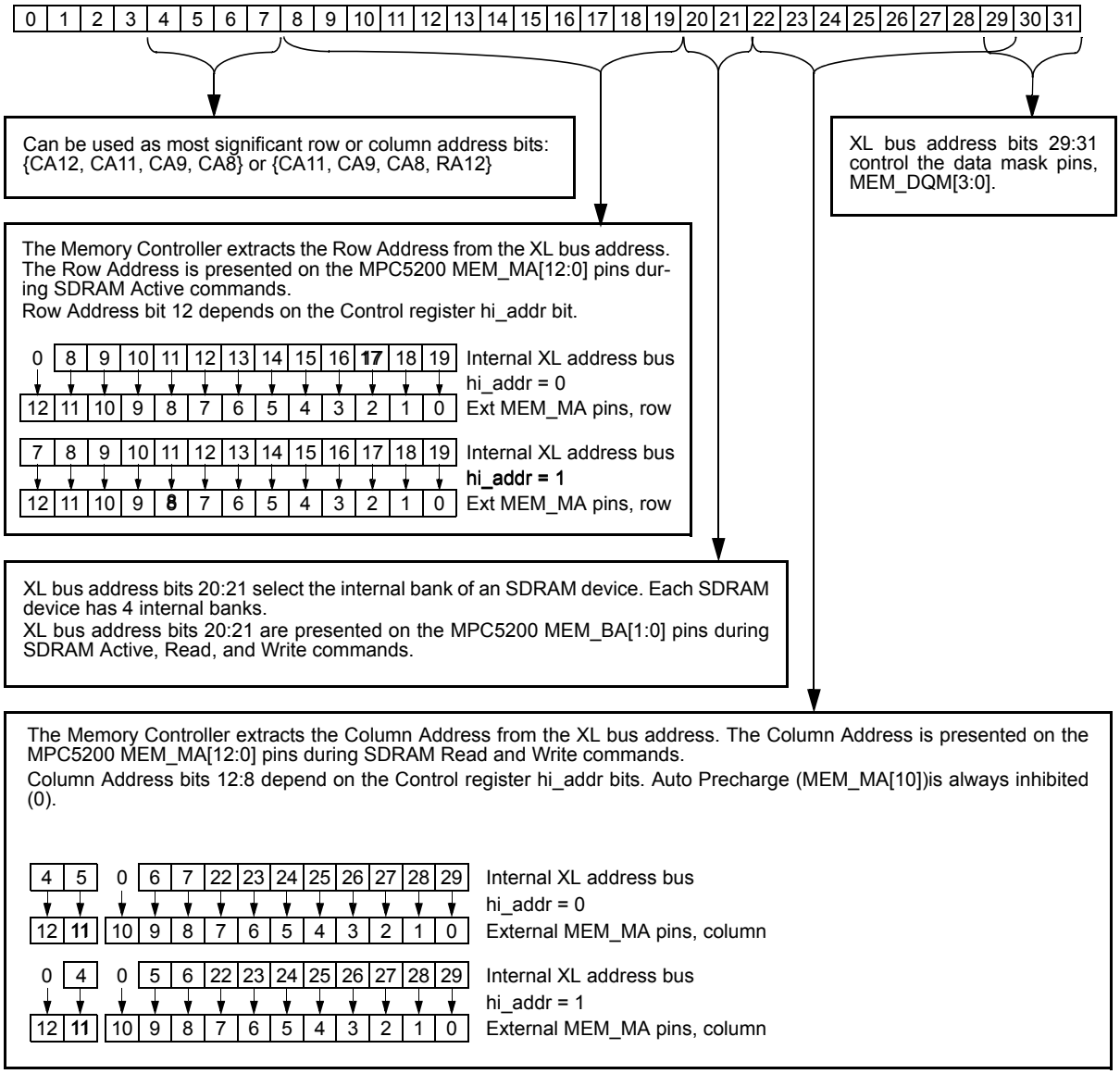


Figure 8-4. Address Bus Mapping

8.8.1 Example—Physical Address Multiplexing

The mapping of XL address bus to memory address bus is shown in Figure 8-4. The default mapping is:

- Row address comes from XLA[8:19]
- Column address comes from XLA[4:7, 22:29]
- Bank address comes from XLA[20:21]

Using the MT46V32M16 DDR SDRAM memory from Micron as an example, the device holds 512Mb organized as 8M x 16bit x 4banks. 2 devices are required to support the MPC5200 32bit memory data bus, giving a total 128MB of address space (assuming just one $\overline{CS}$).

The Micron data sheet shows the following requirements:

- 13 row address bits
- 10 column address bits
- 2 bank select bits

By default, the Memory Controller only provides 12 row address bits and 12 column address bits. To enable the 13th row address bit, the hi_addr bit of the Control register must be set to 1 (MBAR+0x0104, Control[7]). This also reduces the column address width to 11 bits.

Chapter 9
LocalPlus Bus (External Bus Interface)

9.1 Overview

The LocalPlus Bus is the external bus interface of the MPC5200. This multi-function bus system supports interfacing to external Boot ROM or Flash memories, external SRAM memories or other memory mapped devices. The following sections are contained herein:

The MPC5200 offers a shared external 32-bit address/data bus, which supports connections to PCI and ATA compliant devices, as well as memory mapped devices such as Flash memories, ROM, SRAM, gate-array logic, or other simple target (slave) devices with little or no additional circuitry. Separate control signals are used by each interface. The on-chip arbiter (called PCI Arbiter) controls the access to the shared AD bus for the different clients.

NOTE

If the PCI interface is NOT used (and internally disabled) the PCI control pins must be terminated as indicated by the PCI Local Bus specification. PCI control signals always require pull-up resistors on the motherboard (not the expansion board) to ensure that they contain stable values when no agent is actively driving the bus. This includes PCI_FRAME, PCI_TRDY, PCI_IRDY, PCI_DEVSEL, PCI_STOP, PCI_SERR, PCI_PERR, and PCI_REQ.

The PCI signals, which are not used as address in Large Flash mode, are drive low during a Large Flash access. This includes PCI_SERR, PCI_PERR, PCI_IDSEL, PCI_REQ, PCI_GNT and PCI_RESET.

The PCI interface is described in Chapter 10, *PCI Controller*. The ATA compliant interface is described in Chapter 11, *ATA Controller*. The interface for memory mapped devices, called LocalPlus Bus, is described in this chapter. The MPC5200 LocalPlus Controller (LPC) module implements the LocalPlus Bus interface.

The LocalPlus Bus interface provides a high flexibility and all its different operating modes can be selected by means of software configuration and in some cases minimal external logic (in multiplexed mode).

9.2 Features

LocalPlus has the following features:

- Interface to memory mapped or chip selected devices
- Two main modes of operation :
  - non-MUXed Modes
    - Legacy Modes (Address 8, 16, or 24 bits, Data 8 or 16 bits)
    - Most Graphics Mode (Address 24 bits, Data 32 bits)
    - Large Flash Mode (Address 26 bits, Data 8 or 16 bits)
  - MUXed Modes

  - (Address 8, 16, 24 or 25 bits, Data 8,16 or 32 bits, 2 Bank Selects)
- 8 Chip Select (CS) signals
  - Programmable Wait States per CS
  - Programmable Deadcycles per CS
  - Programmable Byte Swapping per CS
- Configurable Boot interface supporting PowerPC architecture code execution
- Dynamic bus sizing on some interfaces
- Support of BURST MODE FLASH devices
- DMA (BestComm) support allows data movement independently from the CPU
- NO support of misaligned accesses

9.3 Interface

The LocalPlus interface consists of:

- Address Bus
- Data Bus
- Chip Select signals CS0-7
- control signals:
  - R/$\overline{W}$ (Read/Write)
  - $\overline{ALE}$ (Address Latch Enable)
  - $\overline{ACK}$ (Acknowledge)
  - $\overline{TS}$ (Transfer Start)
  - $\overline{OE}$ (Output Enable)
  - TSIZ bits (Transfer Size)
  - Bank Select bits
- reference clock PCI_CLOCK

The reference clock PCI_CLOCK is always running, even if the PCI Controller is disabled.

9.3.1 External Signals

The external I/O bus is shared with the PCI AD bus and the ATA bus and requires arbitration for access to the external bus.

Table 9-1. LocalPlus External Signals

Signal	I/O	Definition
$\overline{CS}$ [7:0]	O	Chip Selects (active low), $\overline{CS}$[4] and $\overline{CS}$[5] shared with ATA, $\overline{CS}$[6] and $\overline{CS}$[7] shared with PSC3.
R/$\overline{W}$	O	Read/Write. 1 = Read, 0 = Write
EXT_AD[31:0]	I/O	AD Address / Data bus (bi-directional when used as data; bit 31=msb)
ACK	I/O	External Acknowledge input (non-burst transactions), BURST indication for Most Graphics or Large Flash Modes (Open Drain)
TS	O	Transfer Start (multiplexed transactions only)
OE	O	Output Enable
TSIZ[1:2]	O	Transfer Size (available in MOST Graphics only). **Note:** The MUXed Mode provides 3 bits TSIZ[0:2], which are available on EX_AD[30:28].

9.3.2 Block Diagram

The block diagram of the LocalPlus Controller (LPC) is shown in Figure 9-1. This diagram shows the non-multiplexed implementation of address and data lines.

The LPC is driven by the internal IP bus clock and the PCI_CLOCK. The supported ratios of the IP bus clock to the reference clock PCI_CLOCK (the one externally seen by peripherals) are 4:1, 2:1 and 1:1.

The reference clock is the PCI_CLOCK and all clock counts are referred to this clock. All transitions are synchronized to the rising edge of the PCI_CLOCK.

Start/Stop registers to define the CS address range for each CS output are contained in the MPC5200 MMAP register group, see Section 3.3.3.2, *Boot and Chip Select Addresses*. Registers in the LPC are accessed through the address range specified in the MPC5200 Internal Register Map. For more information, see Section 9.7, *Programmer's Model*. These registers control the operation of a particular CS and peripheral, when a "hit" occurs in the MMAP module for the corresponding CS space.

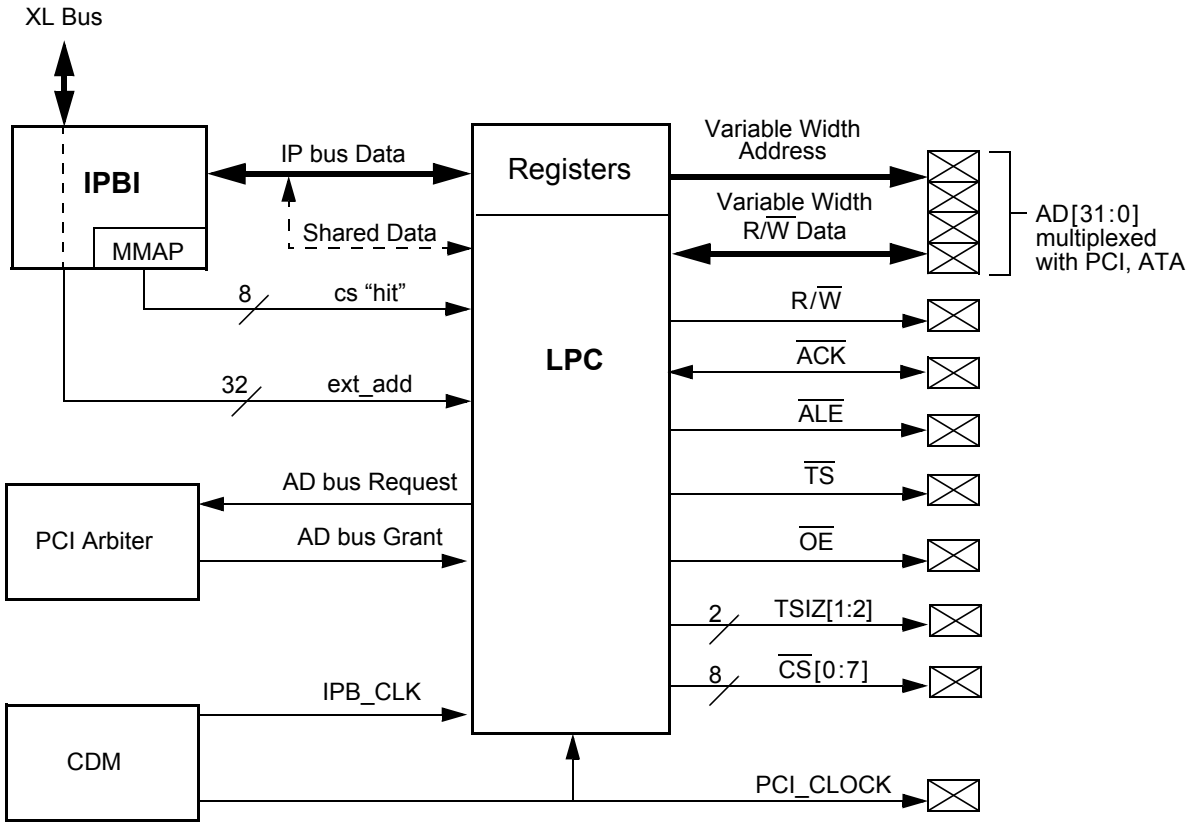


Figure 9-1. LPC Concept Diagram

NOTE

BestComm Interface + FiFo not shown

Not all pins are used in all modes.

For multiplexed bus implementation, external logic is required to capture the address phase as shown in Figure 9-2.

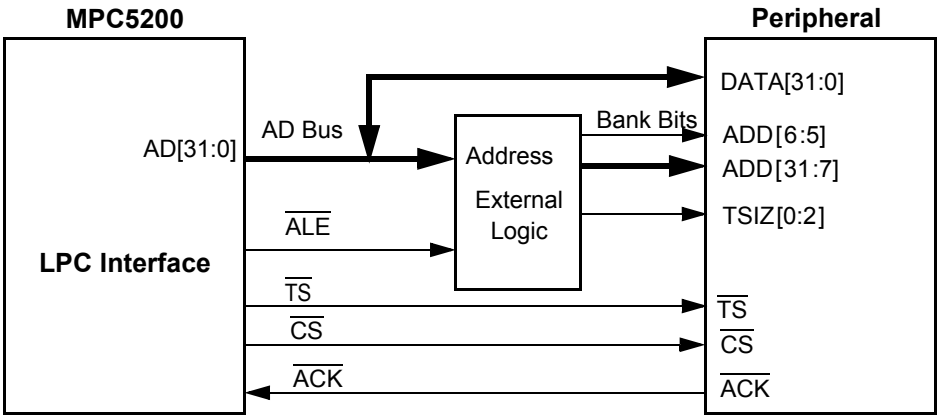


Figure 9-2. Muxed Mode Address Latching

9.4 Modes of Operation

There are 2 primary modes of operation:

- MUXed
- non-MUXed (Legacy, Large Flash, Most/Graphic modes, Burst and Non-Burst)

Within each mode, there is considerable flexibility to control the operation.

Each CS can be programmed to a different mode of operation (MUXed, non-MUXed, number of wait states, byte swapping etc.).

The MPC5200 always begins execution from the release of HRESET on the LocalPlus Bus and from the memory device connected to $\overline{CS}0$.

If an ATA Disk drive is present in the system, 2 CS signals may be taken up by the ATA interface. The ATA CSs can also be programmed to appear on other signals. For more information, see Chapter 11, *ATA Controller*.

MUXed mode allows devices with a larger address range be attached to the LocalPlus bus. In this mode the same 32-bit local bus presents an Address in an address tenure and Data in a data tenure, in a multiplexed fashion (similar to PCI protocol).

MUXed mode provides an $\overline{ALE}$ during the address phase and a $\overline{TS}$ during a separate data phase. This mode requires external logic to latch the address during the address tenure. An $\overline{ACK}$ input is provided and can be asserted to shorten (but not extend) wait states. The MUXed mode is available for all CSs, including CS0 (i.e., Boot Device).

The LocalPlus Bus on MPC5200 provides an Output Enable signal $\overline{OE}$ to achieve a complete glue less interface for most devices.

The logic equation for the internal generation of the $\overline{OE}$ signal is :

$\overline{OE} = \overline{CSx} + (NOT\ R/\overline{W})$

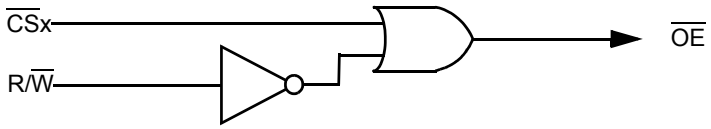


Figure 9-3. Output Enable Signal

MUXed and non MUXed modes support a variety of device configurations and are configurable on a per CS basis.

9.4.1 Non-MUXed Mode

In Non-MUXed mode the 32-bit address/data bus is divided into address and data lines. Eight different partitionings of address and data lines can be configured.

Table 9-2. Non-Muxed Mode Options

Category	Address Size	Data Size	Pins used	Memory size	Comments
Small	8	8	16	256 Bytes	Legacy Mode
Small	8	16	24	256 Bytes	Legacy Mode
Small	16	8	24	64 kBytes	Legacy Mode
Small	16	16	32	64 kBytes	Legacy Mode (BOOT OPTION)
Medium	24	8	32	16 MBytes	Legacy Mode (BOOT OPTION)
MOST/G	24	32	56	16 MBytes	MOST Graphics (BOOT OPTION) Burst support. No PCI or ATA support
Large	26	8	34	64 MBytes	Large Flash Mode (BOOT OPTION). Burst support. No PCI support.
Large	26	16	42	64 MBytes	Large Flash Mode (BOOT OPTION) Burst support. No PCI support

NOTE

The 24-bit data width is **not** supported.

The total pin number requires also the addition of the control signals CS, R/$\overline{W}$, $\overline{ACK}$, $\overline{OE}$, $\overline{TS}$ (MOST/Graphis and Large Flash mode) and TSIZ (MOST/Graphics mode) where available.

The total supported memory size has been calculated taking into account that when accessing 16/32 bit devices A1 and/or A0 can NOT be used.

The above options defined as BOOT Option are selectable via the reset configuration word. Other configurations are possible via software configuration (e.g., 8-bit data and 16-bit address). Figure 9-4 shows the operation of Non-MUXed Read/Write accesses.

TSIZ bits are available for MOST/Graphics mode. They appear on GPIO_WKUP_7 (TSIZ most significant bit, TSIZ 1) and TEST_SEL_1 (TSIZ least significant bit, TSIZ 2). Only TSIZEs of 1, 2, or 4 are supported.

TSIZ[1:2] are driven as follows:

01 = Transaction is 1 byte.

10 = Transaction is 2 bytes.

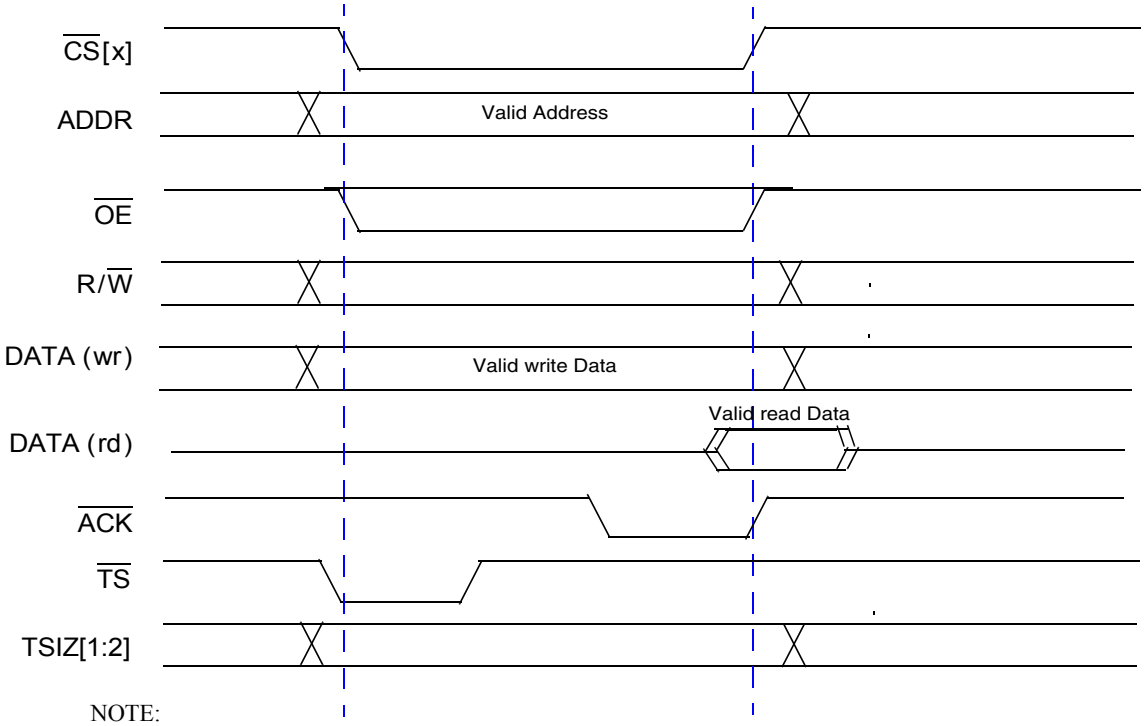
00 = Transaction is 4 bytes.

Other values are invalid and should not be required by the external peripheral!

Table 9-3 describes the various combinations of TSIZ, address and byte lanes for MOST/Graphis mode.

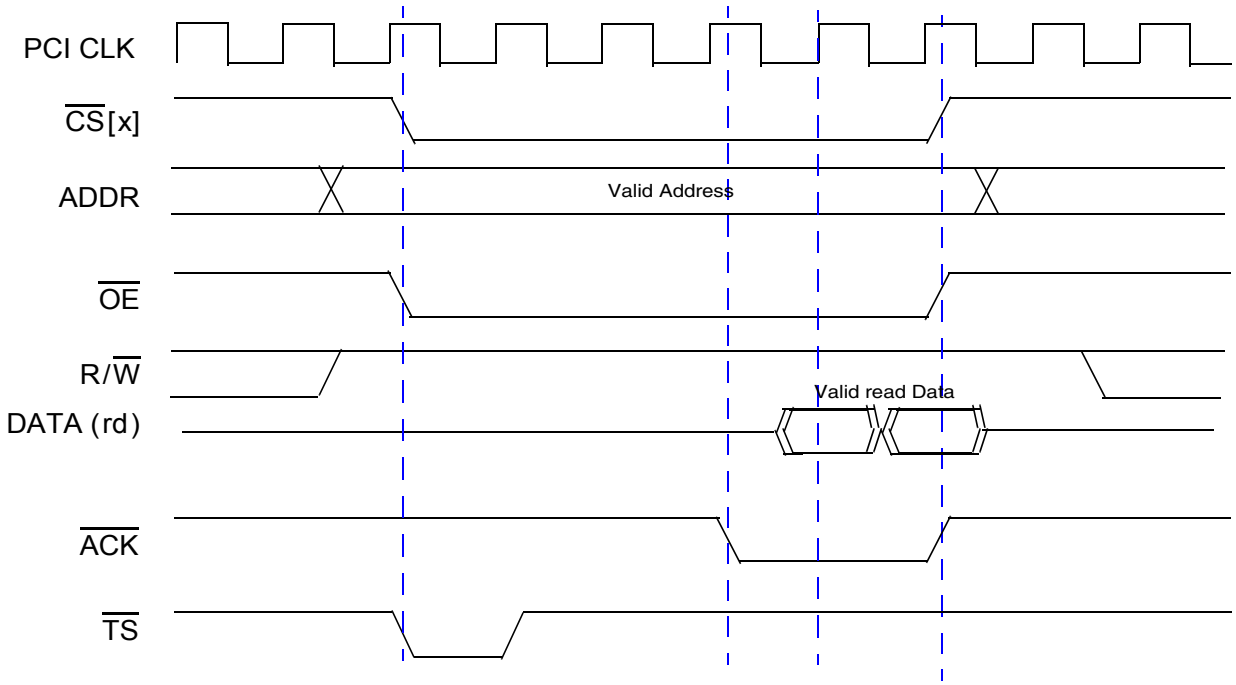
Table 9-3. Non-Muxed Aligned Data Transfers

Transfer Size	TSIZ[1:2]	Addr[1:0]	Data lanes			
			AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]
1 Byte	01	00	Data	--	--	--
		01	--	Data	--	--
		10	--	--	Data	--
		11	--	--	--	Data
2 Bytes	10	00	Data	Data	--	--
		10	--	--	Data	Data
4 Bytes	00	00	Data	Data	Data	Data



NOTE:
1. ACK can shorten the CS pulse width.
2. TS is only available in Large Flash and MOST Graphics mode.
3. TSIZ[1:2] are only available in MOST Graphics mode.

Figure 9-4. Timing Diagram—Non-MUXed Mode



NOTE:
1. ACK is output and indicates the burst.

Figure 9-5. Timing Diagram—Burst Mode

In this mode, the peripheral address and data lines are limited to a total of 32 in Legacy Modes, to 40 or 48 in Large Flash or to 56 in MOST Graphics mode. They are driven/read simultaneously on the external AD bus. A single dedicated R/$\overline{W}$ pin is driven to indicate read or write. An individually dedicated CS pin is driven low while an external access is active.

Wait states are programmable and simply select how many PCI clocks the CS pin (and related signals) remain asserted. Separate values are available for Read cycles versus Write Cycles. These values can be combined to create extremely long (up to 16 bits) Write cycles. Byte lane swapping is separately programmable between Reads versus Writes and can be used to perform Endian conversions. The 24-bit data width is **not** supported.

Peripherals can be marked as read-only or write-only by setting a control bit in the appropriate LPC register. Attempted accesses in violation of this setting are prevented and result in either a Bus Error and/or an Interrupt as controlled by corresponding Enable bits. Each CS pin can be individually enabled/disabled and the entire LPC module has a Master Enable bit. No software reset bit is provided or needed.

The non-multiplexed mode requires no external logic for interfacing to simple devices such as Flash ROM, E2PROM or SRAM. It is faster than the multiplexed mode because data and address are provided in a single tenure. The supported address space is limited by the 26 address lines.

9.4.2 MUXed Mode

In MUXed mode the addresses and data are multiplexed using dual tenure. First, the address is put on the shared address/data bus and $\overline{ALE}$ is asserted. Then the data is driven when the chip select is asserted. Twelve different modes of address and data sizes can be configured:

Table 9-4. MUXed Mode Options

Category	Address Size	Data Size	Memory Size per Bank	Memory Size Total	Comments
Legacy	8	8	256 Bytes	1 kBytes	
Legacy	8	16	256 Bytes	1 kBytes	A0 not used.
Legacy	8	32	256 Bytes	1 kBytes	A0, A1 not used.
Legacy	16	8	64 kBytes	256 kBytes	
Legacy	16	16	64 kBytes	256 kBytes	A0 not used.
Legacy	16	32	64 kBytes	256 kBytes	A0, A1 not used.
Legacy	24	8	16 MBytes	64 MBytes	
Legacy	24	16	16 MBytes	64 MBytes	A0 not used.
Legacy	24	32	16 MBytes	64 MBytes	A0, A1 not used.
Legacy	25	8	32 MBytes	128 MBytes	
Legacy	25	16	32 MBytes	128 MBytes	BOOT
Legacy	25	32	32 MBytes	128 MBytes	BOOT

NOTE

The 24-bit data width is **not** supported.

The total supported Memory space consists of four banks.

Bank select bits are written in a register by the G2_LE processor. They can be used as individual selects or as encoded values. They are presented on the bus during the address tenure as additional upper address bits.

In this mode, an address tenure is generated that can be up to 25 bits of active address. The additional address bits drive:

- a TSIZE value (3 bits)
- a Bank Select value (2 bits)

An ALE signal is asserted (active lo) during this address tenure. ALE width is always one PCI bus clock. The dedicated R/$\overline{W}$ output is also driven with ALE (and throughout the cycle). When ALE negates, the appropriate CS pin asserts (low) and the AD bus enters the data tenure. The CS pin and this data tenure remain active until the programmed wait states expire, or the peripheral responds with an ACK assertion. ACK polarity is active low, but can be programmed to be ignored. The data tenure can contain up to the full 32-bit width. However, the data width is programmable to support dynamically bus-sized transactions.

The MUXed mode requires external logic to latch the address during the address tenure and to decode bank selects if they are encoded. This mode is slower than the non-MUXed mode because data and address are multiplexed in time. The supported address space is limited by the 25 address lines. In MUXed mode, LocalPlus can access up to 128 MBytes of data divided into four banks each of 32 MBytes maximum.

9.4.2.1 Address Tenure

The address is presented on the corresponding AD bus bits up to a maximum of 25 bits (i.e., AD[24:0]). Smaller devices (with address ranges at 8, 16, or 24 respectively) must use the corresponding AD bits, beginning with AD[0]. AD[0] is the least significant address bit. Regardless of address size, the entire AD bus is driven during the address phase.

The Bank Select bits appear on AD[26] (Bank Select most significant bit) and AD[25] (Bank Select least significant bit). These bit values are pre-programmed into the corresponding LPC control register prior to initiating an external transaction.

The TSIZ bits appear on AD[30] (TSIZ most significant bit) to AD[28] (TSIZ least significant bit). These bits are calculated and driven by the LPC based on the internal Byte Lane enables on the IP bus.

NOTE

Only TSIZs of 1, 2, or 4 are supported.

TSIZ [0:2]/AD[30:28] are driven as follows:

001 = Transaction is 1 byte.

010 = Transaction is 2 bytes.

100 = Transaction is 4 bytes.

NOTE

Other values are invalid and should not be required by the external peripheral !

Table 9-5 describes the various combinations of TSIZ, address and byte lanes for 32 bit wide data bus.

Table 9-5. Non-Muxed Aligned Data Transfers

Transfer Size	TSIZ[0:2]	AD[1:0]	Data lanes			
			AD[31:24]	AD[23:16]	AD[15:8]	AD[7:0]
1 Byte	001	00	Data	--	--	--
		01	--	Data	--	--
		10	--	--	Data	--
		11	--	--	--	Data
2 Bytes	010	00	Data	Data	--	--
		10	--	--	Data	Data
4 Bytes	100	00	Data	Data	Data	Data

The ALE signal is active low and remains asserted for 1 external PCI bus clocks. When active any external latch should be transparent.

AD[31] & AD[27] are unused and are driven low by the LPC during the address tenure, they are used as data lines during the data phase in 32-bit modes.

9.4.2.2 Data Tenure

During Data Tenure, the following occurs:

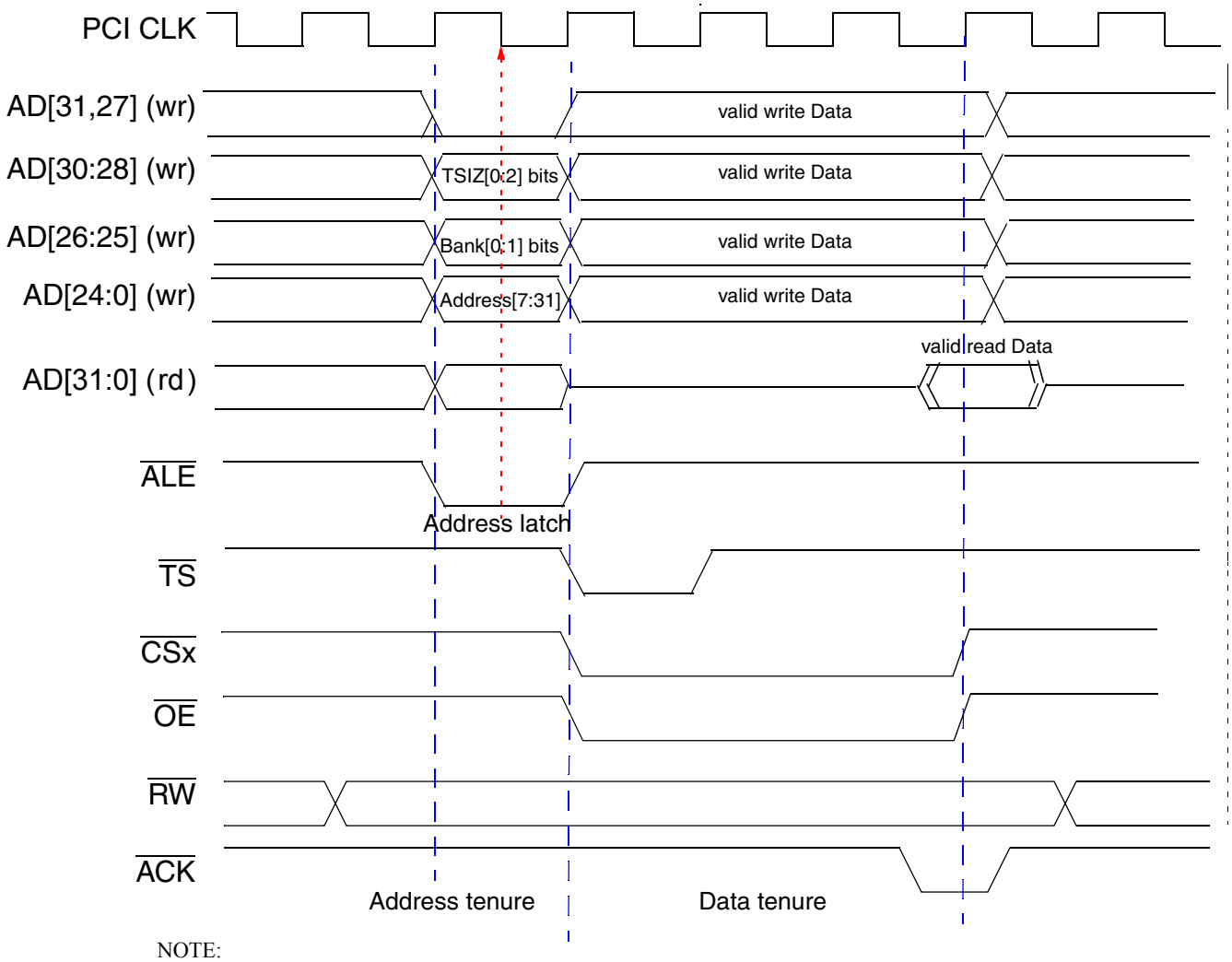
- In the case of a write to the peripheral, the LPC drives the indicated AD data bits.
- In the case of a read, the indicated AD bits are tri-stated by the LPC.

NOTE

AD[0] is treated as the least significant data bit. Any unused data bits (as indicated by the Data Size field in the associated control register) are driven low by the LPC. Therefore, they should NOT be driven by the peripheral or glue chip.

At the first PCI clock edge where the ACK input is detected as asserted, the LPC terminates the transaction and releases the bus on the **next** PCI Bus clock. AD bus control reverts to the PCI Controller, which is then responsible for driving default values on the bus. Obviously, any peripheral device **must** tri-state the AD bus when it is not in use.

Figure 9-6 shows a MUXed transaction type timing diagram.



NOTE:
1. ACK can shorten the CS pulse width.
2. Address should be latched during the low phase of ALE with the negative edge of the PCI clock.

Figure 9-6. Timing Diagram—MUXed Mode

9.5 Configuration

The LPC supports several options in terms of modes, address and data sizes, speed, and configuration which are described below.

9.5.1 Boot Configuration

After power-on reset (POR) the G2_LE processor accesses the local bus to fetch initial code sequences. Chip Select Boot (CS Boot) is dedicated for this purpose. CS Boot and CS0 are physically the same pins. The difference is that CS Boot is impacted by the reset configuration and is enabled after reset.

Several options are also available for boot code fetches. The boot configuration is determined during POR using the reset_configuration word.

The following boot code configuration options are available, see Table 9-6.

- MUXed or non-MUXed mode.
  - In MUXed mode Data bus can be 16- or 32-bits wide.
  - In non-MUXed Legacy mode Data bus can be 8- or 16-bits wide.
  - In non-MUXed MOST Graphics mode Data bus can be 32-bits wide.
  - In non-MUXed Large Flash modes Data bus can be 8- or 16-bits wide.
- The number of wait states during boot can be 4 or 48 PCI bus clock cycles.

- The boot address/exception table can be located at 0x0000 0100 or 0xfff0 0100.

The PowerPC architecture compatible processor core requires 64-bit instruction fetches. During boot code accesses from CS Boot space on-chip logic is provided to perform enough LocalPlus accesses to accumulate 64-bit instructions to be given to the G2_LE processor. For example, before passing the resulting 64-bit instruction to the G2_LE processor, LocalPlus logic does either:

- 8 accesses to an 8-bit device
- 4 accesses to a 16-bit device
- 2 accesses to a 32-bit device

NOTE

The Boot space supports cached instruction reads and "critical doubleword word first" transactions.
The Boot space does NOT support:

- an 8-bit wide MUXed mode configuration during boot.

After boot, CS Boot space can be programmed to act as other MPC5200 Chip Select spaces (CS0-7). This capability is described in the sections below.

9.5.2 Chip Selects Configuration

All Chip Selects CS0-7 have the same functionality. Only one CS can be active at any given time. Multiple CS windows should not overlap. In the case that an address "hit" is located in multiple CS windows, only one CS, the one with the highest priority, becomes active. The CS with the lowest number has the highest priority (CS0 highest priority, CS7 lowest priority).

CS Boot and CS0 are identical with the exception of their control registers contained in the MPC5200 MMAP register group, see Section 3.3.3.2, *Boot and Chip Select Addresses*. CS Boot and CS0 are physically the same pins. The difference is that CS Boot is impacted by the reset configuration and is enabled after reset, so boot is always performed only at CS Boot.

To change from CS Boot to CS0 the CS0 start and stop addresses must be configured and the disable of CS Boot must occur together with the Enable of CS0 (see example).

```
ipbi->control_reg = (ipbi->control_reg & ~CSCTRL_BOOT_EN) | CSCTRL_CS0_EN;
```

Deadcycles from 0 to 3 can be added to any CS read access and will occur in addition to any cycles which already exist. The configuration of Dead cycles are done by the Chip Select Deadcycle Control Register.

Burst Mode operations are supported on all CS and can be configured by the Chip Select Burst Control Register.

CS0-CS7 in MUXed mode:

- Supports 8-, 16- and 32-bit data reads and writes.
- Support of Dynamic bus sizing. This means read and write transactions greater than the defined port size are possible (up to a maximum of 32 bits).
- The LPC Controller creates multiple transactions at the defined port size to satisfy the transaction size requested up to a maximum of 32 bits. Transactions **less** than the defined port size are supported only if the peripheral can decode the TSIZE[0:2] bits, which indicate the current transaction size.
- 64-bit access is not supported. Internal logic is limited to 32-bits accesses.
- Support of Code execution

  The G2_LE processor can execute code from the LP bus from all CS.

CS0-CS7 non-MUXed mode:

- In non-MUXed mode the data port size can be 8, 16 or 32bits.
- Dynamic Bus Sizing for read and write transactions are supported at the defined port sizes. However, transactions that are **less** than the port size fail because no control signals exist to alert the peripheral to the current transaction size. TSIZE[1:2] bits are available in MOST Graphics mode on separate pins.
- Support of Burst access

9.5.3 Reset Configuration

The mode of the LocalPlus interface at boot is controlled by bits in the RST_CONFIG word described in Chapter 4, *Resets and Reset Configuration*. The following 6 RST_CONFIG bits control boot device operation from reset:

- BootType
- BootSize
- BootMostGraphics
- BootLargeFlash
- BootWait

- BootSwap

Table 9-1 describes possible boot settings.

Table 9-6. BOOT_CONFIG (RST_CONFIG) Options

Parameter	If Pulled Down (0)	If Pulled Up (1)	Notes
BootType	non-MUXed boot mode	MUXed boot mode	
BootSize	non-MUXed type: 8-bit data 24-bit address	non-MUXed type: 16-bit data 16-bit address	
	MUXed type: 16-bit data (25 bit address)	MUXed type: 32-bit data (25 bit address)	
BootMostGraphics	-	MostGraphics boot mode.	
LargeFlash	-	Large Flash boot mode	when active BootSize defines data size (8/16)
BootWait	Minimum Wait states 4 ipb_clk cycles	Maximum Wait states: 48 ipb_clk cycles	The ACK input can shorten wait states, if BootDevice supports it.
BootSwap	no Endian swapping applied to read from Boot Device	Standard Endian swapping performed on reads from Boot Device	If swap indicated: 8-bit access = no swap 16-bit access = 2Byte swap 32-bit access = 4Byte swap

9.6 DMA (BestComm) Interface (SCLPC)

The SCLPC interface provides a separate path from BestComm directly (on CommBus) to any peripheral. The supported transactions are limited to 1, 2, 4, or 8 bytes only.

A single FIFO with a size of 512 bytes (32 x 128 bits) supports half duplex operation (Transmit or Receive) only. If software configures a Transmit Packet, the Packet must be complete before a Receive operation can be configured and started.

9.7 Programmer's Model

Table 9-7 through Table 9-12 describe in detail the registers and bit meanings for configuring CS operation. There are eight identical chip select configuration registers, one for each CS output. However, the CS Boot ROM Configuration Register has active defaults for use by BOOTROM on CS0. All other configuration registers power-up disabled and require software intervention before the corresponding CS operates. The Chip Select Control Register is the enable register and the Chip Select Status Register serves as a status register. For Burst Mode the Chip Select Burst Control Register exists and the configuration of Dead cycles are done by the Chip Select Deadcycle Control Register.

NOTE

The address range registers for each CS reside in the MMAP register set rather than in the LPC register set. See Section 3.3.3.2, *Boot and Chip Select Addresses*.

9.7.1 Chip Select/LPC Registers—MBAR + 0x0300

There are 12 32-bit Chip Select/LocalPlus (CS/LP) registers. These registers are located at an offset from MBAR of 0x0300. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0300 + register address**

The following registers are available:

- Section 9-7, *Chip Select 0/Boot Configuration Register* (0x0300)
- Section 9-8, *Chip Select 1 Configuration Register* (0x0304)
- Section 9-9, *Chip Select Control Register* (0x0318)

- Section 9-7, *Chip Select 0/Boot Configuration Register* (0x0300)
- Section 9-10, *Chip Select Status Register* (0x031C)
- Section 9-11, *Chip Select Burst Control Register* (0x0328)
- Section 9-12, *Chip Select Deadcycle Control Register* (0x032C)

9.7.1.1 Chip Select 0/Boot Configuration Register—MBAR + 0x0300

Table 9-7. Chip Select 0/Boot Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WaitP								WaitX							
W																
RESET:	0	0	0	0	0	0	0	0	cfg	cfg	cfg	cfg	cfg	cfg	cfg	cfg

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	MX	Rsvd	AA	CE	AS		DS		Bank		WTyp		WS	RS	WO	RO
W																
RESET:	cfg	1	1	1	cfg	cfg	cfg	cfg	0	0	0	0	0	0	0	1

Bits	Name	Description
0:7	WaitP	Number of wait states to insert. Can be applied as a prescale to WaitX or used by itself, as specified by WTyp bits below. Wait states control how many PCI clocks the corresponding CS pin remains active.
8:15	WaitX	Base number of wait states to insert, or combined with WaitP as specified by WTyp bits below. **cfg operation**—If rstcfg[11] (on pad_eth_03) is zero then 4 wait states are in effect, else 48 wait states are in effect. Wait States equals the number of PCI clocks from CS assertion to when data must be valid from boot device.
16	MX	MX bit specifies whether a transaction operates as multiplexed or non-multiplexed. A multiplexed transaction presents address and data in different tenures. During the address tenure, $\overline{\text{ALE}}$ is asserted. At the end of $\overline{\text{ALE}}$, AD bus is switched to data tenure and $\overline{\text{CS}}$x pin is asserted. 0 = Non-multiplexed 1 = Multiplexed **cfg operation**—If rstcfg[14] on pad_eth_06 is low, boot operation is non-multiplexed (single tenure), else boot operation is multiplexed (dual tenure).
17	—	Reserved
18	AA	ACK Active. This bit defines whether $\overline{\text{ACK}}$ input is active or not. If AA is 1, programmed wait states can be overridden when/if the external device drives the $\overline{\text{ACK}}$ input low. If AA is 0, the $\overline{\text{ACK}}$ input is ignored. Wait states are still in effect. If no $\overline{\text{ACK}}$ is received, cycle terminates at end of wait state period. **Note:** Bit must be set to 0, to use ACK as burst indication signal during a burst transaction.
19	CE	An individual Enable bit—allows CS operation for the corresponding CS pin. CE must be high to allow operation. Chip Select Control Register ME bit must also be high, except when CS[0] is used for boot ROM. 1 = Enable 0 = Disabled, register writes can occur but no external access is generated.

Bits	Name	Description
20:21	AS	Address Size field—defines size of peripheral Address bus (in bytes) and must be consistent with physical connections. 00 = 8 bits 01 = 16 bits 10 = 24 bits 11 = > 25 bits See documentation for Physical Connection requirements. The combination of address size, data size, and transaction type (MX) must be consistent with the peripheral physical connection. In case of a multiplexed transaction, the entire address is driven regardless of address size field. **cfg operation**—If rstcfg[13] on pad_eth_05 is low, then the address size for non-multiplexed boot device is set to 24 bits (AS=10), else the boot device is treated as a 16 bit address (AS=01) device. For multiplexed mode boot devices the maximum 25 bits of address is always driven. This rstcfg bit more particularly affects the DS field below, and can be thought of as the "small" or "big" data size config bit.
22:23	DS	Data Size field—represents the peripheral data bus size (in bytes): 00 = 1 Byte 01 = 2 Bytes 10 = 3 Bytes (**Not Supported**) 11 = 4 Bytes **cfg operation**—If rstcfg[13] on pad_eth_05 is low, then the data size for non-multiplexed boot device is set to 8 bits (DS=00), else the boot device is treated as a 16 bit (DS=01) device. For multiplexed mode boot device the selection is 16 bit data or 32 bit data respectively.
24:25	Bank	Bank bits—are reflected on external AD lines (AD[26:25]) during Address tenure of a multiplexed transaction. Register bit 24 is the msb and appears on AD[26].
26:27	WTyp	Wait state Type bits—define the application of wait states contained in WaitP and WaitX fields, as follows: 00 = WaitX is applied to read and write cycles (WaitP is ignored). 01 = WaitX is applied to Read cycles, WaitP is applied to Write cycles. 10 = WaitX is applied to Reads, WaitP/WaitX (16-bit value) is applied to Writes. 11 = WaitP/Waitx (as a full 16-bit value) is applied to Reads and Writes.
28	WS	Write Swap bit—If high, Endian byte swapping occurs during writes to a peripheral. • For 8-bit peripherals, this bit has no effect. • For 16-bit peripherals, byte swapping can occur. • For 32-bit peripherals (possible in MUXed mode only) byte swap can occur. 1 = swap 0 = NO swap 2-byte swap is AB to BA, 4-byte swap is ABCD to DCBA. **Note:** Transactions at less than the defined port size (i.e., data size) apply swap rules as above, according to the current transaction size.
29	RS	Read Swap bit—Same as WS, but swapping is done when reading data from a peripheral. 1 = swap 0 = NO swap **cfg operation**—If rstcfg[12] on pad_eth_04 is low, data from the boot device is Endian swapped when read. This only has effect for boot devices configured as 16- or 32-bit data size.

Bits	Name	Description
30	WO	Write Only bit—If high, peripheral is treated as a write-only device. An attempted Read access doesn't generate a transaction to the peripheral. Additionally, the Write Only error bit is set.
31	RO	Read Only bit—If high, peripheral is treated as a read-only device. An attempted Write access doesn't generate a transaction to the peripheral. Additionally, the Read Only error bit is set. **Note:** This bit is high from Reset, indicating Boot Device is Read-Only.

Note:

1. The reset values defined as "cfg" depends on the Reset Configuration.
2. Large Flash mode is used, if AS is set to 11 and DS is set to 00 or 01.
3. MOST/Graphics mode is used, if AS is set to 10 and DS is set to 11.

9.7.1.2 Chip Select 1 Configuration Register—MBAR + 0x0304
Chip Select 2 Configuration Register—MBAR + 0x0308
Chip Select 3 Configuration Register—MBAR + 0x030C
Chip Select 4 Configuration Register—MBAR + 0x0310
Chip Select 5 Configuration Register—MBAR + 0x0314
Chip Select 6 Configuration Register—MBAR + 0x0320
Chip Select 7 Configuration Register—MBAR + 0x0324

Table 9-8. Chip Select 1 Configuration Register
Chip Select 2 Configuration Register
Chip Select 3 Configuration Register
Chip Select 4 Configuration Register
Chip Select 5 Configuration Register
Chip Select 6 Configuration Register
Chip Select 7 Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WaitP								WaitX							
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 lsb
R	MX	Rsvd	AA	CE	AS		DS		Bank		WTyp		WS	RS	WO	RO
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	WaitP	Number of Wait States to insert. Can be applied as a prescale to Wait X or used by itself, as dictated by the WTyp bits (see below). Wait States control how many PCI clocks the corresponding CS pin remains active.
8:15	WaitX	The base number of wait states to insert, or combined with WaitP as dictated by the WTyp bits below.

Bits	Name	Description
16	MX	MX bit specifies whether transaction operates as multiplexed or non-multiplexed. A multiplexed transaction presents address and data in different tenures. During the address tenure, $\overline{\text{ALE}}$ is asserted. At the end of $\overline{\text{ALE}}$, AD bus is switched to data tenure and $\overline{\text{CS}}$x pin is asserted. 0 = Non-multiplexed 1 = Multiplexed
17	—	Reserved
18	AA	ACK Active. This bit defines whether $\overline{\text{ACK}}$ input is active or not. If AA is 1, programmed wait states can be overridden when/if the external device drives the $\overline{\text{ACK}}$ input low. If AA is 0, the $\overline{\text{ACK}}$ input is ignored. Wait states are still in effect. If no $\overline{\text{ACK}}$ is received, cycle terminates at end of wait state period. **Note:** Bit must be set to 0, to use ACK as burst indication signal during a burst transaction.
19	CE	Chip Enable—bit allows CS operation for the corresponding CS pin. Must be high to allow operation. Chip Select Control Register ME bit must also be high. Enabled. 0 = Disabled, register writes can occur but no external access is generated.
20:21	AS	Address Size field—defines the peripheral address bus size in bytes, and must be consistent with the physical connections. 00 = 8 bits 01 = 16 bits 10 = 24 bits 11 = > 25 bits **Note:** The combination of address size, data size, and transaction type (MX) must be consistent with the physical peripheral connection. In a multiplexed transaction, the entire address is driven, regardless of the address size field.
22:23	DS	Data Size field—represents the peripheral data bus size (in bytes): 00 = 1 Byte 01 = 2 Bytes 10 = 3 Bytes (**Not Supported**) 11 = 4 Bytes
24:25	Bank	Bank bits—are reflected on external AD lines (AD[26:25]) during address tenure of a multiplexed transaction. Register bit 24 is the msb and appears on AD[26].
26:27	WTyp	Wait state Type bits—define application of wait states contained in WaitP and WaitX fields, as follows: 00 = WaitX is applied to Read and Write cycles (WaitP is ignored) 01 = WaitX is applied to Read cycles, WaitP is applied to Write cycles 10 = WaitX is applied to Reads, WaitP/WaitX (16-bit value) is applied to Writes 11 = WaitP/Waitx (as a full 16-bit value) is applied to Reads and Writes

Bits	Name	Description
28	WS	Write Swap bit—If high, Endian byte swapping occurs during writes to a peripheral. • For 8-bit peripherals, this bit has no effect. • For 16-bit peripherals, byte swapping can occur. • For 32-bit peripherals (possible in MUXed mode only) byte swap can occur. 1 = swap 0 = NO swap 2-byte swap is AB to BA, 4-byte swap is ABCD to DCBA. **Note:** Transactions at less than the defined port size (i.e., data size) apply swap rules as above, according to the current transaction size.
29	RS	Read Swap bit—Same as WS, but swapping is done when reading data from a peripheral. 1 = swap 0 = NO swap **Note:** Transactions at less than the defined port size (i.e., data size) apply swap rules as above, according to the current transaction size.
30	WO	Write Only bit—If high, peripheral is treated as a write-only device. An attempted Read access doesn't generate a transaction to the peripheral. Additionally, the Write Only error bit is set.
31	RO	Read Only bit—If high, peripheral is treated as a read-only device. An attempted Write access doesn't generate a transaction to the peripheral. Additionally, the Read Only error bit is set.
Note: 1. Large Flash mode is used, if AS is set to 11 and DS is set to 00 or 01. 2. MOST Graphics mode is used, if AS is set to 10 and DS is set to 11.		

9.7.1.3 Chip Select Control Register—MBAR + 0x0318

Table 9-9. Chip Select Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							ME	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	ME	Master Enable bit—a global module enable bit. If this bit is low, register access can still occur, but no external transactions are accepted. However, ME does not affect boot ROM operation on $\overline{CS}$[0]. If software wishes to disable $\overline{CS}$[0], it must write 0 to the Chip Select Boot ROM Configuration Register enable bit (CE).
8:31	—	Reserved

9.7.1.4 Chip Select Status Register—MBAR + 0x031C

Table 9-10. Chip Select Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		WOerr	ROerr	Rsvd	CSxerr			Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	—	Reserved
2	WOerr	Write Only error—If 1, it indicates a Read access was attempted on a peripheral marked as write-only. This is a sticky bit and must be written with 1 to be cleared. The CS number that relates to the error is reflected in the CSxerr field.
3	ROerr	Read Only error—If 1, it indicates a Write access was attempted on a peripheral marked as read-only. This is a sticky bit and must be written with 1 to be cleared. The CS number that relates to the error is reflected in the $\overline{\text{CS}}$xerr field.
4	—	Reserved
5:7	CSxerr	Chip Select error—Indicates CS number associated with WOerr or ROerr.
8:31	—	Reserved

9.7.1.5 Chip Select Burst Control Register—MBAR + 0x0328

Table 9-11. Chip Select Burst Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CW7	SLB7	Rsvd	BRE7	CW6	SLB6	Rsvd	BRE6	CW5	SLB5	Rsvd	BRE5	CW4	SLB4	Rsvd	BRE4
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 lsb
R	CW3	SLB3	Rsvd	BRE3	CW2	SLB2	Rsvd	BRE2	CW1	SLB1	Rsvd	BRE1	CW0	SLB0	Rsvd	BRE0
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	CW7	Chip Select 7 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
1	SLB7	Chip Select 7 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
2	—	Reserved
3	BRE7	Chip Select 7 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
4	CW6	Chip Select 6 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
5	SLB6	Chip Select 6 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
6	—	Reserved
7	BRE6	Chip Select 6 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
8	CW5	Chip Select 5 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
9	SLB5	Chip Select 5 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
10	—	Reserved
11	BRE5	Chip Select 5 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
12	CW4	Chip Select 4 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
13	SLB4	Chip Select 4 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
14	—	Reserved

Bits	Name	Description
15	BRE4	Chip Select 4 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
16	CW3	Chip Select 3 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
17	SLB3	Chip Select 3 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
18	—	Reserved
19	BRE3	Chip Select 3 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
20	CW2	Chip Select 2 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
21	SLB2	Chip Select 2 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
22	—	Reserved
23	BRE2	Chip Select 2 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
24	CW1	Chip Select 1 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
25	SLB1	Chip Select 1 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.
26	—	Reserved
27	BRE1	Chip Select 1 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.
28	CW0	Chip Select 0 Cache Wrap capable, set if peripheral burst can perform PPC cache wrap. This bit setting only applies in Large Flash or MOST Graphics Mode.
29	SLB0	Chip Select 0 Short/Long Burst, 0 for Short Burst only, 1 for Long Burst capable. Short burst is 8-bytes, used for Instruction fetches, and CDWF cache line bursts on XLB if cache wrap not capable. Long Burst capable means that peripheral can do 32-byte burst which hardware will generate for cache line aligned XLB bursts (and CDWF if peripheral tagged as cache wrap capable also). This bit setting only applies in Large Flash or MOST Graphics Mode.

Bits	Name	Description
30	—	Reserved
31	BRE0	Chip Select 0 Burst Read Enable, 1 to enable peripheral bursting for given chip select. Must be set to enable any Bursting reads. This bit setting only applies in Large Flash or MOST Graphics Mode.

Note:

1. CDWF is defined as "critical doubleword word first".
2. The bits for Chip Select 0 (CS0) control CS Boot too.
3. With a clock ratio 1:1:1 (66:66:66 MHz) it is not possible to burst in Large Flash mode.

9.7.1.6 Chip Select Deadcycle Control Register—MBAR + 0x032C

Table 9-12. Chip Select Deadcycle Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		DC7		Reserved		DC6		Reserved		DC5		Reserved		DC4	
W																
RESET:	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved		DC3		Reserved		DC2		Reserved		DC1		Reserved		DC0	
W																
RESET:	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

Bits	Name	Description
0:1	—	Reserved
2:3	DC7	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 7 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
4:5	—	Reserved
6:7	DC6	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 6 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
8:9	—	Reserved
10:11	DC5	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 5 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
12:13	—	Reserved
14:15	DC4	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 4 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
16:17	—	Reserved

Bits	Name	Description
18:19	DC3	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 3 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
20:21	—	Reserved
22:23	DC2	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 2 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
24:25	—	Reserved
26:27	DC1	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 1 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.
28:29	—	Reserved
30:31	DC0	Deadcycles can be specified as 0 to 3. Dead cycles will be added to the end of Chip Select 0 read access and will occur in addition to any cycles which may already exist. These cycles are to provide peripheral additional time to tri-state it's bus after a read operation. This is for all access types.

NOTE

Deadcycle counter is only used, if no arbitration to an other module (ATA or PCI) of the shared local bus happens. If an arbitration happens the bus can be dirven within 4 IPB clocks by an other module.

9.7.2 SCLPC Registers—MBAR + 0x3C00

There are 6 32-bit BestComm Registers for the LocalPlus (SCLPC). These registers are located at an offset from MBAR of 0x3C00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3C00 + register address**

The following registers are available:

- Section 9-13, *SCLPC Packet Size Register* (0x3C00)
- Section 9-14, *SCLPC Start Address Register* (0x3C04)
- Section 9-15, *SCLPC Control Register* (0x3C08)
- Section 9-16, *SCLPC Enable Register* (0x3C0C)
- Section 9-17, *SCLPC Bytes Done Status Register* (0x3C14)

9.7.2.1 SCLPC Packet Size Register—MBAR + 0x3C00

Table 9-13. SCLPC Packet Size Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							0	Packet Size							
W								Restart								
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 lsb
R	Packet Size															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	Restart	Once all registers have been programmed, software writes a 1 to this bit to begin a transfer. It will auto-clear and always reads back as zero.
8:31	Packet Size	This 24-bit field represents the number of bytes SCLPC is to transact before going idle and waiting for a Restart. **Note:** The co-location of Restart bit and Packet_Size field allows Software to both Restart a transaction AND change the Packet_Size in a single write. Maximum packet size is 16M-1 bytes.

9.7.2.2 SCLPC Start Address Register—MBAR + 0x3C04

Table 9-14. SCLPC Start Address Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Start Address															
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 lsb
R	Start Address															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:31	Start Address	Address of the first byte in the packet to be sent. This value must be aligned with the "BPT" (Bytes Per Transaction) field, described below. This address will appear directly at the peripheral and is completely independent of XLB address decoding logic.

9.7.2.3 SCLPC Control Register—MBAR + 0x3C08

Table 9-15. SCLPC Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					CSX			Reserved						Flush	RWb
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 lsb
R	Reserved							DAI	Reserved				BPT			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:4	—	Reserved
5:7	CSX	This field should be written with the Chip Select number associated with each DMA transaction. **Note:** LPC configuration registers associated with this CS also affect SCLPC transactions. The two work together.
8:13	—	Reserved
14	Flush	If set to 1, enables the assertion of SCLPC requestor at the completion of a *Read* Packet, regardless of the actual state of the physical fifo ALarm. Requestor will de-assert once the fifo goes empty. This is the fix for the familiar "Stale Read Data" fifo problem.
15	RWb	Read - Write bar. Controls direction of DMA transaction. 1 = SCLPC will read from the peripheral, i.e. Fifo Receive 0 = SCLPC will write to the peripheral, i.e. Fifo Transmit

Bits	Name	Description
16:22	—	Reserved
23	DAI	Disable Auto Increment. Normally, SCLPC and LPC will present sequential incrementing addresses to the peripheral as the Packet proceeds. If the peripheral is operating as a single address Fifo, then the DAI bit should be set to 1. When set, addresses to the peripheral will be stuck at Start_Address for every transaction. For DAI operation, the BPT field *MUST* be set to the port size of the peripheral.
24:27	—	Reserved
28:31	BPT	Bytes Per Transaction. Indicates number of bytes per transaction. The "only" valid entries in this field are decimal/hex 1, 2, 4, or 8 bytes (i.e. binary 0001, 0010, 0100, 1000). BPT should not be set to less than the peripheral port size, but certainly can be set to larger than the peripheral port size. The higher the BPT value, the greater the throughput. **Note:** Start_Address and Packet_Size values *must* be aligned/multiples of BPT. For DAI operation, BPT must be set to the peripheral port size.

9.7.2.4 SCLPC Enable Register—MBAR + 0x3C0C

Table 9-16. SCLPC Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							RC	Reserved							RF
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 lsb
R	Reserved						AIE	NIE	Reserved							ME
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	RC	Reset Controller. This bit allows for a Software reset of the SCLPC state machine. Writing a 1 to this bit will reset the SCLPC state machine. Reset will be maintained as long as this bit is high. Software must write this bit low to release the reset and start operation. **Note:** 1. Although RC does *not* reset this register interface, it does clear interrupt and interrupt status conditions. 2. Never reset the SCLPC Controller during a transaction (tx or rx).
8:14	—	Reserved
15	RF	Reset Fifo. This is the Fifo software reset bit. Writing a 1 to this bit will reset the SCLPC Fifo. The Fifo must not be in reset for normal operation. Software reset of the Fifo will clear the fifo of data, reset its read/write pointers, but *not* disturb previously programmed Alarm and Granularity settings. **Note:** Good Practice would be for software to set and clear the RC and RF bits prior to programming and starting a Packet.
16:21	—	Reserved

Bits	Name	Description
22	AIE	Abort Interrupt Enable. If set, and a fifo error occurs during packet transmission, a cpu interrupt from SCLPC will be generated. In any case, the Packet will be terminated and an Abort Status bit will be set. **Note:** This bit does *not* affect the Requestor to BestComm in any way.
23	NIE	Normal Interrupt Enable. This bit, if set enables a cpu interrupt to occur at the end of a normally terminated Packet. There is also a NT status bit which sets in any case. **Note:** This bit does *not* affect the Requestor to BestComm in any way.
24:30	—	Reserved
31	ME	Master Enable. This bit must be set to 1 to allow a Restart to be generated to the SCLPC state machine. Restart is achieved by writing 1 to Byte 0 of the Packet_Size register. This ME bit must also be set for a Restart to occur. **Note:** ME being low (inactive) will also clear Interrupt and Interrupt status. But it does *NOT* affect the BestComm Requestor.

9.7.2.5 SCLPC Bytes Done Status Register—MBAR + 0x3C14

Table 9-17. SCLPC Bytes Done Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			AT	Reserved			NT	Bytes Done							
W				rwc				rwc	Read Only							
RESET:	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Bytes Done															
W	Read Only															
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit does not reset to a defined value.

Bits	Name	Description
0:2	—	Reserved
3	AT	Abort Termination. This bit will be set to 1 if the Packet has terminated abnormally (which is only possible if a fifoError occurred). **Note:** This bit is ANDed with the AIE bit above to generate a single CPU interrupt signal to the core. This bit is "sticky write to 1" for clearing the bit and clearing the interrupt. **Note:** This bit (and any interrupt) is also cleared if; 1) RC bit is set, 2) ME bit is clear, or 3) Restart occurs.
4:6	—	Reserved

Bits	Name	Description
7	NT	Normal Termination. This bit is set to 1 whenever a complete Packet has been transferred successfully. **Note:** This bit is ANDed with the NIE bit above to generate a single CPU interrupt signal to the core. This bit is "sticky write to 1" for clearing the bit and clearing the interrupt.
8:31	Bytes Done	Bytes Done is updated dynamically by the SCLPC state machine to represent the actual number of bytes transmitted at a given point in time. At the normal conclusion of a Packet, the bytes_done field should match the Packet_Size field.

9.7.3 SCLPC FIFO Registers—MBAR + 0x3C40

LPC uses a single FIFO that changes direction based on the Rx/Tx mode. Software controls direction change and flushes FIFO before changing directions. FIFO memory is 512Bytes (32 x 128).

LPC FIFO is controlled by six 32-bit registers. These registers are located at an offset from MBAR of 0x3C40. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3C40 + register address**

Hyperlinks to the LPC FIFO registers are provided below:

- Section 9-18, *LPC Rx/Tx FIFO Data Word Register* (0x3C40)
- Section 9-19, *LPC Rx/Tx FIFO Status Register* (0x3C44)
- Section 9-20, *LPC Rx/Tx FIFO Control Register* (0x3C48)
- Section 9-21, *LPC Rx/Tx FIFO Alarm Register* (0x3C4C)
- Section 9-22, *LPC Rx/Tx FIFO Read Pointer Register* (0x3C50)
- Section 9-23, *LPC Rx/Tx FIFO Write Pointer Register* (0x3C54)

9.7.3.1 LPC Rx/Tx FIFO Data Word Register—MBAR + 0x3C40
LPC_rx/tx_fifo_data_word_register

Table 9-18. LPC Rx/Tx FIFO Data Word Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_Data_Word															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	FIFO_Data_Word															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit does not reset to a defined value.

Bits	Name	Description
0:31	FIFO_Data_Word	The FIFO data port. Reading from this location "pops" data from the FIFO, writing "pushes" data into the FIFO. During normal operation the BestComm Controller pushes data here. **Note:** ONLY full word access is allowed. If all byte enables are not asserted when accessing this location, a FIFO error flag is generated.

9.7.3.2 LPC Rx/Tx FIFO Status Register—MBAR + 0x3C44

Table 9-19. LPC Rx/Tx FIFO Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									Err	UF	OF	Full	HI	LO	Emty
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:8	—	Reserved
9	Err	Error—flag bit is essentially the logical "OR" of other flag bits and can be polled for detection of any FIFO error. After clearing the offending condition, writing 1 to this bit clears flag.
10	UF	UnderFlow—flag indicates read pointer has surpassed the write pointer. FIFO was read beyond empty. Resetting FIFO clears this condition; writing 1 to this bit clears flag.
11	OF	OverFlow—flag indicates write pointer surpassed read pointer. FIFO was written beyond full. Resetting FIFO clears this condition; writing 1 to this bit clears flag.
12	Full	FIFO full—this is NOT a sticky bit or error condition. Full indication tracks with FIFO state.
13	HI	High—FIFO requests attention, because high level alarm is asserted. To clear this condition, FIFO must be read to a level below the setting in granularity bits.
14	LO	Low—FIFO requests attention, because Low level alarm is asserted. To clear this condition, FIFO must be written to a level in which the space remaining is less than the granularity bit setting.
15	Emty	FIFO empty—this is NOT a sticky bit or error condition. Full indication tracks with FIFO state.
16:31	—	Reserved

9.7.3.3 LPC Rx/Tx FIFO Control Register—MBAR + 0x3C48

Table 9-20. LPC Rx/Tx FIFO Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		WFR	Reserved		GR			Reserved							
W																
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	—	Reserved
2	WFR	When bit sets, FIFO Controller assumes next data write is End of Frame (EOF). **Note:** This module does not support Framing. This bit should remain low.
3:4	—	Reserved
5:7	GR	Granularity—bits control high "watermark" point at which FIFO negates Alarm condition (i.e., request for data). It represents the number of free bytes times 4. 000 = FIFO waits to become completely full before stopping data request. 001 = FIFO stops data request when only one long word of space remains.
8:31	—	Reserved

9.7.3.4 LPC Rx/Tx FIFO Alarm Register—MBAR + 0x3C4C

Table 9-21. LPC Rx/Tx FIFO Alarm Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved							Alarm								
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:22	—	Reserved
23:31	Alarm	User writes these bits to set low level "watermark", which is the point where FIFO asserts request for BestComm Controller data filling. Value is in bytes. For example, with Alarm = 32, alarm condition occurs when FIFO contains 32 Bytes or less. Once asserted, alarm does not negate until high level mark is reached, as specified by FIFO control register granularity bits.

9.7.3.5 LPC Rx/Tx FIFO Read Pointer Register—MBAR + 0x3C50

Table 9-22. LPC Rx/Tx FIFO Read Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved							ReadPtr								
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:22	—	Reserved
23:31	ReadPtr	Value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. Value represents the Read address presented to the FIFO RAM.

9.7.3.6 LPC Rx/Tx FIFO Write Pointer Register—MBAR + 0x3C54

Table 9-23. LPC Rx/Tx FIFO Write Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved							WritePtr								
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:22	—	Reserved
23:31	WritePtr	Value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. Value represents the Read address presented to the FIFO RAM.

Notes

Chapter 10
PCI Controller

10.1 Overview

The Peripheral Component Interface (PCI) Bus is a high-performance bus with multiplexed address and data lines. It is especially suitable for high data-rate applications.

The MPC5200 PCI Controller module supports a 32-bit PCI initiator and target interface. As a target, access to the internal XL bus is supported. As an initiator, the PCI controller is coupled directly to the XL bus (as a slave) and available on the Communication Sub-System as a Multi-Channel DMA peripheral.

The 32-bit multiplexed address/data is shared with the ATA Controller and LocalPlus Controllers. However, control signals are on separate pins and only one operation (PCI, ATA, or LocalPlus) can be done at any given time.

The LocalPlus Large Flash and Most/Graphic interfaces are not compatible with any PCI operation. When these interfaces are needed, the PCI internal controller **must** be disabled by setting bit 16 (PCI_DIS) of the GPS Configuration register. Section 7.3.2.1.1, *GPS Port Configuration Register—MBAR + 0x0B00*

The MPC5200 contains PCI central resource functions such as the PCI Arbiter (Section 10.5, *PCI Arbiter*) and PCI reset control. The PCI bus clock is ***always*** sourced from the MPC5200 and either equal to 1, 1/2 the frequency of the Slave bus clock (IP bus clock) or 1/4 the frequency of the XLB clock. Even when the PCI internal controller is disabled, the PCI clock is sourced by the MPC5200.

A PCI reset signal is provided and implemented as an open-drain pin. An external (on board) pull-up resistor (e.g. 5.6 kOhm) is then required to ensure proper operation.

NOTE

If the PCI interface is NOT used (and internally disabled) the PCI control pins must be terminated as indicated by the PCI Local Bus specification. PCI control signals always require pull-up resistors on the motherboard (not the expansion board) to ensure that they contain stable values when no agent is actively driving the bus. This includes PCI_FRAME, PCI_TRDY, PCI_IRDY, PCI_DEVSEL, PCI_STOP, PCI_SERR, PCI_PERR, and PCI_REQ.

10.1.1 Features

- Supports system clock: Slave (IP) bus (internal peripheral slave bus) to PCI bus frequency ratios 1:1, 2:1. Or the XLB to PCI bus frequency ratio 4:1 (e.g. PCI runs at 33 MHz while the XLB bus runs at 132 MHz).
- Compatible with PCI 2.2 specification
- PCI initiator and target operation
- Fully synchronous design
- 32-bit PCI Address/Data bus
- PCI 2.2 Type 0 Configuration Space header
- Supports the PCI 16/8 clock rule
- PCI master Multi-Channel DMA or CPU access to PCI Bus
- High transfer rates at 66Mhz PCI clock, 512 byte buffer
- PCI to system bus address translation
- Target response is medium $\overline{\text{DEVSEL}}$ generation
- Initiator latency time-outs are **NOT** supported.
- Automatic retry of target disconnects
- Fast Back-to-Back transactions are **NOT** supported.

NOTE

The corresponding FC bit in the Configuration Status Register is fixed to ‘1’ indicating the opposite. Nonetheless no Fast Back-to-Back transaction is supported.

10.1.2 Block Diagram

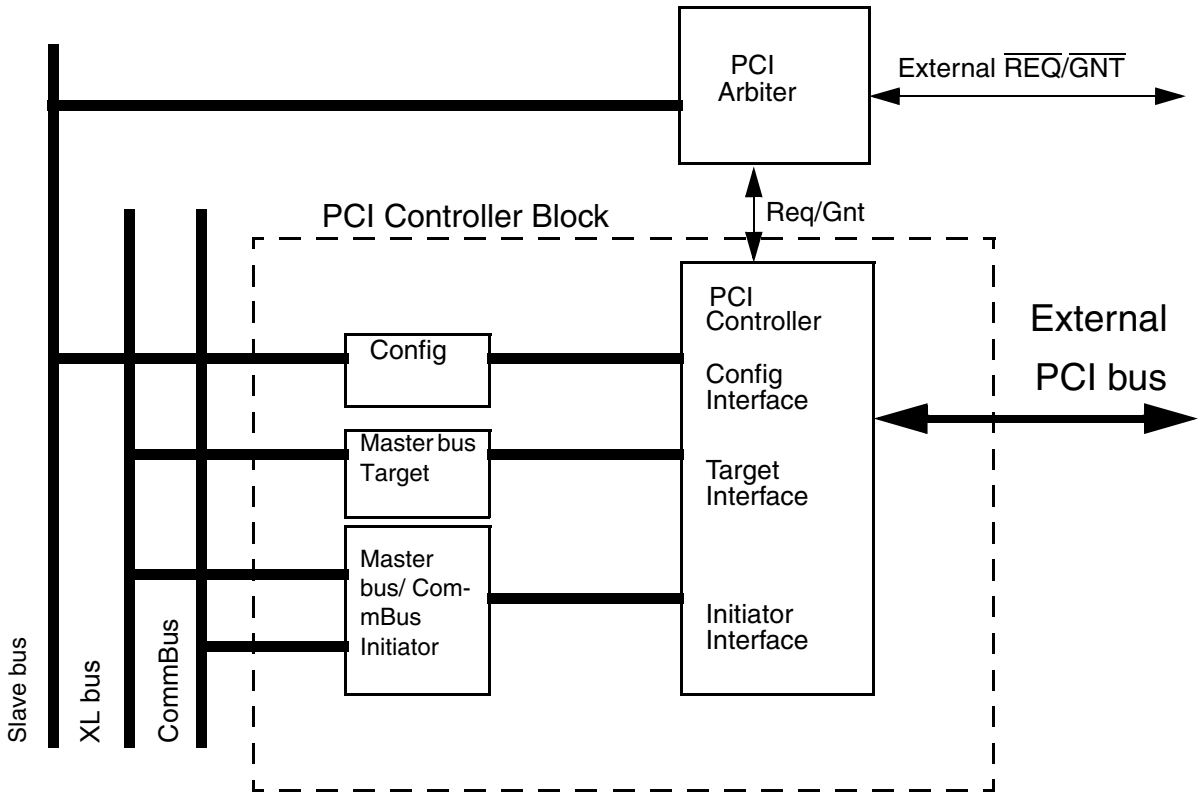


Figure 10-1. PCI Block Diagram

10.2 PCI External Signals

Table 10-1. PCI External Signals

Signal	I/O	Definition
AD[31:0]	I/O	Multiplexed Address and Data Bus (Shared with ATA and LPC). AD31 is the most significant bit while AD0 is the least significant as per the PCI specification. The entire PCI external bus is little Endian ordered.
PCI_CXBE[3:0]	I/O	Command/Bytes Enables
PCI_DEVSEL	I/O	Device Select
PCI_FRAME	I/O	Frame
PCI_IDSEL	I	Initialization Device Select
PCI_IRDY	I/O	Initiator Ready
PCI_PAR	I/O	Parity
PCI_CLK	O	PCI Clock
PCI_PERR	I/O	Parity Error
PCI_RST	O	PCI Reset
PCI_SERR	I/O	System Error
PCI_STOP	I/O	Stop
PCI_TRDY	I/O	Target Ready

For detailed description of the PCI bus signals, see the *PCI Local Bus Specification, Revision 2.2*.

10.2.1 PCI_AD[31:0] - Address/Data Bus

The PCI_AD[31:0] lines are a time multiplexed address data bus. The address is presented on the bus during the address phase while the data is presented on the bus during one or more data phases.

10.2.2 PCI_CXBE[3:0] - Command/Byte Enables

The PCI_CXBE[3:0] lines are time multiplexed. The PCI command is presented during the address phase and the byte enables are presented during the data phase.

10.2.3 PCI_DEVSEL - Device Select

The PCI_DEVSEL signal is asserted active low when MPC5200 decodes that it is the target of a PCI transaction from the address presented on the PCI bus during the address phase.

10.2.4 PCI_FRAME - Frame

The PCI_FRAME signal is asserted by a PCI initiator to indicate the beginning of a transaction. It is deasserted when the initiator is ready to complete the final data phase.

10.2.5 PCI_IDSEL - Initialization Device Select

The PCI_IDSEL signal is asserted during a PCI Type 0 Configuration Cycle to address the PCI Configuration header.

10.2.6 PCI_IRDY - Initiator Ready

The PCI_IRDY signal is asserted to indicate that the PCI initiator is ready to transfer data. During a write operation, assertion indicates that the master is driving valid data on the bus. During a read operation, assertion indicates that the master is ready to accept data.

10.2.6.1 PCI_PAR - Parity

The PCI_PAR signal indicates the parity of data on the PCI_AD[31:0] and PCI_CXBE[3:0] lines.

10.2.7 PCI_CLK - PCI Clock

The PCI_CLK signal is the clock for the internal PCI Controller and the external PCI system. The PCI clock is also used as reference clock for the LocalPlus synchronous interfaces (Burst Flash, ATA). The PCI_CLK is **always** (at all time) sourced by the MPC5200.

10.2.8 PCI_PERR - Parity Error

The PCI_PERR signal is asserted when a data phase parity error is detected if enabled.

10.2.9 PCI_RST - Reset

The PCI_RST signal is asserted active low by MPC5200 to reset the PCI bus. This signal is asserted after MPC5200 reset and must be negated (see <st-bold>10.3.2.1 Global Status/Control Register PCIGSCR(RW) —MBAR + 0x0D60) to enable usage of the PCI bus. An external shared pull-up resistor is needed connected to this pin.

10.2.10 PCI_SERR - System Error

The PCI_SERR signal, if enabled, is asserted by the MPC5200 *only* when an address phase parity error is detected.

10.2.11 PCI_STOP - Stop

The PCI_STOP signal is asserted by the currently addressed target to indicate that it wishes to stop the current transaction.

10.2.12 PCI_TRDY - Target Ready

The PCI_TRDY signal is asserted by the currently addressed target to indicate that it is ready to complete the current data phase.

10.3 Registers

MPC5200 has several sets of registers that control and report status for the different interfaces to the PCI controller: PCI Type 0 Configuration Space Registers, General Status/Control Registers, and Communication Sub-System Interface Registers. All of these registers are accessible as offsets of MBAR (the PCI interface is located starting at offset 0x0D00 relative to the MBAR register's value, while the BestComm interface starts at offset 0x3800). As an XL bus master, an external PCI bus master can access MBAR space for register updates and the internal SRAM.

NOTE

$\overline{\text{PCI\_RST}}$ is controlled by a bit in the register space and must first be cleared before external PCI devices wake-up. In other words, an external PCI master cannot load configuration software across the PCI bus until this bit is cleared by internal means.

All registers are accessible at an offset of MBAR in the memory space. There are two module offsets for PCI configuration space. One is allocated to the Communication Sub-System Interface registers and the other to all other PCI Controller Registers including the standard Type 0 PCI Configuration Space. Software reads from unimplemented registers return 0x00000000 and writes have no effect. See Section 3.2, *Internal Register Memory Map* for module offsets and descriptions of module responses.

Table 10-2. PCI Register Map

Register Offset	Mnemonic	Name
PCI Type 0 Configuration Registers		
0x00	PCIIDR	Device ID/Vendor ID
0x04	PCISCR	Status/Command
0x08	PCICCRIR	Class Code/Revision ID
0x0C	PCICR1	Configuration 1Register
0x10	PCIBAR0	Base Address Register 0
0x14	PCIBAR1	Base Address Register 1
0x18	Reserved	
...		
0x24		
0x28	PCICCPR	Cardbus CIS Pointer
0x2C	PCISID	Subsystem ID/Subsystem Vendor ID
0x30	PCIERBAR	Expansion ROM
0x34	PCICPR	Capabilities Pointer
0x38	Reserved	
0x3C	PCICR2	Configuration 2 Register
0x40	Reserved	
...		
0x5C		
General Control/Status Registers		
0x60	PCIGSCR	Global Status/Control Register
0x64	PCITBATR0	Target Base Address Translation Register 0
0x68	PCITBATR1	Target Base Address Translation Register 1
0x6C	PCITCR	Target Control Register
0x70	PCIIW0BTAR	Initiator Window 0 Base/Translation Address Register

Table 10-2. PCI Register Map (continued)

Register Offset	Mnemonic	Name
0x74	PCIIW1BTAR	Initiator Window 1 Base/Translation Address Register
0x78	PCIIW2BTAR	Initiator Window 2 Base/Translation Address Register
0x7C	Reserved	
0x80	PCIIWCR	Initiator Window Configuration Register
0x84	PCIICR	Initiator Control Register
0x88	PCIISR	Initiator Status Register
0x8C	PCIARB	PCI Arbiter Register
0x90	Reserved	
...		
0xF4		
0xF8	PCICAR	Configuration Address Register
0xFC	Reserved	

Table 10-3. PCI Communication System Interface Register Map

Register Offset	Mnemonic	Name
0x00	PCITPSR	Tx Packet Size
0x04	PCITSAR	Tx Start Address
0x08	PCITTCR	Tx Transaction Control Register
0x0C	PCITER	Tx Enables
0x10	PCITNAR	Tx Next Address
0x14	PCITLWR	Tx Last Word
0x18	PCITDCR	Tx Done Counts
0x1C	PCITSR	Tx Status
0x20	Reserved	
...		
0x3C		
0x40	PCITFDR	Tx FIFO Data
0x44	PCITFSR	Tx FIFO Status
0x48	PCITFCR	Tx FIFO Control
0x4C	PCITFAR	Tx FIFO Alarm
0x50	PCITFRPR	Tx FIFO Read Pointer
0x54	PCITFWPR	Tx FIFO Write Pointer

Table 10-3. PCI Communication System Interface Register Map (continued)

Register Offset	Mnemonic	Name
0x58	Reserved	
...		
0x7C		
0x80	PCIRPSR	Rx Packet Size
0x84	PCIRSAR	Rx Start Address
0x88	PCIRTCR	Rx Transaction Control Register
0x8C	PCIRER	Rx Enables
0x90	PCIRNAR	Rx Next Address
0x94	PCIRLWR	Rx Last Word
0x98	PCIRDCR	Rx Done Counts
0x9C	PCIRSR	Rx Status
0xA0	Reserved	
...		
0xBC		
0xC0	PCIRFDR	Rx FIFO Data
0xC4	PCIRFSR	Rx FIFO Status
0xC8	PCIRFCR	Rx FIFO Control
0xCC	PCIRFAR	Rx FIFO Alarm
0xD0	PCIRFRPR	Rx FIFO Read Pointer
0xD4	PCIRFWPR	Rx FIFO Write Pointer
0xD8	Reserved	
...		
0xFC		

10.3.1 PCI Controller Type 0 Configuration Space

MPC5200 supplies a type 0 PCI Configuration Space header. These registers are accessible as an offset from MBAR (Section 3.2, *Internal Register Memory Map*) or through externally mastered PCI Configuration Cycles.

NOTE

The internal PCI controller can discover itself (by means of connecting an AD line [preferably AD24 to AD31]to the PCI _IDSEL input). It is essential, when the PCI interface is used as a Target, to enable the internal PCI controller to access via the external PCI bus its own PCI registers. This is the only available way in order to clear any error flag RWC bit (Read/WriteClear bit).

Reg Addr	PCI DWord Offset	Reg	[31:24]	[23:16]	[15:8]	[7:0]
0x100	0x00	PCIIDR	Device ID		Vendor ID	
0x104	0x01	PCISCR	Status		Command	
0x108	0x02	PCICCRIR	Class Code			Revision ID

0x10C	0x03	PCICR1	BIST	Header Type	Latency Timer	Cache Line Size
0x110	0x04	PCIBAR0	BAR0			
0x114	0x05	PCIBAR0	BAR1			
0x118	0x06		Reserved			
...	...					
0x124	0x09					
0x128	0x0A	PCICCPR	CardBus CIS Pointer			
0x12C	0x0B	PCISID	Subsystem ID		Subsystem Vendor ID	
0x130	0x0C		Expansion ROM Base Address			
0x134	0x0D		Reserved			Cap_Ptr
0x138	0x0E		Reserved			
0x13C	0x0F	PCICR2	Min_Gnt	Max_Lat	Int Pin	Int Line
na	0x10		Reserved			
na	...					
na	0x3F					

PCI Dword Reserved space (0x10 - 0x3F) can be accessed only from an external PCI Configuration access.

NOTE

A PCI Double Word (DWORD) is a 32 bit long word. A PowerPC Double Word is instead a 64 bit word (according to the EABI rule) while a Word is a 32 bit value. In the following PCI Configuration space a DWORD refers **always** to a 32 bit word.

10.3.1.1 Device ID/ Vendor ID Registers PCIIDR(R) —MBAR + 0x0D00

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Device ID															
W																
RESET	0x5803															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Vendor ID															
W																
RESET	0x1057															

Bits	Name	Description
0:15	Device ID	This field is read-only and represents the PCI Device Id assigned to MPC5200 Its value is: 0x5803.
16:31	Vendor ID	This field is read-only and represents the PCI Vendor Id assigned to MPC5200 Its value is: 0x1057.

10.3.1.2 Status/Command Registers PCISCR(R/RW/RWC) —MBAR + 0x0D04

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PE	SE	MA	TR	TS	DT		DP	FC	R	66M	C	Reserved			
W	rwc	rwc	rwc	rwc	rwc			rwc								
RESET:	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved						F	S	St	PER	V	MW	Sp	B	M	IO
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits 31-27 and 24 are read-write-clear (RWC).

- Hardware can set RWC bits, but cannot clear them.
- Only PCI configuration cycles can clear RWC bits that are currently set by writing a 1 to the bit location. Writing a 1 to a RWC bit that is currently a 0 or writing a 0 to any RWC bit has no effect.

Bits	Name	Description
0	Parity Error Detected (PE)	This bit is set when a parity error is detected, even if the Parity Error Response bit in the Command Register (bit 6) is disabled. A CPU interrupt will be generated if the PCIGSCR[PEE] bit is set. This register is read-write-clear (RWC) via PCI configuration cycles.
1	System Error Signalled (SE)	This bit is set whenever MPC5200 generates a PCI System Error on the $\overline{\text{SERR}}$ line. This register is read-write-clear (RWC) via PCI configuration cycles.
2	Master Abort Received (MA)	This bit is set whenever MPC5200 is the PCI master and terminates a transaction (except for Special Cycle) with a Master-Abort. This register is read-write-clear (RWC) via PCI configuration cycles.
3	Target Abort Received (TR)	This bit is set whenever MPC5200 is the PCI master and a transaction is terminated by a Target Abort from the currently-addressed target. This register is read-write-clear (RWC) via PCI configuration cycles.
4	Target Abort Signalled (TS)	This bit is set whenever MPC5200 is the PCI target and it terminates a transaction with a Target Abort. This register is read-write-clear (RWC) via PCI configuration cycles.
5:6	DEVSEL# Timing (DT)	Fixed to 01. These bits encode a medium $\overline{\text{DEVSEL}}$ timing. This defines the slowest $\overline{\text{DEVSEL}}$ timing when MPC5200 is the PCI target (except configuration accesses).
7	Master Data Parity Error (DP)	This bit applies only when MPC5200 is PCI master and is set only if the following conditions are met: • MPC5200-as-master sets $\overline{\text{PERR}}$ itself during a read or detected it asserted by the target during a write • The Parity Error Response bit in the Command Register, bit 6, is set to 1 This register is read-write-clear (RWC) via PCI configuration cycles.
8	Fast Back-to-Back Capable (FC)	Fixed to 1. The MPC5200 PCI controller does **NOT** support Fast Back-to-Back transactions.

Bits	Name	Description
9	Reserved (R)	Fixed to 0. Prior to the 2.2 PCI Spec, this was the UDF (User Defined Features) Supported bit. 1 = Supported User Defined Features 0 = Does not support UDF
10	66 MHz Capable (66M)	Fixed to 1. This bit indicates that the PCI controller is 66 MHz capable.
11	Capabilities List (C)	Fixed to 0. This bit indicates that the PCI controller does not implement the New Capabilities List Pointer Configuration Register in DWORD 13 of the Configuration Space.
12:21	Reserved	These bits are reserved.
22	Fast Back-to-Back Transfer Enable (F)	The MPC5200 PCI controller does **NOT** support Fast Back-to-Back transactions. Setting this bit has no effect.
23	$\overline{\text{SERR}}$ enable (S)	This bit is an enable bit for the $\overline{\text{SERR}}$ driver. A value of zero disables the $\overline{\text{SERR}}$ driver. A value of 1 enables the $\overline{\text{SERR}}$ driver. Note: Address parity errors are reported only if this bit and bit 6 are 1. This bit is programmable (read/write from both the IP bus and PCI bus Configuration cycles).
24	Address and Data Stepping (St)	Fixed to 0. This bit indicates that the PCI controller never uses address/data stepping. Initialization software should write a 0 to this bit location.
25	Parity Error Response (PER)	This bit controls the device's response to parity errors. When set and a parity error is detected, the PCI controller asserts $\overline{\text{PERR}}$. When the bit is "0", the device sets its Detected Parity Error status bit (bit 0) in the event of a parity error, but does not assert $\overline{\text{PERR}}$. This bit is programmable (read/write from both the IP bus and PCI bus Configuration cycles).
26	VGA Palette Snoop Enable (V)	Fixed to 0. This bit indicates that the PCI controller is not VGA compatible. Initialization software should write a 0 to this bit location.
27	Memory Write and Invalidate Enable (MW)	This bit is an enable for using the Memory Write and Invalidate command. When this bit is 1, MPC5200-as-master may generate the command. When it is 0, Memory Write must be used instead. This bit is programmable (read/write from both the IP bus and PCI bus Configuration cycles).
28	Special Cycle Monitor or Ignore (Sp)	This bit is to determine whether or not to ignore PCI Special Cycles. Since MPC5200-as-target does not recognize messages delivered via the Special Cycle operation, a value of 1 should never be programmed to this register. This bit, however, is programmable (read/write from both the IP bus and PCI bus Configuration cycles).
29	Bus Master Enable (B)	This bit indicates whether or not MPC5200 has the ability to serve as a master on the PCI bus. A value of 1 indicates this ability is enabled. If MPC5200 is used as a master on the PCI bus (via XL bus or CommBus), a 1 should be written to this bit during initialization. Even if set to 0, a transaction initiated by an internal master (the core, BestComm) is allowed to take place. It is meant to be read by configuration software. This bit is programmable (read/write from both the IP bus and PCI bus Configuration cycles).

Bits	Name	Description
30	Memory Access Control (M)	This bit controls the PCI controller's response to Memory Space accesses. A value of 0 disables the response. A value of 1 allows the controller to recognize a Memory access. This bit is programmable (read/write from both the IP bus and PCI bus Configuration cycles).
31	IO access Control (IO)	Fixed to 0. This bit is not implemented because there is no MPC5200 IO type space accessible from the PCI bus. The PCI base address registers are Memory address ranges only. Initialization software should write a 0 to this bit location.

10.3.1.3 Revision ID/ Class Code Registers PCICCRIR(R) —MBAR + 0x0D08

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Class Code															
W																
RESET	0x0680															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Class Code (continued)								Revision ID							
W																
RESET	0x00								0x00							

Bits	Name	Description
0:23	Class Code	This field is read-only and represents the PCI Class Code assigned to MPC5200 Its value is: 0x068000. (Other bridge device)
24:31	Revision ID	This field is read-only and represents the PCI Revision Id for this version of MPC5200. Its value is: 0x00.

10.3.1.4 Configuration 1 Register PCICR1(R/RW) —MBAR + 0x0D0C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BIST								Header Type							
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 lsb
R	Lat timer[7:3]					Lat Timer[2:0]			Reserved				Cache Line Size			
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Built-In Self Test (BIST)	Fixed to 0x00. The PCI controller does not implement the Built-In Self Test register. Initialization software should write a 0x00 to this register location.
8:15	Header Type	Fixed to 0x00. The PCI controller implements a Type 0 PCI Configuration Space Header. Initialization software should write a 0x00 to this register location.
16:23	Latency Timer	This register contains the latency timer value, in PCI clocks, used when MPC5200 is the PCI master. The lower three bits of the register are hardwired low and the upper five bits are programmable (read/write from both the IP bus and PCI bus Configuration cycles). **Note:** The MPC5200 does NOT support initiator latency time-outs, the internal PCI Arbiter does not support preemption of the internal masters XIPCI or SCPCI. The internal master is granted until the transaction has been completed. The Latency Timer (LT) cannot terminate any transfer.
28:31	Cache Line Size	The four lower bits of this register are programmable (read/write from both the IP bus and PCI bus Configuration cycles). The value programmed specifies the cacheline size in units of DWORDs.

10.3.1.5 Base Address Register 0 PCIBAR0(RW) —MBAR + 0x0D10

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Base Address 0														Reserved	
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 lsb
R	Reserved												pref	range		IO/M#
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:13	Base Address Register 0 (BAR0)	MPC5200 PCI Base Address Register 0 (256Kbyte). Applies only when MPC5200 is target. These bits are programmable (read/write from both the IP bus and PCI bus Configuration cycles). This BAR register *should* be used to point at the internal MPC5200 register space (MBAR)
14:27	Reserved	These bits are reserved.
28	prefetchable access (pref)	Fixed to 0. This bit indicates that the memory space defined by BAR0 is NOT prefetchable. Configuration software should write a 0 to this bit location.
29:30	range	Fixed to 00. This register indicates that base address 0 is 32 bits wide and can be mapped anywhere in 32-bit address space. Configuration software should write 00 to these bit locations.
31	IO or Memory Space (IO/M#)	Fixed to 0. This bit indicates that BAR0 is for memory space. Configuration software should write a 0 to this bit location. 0 = Memory 1 = I/O

10.3.1.6 Base Address Register 1 PCIBAR1(RW) —MBAR + 0x0D14

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Base Address 1		Reserved													
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 lsb
R	Reserved												pref	range		IO/M#
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bits	Name	Description
0:1	Base Address Register 1 (BAR1)	MPC5200 PCI Base Address Register 1 (1Gbyte). Applies only when MPC5200 is target. These bits are programmable (read/write from both the IP bus and PCI bus Configuration cycles). This BAR register shall be used to point at the local SDRAM/DDR Memory Space. **Note:** The address 'Window' is much larger than the maximum theoretically supported physical memory. **Note:** This register should not point to the LocalPlus Memory Space. This is not supported.
2:27	Reserved	These bits are reserved.
28	prefetchable access (pref)	Fixed to 1. This bit indicates that the memory space defined by BAR1 is prefetchable. Configuration software should write a 1 to this bit location.
29:30	range	Fixed to 00. This register indicates that base address 1 is 32 bits wide and can be mapped anywhere in 32-bit address space. Configuration software should write 00 to these bit locations.
31	IO or Memory Space (IO/M#)	Fixed to 0. This bit indicates that BAR1 is for memory space. Configuration software should write a 0 to this bit location. 0 = Memory 1 = I/O

10.3.1.7 CardBus CIS Pointer Register PCICCPR(RW) —MBAR + 0x0D28

This optional register contains the pointer to the Card Information Structure (CIS) for the CardBus card. All 32 bits of the register are programmable by the Slave bus. It can only be read from the PCI Bus. Its reset value is 0x00000000.

10.3.1.8 Subsystem ID/ Subsystem Vendor ID Registers PCISID(R)—MBAR + 0x0D2C

The Subsystem Vendor ID register contains the 16-bit manufacturer identification number of the add-in board or subsystem that contains this PCI device. The Subsystem ID register contains the 16-bit subsystem identification number of the add-in board or subsystem that contains this PCI device. A value of zero in these registers indicates there isn't a Subsystem Vendor and Subsystem ID associated with the device. If used, software must write to these registers before any PCI bus master reads them.

All 32 bits of the register are programmable by the Slave bus. They can only be read from the PCI Bus. The reset value is 0x00000000.

10.3.1.9 Expansion ROM Base Address PCIERBAR(R) —MBAR + 0x0D30

Not implemented. Fixed to 0x00000000.

10.3.1.10 Capabilities Pointer (Cap_Ptr) PCICPR(R)—MBAR + 0x0D34

Not implemented. Fixed to 0x00000000.

10.3.1.11 Configuration 2 Register PCICR2 (R/RW) —MBAR + 0x0D3C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Maximum Latency								Minimum Grant							
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 lsb
R	Interrupt Pin								Interrupt Line							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Maximum Latency (Max_Lat)	Specifies how often, in units of 1/4 microseconds, the PCI controller would like to have access to the PCI bus as master. A value of zero indicates the device has no stringent requirement in this area. The register is read/write to/from the Slave bus, but read only from the PCI bus. **Note:** The MPC5200 does NOT support initiator latency time-outs, the internal PCI Arbiter does not support preemption of the internal masters XIPCI or SCPCI. The internal master is granted until the transaction has been completed. The Latency Timer (LT) cannot terminate any transfer.
8:15	Minimum Grant (Min_Gnt)	The value programmed to this register indicates how long the PCI controller as master would like to retain PCI bus ownership whenever it initiates a transaction. The register is programmable from the Slave bus, but read only from the PCI bus.
16:23	Interrupt Pin	Fixed to 0x00. Indicates that this device does **NOT** use an interrupt request pin.
24:31	Interrupt Line	Fixed to 0x00. The Interrupt Line register stores a value that identifies which input on a PCI interrupt controller the function's PCI interrupt request pin. Since no interrupt request pin is used, as specified in the Interrupt Pin register, this register has no function.

10.3.2 General Control/Status Registers

The General Control/Status Registers primarily address the configurability of the XL bus Initiator and Target Interfaces, though some also address global options which affect the Multi-Channel DMA interface (BestComm). These registers are accessed primarily internally as offsets of MBAR, but can also be accessed by an external PCI master if PCI base and Target base address registers are configured to access the space. See Section 10.6.2, *Address Maps* on configuring address windows.

10.3.2.1 Global Status/Control Register PCIGSCR(RW) —MBAR + 0x0D60

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	BM	PE	SE	Rsvd	xlb_clk to PCI_CLK differential			Reserved					ipg_clk to PCI_CLK differential		
W		rwc	rwc	rwc												
RESET	0	0	0	0	0	x	x	x	0	0	0	0	0	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Rsvd	BME	PEE	SEE	Reserved											PR
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Name	Description
0	Reserved	Unused bit. Software should write zero to this register.
1	Broken Master Detected (BM)	This bit is set when the PCI Arbiter detects a broken external PCI master. **Note:** In case of broken master detection the external PCI request will be ignored until external deassertion of PCI request or until a software reset (PCI Arbiter Softreset) or by Hardreset is detected. After broken master detection (PCI bus idle for 16 clocks) the arbiter will ignore any FRAME# assertion. A CPU interrupt will be generated if the PCIGSCR[BME] bit is set. This is a RWC (Read/WriteClear) bit: to clear it, software must write a '1' at this position.
2	PERR Detected (PE)	This bit is set when the PCI Parity Error line, $\overline{\text{PERR}}$, asserts (any device). A CPU interrupt will be generated if the PCIGSCR[PEE] bit is set. This is a RWC (Read/WriteClear) bit: to clear it, software must write a '1' at this position.
3	SERR Detected (SE)	This bit is set when a PCI System Error line, $\overline{\text{SERR}}$, asserts (any device). A CPU interrupt will be generated if the PCIGSCR[SEE] bit is set. This is a RWC (Read/WriteClear) bit: to clear it, software must write a '1' at this position.
4	Reserved	Unused bit. Software should write zero to this register.
5:7	xlb_clk to PCI_CLK differential (read only)	This bit field stores the XL bus clock to the PCI clock divide ratio. This field is read-only and the reset value is determined by the PLL multiplier (either 1, 2, or 4). Software can read these bits to determine a valid ratio. If the register contains a differential value that does not reflect the PLL settings, the PCI controller could malfunction.
8:12	Reserved	Unused bits. Software should write zero to this register.
13:15	ipg_clk to PCI_CLK differential (read only)	This bit field stores the Slave bus clock to the PCI clock divide ratio. This field is read-only and the reset value is determined by the PLL multiplier (either 1, 2, or 4). Software can read these bits to determine a valid ratio. If the register contains a differential value that does not reflect the PLL settings, the PCI controller could malfunction.
16	Reserved	Unused bit. Software should write zero to this register.
17	Broken Master Interrupt Enable (BME)	This bit enables CPU Interrupt generation when a broken Master is detected. When enabled, software must clear the BM status bit to clear the interrupt condition.
18	Parity Error Interrupt Enable (PEE)	This bit enables CPU Interrupt generation when the PCI Parity Error signal, $\overline{\text{PERR}}$, is sampled asserted. When enabled and $\overline{\text{PERR}}$ asserts, software must clear the PE status bit to clear the interrupt condition.
19	System Error Interrupt Enable (SEE)	This bit enables CPU Interrupt generation when a PCI System Error is detected on the $\overline{\text{SERR}}$ line. When enabled and $\overline{\text{SERR}}$ asserts, software must clear the SE status bit to clear the interrupt condition.
20:30	Reserved	Unused bits. Software should write zero to this register.
31	PCI Reset (PR)	This bit controls the external PCI $\overline{\text{RST}}$. When this bit is cleared, the external PCI $\overline{\text{RST}}$ deasserts. Setting this bit does not reset the internal PCI controller. The application software must not initiate PCI transactions while this bit is set. It is recommended that this bit be programmed last. The reset value of the bit is 1 (PCI $\overline{\text{RST}}$ asserted). **Note:** A global PCI reset should be asserted just by the MPC5200 controller. Any external common reset controller signal will be ignored by the internal PCI controller.

10.3.2.2 Target Base Address Translation Register 0 PCITBATR0(RW) —MBAR + 0x0D64

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Base Address Translation 0														Reserved	
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 lsb
R	Reserved															En
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:13	Base Address Translation 0	This base address register corresponds to a hit on the BAR0 in MPC5200 PCI Type 0 Configuration space register from PCI space. When there is a hit on MPC5200 PCI BAR0 (MPC5200 as Target), the upper 14 bits of the external PCI address (256Kbyte boundary) are written over by this register value to address some space in MPC5200. In normal operation, this value should be written during the initialization sequence only to point to the internal Register space.
14:30	Reserved	Unused bits. Software should write zero to this register.
31	Enable 0	This bit enables a transaction in BAR0 space. If this bit is zero and a hit on MPC5200 PCIBAR0 occurs, the target interface gasket will abort the PCI transaction.

10.3.2.3 Target Base Address Translation Register 1 PCITBATR1(RW) —MBAR + 0x0D68

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Base Address Translation 1		Reserved													
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 lsb
R	Reserved															En
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	Base Address Translation 1	This base address register corresponds to a hit on the BAR1 in MPC5200 PCI Type 0 Configuration space register (PCI space). When there is a hit on MPC5200 PCI BAR1 (MPC5200 as Target), the upper 2 bits of the external PCI address (1Gbyte boundary) are written over by this register value to address some 1Gbyte space in MPC5200. This register can be reprogrammed to move the window of MPC5200 address space accessed during a hit in PCIBAR1. It should be written by software during initialization to point to the internal SDR/DDR memory space. **Note:** This register should not point to the LocalPlus Memory Space. This is not supported.
2:30	Reserved	Unused bits. Software should write zero to this register.
31	Enable 1	This bit enables a transaction in BAR1 space. If this bit is zero and a hit on MPC5200 PCI BAR1 occurs, the target interface gasket will abort the PCI transaction.

10.3.2.4 Target Control Register PCITCR(RW) —MBAR + 0x0D6C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							LD	Reserved							P
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	Reserved	Unused bits. Software should write zero to this register.
7	Latrule Disable (LD)	This control bit applies only when MPC5200 is Target. When set, it prevents the PCI Controller from automatically issuing a retry disconnect due to the PCI 16/8 clock rule. The bit must be set before the 15th PCI clock for the first transfer and before the 7th clock for other transfers.
8:14	Reserved	Unused bits. Software should write zero to this register.
15	Prefetch Reads (P)	This bit controls fetching a line from memory in anticipation of a request from the external master. The target interface will continue to prefetch lines from memory as long as PCI_FRAME is asserted and there is space to store the data in the target read buffer. **Note:** This bit only applies to PCI reads in the address range for BAR 1 (prefetchable memory). **Note:** Prefetching is performed in response to a PCI memory-read-multiple command even if this bit is cleared.
16:31	Reserved	Unused bits. Software should write zero to this register.

10.3.2.5 Initiator Window 0 Base/Translation Address Register PCIIW0BTAR(RW)—MBAR + 0x0D70

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Window 0 Base Address								Window 0 Address Mask							
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 lsb
R	Window 0 Translation Address								Reserved							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Window 0 Base Address	One of three base address registers to determine an XL bus hit on PCI. At most, the upper byte of the address is decoded. The Window 0 Address Mask register determines what bits of this register to compare the XL bus address against to generate the hit. **Note:** The smallest possible Window is a 16 MByte block.
8:15	Window 0 Address Mask	The Window 0 Address Mask Register masks the corresponding XL bus base address bit of the base address for Window 0 (Window 0 Base Address) to instruct the address decode logic to ignore or “don’t care” the bit. If the base address mask bit is set, the associated base address bit of Window 0 is ignored when generating the PCI hit. Bit 16 masks bit 24, bit 17 masks bit 25, and so on. 0 Corresponding address bit is used in address decode 1 Corresponding address bit is ignored in address decode For XLB accesses to Window 0 address range, this byte also determines which upper 8 bits of the XLB address to pass on for presentation as a PCI address. Any address bit used to decode the XLB address, indicated by a “0”, will be translated. This provides a way to overlay a PCI page address onto the XLB address. A “1” in the Address Mask byte indicates that the XLB address bit will be passed to PCI unaltered.
16:23	Window 0 Translation Address	For any translated bit (described above), the corresponding value here will be driven onto the PCI address bus for the XL bus Window 0 address hit. **Note:** The Window Translation operation can not be turned off. If a direct mapping from XLB to PCI space is desired, program the same value to both the Window Base Address Register and Window Translation Address Register.
24:31	Reserved	Unused bits. Software should write zero to this register.

10.3.2.6 Initiator Window 1 Base/Translation Address Register PCIIW1BTAR(RW) —MBAR + 0x0D74

Table 1.

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Window 1 Base Address								Window 1 Address Mask							
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 lsb

R	Window 1 Translation Address								Reserved							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.2.7 Initiator Window 2 Base/Translation Address Register PCIIW2BTAR(RW) —MBAR + 0x0D78

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Window 2 Base Address								Window 2 Address Mask							
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 lsb
R	Window 2 Translation Address								Reserved							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.2.8 Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				Window 0 Control				Reserved				Window 1 Control			
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 lsb
R	Reserved				Window 2 Control				Reserved							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	Reserved	Reserved register. Write a zero to this register.
4:7	Window 0 Control [3:0]	Bit[3] - IO/M#. 0 = Window is mapped to PCI memory 1 = Window is mapped to PCI I/O Bit[2:1] - PCI Read Command (PRC). If bit[3] is programmed memory, “0”, then these bits are used to determine the type of PCI memory command to issue. See Table 10-15. If bit[3] is set to “1”, the value of these bits are meaningless. 00 = PCI Memory Read 01 = PCI Memory Read Line 10 = PCI Memory Read Multiple 11 = Reserved **Note:** A PCI write command is automatically detected and needs not to be explicitly configured. No PCI Write and Invalidate command is allowed in any case with this interface. Bit[0] - Enable. This bit is set to indicate the address registers that control the XLB initiator interface access to PCI initialized and will be used. The PCI Controller can begin to decode XLB PCI accesses. 0 = Do not decode XLB PCI accesses to Window 1 = Registers initialized - decode accesses to Window

8:11	Reserved	Reserved register. Write a zero to this register.
12:15	Window 1Control [3:0]	Bit[3] - IO/M#. Bit[2:1] - PRC. Bit[0] - Enable.
16:19	Reserved	Reserved register. Write a zero to this register.
20:23	Window 0 Control [3:0]	Bit[3] - IO/M#. Bit[2:1] - PRC. Bit[0] - Enable.
24:31	Reserved	Reserved register. Write a zero to this register.

10.3.2.9 Initiator Control Register PCIICR(RW) —MBAR + 0x0D84

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					REE	IAE	TAE	Reserved							
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 lsb
R	Reserved								Maximum Retries							
W																
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bits	Name	Description
0:4	Reserved	Unused bits. Software should write zero to this register.
5	Retry Error Enable (RE)	This bit enables CPU Interrupt generation in the case of Retry Error termination of a packet transmission. It may be desirable to mask CPU interrupts, but in such a case, software should poll the status bits to prevent a possible lock-up condition.
6	Initiator Abort Enable (IAE)	This bit enables CPU Interrupt generation in the case of Initiator Abort termination of a packet transmission. It may be desirable to mask CPU interrupts, but in such a case, software should poll the status bits to prevent a possible lock-up condition.
7	Target Abort Enable (TAE)	This bit enables CPU Interrupt generation in the case of Target Abort termination of a packet transmission. It may be desirable to mask CPU interrupts, but in such a case, software should poll the status bits to prevent a possible lock-up condition.
8:23	Reserved	Unused bits. Software should write zero to this register.
24:31	Maximum Retries	This bit field controls the maximum number of automatic PCI retries or master latency time-outs to permit per transaction. The retry counter is reset at the beginning of each transaction (i.e. it is not cumulative). Setting the Maximum Retries to 0x00 allows infinite automatic retry cycles and latency time-outs before the transaction will be abort and send back an error on XLB. A slow or malfunctioning Target might issue infinite retry disconnects or hold the data tenure open indefinitely, and therefore, permanently tie up the PCI bus if no Target Abort occurs.

10.3.2.10 Initiator Status Register PCIISR(RWC) —MBAR + 0x0D88

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					RE	IA	TA	Reserved							
W						rwc	rwc	rwc								
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:4	Reserved	Unused bits. Software should write zero to this register.
5	Retry Error (RE)	This flag is set if Max_Retries is set to a finite value (0x01 through 0xff) and the Target has performed Max_Retries number of retry disconnects for a single transaction. A retry error would generally indicate a broken or improperly accessed Target. A CPU interrupt will be generated if PCIICR[RE] bit is set. This is a RWC (Read/WriteClear) bit: to clear it, software must write a ‘1’ at this position.
6	Initiator Abort (IA)	This flag bit is set if the PCI controller issues an Initiator Abort flag. This indicates that no Target responded by asserting $\overline{\text{DEVSEL}}$ within the time allowed for subtractive decoding. A CPU interrupt will be generated if the PCIICR[IAE] bit is set. This is a RWC (Read/WriteClear) bit: to clear it, software must write a ‘1’ at this position.
7	Target Abort (TA)	This flag bit is set if the addressed PCI Target has signalled an Abort. A CPU interrupt will be generated if the PCIICR[TAE] bit is set. It is up to application software to query the Target’s status register and determine the source of the error. This is a RWC (Read/WriteClear) bit: to clear it, software must write a ‘1’ at this position.
8:31	Reserved	Unused bits. Software should write zero to this register.

10.3.2.11 PCI Arbiter Register PCIARB(RW) —MBAR + 0x0D8C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							ASR	Reserved							
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	Reserved	Unused bits. Software should write zero to this register.

7	PCI Arbiter Soft Reset (ASR)	This bit puts the PCI Arbiter in a reset condition. 1 = reset the PCI Arbiter 0 = release the PCI Arbiter **Note:** Resetting the PCI arbiter will disrupt any related transaction in progress and should be reserved only for error conditions, or when it is known that no PCI or AD bus transactions are in progress.
8:31	Reserved	Unused bits. Software should write zero to this register.

10.3.2.12 Configuration Address Register PCICAR (RW) —MBAR + 0x0DF8

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	E	Reserved							Bus Number							
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 lsb
R	Device Number					Function Number			dword						Reserved	
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	Enable (E)	The enable flag that controls configuration space mapping. When enabled, subsequent access to initiator window space defined as I/O in the PCIIWCR is translated into a PCI configuration access using the Configuration Address Register information (Section 10.6, *Application Information*). When disabled, a read or write to the window is passed through to the PCI bus as an I/O transaction using the. 1 = Enabled 0 = Disabled
1:7	Reserved	Unused bits. Software should write zero to this register.
8:15	Bus Number	This register field is an encoded value used to select the target bus of the configuration access. For target devices on the PCI bus connected to MPC5200, this field should be set to 0x00.
16:20	Device Number	This field is used to select a specific device on the target bus.
21:23	Function Number	This field is used to select a specific function in the requested device. Single-function devices should respond to function number 0b000.
24:29	dword	This field is used to select the dword address offset in the configuration space of the target device.
30:31	Reserved	Unused bits. Software should write zero to this register.

10.3.3 Communication Sub-System Interface Registers

The Communication Sub-System/Multi-Channel DMA interface (also shortly referred to as SCPCI) has separate control registers for transmit and receive operations.

10.3.3.1 Multi-Channel DMA Transmit Interface

PCI Tx is controlled by 14 ‘32-bit’ registers. These registers are located at an offset 0x3800 from MBAR. Register addresses are relative to this offset.

10.3.3.1.1 Tx Packet Size PCITPSR(RW) —MBAR + 0x3800

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Packet_Size[16:2]														Packet_Size[1:0]	
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	Packet_Size	User writes the number of bytes for transmit controller to send over PCI.The two low bits are hardwired low; only 32-bit data transfers to the FIFO are allowed. Writing to this register also completes a Restart Sequence as long as the Master Enable bit, PCITER[ME], is high and Reset Controller bit, PCITER[RC], is low.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.1.2 Tx Start Address PCITSAR(RW) —MBAR + 0x3804

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

Bits	Name	Description
0:31	Start_Add	User writes the PCI address to be presented for the first DWORD (32 bit) of a PCI packet. The PCI Tx controller will track and calculate the necessary address for subsequent transactions (addressing is assumed to be sequential from the start address).

10.3.3.1.3 Tx Transaction Control Register PCITTCR(RW) —MBAR + 0x3808

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				PCI_cmnd				Max_Retries							
W																
RESET	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved					Max_Beats			Reserved			W	Reserved			DI
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	Reserved	Unused. Software should write zero to these bits.
4:7	PCI_cmd	The user writes this field with the desired PCI command to present during the address phase of each PCI transaction. The default is Memory Write. This field is not checked for consistency and if written to an illegal value, unpredictable results will occur. If not using the default value, the user should write this register only once prior to any packet Restart.
8:15	Max_Retries	The user writes this field with the maximum number of retries to permit “perpacket”. The retry counter is reset when the packet completes normally or is terminated by a master abort, target abort, or an abort due to exceeding the retry limit. A slow or malfunctioning Target might issue infinite disconnects and therefore permanently tie up the PCI bus. A finite (0x01 to 0xfe) Max_Retries value will detect this condition and generate an interrupt. Setting Max_Retries to 0x00 or 0xff will not generate any interrupt.
16:20	Reserved	Unused bits. Software should write zero to these bits.
21:23	Max_Beats	The user writes this register with the desired number of PCI data beats to attempt on each PCI transaction. The default setting of 0 represents the maximum of eight beats per transaction. The transmit controller will wait until sufficient bytes are in the Transmit FIFO to support the indicated number of beats (NOTE: Each beat is four bytes). In the case that a packet is nearly complete and less than the Max_Beats number of bytes remain to complete the packet, the Transmit Controller will issue single-beat transactions automatically until the packet is finished.
24:26	Reserved	Unused. Software should write zero to these bits.
27	Word Transfer (W)	The user writes this register to disable the two high byte enables of the PCI bus during scpci initiated write transactions. The default setting is 0, enable all 4 byte enables.
28:30	Reserved	Unused. Software should write zero to these bits.
31	Disable address Incrementing (DI)	The user writes this register to disable PCI address incrementing between transactions. The default setting is 0, incrementing the address by 4 (4 byte data bus). **Note:** This feature is recommended when an external FIFO (with a fixed address) must be written.

10.3.3.1.4 Tx Enables PCITER(RW)—MBAR + 0x380C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RC	RF	Rsvd	CM	BE	Reserved		ME	Reserved		FEE	SE	RE	TAE	IAE	NE
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	Reset Controller (RC)	User writes this bit high to put Transmit Controller in a reset state. Other register bits are not affected. This Reset is intended for recovery from an error condition or to reload the Start Address when Continuous mode is selected. This Reset bit does not prohibit register access but it must be negated in order to initiate a Restart sequence (i.e. writing the Packet_Size register). If it is used to reload a Start Address then the Start_Add register must be written prior to asserting this Reset bit.
1	Reset FIFO (RF)	The FIFO will be reset and flushed of any existing data when set high. The Reset Controller bit and the Reset FIFO bit operate independently but clearly both must be low for normal operation.
2	Reserved	Unused bit. Software should write a zero to this bit.
3	Continuous mode (CM)	User writes this bit high to activate Continuous mode. In Continuous mode the Start_Add value is ignored at each packet restart and the PCI address is auto-incremented from one packet to the next. Also, the Packets_Done status byte will become active, indicating how many packets have been transmitted since the last Reset Controller condition. If the Continuous bit is low, software is responsible for updating the Start_Add value at each packet Restart.
4	Bus error Enable (BE)	User writes this bit high to enable Bus Error indications. Section 10.3.3.1.8, *Tx Status PCITSR(RWC) —MBAR + 0x381C* for Bus Error descriptions. Normally this bit will be low (negated) since illegal Slave bus accesses are not destructive to register contents (although it may indicate broken software). This bit does not affect interrupt generation.
5:6	Reserved	Unused. Software should write zero to these bits.
7	Master Enable (ME)	This is the Transmit Controller master enable signal. User must write it high to enable operation. It can be toggled low to permit out-of-order register updates prior to generating a Restart sequence (in which case transmission will begin when Master Enable is written back high), but it should not be used as such in Continuous mode because it has the side effect of resetting the Packets_Done status counter.
8:9	Reserved	Unused. Software should write zero to these bits.
10	FIFO Error Enable (FEE)	User writes this bit high to enable CPU Interrupt generation in the case of FIFO error termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
11	System error Enable (SE)	User writes this bit high to enable CPU Interrupt generation in the case of system error termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case someone should be polling the status bits to prevent a possible lock-up condition.

Bits	Name	Description
12	Retry abort Enable (RE)	User writes this bit high to enable CPU Interrupt generation in the case of retry abort termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
13	Target Abort Enable (TAE)	User writes this bit high to enable CPU Interrupt generation in the case of target abort termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
14	Initiator Abort Enable (IAE)	User writes this bit high to enable CPU Interrupt generation in the case of initiator abort termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
15	Normal termination Enable (NE)	User writes this bit high to enable CPU Interrupt generation at the conclusion of a normally terminated packet transmission. This may or may not be desirable depending on the nature of program control by Multi-Channel DMA or the processor core.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.1.5 Tx Next Address PCITNAR(R) —MBAR + 0x3810

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

Bits	Name	Description
0:31	Next_Address	This status register contains the next (unwritten) PCI address and is updated at the successful completion of each PCI data beat. It represents a byte address and is updated with the user-written Start_Add value whenever the Start_Add is reloaded. It is intended to be accurate even in the case of abnormal terminations on the PCI bus.

10.3.3.1.6 Tx Last Word PCITLWR(R) —MBAR + 0x3814

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

Bits	Name	Description
0:31	Last_Word	This status register indicates the last 32-bit data fetched from the FIFO and is designed for the case in which an abnormal PCI termination has corrupted the integrity of the FIFO data (for that word).

10.3.3.1.7 Tx Done Counts PCITDCR(R) —MBAR + 0x3818

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Bytes_Done															
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 lsb
R	Packets_Done															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	Bytes_Done	This status register indicates the number of bytes transmitted since the start of a packet. It is updated at the end of each successful PCI data beat. For normally terminated packets the Bytes_Done value and the Packet_Size values will be equal. If Continuous Mode is active the Bytes_Done value will read zero at the end of a successful packet and the Packets_Done field will be incremented.
16:31	Packets_Done	This status register indicates the number of packets transmitted and is active only if continuous mode is in effect. The counter is reset if the following occurs: • Reset Controller bit, PCITER[RC], is asserted (normal way to restart continuous mode) • Master Enable bit, PCITER[ME], becomes negated Master enable can reset Packets_Done status without disturbing continuous mode addressing. At any point in time, the total number of Bytes transmitted can be calculated as: (Packets_Done x Packet_Size) + Bytes_Done assuming Packet_Size is the same for all restart sequences

10.3.3.1.8 Tx Status PCITSR(RWC) —MBAR + 0x381C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							NT	BE3	BE2	BE1	FE	SE	RE	TA	IA
W								rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	Reserved	Unused. Software should write zero to these bits.
7	Normal Termination (NT)	This flag is set when any packet terminates normally. It is NOT set for abnormally terminated packets. **Note:** Flag does not **require** clearing, but does not clear until 1 is written, in which case 0 is read back (i.e., negated). The following flag bits operate similarly.
8	Bus Error type 3 (BE3)	This flag is set whenever a Slave bus transaction attempts to write to a Read-Only register. This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out. No register bit corruption occurs for this (or any other) bus error case.
9	Bus Error type 2 (BE2)	This flag is set whenever a Slave bus transaction attempts to write to a Reserved register (an entire 32-bit register, not just a Reserved bit or byte). This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out.
10	Bus Error type 1 (BE1)	This flag is set whenever a Slave bus transaction attempts to read a Reserved register (an entire 32-bit register, not just a Reserved bit or byte). This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out.
11	FIFO Error (FE)	This flag is set whenever the Transmit FIFO asserts its FIFO Error output. A CPU interrupt will be generated if the FIFO Error Enable (FEE) bit is set. The source of the error must be determined by reading the FIFO Error status register. Also, the error condition must be cleared at the FIFO prior to clearing this Sticky bit or this flag will continue to assert.
12	System Error (SE)	This flag is set in response to the Transmit Controller entering an illegal state. A CPU interrupt will be generated if the System error Enable (SE) bit is set. In normal operation this should never occur. The only recovery is to assert the Reset Controller bit, PCITER[RC], and clear this flag.
13	Retry Error (RE)	This flag is set if Max_Retries is set to a finite value (0x01 to 0xff) and the PCI transaction has performed retries in excess of the setting. A CPU interrupt will be generated if the Retry error Enable (RE) bit is set. The retry counter is reset at the beginning of each transaction (i.e. it is not cumulative throughout a packet) and would generally indicate a broken or improperly accessed Target.
14	Target Abort (TA)	This flag bit is set if the PCI controller has issued a Target Abort (which means the addressed PCI Target has signalled an Abort). A CPU interrupt will be generated if the Target Abort Enable (TAE) bit is set. It is up to application software to query the Target's status register and determine the source of the error. The coherency of the Transmit FIFO data and the Transmit Controller's status registers (Next_Address, Bytes_Done, etc.) should remain valid.

Bits	Name	Description
15	Initiator Abort (IA)	This flag bit is set if the PCI controller issues an Initiator Abort flag. This indicates that no Target responded but further status information can be read from the PCI Configuration interface. A CPU interrupt will be generated if the Initiator Abort error Enable (IAE) bit is set. The coherency of the Transmit FIFO data and the Transmit Controller's status registers (Next_Address, Bytes_Done, etc.) should remain valid.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.1.9 Tx FIFO Data Register PCITFDR(RW) —MBAR + 0x3840

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_Data_Word															
W																
RESET	uninitialized random 16 bit value															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	FIFO_Data_Word															
W																
RESET	uninitialized random 16 bit value															

Bits	Name	Description
0:31	FIFO_Data_Word	This is the data port to the FIFO. Reading from this location will "pop" data from the FIFO, writing data will "push" data into the FIFO. During normal operation the Multi-Channel DMA controller will be pushing data here. The PCI controller will pop data for transmission from a dedicated peripheral port, so the user program should not be reading here. At reset any uninitialized random 32 bit value is read at this address. A FIFO reset must be always performed before first accessing the FIFO. **Note:** Only full 32-bit accesses are allowed. If all Byte enables are not asserted when accessing this location, FIFO data will be corrupted.

10.3.3.1.10 Tx FIFO Status Register PCITFSR(R/RWC) —MBAR + 0x3844

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									RXW	UF	OF	FR	Full	Alarm	Empty
W										rwc	rwc	rwc				
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1

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

Bits	Name	Description
0:8	Reserved	Unused. Software should write zero to these bits.
9	Receive Wait Condition (RXW)	This flag bit indicates that the ipf_rcv bus is incurring wait states because there is not enough room in the FIFO to accept the data without causing overflow. This bit will cause the error outputs (fifoError, ipf_rcv_error, ipf_xmit_error) to assert unless the RXW_MASK bit in the FIFO Control register is set. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.
10	UnderFlow (UF)	This flag bit indicates that the read pointer has surpassed the write pointer. In other words the FIFO has been read beyond Empty. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.
11	OverFlow (OF)	This flag bit indicates that the write pointer has surpassed the read pointer. In other words the FIFO has been written beyond Full. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.
12	Frame Ready (FR)	The FIFO has a complete Frame of data ready for transmission. This module does not provide support for Data Framing applications, so this bit should be ignored.
13	Full	The FIFO is Full. This is not a sticky bit or error condition. The Full indication tracks with the state of the FIFO.
14	Alarm	When the FIFO pointer is at or below the Alarm “watermark”, as written by the user according to the Alarm and Control registers settings, this bit is set, automatically signalling to the DMA engine the need to re-fill the FIFO. By writing a ‘1’ to this bit software can enforce a re-evaluation of the ‘alarm’ condition.
15	Empty	The FIFO is empty. This is not a sticky bit or error condition.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.1.11 Tx FIFO Control Register PCITFCR(RW) —MBAR + 0x3848

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					GR			IP_MASK	FAE_MASK	RXW_MASK	UF_MASK	OF_MASK	Reserved		
W																
RESET	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0

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

Bits	Name	Description
0:4	Reserved	Unused. Software shall write zero to these bits. (R/W)
5:7	Granularity (GR)	Granularity bits control high “watermark” point at which FIFO negates Alarm condition (i.e., request for data). It represents the number of free Bytes, which is given by the granularity value multiplied by 4. **Note:** A granularity setting of zero should be avoided because it means the Alarm bit (and the Requestor signal) will not negate until the FIFO is completely full. The Multi-Channel DMA module may perform up to 2 additional data writes after the negation of a Requestor due to its internal pipelining

Bits	Name	Description
8	IP_MASK	Illegal Pointer Mask When this bit is set, the FIFO controller masks the Status register's IP bit from generating an error.
9	FAE_MASK	When this bit is set, the FIFO controller masks the Status Register's FAE bit from generating an error.
10	RXW_MASK	When this bit is set, the FIFO controller masks the Status Register's RXW bit from generating an error. (To help with backward compatibility, this bit is asserted at reset.)
11	UF_MASK	When this bit is set, the FIFO controller masks the Status Register's UF bit from generating an error.
12	OF_MASK	When this bit is set, the FIFO controller masks the Status Register's OF bit from generating an error.
13:15	Reserved	Unused. Software should write zero to these bits.
16:31	Reserved	Unused. Software should write zero to these bits. (R/W)

10.3.3.1.12 Tx FIFO Alarm Register PCITFAR(RW) —MBAR + 0x384C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				Alarm				Alarm							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	Reserved	Unused. Software should write zero to these bits.
20:31	Alarm	User writes these bits to set low level "watermark", which is the point where FIFO asserts request for Multi-Channel DMA controller data filling. Value is in Bytes. For example, with Alarm = 32, alarm condition occurs when FIFO contains less than 32 Bytes. Once asserted, alarm does not negate until high level mark is reached, as specified by FIFO control register granularity (GR) bits. **Note:** An Alarm setting less than the value of Max_Beats x 4 should be avoided. The transmit operation waits for the data to be stored in the FIFO before transmission onto the PCI bus. (e.g. A Max_setting of 0 represents eight beats (32-bits each) per transaction. The value of Alarm is in bytes. Ex: the value programmed to the Alarm register should be at least 0x20 (32 bytes) for the Multi-Channel DMA to continue to write enough data to complete at least one PCI burst.) **Note:** TX PCI FIFO is 512 bytes deep.

10.3.3.1.13 Tx FIFO Read Pointer Register PCITFRPR(RW) —MBAR + 0x3850

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				ReadPtr											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	Reserved	Unused. Software should write zero to these bits.
20:31	ReadPtr	This value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. The value represents the Read address presented to the FIFO RAM.

10.3.3.1.14 Tx FIFO Write Pointer Register PCITFWPR(RW) —MBAR + 0x3854

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				WritePtr											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	Reserved	Unused bits. Software should write zero to these bits.
20:31	WritePtr	Value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. Value represents the Write address presented to the FIFO RAM.

This marks the end of the PCI Multi-Channel DMA Transmit Interface description.

10.3.3.2 Multi-Channel DMA Receive Interface

PCI Rx is controlled by 13 32-bit registers. These registers are located at an offset from MBAR. Register addresses are relative to this offset.

10.3.3.2.1 Rx Packet Size PCIRPSR(RW) —MBAR + 0x3880

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Packet_Size[16:2]														Packet_Size[1:0!]	
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	Packet_Size	The user writes this register with the number of bytes for Receive Controller to fetch over PCI. The two low bits are hardwired low; only 32-bit data transfers to the FIFO are allowed. Writing to this register also completes a Restart Sequence as long as Master Enable bit, PCIRER[ME], is high and Reset Controller bit, PCIRER[RC], is low.
16:31	Reserved	Unused bits. Software should write zero to these bits. No Bus Error is generated

10.3.3.2.2 Rx Start Address PCIRSAR (RW) —MBAR + 0x3884

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

Bits	Name	Description
0:31	Start_Add	The user writes this register with the desired Starting Address for the current packet. This is the address which will be first presented on the external PCI bus and then auto-incremented as necessary. This register will not increment as the PCI packet proceeds.

10.3.3.2.3 Rx Transaction Control Register PCIRTCR(RW) —MBAR + 0x3888

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				PCI_cmnd				Max_Retries							
W																
RESET	0	0	0	0	1100				0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved			FB	R	Max_Beats			Reserved			W	Reserved			DI
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	Reserved	Unused. Software should write zero to these bits.
4:7	PCI_cmd	The user writes this field with the desired PCI command to present during the address phase of each PCI transaction. The default is Memory Read Multiple. This field is not checked for consistency and if written to an illegal value, unpredictable results will occur. If not using the default value, the user should write this register only once prior to any packet Restart.
8:15	Max_Retries	The user writes this field with the maximum number of retries to permit “per packet”. The retry counter is reset when the packet completes normally or is terminated by a master abort, target abort, or an abort due to exceeding the retry limit. A slow or malfunctioning Target might issue infinite disconnects and therefore permanently tie up the PCI bus. A finite (0x01 to 0xfe) Max_Retries value will detect this condition and generate an interrupt. Setting Max_Retries to 0x00 or 0xff will not generate any interrupt.
16:18	Reserved	Unused. Software should write zero to these bits.
19	Full Burst (FB)	This is the Full Burst bit. If Full Burst is set, no check of the Receive Fifo emptiness is done and the PCI transaction is immediately started when Packet_Size register is written (and SCPCI RX gains the PCI bus). The PCI transaction will continue with multiple data beats UNTIL THE FULL PACKET IS TRANSFERRED (up to 65K bytes). The Full Burst operation avoids latency time-out and will not relinquish the bus until all Packet Bytes are received. **Note:** All Fifo checks (by scpci Rx) are disabled in this mode. It is up to the Multi-Channel DMA to keep the Rx Fifo from being overrun by the continuous incoming PCI burst data. **Note:** It is recommended to use the Full Burst mode only for transactions where more than 32 Bytes should be received. **Note:** Max_Beats must be set to 0.
20	Reserved	Unused. Software should write zero to this bit.
21:23	Max_Beats	The user writes this register with the desired number of PCI data beats to attempt on each PCI transaction. The default setting of 0 represents the maximum of eight beats per transaction. The receive controller will wait until sufficient space is in the Receive FIFO to support the indicated number of beats (Note: Each beat is four bytes). In the case that a packet is nearly complete and less than the Max_Beats number of bytes remain to complete the packet, the Receive Controller will issue single-beat transactions automatically until the packet is finished.
24:26	Reserved	Unused. Software should write zero to these bits.
27	Word Transfer (W)	The user writes this register to disable the two high byte enables of the PCI bus during initiated read transactions. The default setting is 0, enable all 4 byte enables.
28:30	Reserved	Unused. Software should write zero to these bits.
31	Disable address Incrementing (DI)	The user writes this register to disable PCI address incrementing between transactions. The default setting is 0, increment address by 4 (4 byte data bus). **Note:** This feature is recommended when reading from an external FIFO (having a fixed address).

10.3.3.2.4 Rx Enables PCIRER (RW) —MBAR + 0x388C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RC	RF	FE	CM	BE	Reserved		ME	Reserved		FEE	SE	RE	TAE	IAE	NE
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	Reset Controller (RC)	User writes this bit high to put Receive Controller in a reset state. Note that other register bits are not affected. This Reset is intended for recovery from an error condition or to reload the Start Address when Continuous mode is selected. This Reset bit does not prohibit register access but it must be negated in order to initiate a Restart sequence (i.e. writing the Packet_Size register). If it is used to reload a Start Address then the Start_Add register must be written prior to asserting this Reset bit.
1	Reset FIFO (RF)	The FIFO will be reset and flushed of any existing data when set high. The Reset Controller bit and the Reset FIFO bit operate independently, but clearly both must be low for normal operation.
2	FE	Flush enable. This is an important bit which causes a flush signal to be generated to the Receive FIFO Controller when the end of the current packet occurs. This Flush is necessary to insure that the Multi-Channel DMA will get all data left in the Receive FIFO. FE is active high.
3	Continuous mode (CM)	User writes this bit high to activate Continuous mode. In Continuous mode the Start_Add value is ignored at each packet restart and the PCI address is auto-incremented from one packet to the next. Also, the Packets_Done status byte will become active, indicating how many packets have been received since the last Reset Controller condition. If the Continuous bit is low, software is responsible for updating the Start_Add value at each packet Restart.
4	Bus error Enable (BE)	User writes this bit high to enable Bus Error indications. Section 10.3.3.2.8, *Rx Status PCIRSR (R/sw1) —MBAR + 0x389C* for Bus Error descriptions. Normally this bit will be 0 since illegal Slave bus accesses are not destructive to register contents, although it may indicate broken software. Note that this bit does not affect interrupt generation.
5:6	Reserved	Unused. Software should write zero to these bits.
7	Master Enable (ME)	This is the Receive Controller master enable signal. User must write it high to enable operation. It can be toggled low to permit out-of-order register updates prior to generating a Restart sequence (in which case transmission will begin when Master Enable is written back high), but it should not be used as such in Continuous mode because it has the side effect of resetting the Packets_Done status counter.
8:9	Reserved	Unused. Software should write zero to these bits.
10	FIFO Error Enable (FEE)	User writes this bit high to enable CPU Interrupt generation in the case of FIFO error termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.

Bits	Name	Description
11	System error Enable (SE)	User writes this bit high to enable CPU Interrupt generation in the case of system error termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case someone should be polling the status bits to prevent a possible lock-up condition.
12	Retry abort Enable (RE)	User writes this bit high to enable CPU Interrupt generation in the case of retry abort termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case, software should poll the status bits to prevent a possible lock-up condition.
13	Target Abort Enable (TAE)	User writes this bit high to enable CPU Interrupt generation in the case of target abort termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
14	Initiator Abort error Enable (IAE)	User writes this bit high to enable CPU Interrupt generation in the case of initiator abort error termination of a packet transmission. It may be desirable to mask CPU interrupts in the case that Multi-Channel DMA is controlling operation, but in such a case software should poll the status bits to prevent a possible lock-up condition.
15	Normal termination Enable (NE)	User writes this bit high to enable CPU Interrupt generation at the conclusion of a normally terminated packet transmission. This may or may not be desirable depending on the nature of program control by Multi-Channel DMA or the processor core.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.2.5 Rx Next Address PCIRNAR(R) —MBAR + 0x3890

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

Bits	Name	Description
0:31	Next_Address	This status register contains the next (unread) PCI address and is updated at the successful completion of each PCI data beat. It represents a Byte address and is updated with a user-written Start_Add value when Start_Add is reloaded. This register is intended to be accurate even if an abnormal PCI bus termination occurs.

10.3.3.2.6 Rx Last Word PCIRLWR(R) —MBAR + 0x3894

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

Bits	Name	Description
0:31	Last_Word	This status register indicates the last 32-bit data fetched from the FIFO and is designed for the case in which an abnormal PCI termination has corrupted the integrity of the FIFO data (for that word).

10.3.3.2.7 RxDone Counts PCIRDCR(R) —MBAR + 0x3898

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Bytes_Done															
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 lsb
R	Packets_Done															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	Bytes_Done	This status register indicates the number of Bytes received since the start of a packet. It is updated at the end of each successful PCI data beat. For normally terminated packets, the Bytes_Done value and the Packet_Size values are equal. If continuous mode is active, the Bytes_Done value reads 0 at the end of a successful packet and the Packets_Done field is incremented.
16:31	Packets_Done	This status register indicates the number of packets received. It is active only if continuous mode is in effect. If the following occurs, the counter is reset: • Reset Controller bit, PCIRER[RC], is asserted (normal way to restart continuous mode) • Master Enable bit, PCIRER[ME], is negated In this way, master enable can be used to reset Packets_Done status without disturbing continuous mode addressing. At any point in time the total number of Bytes received can be calculated as: (Packets_Done x Packet_Size) + Bytes_Done This assumes Packet_Size is the same for all restart sequences.

10.3.3.2.8 Rx Status PCIRSR (R/sw1) —MBAR + 0x389C

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							NT	BE3	BE2	BE1	FE	SE	RE	TA	IA
W								rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc	rwc
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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	Reserved	Unused. Software should write zero to these bits.
7	Normal Termination (NT)	This flag is set when any packet terminates normally. It is not set in the case of an abnormally terminated packet. It does not require clearing but will not clear until it is written to a one (in which case it will now read back as zero, i.e. negated). >ALL THE FOLLOWING FLAG BITS OPERATE SIMILARLY<
8	Bus Error type 3 (BE3)	This flag is set whenever a Slave bus transaction attempts to write to a Read-Only register. This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out. No corruption of the register bits occur for this (or any other) Bus Error case.
9	Bus Error type 2 (BE2)	This flag is set whenever a Slave bus transaction attempts to write to a Reserved register (an entire 32-bit register, not just a Reserved bit or byte). This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out.
10	Bus Error type 1 (BE1)	This flag is set whenever a Slave bus transaction attempts to read a Reserved register (an entire 32-bit register, not just a Reserved bit or byte). This flag bit is set regardless of the Bus error Enable bit (BE). If software is polling this Byte and wishes to disregard this error it must mask this bit out.
11	FIFO Error (FE)	This flag is set whenever the Receive FIFO asserts its FIFO Error output. A CPU interrupt will be generated if the FIFO Error Enable (FEE) bit is set. The source of the error must be determined by reading the FIFO Error status register. Also, the error condition must be cleared at the FIFO prior to clearing this Sticky bit or this flag will continue to assert.
12	System Error (SE)	This flag is set in response to the Transmit Controller entering an illegal state. A CPU interrupt will be generated if the System error Enable (SE) bit is set. In normal operation this should never occur. The only recovery is to assert the Reset Controller bit, PCIRER[RC], and clear this flag.
13	Retry Error (RE)	This flag is set if Max_Retries is set to a finite value (0x01 to 0xff) and the PCI transaction has performed retries in excess of the setting. A CPU interrupt will be generated if the Retry error Enable (RE) bit is set. The retry counter is reset at the beginning of each transaction (i.e. it is not cumulative throughout a packet) and would generally indicate a broken or improperly accessed Target.
14	Target Abort (TA)	This flag bit is set if the PCI controller has issued a Target Abort (which means the addressed PCI Target has signalled an Abort). A CPU interrupt will be generated if the Target Abort Enable (TAE) bit is set. It is up to application software to query the Target's status register and determine the source of the error. The coherency of the Receive FIFO data and the Receive Controller's status registers (Next_Address, Bytes_Done, etc.) should remain valid.
15	Initiator Abort (IA)	This flag bit is set if the PCI controller issues an Initiator Abort flag. This indicates that no Target responded but further status information can be read from the PCI Configuration interface. A CPU interrupt will be generated if the Initiator Abort error Enable (IAE) bit is set. The coherency of the Receive FIFO data and the Receive Controller's status registers (Next_Address, Bytes_Done, etc.) should remain valid.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.2.9 Rx FIFO Data Register PCIRFDR(RW) —MBAR + 0x38C0

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_Data_Word															
W																
RESET	uninitailized random 16 bit value															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	FIFO_Data_Word															
W																
RESET	uninitalized random 16 bit value															

Bits	Name	Description
0:31	FIFO_Data_Word	FIFO data port—Reading from this location “pops” data from the FIFO; writing “pushes” data into the FIFO. During normal operation the Multi-Channel DMA controller pops data here. The receive controller pushes data. Therefore, user programs should not write here. At power on reset an uninitialized random value is read at this register. A FIFO reset must be always performed before first accessing the FIFO. **Note:** Only full 32-bit accesses are allowed. If all Byte enables are not asserted when accessing this location, FIFO data will be corrupted.

10.3.3.2.10 Rx FIFO Status Register PCIRFSR(R/sw1) —MBAR + 0x38C4

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									RXW	UF	OF	FR	Full	Alarm	Empty
W										rwc	rwc	rwc				
RESET	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 lsb
R	Reserved															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:8	Reserved	Unused byte. Software should write zero to these bits.
9	Receive Wait Condition (RXW)	This flag bit indicates that the ipf_rcv bus is incurring wait states because there is not enough room in the FIFO to accept the data without causing overflow. This bit will cause the error outputs (fifoError, ipf_rcv_error, ipf_xmit_error) to assert unless the RXW_MASK bit in the FIFO Control register is set. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.
10	UnderFlow (UF)	This flag bit indicates that the read pointer has surpassed the write pointer. In other words the FIFO has been read beyond Empty. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.
11	OverFlow (OF)	This flag bit indicates that the write pointer has surpassed the read pointer. In other words the FIFO has been written beyond Full. Resetting the FIFO will clear this condition and the flag bit is cleared by writing a one to its bit position.

Bits	Name	Description
12	Frame Ready (FR)	The FIFO has a complete Frame of data ready for transmission. This module does not provide support for Data Framing applications, so this bit should be ignored.
13	Full	The FIFO is Full. This is not a sticky bit or error condition. The Full indication tracks with the state of the FIFO.
14	Alarm	When the FIFO pointer is at or above the Alarm “watermark”, as written by the user according to the Alarm and Control registers settings, the Alarm bit is asserted, thus automatically signalling to the DMA engine that the FIFO needs to be ‘emptied’. By writing a ‘1’ to this location software can enforce re-evaluation of the alarm condition.
15	Empty	The FIFO is empty. This is not a sticky bit or error condition.
16:31	Reserved	Unused. Software should write zero to these bits.

10.3.3.2.11 Rx FIFO Control Register PCIRFCR(RW) —MBAR + 0x38C8

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					GR			IP_MASK	FAE_MASK	RXW_MASK	UF_MASK	OF_MASK	Reserved		
W																
RESET	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0

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

Bits	Name	Description
0:4	Reserved	Unused. Software shall write zero to these bits. (R/W)
5:7	Granularity (GR)	Granularity bits control high “watermark” point at which FIFO negates Alarm condition (i.e., request for data). It represents the number of free Bytes times 4. **Note:** A granularity setting of zero should be avoided because it means the Alarm bit (and the Requestor signal) will not negate until the FIFO is completely full. The Multi-Channel DMA module may perform up to 2 additional data writes after the negation of a Requestor due to its internal pipelining
8	IP_MASK	Illegal Pointer Mask When this bit is set, the FIFO controller masks the Status register’s IP bit from generating an error.
9	FAE_MASK	When this bit is set, the FIFO controller masks the Status Register’s FAE bit from generating an error.
10	RXW_MASK	When this bit is set, the FIFO controller masks the Status Register’s RXW bit from generating an error. (To help with backward compatibility, this bit is asserted at reset.)
11	UF_MASK	When this bit is set, the FIFO controller masks the Status Register’s UF bit from generating an error.
12	OF_MASK	When this bit is set, the FIFO controller masks the Status Register’s OF bit from generating an error.

Bits	Name	Description
13:15	Reserved	Unused. Software should write zero to these bits.
16:31	Reserved	Unused. Software shall write zero to these bits. (R/W)

10.3.3.2.12 Rx FIFO Alarm Register PCIRFAR(RW) —MBAR + 0x38CC

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				Alarm				Alarm							
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	Reserved	Unused. Software should write zero to these bits.
20:31	Alarm [11:0]	User writes these bits to set the low level watermark, which is the point at which the FIFO asserts its request for data emptying to the Multi-Channel DMA controller. This value is in bytes. For example, with Alarm = 32, the alarm condition will occur when the FIFO has 32 or less free bytes in it. The alarm, once asserted, will not negate until the high level mark is reached, as specified by the Granularity bits in the Rx FIFO Control Register. **Note:** The PCI RX FIFO is 512 bytes deep.

10.3.3.2.13 Rx FIFO Read Pointer Register PCIRFRPR(RW) —MBAR + 0x38D0

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				ReadPtr											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	Reserved	Unused. Software should write zero to these bits.
20:31	ReadPtr	This value is maintained by the FIFO hardware and is not normally written. It can be adjusted in special cases but will disrupt the integrity of the data flow. This value represents the Read address being presented to the FIFO RAM.

10.3.3.2.14 Rx FIFO Write Pointer Register PCIRFWPR (RW) —MBAR + 0x38D4

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				WritePtr											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
12:19	Reserved	Unused. Software should write zero to these bits.
20:31	WritePtr	This value is maintained by the FIFO hardware and is not normally written. It can be adjusted in special cases but will of course disrupt the integrity of the data flow. This value represents the Write address being presented to the FIFO RAM.

This marks the end of the PCI Multi-Channel DMA Receive Interface description.

10.4 Functional Description

The MPC5200 PCI module provides both master and target PCI bus interfaces as shown in Figure 10-1. The internal *PCI master*, or *initiator*, interface is accessible by any XL bus master such as the processor core and also provides a DMA interface (for BEstComm) through the Communication Sub-System, which can be accessed by the Multi-Channel DMA engine. The internal *PCI target* interface provides external PCI masters access into two memory windows of MPC5200 address space. PCI arbitration is handled external to this module, by the MPC5200 internal PCI arbiter.

NOTE

Only the internal PCI arbiter of the MPC5200 can be used as PCI arbiter for the PCI bus. An external PCI arbiter cannot be used.

The registers, described in Section 10.3, *Registers*, control and provide information about these multiple interfaces. An additional Configuration interface allows internal access through the Slave bus(also referred to as IP bus) to the PCI Type 0 Configuration registers, which are accessible to both MPC5200 and external masters through the PCI bus.

The following sections describe the operation of the PCI module.

10.4.1 PCI Bus Protocol

This section will provide a simple overview of the PCI bus protocol, including some details of MPC5200 implementation. For details regarding PCI bus operation, refer to the *PCI Local Bus Specification, Revision 2.2*.

10.4.1.1 PCI Bus Background

The PCI interface is synchronous and is best used for bursting data in large chunks. Its maximum theoretical bandwidth approaches 266 Megabytes per second for the 32-bit implementation running at 66MHz. A system will contain one device that is responsible for configuring all other devices on the bus upon reset. Each device has 256 bytes of configuration space that define individual requirements to the system controller. These registers are read and written through a “configuration access” command. A PCI transfer is started by the **master** and is directed toward a specific **target**. A provision is made for broadcasting to several targets through the “special command.” Data is transferred through the use of memory and IO read and write commands.

Table 10-4. PCI Command encoding

C/BE[3:0]	Command Type
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved
0110	Memory Read
0111	Memory Write
1000	Reserved
1001	Reserved
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

10.4.1.2 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst. A burst is composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, and $\overline{\text{TRDY}}$. An initiator asserts $\overline{\text{FRAME}}$ to indicate the beginning of a PCI bus transaction and negates $\overline{\text{FRAME}}$ to indicate the end of a PCI bus transaction. An initiator negates $\overline{\text{IRDY}}$ to force wait cycles. A target negates $\overline{\text{TRDY}}$ to force wait cycles.

The PCI bus is considered idle when both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated. The first clock cycle in which $\overline{\text{FRAME}}$ is asserted indicates the beginning of the address phase. The address and bus command code are transferred in that first cycle. The next cycle begins the first of one or more data phases. Data is transferred between initiator and target in each cycle that both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted. Wait cycles may be inserted in a data phase by the initiator (by negating $\overline{\text{IRDY}}$) or by the target (by negating $\overline{\text{TRDY}}$).

Once an initiator has asserted $\overline{\text{IRDY}}$, it cannot change $\overline{\text{IRDY}}$ or $\overline{\text{FRAME}}$ until the current data phase completes regardless of the state of $\overline{\text{TRDY}}$. Once a target has asserted $\overline{\text{TRDY}}$ or $\overline{\text{STOP}}$, it cannot change $\overline{\text{DEVSEL}}$,$\overline{\text{TRDY}}$, or $\overline{\text{STOP}}$ until the current data phase completes. In simpler terms, once an initiator or target has committed to the data transfer, it cannot back out.

When the initiator intends to complete only one more data transfer (which could be immediately after the address phase), $\overline{\text{FRAME}}$ is negated and $\overline{\text{IRDY}}$ is asserted (or kept asserted) indicating the initiator is ready. After the target indicates the final data transfer (by asserting $\overline{\text{TRDY}}$), the PCI bus may return to the idle state (both $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are negated).

NOTE

No Fast Back-to-Back transactions are supported by the MPC5200.

10.4.1.3 PCI Transactions

The figures in this section show the basic “memory read” and “memory write” command transactions.

Figure 10-2 shows a PCI burst read transaction (2-beat). The signal $\overline{\text{FRAME}}$ is driven low to initiate the transfer. Cycle 1 is the address phase with valid address information driven on the AD bus and a PCI command driven on the $\overline{\text{C/BE}}$ bus. In cycle 2, the AD bus is in a turnaround cycle because of the read on a muxed bus. The byte enables, which are active low, are driven onto the $\overline{\text{C/BE}}$ bus in this clock. Any combination of byte enables can be asserted (none may be asserted). A target will respond to an address phase by driving the $\overline{\text{DEVSEL}}$ signal. The specification allows for four types of decode operations. The target can drive $\overline{\text{DEVSEL}}$ in 1, 2 or 3 clocks depending on whether the target is a fast, medium or slow decode device. A single device is allowed to drive $\overline{\text{DEVSEL}}$ should another agent fail to respond by the fourth clock. This is called “subtractive decoding” in PCI terminology.

A valid transfer occurs when both IRDY and TRDY are asserted. If either are negated during a data phase, it is considered a wait state. The target asserts a wait state in cycles 3 and 5 of Figure 10-2. A master indicates that the final data phase is to occur by negating FRAME. The final data phase occurs in cycle 6. Another agent cannot start an access until cycle 8.

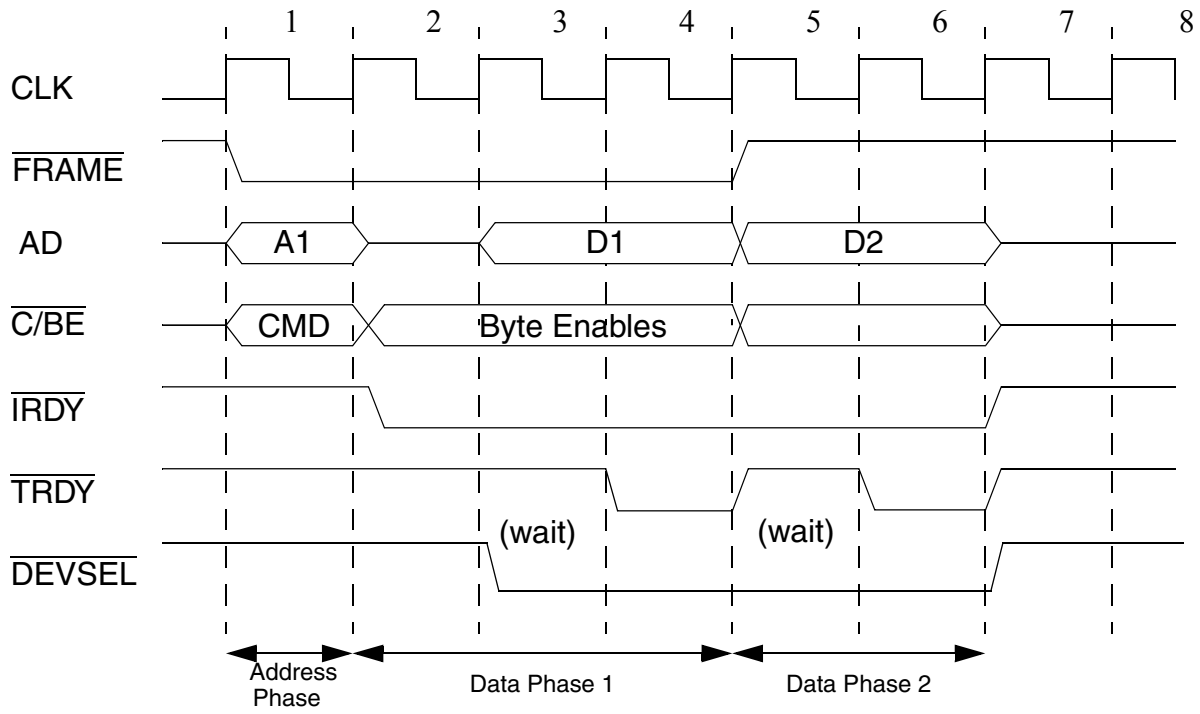


Figure 10-2. PCI Read Terminated by Master

Figure 10-3 shows a write cycle which is terminated by the target. In this diagram the target responds as a slow device, driving DEVSEL in cycle 4. The first data is transferred in cycle 4. The master inserts a wait state at cycle 5. The target indicates that it can accept only one more transfer by asserting both TRDY and STOP at the same time in cycle 5. The signal STOP must remain asserted until FRAME negates. The final data phase does not have to transfer data. If STOP and IRDY are both asserted while TRDY is negated, it is considered a target disconnect without a transfer. See the PCI specification for more details.

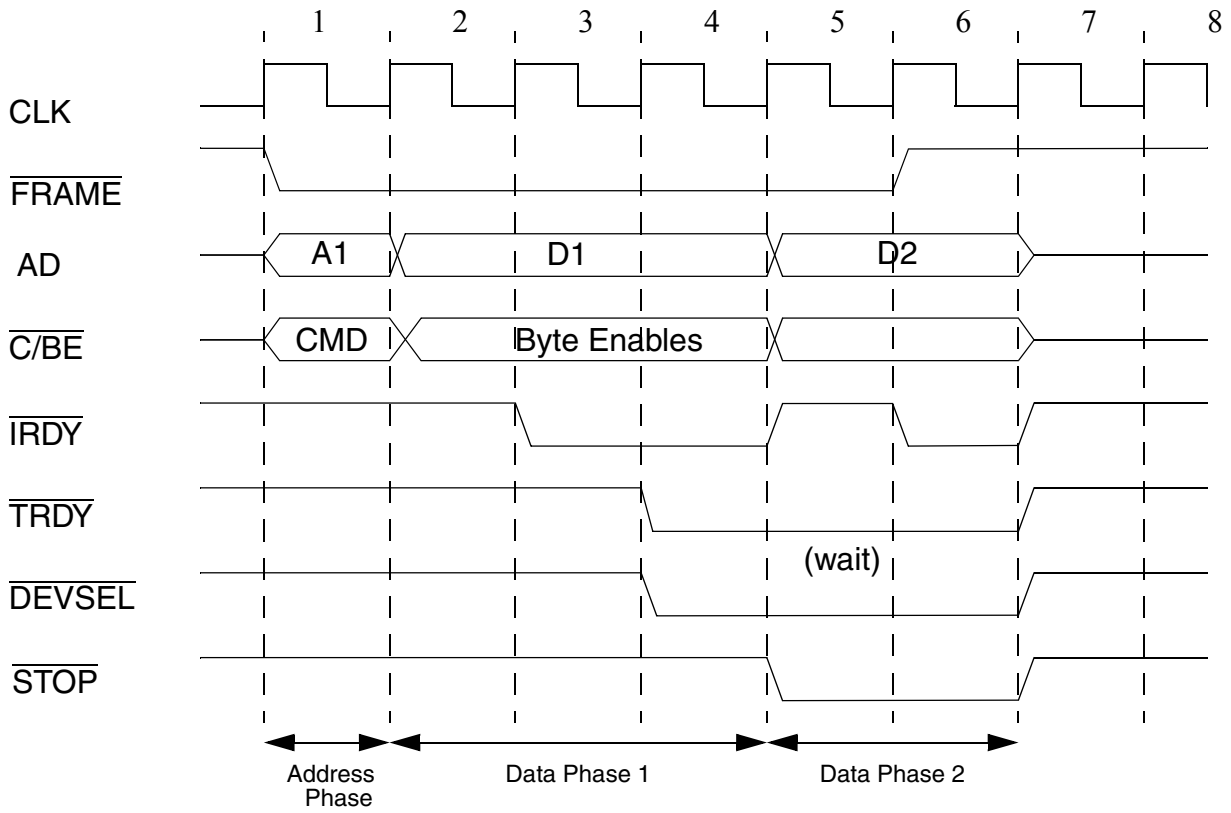


Figure 10-3. PCI Write Terminated by Target

10.4.1.4 PCI Bus Commands

PCI supports a number of different commands. These commands are presented by the initiator on the $\overline{\text{C/BE[3:0]}}$ lines during the address phase of a PCI transaction.

Table 10-5. PCI Bus Commands

C/BE[3:0]	PCI Bus Command	MPC5200 supports as Initiator	MPC5200 supports as Target	Definition
0000	Interrupt Acknowledge	Yes	No	The interrupt acknowledge command is a read (implicitly addressing an external interrupt controller). Only one device on the PCI bus should respond to the interrupt acknowledge command.
0001	Special Cycle	Yes	No	The Special Cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus.
0010	I/O-read	Yes	No	The I/O-read command accesses agents mapped into the PCI I/O space.
0011	I/O-write	Yes	No	The I/O-write command accesses agents mapped into the PCI I/O space.
0100	Reserved	No	No	--
0101	Reserved	No	No	--
0110	Memory-read	Yes	Yes	The memory read command accesses agents mapped into PCI memory space.

Table 10-5. PCI Bus Commands (continued)

C/BE[3:0]	PCI Bus Command	MPC5200 supports as Initiator	MPC5200 supports as Target	Definition
0111	Memory-write	Yes	Yes	The memory write command accesses agents mapped into PCI memory space.
1000	Reserved	No	No	--
1001	Reserved	No	No	--
1010	Configuration read	Yes	Yes	The configuration read command accesses the 256 byte configuration space of a PCI agent.
1011	Configuration write	Yes	Yes	The configuration read command accesses the 256 byte configuration space of a PCI agent.
1100	Memory read multiple	Yes	Yes	For MPC5200, the memory read multiple command functions the same as the memory read command. Cache line wrap is implemented when XLB is the transaction initiator and it also wraps.
1101	Dual address cycle	No	No	The dual address cycle command is used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices. MPC5200 device does not respond to this command.
1110	Memory read line	Yes	Yes	The memory read line command indicates that an initiator is requesting the transfer of an entire cache line.For MPC5200, the memory read line functions the same as the memory read command. Cache line wrap is not implemented.
1111	Memory write and invalidate	Yes (DMA access only)	Yes	The memory write and invalidate command indicates that an initiator is transferring an entire cache line, and, if this data is in any cacheable memory, that cache line needs to be invalidated. The memory write and invalidate functions the same as the memory write command. Cache line wrap is implemented. Software must make sure that the cache line register and max_beats register are set to the same value and the packet size must be a multiple of the cache line size. This instruction is supported only by the TX SCPCI initiator interface and when the MPC5200 acts as a target.

Though MPC5200 supports many PCI commands as an initiator, the Communication Sub-System Initiator interface is intended to use PCI Memory Read, and Memory Write commands.

10.4.1.5 Addressing

PCI defines three physical address spaces: PCI memory space, PCI I/O space, and PCI configuration space. Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding: positive decoding and subtractive decoding. The address space which is accessed depends primarily on the type of PCI command that is used.

10.4.1.5.1 Memory space addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address: linear incrementing(AD[1:0] = 0b00) and cache wrap mode (AD[1:0] = 0b10). The other two AD[1:0] encodings (0b01 and 0b11) are reserved.

For linear incrementing mode, the memory address is encoded/decoded using AD[31:2]. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4 byte data width per data phase is implied). Note, the two low-order bits of the address are still included in all the parity calculations.

MPC5200 supports both linear incrementing and cache wrap mode as an initiator. For memory transactions, when an XLB burst transaction is wrapped, the cache wrap mode is automatically generated. For zero-word-aligned bursts and single-beat transactions, MPC5200 drives AD[1:0] to 0b00. As a target, the MPC5200 treats cache wrap mode as a reserved memory mode. MPC5200 will return the first beat of data and then signal a disconnect without data on the second data phase.

10.4.1.5.2 I/O space addressing

For PCI I/O accesses, all 32 address signals are used to provide an address with granularity of a single byte. Once a target has claimed an I/O access, it must determine if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range of the target, the entire access cannot complete. In this case, the target does not transfer any data, and terminates the transaction with a target-abort.

Table 10-6. PCI I/O space byte decoding

Access Size	AD[1:0]	C/BE[3:0]	Data
8-bit	00	xxx0	AD[7:0]
	01	xx01	AD[15:8]
	10	x011	AD[23:16]
	11	0111	AD[31:24]
16-bit	00	xxx0	AD[15:0]
	01	xx01	AD[23:8]
	10	x011	AD[31:16]
24-bit	00	xxx0	AD[23:0]
	01	xx01	AD[31:8]
32-bit	00	xxx0	AD[31:0]

10.4.1.5.3 Configuration space addressing and transactions

PCI supports two types of configuration accesses. Their primary difference is the format of the address on the AD[31:0] signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase: type 0 (AD[1:0] = 0b00) or type 1 (AD[1:0] = 0b01). Both address formats identify a specific device and a specific configuration register for that device.

Type 0 configuration accesses are used to select a device on the local PCI bus. They do not propagate beyond the local PCI bus and are either claimed by a local device or terminated with a master-abort. Type 1 configuration accesses are used to target a device on a subordinate bus through a PCI-to-PCI bridge. Type 1 accesses are ignored by all targets except PCI-to-PCI bridges that pass the configuration request to another PCI bus.

When the controller initiates a configuration access on the PCI bus, it places the configuration address information on the AD bus and the configuration command on the $\overline{\text{C/BE}}$[3:0] bus. A Type 0 configuration transaction is indicated by setting AD[1:0] to 0b00 during the address phase. The bit pattern tells the community of devices on the PCI bus that the bridge that “owns” that bus has already performed the bus number comparison and verified that the request targets a device on its bus. Figure 10-4 shows the contents of the AD bus during the address phase of the Type 0 configuration access.

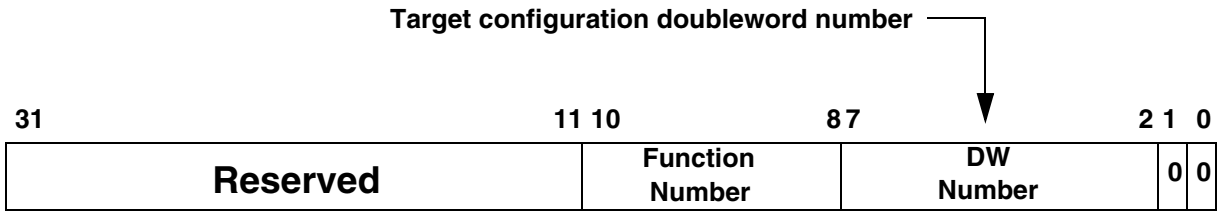


Figure 10-4. Contents of the AD Bus During Address Phase of a Type 0 Configuration Transaction

Address bits [10:8] identify the target function and bits AD[7:2] select one of the 64 configuration dwords within the target function's configuration space. For Type 0 configuration transactions, the target device's IDSEL pin must be asserted. The upper 21 address lines are commonly used as IDSELs since they are not used during the address phase of a type 0 configuration transaction.

If the target bus is a bus that is subordinate to the local PCI bus (bus 0), the configuration transaction is still initiated on bus 0, but indicates that none of the devices on this bus are the target of the transaction. Rather, only PCI-to-PCI bridges residing on the bus should pay attention to the transaction because it targets a device on a bus further out in the hierarchy beyond a PCI-to-PCI bridge that is attached to the local PCI bus (bus 0). This is accomplished by initiating a Type 1 configuration transaction (setting AD[1:0] to 01b during the address phase). This pattern instructs all functions other than PCI-to-PCI bridges that the transaction is not for any of them. Figure 10-5 illustrates the contents of the AD bus during the address phase of the Type 1 configuration access.

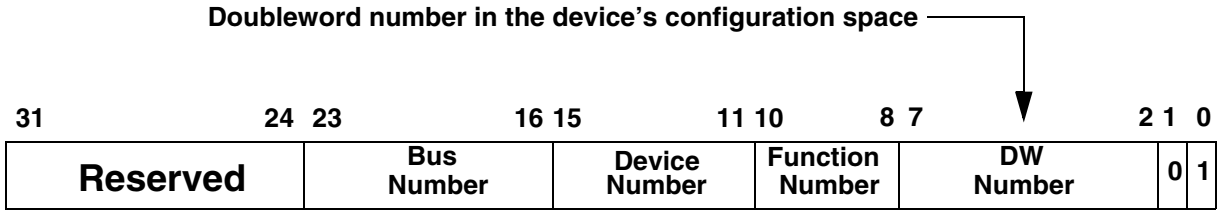


Figure 10-5. Contents of the AD Bus During Address Phase of a Type 1 Configuration Transaction

During the address phase of a Type 1 configuration access, the information on the AD bus if formatted as follows:

- AD[1:0] contain a 01b, identifying this as a Type 1 configuration access.
- AD[7:2] identifies one of 64 configuration dwords within the target devices's configuration space.
- AD[10:8] identifies one of the eight functions within the target physical device.
- AD[15:11] identifies one of 32 physical devices. This field is used by the bridge to select which device's IDSEL line to assert.
- AD[23:16] identifies one of 256 PCI buses in the system.
- AD[31:24] are reserved and are cleared to zero.

During a Type 1 configuration access, PCI devices ignore the state of their IDSEL inputs. When any PCI-to-PCI bridge latches a Type 1 configuration access (command = configuration read or write and AD[1:0] = 01b) on its primary side, it must determine whether the bus number field on the AD bus matches the number of its secondary bus or if it's within the range of its subordinate buses. If the bus number matches, it should claim and pass the configuration access onto its secondary bus as a Type 0 configuration access, decoding the device number to select one of the IDSEL lines. If the bus number isn't equal to its secondary bus, but is within the range of buses that are subordinate to the bridge, the bridge claims and passes that access through as a Type 1 access.

10.4.1.5.4 Address decoding

For positive address decoding, an address hits when the address on the address bus matches an assigned address range. Multiple devices on the same PCI bus may use positive address decoding, though there can not be any overlap in the assigned address ranges.

For subtractive address decoding, an address hits when the address on the address bus does not match any address range for any of the PCI devices on the bus. Only one device on a PCI bus may use subtractive address decoding, and its use is optional.

10.4.2 Initiator Arbitration

There are three possible internal initiator sources - CommBus Transmit, CommBus Receive, or the XL bus (from Internal System Arbiter). Custom interface logic arbitrates and provides mux select control for these sources to the PCI controller. Figure 10-6 illustrates the arbitration block connection.

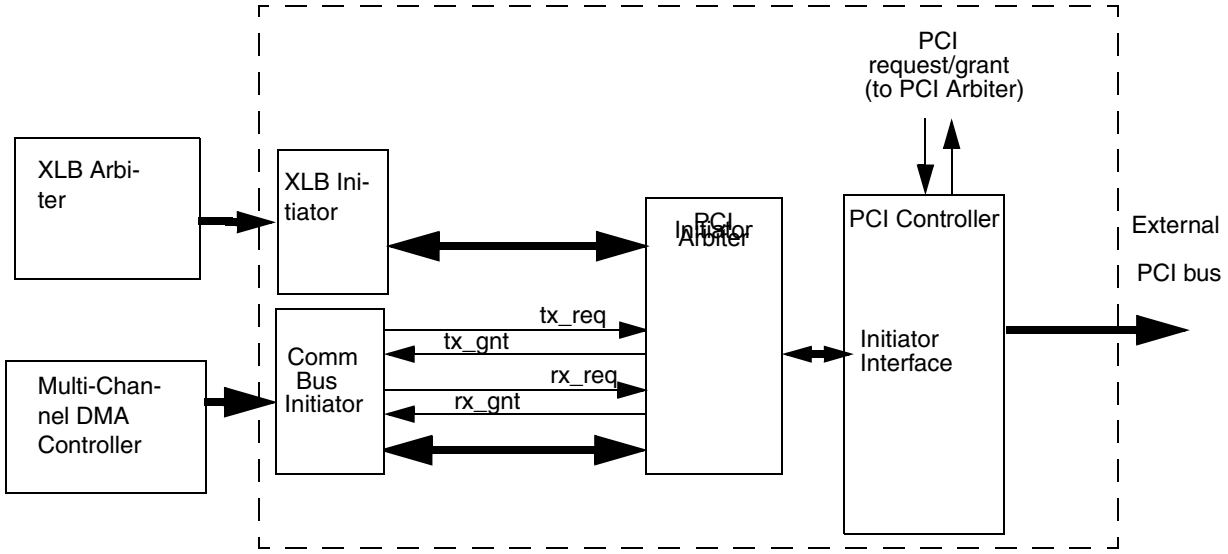


Figure 10-6. Initiator Arbitration Block Diagram

10.4.2.1 Priority Scheme

The PCI Initiator arbiter uses the following fixed priority scheme.

1. XL bus Initiator
2. CommBus Transmit (Tx)
3. CommBus Receive (Rx) (lowest)

10.4.3 Configuration Interface

The PCI bus protocol requires the implementation of a standardized set of registers for most devices on the PCI bus. MPC5200 implements a Type 0 Configuration register set or header. They are described in Section 10.3.1, *PCI Controller Type 0 Configuration Space*. These registers are primarily intended to be read or written by the PCI configuring master at initialization time through the PCI bus. MPC5200 provides internal access to these registers through a Slave bus interface. As with most MPC5200 registers, they are accessible by software in the address space at offsets of MBAR. Internal accesses to the Type 0 Configuration header do not require PCI arbitration when they are accessed as offsets of MBAR and are allowed to execute regardless of whether any write data is posted in the PCI Controller.

If MPC5200 is the configuring master, the Slave bus interface should be used to configure the PCI Controller. An external master would configure the PCI controller through the external PCI bus.

More information on the standard PCI Configuration register can be found in the PCI 2.2 specification.

10.4.4 XL bus Initiator Interface

The XL bus Initiator Interface provides access to the PCI bus for XL bus masters, primarily the processor core. This interface is accessed through three windows in MPC5200 address space set up by base address and base address mask registers (Section 10.3.2.5, *Initiator Window 0 Base/Translation Address Register PCIIW0BTAR(RW)—MBAR + 0x0D70*). The base address registers must be enabled by setting their respective Enable bits in the Section 10.3.2.8, *Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80*. Accesses to this area are translated into PCI transactions on the PCI bus. See Section 10.6.2, *Address Maps* for examples on setting up address windows.

The particular type of PCI transaction generated is determined by the PCI configuration bits associated with the address window (PCIIWCR). For example, the user might set one window to do PCI memory read multiple accesses, one window for PCI I/O accesses, and the other window to do non-prefetchable (memory-mapped I/O) PCI memory accesses. Table 10-15 for command translation.

In addition to the configurable address window mapping logic, the register interface provides a Configuration Address Register, which provides the ability to generate Configuration, Interrupt Acknowledge and Special Cycles. External PCI devices should be configured through this interface. Section 10.4.4.2, *Configuration Mechanism* for configuration, interrupt acknowledge, and special cycle command support.

The PCI XLB Initiator interface supports all XLB transactions, including single-beat transfers and bursts (32 bytes). Single-beat 64-bit data transactions are automatically translated into 2-beats burst transfers on the PCI bus. Standard XL bus burst transactions are supported as well, however, buffering is implemented to boost performance during writes and avoid deadlock scenario for all reads and memory writes. If the target for an XL bus read from PCI disconnects part way through the burst, MPC5200 may have to handle a local memory access from an alternate PCI master before the disconnected transfer can continue.

XLB initiator read requests are decoded into four types: PCI Memory, I/O, Configuration, and Interrupt Acknowledge. The PCI Controller must first gain access to the PCI bus before acknowledging the XLB read request. The specific timing of the address acknowledge is dependent upon the type of transfer.

When the XL bus requests burst data from PCI space, the data received from PCI is stored in a buffer until all requested data has been latched. The PCI Controller does not terminate the address tenure of the XLB transaction until all requested data is latched. This is because PCI targets are allowed to disconnect in the middle of a transfer, and the XL bus requires burst transfers to be atomic. If the PCI target disconnects in the middle of the data transfer and an alternate PCI master acquires the bus and initiates a local memory access, the Controller retries the internal read transaction on the XL bus. The PCI Controller continues to request mastership of the PCI bus until the original request is completed.

For example, if the XL bus initiates a burst read, and the PCI target disconnects after transferring the first half of the burst, MPC5200 re-arbitrates for the PCI bus, and when granted, initiates a new transaction with the address of the third beat of the burst (4-beat XLB bus bursts). If an alternate PCI master requests data from local memory while the PCI Controller is waiting for the PCI bus grant, the PCI controller retries the XLB bus transaction to allow the PCI-initiated transaction to complete and the read buffer will be emptied.

PCI critical-word-first (CWF) burst operation (i.e. cache line wrap burst) is supported and the 2-bit cache line wrap address mode is driven on the address bus when the XLB bus starts the burst at a non-zero-word-first address. Note that this option is only provided as a means to support memory targets that support cache-line wrap.

NOTE

A processor is not permitted to cache from any external memory targets residing on the PCI bus. This was allowed previously in the PCI spec. 2.1. The PCI spec. 2.2. took this requirements away.

XL bus writes are decoded into PCI memory, PCI I/O, PCI configuration, or special cycles. If the transaction decodes into an I/O, configuration, or special cycle, the write is connected. The PCI controller gains access to the PCI bus and successfully transfers the data before it asserts address acknowledge to the XL bus. If the address maps to PCI memory space, the XLB address tenure is immediately acknowledged and write data is posted.

A 32-byte buffer is used to post memory writes from XLB to PCI. Buffering minimizes the effect of the slower PCI bus on the higher-speed XL bus. It may contain single-beat XLB write transactions or a single burst. After the XL bus write data is latched internally, the bus is available for subsequent transactions without having to wait for the write to the PCI target to complete. If a subsequent XLB write request to the PCI bus comes in, the data transfer is delayed until all previous writes to the PCI bus are completed. Only when the write buffer is empty can burst data from the XL bus be posted.

10.4.4.1 Endian Translation

The PCI bus is inherently little endian in its byte ordering. The internal XLB bus, however, is big endian. XLB bus transactions are limited to 1, 2, 3, 4, 5, 6, 7, 8, or 32 byte (burst) transactions within the data bus byte lanes on any 32-bit address boundary for burst transfers. Table 10-7 shows the byte lane mapping between the two buses.

Table 10-7. XLB bus to PCI Byte Lanes for Memory[a] Transactions

XL bus										PCI Bus					
A [29:31]	TSIZ [0:2]	Data Bus Byte Lanes								AD [2:0]	BE [3:0]	31:24	23:16	15:8	7:0
		0	1	2	3	4	5	6	7						
000	001	OP7	--	--	--	--	--	--	--	000	1110	--	--	--	OP7
001	001	--	OP7	--	--	--	--	--	--	000	1101	--	--	OP7	--
010	001	--	--	OP7	--	--	--	--	--	000	1011	--	OP7	--	--
011	001	--	--	--	OP7	--	--	--	--	000	0111	OP7	--	--	--
100	001	--	--	--	--	OP7	--	--	--	100	1110	--	--	--	OP7

Table 10-7. XLB bus to PCI Byte Lanes for Memory[a] Transactions (continued)

XL bus										PCI Bus					
A [29:31]	TSIZ [0:2]	Data Bus Byte Lanes								AD [2:0]	BE [3:0]	31:24	23:16	15:8	7:0
		0	1	2	3	4	5	6	7						
101	001	--	--	--	--	--	OP7	--	--	100	1101	--	--	OP7	--
110	001	--	--	--	--	--	--	OP7	--	100	1011	--	OP7	--	--
111	001	--	--	--	--	--	--	--	OP7	100	0111	OP7	--	--	--
000	010	OP6	OP7	--	--	--	--	--	--	000	1100	--	--	OP7	OP6
001	010	--	OP6	OP7	--	--	--	--	--	000	1001	--	OP7	OP6	--
010	010	--	--	OP6	OP7	--	--	--	--	000	0011	OP7	OP6	--	--
011	010	--	--	--	OP6	OP7	--	--	--	000	0111	OP6	--	--	--
										100	1110	--	--	--	OP7
100	010	--	--	--	--	OP6	OP7	--	--	100	1100	--	--	OP7	OP6
101	010	--	--	--	--	--	OP6	OP7	--	100	1001	--	OP7	OP6	--
110	010	--	--	--	--	--	--	OP6	OP7	100	0011	OP7	OP6	--	--
000	011	OP5	OP6	OP7	--	--	--	--	--	000	1000	--	OP7	OP6	OP5
001	011	--	OP5	OP6	OP7	--	--	--	--	000	0001	OP7	OP6	OP5	--
010	011	--	--	OP5	OP6	OP7	--	--	--	000	0011	OP6	OP5	--	--
										100	1110	--	--	--	OP7
011	011	--	--	--	OP5	OP6	OP7	--	--	000	0111	OP5	--	--	--
										100	1100	--	--	OP7	OP6
100	011	--	--	--	--	OP5	OP6	OP7	--	100	1000	--	OP7	OP6	OP5
101	011	--	--	--	--	--	OP5	OP6	OP7	00	0001	OP7	OP6	OP5	--
000	100	OP4	OP5	OP6	OP7	--	--	--	--	00	0000	OP7	OP6	OP5	OP4
001	100	--	OP4	OP5	OP6	OP7	--	--	--	000	0001	OP6	OP5	OP4	--
										100	1110	--	--	--	OP7
010	100	--	--	OP4	OP5	OP6	OP7	--	--	000	0011	OP5	OP4	--	--
										100	1100	--	--	OP7	OP6
011	100	--	--	--	OP4	OP5	OP6	OP7	--	000	0111	OP4	--	--	--
										100	1000	--	OP7	OP6	OP5
100	100	--	--	--	--	OP4	OP5	OP6	OP7	100	0000	OP7	OP6	OP5	OP4
000	101	OP3	OP4	OP5	OP6	OP7	--	--	--	000	0000	OP6	OP5	OP4	OP3
										100	1110	--	--	--	OP7
001	101	--	OP3	OP4	OP5	OP6	OP7	--	--	000	0001	OP5	OP4	OP3	--
										100	1100	--	--	OP7	OP6
010	101	--	--	OP3	OP4	OP5	OP6	OP7	--	000	0011	OP4	OP3	--	--
										100	1000	--	OP7	OP6	OP5

Table 10-7. XLB bus to PCI Byte Lanes for Memory[a] Transactions (continued)

XL bus										PCI Bus					
A [29:31]	TSIZ [0:2]	Data Bus Byte Lanes								AD [2:0]	BE [3:0]	31:24	23:16	15:8	7:0
		0	1	2	3	4	5	6	7						
011	101	--	--	--	OP3	OP4	OP5	OP6	OP7	000	0111	OP3	--	--	--
										100	0000	OP7	OP6	OP5	OP4
000	110	OP2	OP3	OP4	OP5	OP6	OP7	--	--	000	0000	OP5	OP4	OP3	OP2
										100	1100	--	--	OP7	OP6
001	110	--	OP2	OP3	OP4	OP5	OP6	OP7	--	000	0001	OP4	OP3	OP2	--
										100	1000	--	OP7	OP6	OP5
010	110	--	--	OP2	OP3	OP4	OP5	OP6	OP7	000	0011	OP3	OP2	--	--
										100	0000	OP7	OP6	OP5	OP4
000	111	OP1	OP2	OP3	OP4	OP5	OP6	OP7	--	000	0000	OP4	OP3	OP2	OP1
										100	1000	--	OP7	OP6	OP5
001	111	--	OP1	OP2	OP3	OP4	OP5	OP6	OP7	000	0001	OP3	OP2	OP1	--
										100	0000	OP7	OP6	OP5	OP4
000	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7	000	0000	OP3	OP2	OP1	OP0
										100	0000	OP7	OP6	OP5	OP4

[a] The byte lane translation will be similar for other types of transactions. However, the PCI address may be different as explained in Section 10.4.1.5, *Addressing*.

10.4.4.2 Configuration Mechanism

In order to support both Type 0 and Type 1 configuration transactions, MPC5200 provides the 32 bit Configuration Address Register (CAR), located at module address 0x1F8. The register specifies the target PCI bus, device, function, and configuration register to be accessed. A read or a write to the MPC5200 window defined as PCI I/O space, in PCIIWCR, causes the host bridge to translate the access into a PCI configuration cycle if the enable bit in the Configuration Address Register is set and the device number does not equal 0b1_1111. For space to be defined as I/O space, the accessed space (one of the initiator Windows) must be programmed as I/O, not memory. Section 10.3.2.8, *Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80*.

The format of the Configuration Address Register is shown in Section 10.3.2.12, *Configuration Address Register PCICAR (RW) —MBAR + 0x0DF8*. When MPC5200 detects an access to an I/O Window, it checks the enable flag and the device number in the Configuration Address Register. If the enable bit is set, and the device number is not 0b1_1111, the MPC5200 performs a configuration cycle translation function and runs a configuration read or configuration write transaction on the PCI bus. The device number 0b1_1111 is used for performing interrupt acknowledge and Special Cycle transactions. See Section 10.4.4.2.3, *Interrupt Acknowledge Transactions* and Section 10.4.4.2.4, *Special Cycle Transactions* for more information. If the bus number corresponds to the local PCI bus (bus number = 0x00), a Type 0 configuration cycle transaction is performed. If the bus number indicates a remote PCI bus, MPC5200 performs a Type 1 configuration cycle translation. If the enable bit is not set, the access to the Configuration Window is passed through to the PCI bus as a I/O space transaction at the internal address (window translation applies).

Note that the PCI data byte enables ($\overline{\text{C/BE}}$[3:0]) are determined by the size access to the Window.

10.4.4.2.1 Type 0 Configuration Translation

Figure 10-7 shows the Type 0 translation function performed on the contents of the Configuration Address Register to the AD[31:0] signals on the PCI bus during the address phase of the configuration cycle (only applies when the Enable bit in the Configuration Address Register is set).

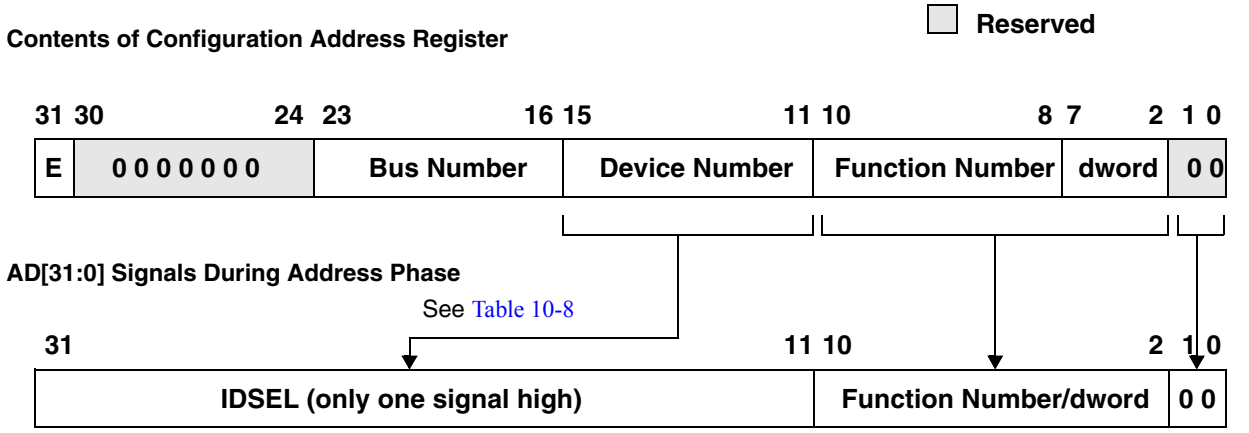


Figure 10-7. Type 0 Configuration Translation

For Type 0 configuration cycles, MPC5200 translates the device number field of the Configuration Address Register into a unique IDSEL line shown in Table 10-8. (allows for 21 different devices).

Table 10-8. Type 0 Configuration Device Number to IDSEL Translation

Device Number		IDSEL
Binary	Decimal	
0b0_0000-0b0_1001	0-9	-
0b0_1010	10	AD31
0b0_1011	11	AD11
0b0_1100	12	AD12
0b0_1101	13	AD13
0b0_1110	14	AD14
0b0_1111	15	AD15
0b1_0000	16	AD16
0b1_0001	17	AD17
0b1_0010	18	AD18
0b1_0011	19	AD19
0b1_0100	20	AD20
0b1_0101	21	AD21
0b1_0110	22	AD22
0b1_0111	23	AD23
0b1_1000	24	AD24
0b1_1001	25	AD25
0b1_1010	26	AD26
0b1_1011	27	AD27
0b1_1100	28	AD28

Table 10-8. Type 0 Configuration Device Number to IDSEL Translation (continued)

Device Number		IDSEL
Binary	Decimal	
0b1_1101	29	AD29
0b1_1110	30	AD30
0b1_1111	31	-

NOTE: Device numbers 0b0_0000 to 0b0_1001 are reserved. Programming to these values and issuing a configuration transaction will result in a PCI configuration cycle with AD31-AD11 driven low.

MPC5200 can issue PCI configuration transactions to itself. A Type 0 configuration initiated by MPC5200 can access its own configuration space by asserting its IDSEL input signal. This is the only way MPC5200 can clear its own status register bits (read-write-clear).

For Type 0 translations, the function number and dword fields are copied without modification onto the AD[10:2] signals and AD[1:0] are driven low during the address phase.

10.4.4.2.2 Type 1 Configuration Translation

For Type 1 translations, the 30 high-order bits of the Configuration Address Register are copied without modification onto the AD[31:2] signals during the address phase. The AD[1:0] signals are driven to 0b01 during the address phase to indicate a Type 1 configuration cycle.

10.4.4.2.3 Interrupt Acknowledge Transactions

When MPC5200 detects a **read** from an I/O-defined Window (Section 10.3.2.8, *Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80*), it checks the enable flag, bus number, and the device number in the Configuration Address Register (Section 10.3.2.12, *Configuration Address Register PCICAR (RW) —MBAR + 0x0DF8*). If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), and the device number is all 1's (device number = 0b1_1111), then an interrupt acknowledge transaction is initiated. If the bus number indicates a subordinate PCI bus (bus number != 0x00), a Type 1 configuration cycle is initiated, similar to any other configuration cycle for which the bus number does not match. The function number and dword values are ignored.

The interrupt acknowledge command (0b0000) is driven on the $\overline{\text{C/BE}}$[3:0] signals and the address bus is driven with a stable pattern during the address phase, but a valid address is not driven. The address of the target device during an interrupt acknowledge is implicit in the command type. Only the system interrupt controller on the PCI bus should respond to the interrupt acknowledge and return the interrupt vector on the data bus during the data phase. The size of the interrupt vector returned is indicated by the value driven on the $\overline{\text{C/BE}}$[3:0] signals.

10.4.4.2.4 Special Cycle Transactions

When the MPC5200 detects a **write** to an I/O-defined Window (Section 10.3.2.8, *Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80*), it checks the enable flag, bus number, and the device number in the Configuration Address Register (Section 10.3.2.12, *Configuration Address Register PCICAR (RW) —MBAR + 0x0DF8*). If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), and the device number is all 1's (device number = 0b1_1111), then a Special Cycle transaction is initiated. If the bus number indicates a subordinate PCI bus (bus number != 0x00), a Type 1 configuration cycle is initiated, similar to any other configuration cycle for which the bus number does not match. The function number and dword values are ignored.

The Special Cycle command (0b0001) is driven on the $\overline{\text{C/BE}}$[3:0] signals and the address bus is driven with a stable pattern during the address phase, but contains no valid address information. The Special Cycle command contains no explicit destination address, but broadcast to all agents on the same bus segment. Each receiving agent must determine whether the message is applicable to it. PCI agent will never assert $\overline{\text{DEVSEL}}$ in response to a Special Cycle command. Master Abort is the normal termination for a Special Cycle and no errors are reported for this case of Master Abort termination. This command is basically a broadcast to all agents, and interested agents accept the command and process the request.

NOTE

Special Cycle commands do not cross PCI-to-PCI bridges. If a master wants to generate a Special Cycle command on a specific bus in the hierarchy that is not its local bus, it must use a Type 1 configuration write command to do so. Type 1 configuration write commands can traverse PCI-to-PCI bridges in both directions for the purpose of generating Special Cycle commands on any bus in the hierarchy and are restricted to a single data phase in length. However, the master must know the specific bus on which it desires to generate the Special Cycle command and cannot simply do a broadcast to one bus and expect it to propagate to all buses.

During the data phase, AD[31:0] contain the Special Cycle message and an optional data field. The Special Cycle message is encoded on the 16 least significant bits (AD[15:0]) and the optional data field is encoded on the most significant bits (AD[31:16]). The Special Cycle message encodings are assigned by the PCI SIG Steering Committee. The current list of defined encodings are provided in Table 10-9.

Table 10-9. Special Cycle Message Encodings

AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 architecture-specific
0x0003-0xFFFF	reserved

10.4.4.3 Transaction Termination

If the PCI cycle Master Aborts, interface will return 0xFFFFFFFF as read data, but complete without error. It will issue an interrupt to the internal interrupt controller if enabled.

For abnormal transaction termination during an XL bus-initiated transaction (unsupported transfer types, retry limit reached, or target abort), an error is generated. It will issue an interrupt to the MPC5200 Interrupt controller if such interrupts are enabled.

Transfers that cross the 32-bit boundary (greater than 4 bytes) to a PCI non-memory address range result in a transfer error. The space is defined as nonmemory if the IO/M# configuration bit associated with that window is programmed “0”.

Table 10-10. Unsupported XLB Transfers

XLB Transaction	PCI Address Space
Burst (32-byte)	Nonmemory
> 4 byte Single Beat	Nonmemory
4 byte Single Beat at a[29:31] 001, 010, or 011	Nonmemory
3 byte Single Beat at a[29:31] 010 or 011	Nonmemory
2 byte Single Beat at a[29:31] 011	Nonmemory

10.4.5 XL bus Target Interface

- The target interface can issue target abort, target retry, and target disconnect terminations.
- The target interface does NOT support fast back-to-back cycles.
- No support of dual address cycles as a PCI target.
- Target transactions are not snooped by the processor.
- Medium device selection timing
- Three 32-byte buffers enhance data throughput.

The XLB Target Interface provides access for external PCI masters to two windows of MPC5200 address space. Target Base Address Translation Registers 0 and 1 allow the user to map PCI address hits on MPC5200 PCI Base Address Registers to areas in the internal address space. All of these registers must be enabled for this interface to operate.

Upon detection of a PCI address phase, the PCI controller decodes the address and bus command to determine if the transaction is for local memory (BAR0 or BAR1hit). If the transaction falls within MPC5200 PCI space (a PCI memory space *only*), the PCI Controller target interface asserts $\overline{\text{DEVSEL}}$, latches the address, decodes the PCI bus command, and forwards them to the internal control unit. On writes, data is forwarded along with the byte enables to the internal gasket. On reads, four bytes of data are provided to the PCI bus and the byte enables determine which byte lanes contain meaningful data. If no byte enables are asserted, MPC5200 completes a read access with valid data and completes a write access by discarding the data internally. All target transactions will be translated into XL bus master transactions.

There are two address translation registers that must be initialized before data transfer can begin. These address registers correspond to BAR0 and BAR1 in MPC5200 PCI Type 00h Configuration space register (PCI space). When there is a hit on MPC5200 PCI base address ranges (0 or 1), the upper bits of the address are written over by this register value to address some space in MPC5200. One 256Kbyte base address range (BAR0) maps to non-prefetchable local memory and one 1Gbyte range (BAR1) targeted to prefetchable memory.

10.4.5.1 Reads from Local Memory

MPC5200 can provide continuous data to a PCI master using two 32-byte buffers. The PCI controller bursts reads internally at each 32-byte PCI address boundary. The data is stored in the first 32-byte buffer until either the PCI master flushes the data or the transaction terminates ($\overline{\text{FRAME}}$ deasserts). For prefetchable memory (BAR1 space), the next line can be fetched from memory in anticipation of the next PCI request (speculative read) and stored in the second buffer. Prefetching is performed for BAR1-addressed transactions if the PCI command is a Memory-Read-Multiple or the prefetch bit is set in the Section 10.3.2.4, *Target Control Register PCITCR(RW) —MBAR + 0x0D6C*.

10.4.5.2 Local Memory Writes

A 32-byte write buffer is implemented to improve data throughput. This allows a write operation to be "*posted*", that is to successfully complete even when the PCI internal controller is requesting access to the local memory. In other words, data is latched while waiting for internal access to local memory to complete. While PCI burst transactions are accepted, writes are sent out on the internal bus as single-beat.

NOTE

Before a read from XLB to PCI or PCI to XLB can complete, all posted writes are flushed.

If the PCI controller aborts the transaction in the middle of PCI burst due to internal conflicts, the external master recognizes some of the data as transferred. (Subsequent transfers of a burst will be aborted on PCI bus). The external PCI master must query the "Target abort signalled" bit in the PCI Type 00h configuration status register to determine if a target abort occurred.

10.4.5.3 Data Translation

The XL bus supports misaligned operations, however, it is strongly recommended that software attempt to transfer contiguous code and data where possible. Non-contiguous transfers degrade performance.

PCI-to-XLB transaction data translation is shown in Table 10-11 and Table 10-12.

Table 10-11. Aligned PCI to XL bus Transfers

PCI Bus						XL bus								
BE [3:0]	AD[2:0]	31:24	23:16	15:8	7:0	A[29:31]	Data Bus Byte Lanes							
							0	1	2	3	4	5	6	7
1110	000				OP3	000	OP3							
1101	000			OP3		001		OP3						
1011	000		OP3			010			OP3					
0111	000	OP3				011				OP3				
1110	100				OP3	100					OP3			
1101	100			OP3		101						OP3		
1011	100		OP3			110							OP3	
0111	100	OP3				111								OP3
1100	000			OP3	OP2	000	OP2	OP3						
1001	000		OP3	OP2		001		OP2	OP3					
0011	000	OP3	OP2			010			OP2	OP3				
1100	100			OP3	OP2	100					OP2	OP3		
1001	100		OP3	OP2		101						OP2	OP3	
0011	100	OP3	OP2			110							OP2	OP3
1000	000		OP3	OP2	OP1	000	OP1	OP2	OP3					
0001	000	OP3	OP2	OP1		000		OP1	OP2	OP3				
1000	100		OP3	OP2	OP1	100					OP1	OP2	OP3	
0001	100	OP3	OP2	OP1		101						OP1	OP2	OP3
0000	000	OP3	OP2	OP1	OP0	000	OP0	OP1	OP2	OP3				
0000	100	OP3	OP2	OP1	OP0	100					OP0	OP1	OP2	OP3

Table 10-12. Non-contiguous PCI to XL bus Transfers (require two XLB bus accesses)

PCI Bus						XL bus								
BE [3:0]	AD[2:0]	31:24	23:16	15:8	7:0	A[29:31]	Data Bus Byte Lanes							
							0	1	2	3	4	5	6	7
1010	000		OP3		OP2	000	OP2							
						010			OP3					
1010	100		OP3		OP2	100					OP2			
						110							OP3	
0110	000	OP3			OP2	000	OP2							
						011				OP3				
0110	100	OP3			OP2	100					OP2			
						111								OP3
0101	000	OP3		OP2		001		OP2						
						011				OP3				
0101	100	OP3		OP2		101						OP2		
						111								OP3
0010	000	OP3	OP2		OP1	000	OP1							
						010			OP2	OP3				
0010	100	OP3	OP2		OP1	100					OP1			
						110							OP2	OP3
0100	000	OP3		OP2	OP1	000	OP1	OP2						
						011				OP3				
0100	100	OP3		OP2	OP1	100					OP1	OP2		
						111								OP3

10.4.5.4 Target Abort

A target abort will occur if the PCI address falls within a base address window (BAR0 or BAR1) that has not been enabled. Section 10.3.2.2, *Target Base Address Translation Register 0 PCITBATR0(RW) —MBAR + 0x0D64* and Section 10.3.2.3, *Target Base Address Translation Register 1 PCITBATR1(RW) —MBAR + 0x0D68*.

10.4.5.5 Latrule Disable

The latrule disable bit in the interface control register, Section 10.3.2.4, *Target Control Register PCITCR(RW) —MBAR + 0x0D6C*, prevents the PCI controller from automatically disconnecting a target transaction due to the PCI 16/8 clock rule. With this bit set, it is possible to hang the PCI bus if the internal bus does not complete the data transfer.

10.4.6 Communication Sub-System Initiator Interface

This interface provides for high-speed, autonomous DMA transactions to PCI with the PCI Controller operating as a standard Communication Sub-System peripheral. Full duplex operation is supported and direct XL bus transactions can also be interleaved while CommBus transactions are in progress. Internal arbitration will occur continuously to support transaction interleaving. (Section 10.4.2, *Initiator Arbitration*.) Multi-Channel DMA operation operates independently of the XL bus. Non-PCI transactions on the XL bus will have 100% bandwidth available to them during PCI Multi-Channel DMA activities. In general, this block will be used by functions in the Multi-Channel DMA API.

The Communication Sub-System Initiator Interface consists of Receive and Transmit FIFOs, integrated as separate Multi-Channel DMA peripherals. Therefore, it is generally controlled by the Multi-Channel DMA controller through a pre-described program loop. As with all Communication Sub-System peripherals, it can be accessed and controlled directly through the Slave bus interface if desired, but this path does not generally lend itself to high throughput.

The Transmit and Receive FIFOs are 512 bytes deep and support PCI bursts up to 8 beats, each beat being a 32 bit word. The burst size is programmable. The general approach is to write a PCI command and address to the control register along with the number of bytes to be transmitted (Packet_Size).

When transmitting data, the module will wait for the Transmit FIFO to fill at least to the minimum number of bytes required to perform the programmed burst; then it begins transmitting the data onto the PCI bus. Multi-Channel DMA must handle filling the Transmit FIFO to support the specified number of bytes. Transmission will continue until the specified number of bytes have been sent.

When reading data, the module will check that enough space is available in the Receive FIFO and immediately begin PCI read transactions. Multi-Channel DMA must handle emptying the Receive FIFO to support the specified number of bytes. Transmission will continue until the specified number of bytes have been received. To avoid stale data while receiving the last burst flushing of the RX FIFO can be forced with the set of the flush bit FE. Section 10.3.3.2.4, *Rx Enables PCIRER (RW) —MBAR + 0x388C*

At this point, software must restart the procedure by at least re-writing the Packet_Size register. Each transmission of the specified number of bytes is considered a “packet”. A new packet can be instructed to continue at the last valid PCI address or software may choose to write a new starting address. The largest burst size is 8 PowerPC words and the largest *Packet_Size* is 65,535 bytes, so a packet will typically consist of many PCI data bursts.

The Transmit Controller will wait until sufficient bytes are in the Transmit FIFO to support a full burst and will continue in this mode until the entire packet is transmitted. Similarly, the Receive Controller will stall until sufficient space is available in the Receive FIFO to support a full burst. If the packet is nearly done and the number of bytes remaining to complete the packet is less than Max_beats, the remaining data will be performed as single-beat PCI transactions.

10.4.6.1 Access Width

This Multi-Channel DMA module primarily performs 32-bit data accesses to and from PCI, even though some signals are referred to in bytes. The two least significant bits of the PCITPSR and PCIRPSR value are ignored. All PCI byte enables are enabled during these types of accesses. Additionally, the FIFOs should only be accessed using 32-bit accesses.

The Communication Sub-System interface optionally supports 16 bit accesses on the PCI bus. Since reads and writes to and from the FIFO require 32-bit accesses, using this option requires padding the remaining 16 bits of data.

10.4.6.2 Addressing

The Communication Sub-System Initiator interface does not use the addressing windows that are set up for the XL bus Initiator Interface. Instead, the Tx Start Address register and Rx Start Address register are used. Software programs these registers with the initial starting address for the packet. The module contains an internal counter which will present the incremented PCI address at the beginning of each successive burst for packet transfers.

10.4.6.3 Data Translation

The PCI bus is inherently little endian in its byte ordering. The Comm bus however is big endian. Table 10-13 shows the byte lane mapping between the two buses. Since this interface only allows 32-bit accesses, there is only one entry.

Table 10-13. Comm bus to PCI Byte Lanes for Memory[a] Transactions

Transfer	Comm bus						PCI data bus					
	cAddress [1:0]	cByte Enable [3:0]	Data Bus				PCI_ AD [1:0]	BE [3:0]	Data Bus			
			31: 24	23: 16	15:8	7:0			31:24	23:16	15:8	7:0
long	00	1111	OP0	OP1	OP2	OP3	00	0000	OP3	OP2	OP1	OP0

[a] The byte lane translation will be similar for other types of transactions. However, the PCI address may be different as explained in Section 10.4.1.5, *Addressing*.

10.4.6.4 Initialization

The following list is the recommended procedure for setting up either the Transmit or Receive controller.

1. Set the Start Address

2. Set the PCI command, Max_Retries, and Max_Beats
3. Set mode, Continuous or Non-continuous
4. Reset the FIFO
5. Set the FIFO Alarm and Granularity fields
6. Set the Master Enable bit (eventually enable the wanted interrupt in case of errors or even of a normal termination)
7. Set the Reset Controller bit low
8. Setup the BestComm (eventually passing the parameters to the task if needed, enabling, if required, the Task to interrupt the Core when finished, etc.)
9. Start the Task(s). It is not strictly necessary to start a PCI RX or TX task before starting the PCI to transmit/receive as one will 'wait' for the other to fill the data in or out of the FIFO.
10. Write the Packet Size value to fire off the transfer

10.4.6.5 Restart and Reset

A Restart sequence (namely writing of the Packet Size register) is required whenever the controller ends a packet transmission, either normally or abnormally. In non-continuous mode, a new Start_Add address is generally required since this value is re-used as the start of the next packet once it is Restarted. In Continuous mode, the Start_Add value is not reused. Instead, the next packet begins where the last one left off, but a Restart sequence is still required to get this next packet started.

Writing a non-zero value to the Packet_Size register generates a Restart pulse to the controller. Depending on the desired mode of operation other register accesses may be required, as described in the following paragraphs.

If Continuous mode is not selected, operation is fairly straight forward. Upon packet termination, Restart will not occur until Packet_Size is written with a non-zero value, even if the packet size is the same it must be re-written. Master Enable bit was previously high and can remain so. Reset Controller bit was previously low and can remain so. Toggling the Master Enable or Reset bit is unnecessary but would not disrupt the transmit controller. If any other Control values, e.g. Start_Add, are to be changed they should be written either prior to writing the Packet_Size value or written while the Master Enable bit is negated and the Reset Controller bit is negated. The recommended approach is to write the control values in order (Packet_Size must be last) and not toggle the Master Enable bit. The Reset bit should remain negated.

If Continuous mode is active, basic operation is still straight forward. A Restart is achieved by writing the Packet_Size register to a non-zero value (just as before). However, the Master Enable and Reset bits must **not** toggle in this case. If the Master Enable bit goes low the Packets_Done counter will be reset. If the Reset bit goes high the Start_Add value will be re-loaded and subsequent transactions will begin at this address. Therefore, the Master Enable bit can be used to reset the Packets_Done counter but without disturbing the current PCI address. The Reset Controller bit will reset the counter and reload the Start_Add value into the transmit controller, thus achieving a total restart of a continuous mode sequence. In any case, it is still required that the Packet_Size register be written to complete a Restart sequence.

The Master Enable bit, if negated, will prevent a Restart sequence but allows Control values to be updated without order dependency. A side effect is to reset the Packets_Done counter and status, which is a concern in continuous mode only.

The Reset bit (RC bit of the RX/TX Enables register, **NOT** the external PCI RESET line), if asserted, will force a Reset of the controller. All continuous mode effects will be reset and the Start_Add value is re-loaded. However, the Reset bit must be negated while the required write to the Packet_Size register is accomplished. The Reset bit provides the only means to re-load the Start_Add value into the transmit controller while Continuous mode is active. In either mode it provides a means to clear the transmit controller in cases of abnormal termination. Note, a new Start_Add value must be written prior to setting the Reset bit.

10.4.6.6 PCI Commands

The expected PCI commands are Memory Write for transmit and Memory Read for receive. These are independent of cache or line size. This permits the number of data beats per transaction to be flexible. If any requirements exist on number of data beats, then the software must carefully consider the possibilities. If the Max_Beats setting does not divide properly into the Packet_Size setting then the packet will end up with one or more single-beat transaction(s). Setting Max_Beats to 1 will force all transactions to be single-beat but will affect throughput.

In normal operation, all PCI byte enables will be asserted for PCI transactions through this interface, except if the 16-bit Word register bit is set in the Section 10.3.3.1.3, *Tx Transaction Control Register PCITTCR(RW) —MBAR + 0x3808* or Section 10.3.3.2.3, *Rx Transaction Control Register PCIRTCR(RW) —MBAR + 0x3888*, in which case $\overline{BE}[3:0]$ = 1100.

Configuration writes to an external target *should* be handled exclusively by the XL bus Initiator interface.

10.4.6.7 FIFO Considerations

Careful consideration must also be given to filling and counting bytes of the Transmit FIFO and emptying and counting bytes of the Receive FIFO. This operation is expected to be accomplished through Multi-Channel DMA which can also perform the register writes to the controller, including necessary Restart sequences.

10.4.6.8 Alarms

The FIFO alarm registers allow software to control when the DMA fills or empties the appropriate FIFO.

10.4.6.9 Bus Errors

Since Bus Errors are particular to the module register set and that register set includes both Transmit and Receive Controller and FIFO settings, the Bus Error status bits and Bus error Enable bit(s) are duplicated in the Transmit and Receive register groupings. Clearing or setting one will clear or set the other. From a software point of view, then, they can be treated separately or together, as desired.

10.4.7 PCI - Supported Clock Ratios

MPC5200 supports the following XLB:IP:PCI clock ratios.

Table 10-14. XLB:IP:PCI Clock Ratios

XLB:IP:PCI	XLB CLK	IP CLK	PCI CLK
4:4:2	132 MHz	132 MHz	66 MHz
4:4:1	132 MHz	132 MHz	33 MHz
4:2:2	132 MHz	66 MHz	66 MHz
4:2:1	132 MHz	66 MHz	33 MHz
2:2:2	66 MHz	66 MHz	66 MHz
2:2:1	66 MHz	66 MHz	33 MHz
2:1:1	66 MHz	33 MHz	33 MHz

10.4.8 Interrupts

10.4.8.1 PCI Bus Interrupts

MPC5200 does not generate interrupts on the PCI bus interrupt lines INTA - INTD.

10.4.8.2 Internal Interrupt

The PCI module is capable of generating 3 interrupts to MPC5200 interrupt controller in MPC5200 SIU. Each interrupt can be enabled for a variety of conditions, mostly error conditions. For the XL bus Initiator interface, the internal interrupt can be enabled for Retry errors, Target Aborts and Initiator (Master) Aborts. See Section 10.3.2.9, *Initiator Control Register PCIICR(RW) —MBAR + 0x0D84* and Section 10.3.2.10, *Initiator Status Register PCIISR(RWC) —MBAR + 0x0D88* for more information. For the Comm bus Initiator interface, an internal interrupt can be enabled for FIFO errors and Normal Termination of a packet transfer for either the Receive (rx) or Transmit (tx) interface. For more information, see the Enable and Status registers for the Comm bus Transmit and Receive interfaces, Section 10.3.3.1, *Multi-Channel DMA Transmit Interface* and Section 10.3.3.2, *Multi-Channel DMA Receive Interface*.

10.5 PCI Arbiter

The PCI Arbiter is a separate module, it is not part of the PCI Controller module. The 32-bit multiplexed PCI A/D bus is shared with the ATA Controller and LocalPlus Controller. The on-chip arbiter (called PCI Arbiter) controls the access to the AD bus for the different clients:

- PCI clients
  - XIPCI (XLB-PCI interface)
  - SCPCI (BestComm-PCI interface)
  - external PCI
- non-PCI clients
  - LPC (LocalPlus bus interface)
  - SCLPC (BestComm LocalPlus bus interface)
  - ATA

One pair **only** of external PCI REQ#/GNT# signals is supported by the PCI Arbiter. By an external Priority Encoder multiple external masters could be connected. The PCI bus clock is always sourced from the MPC5200.

The PCI Arbiter implements a Round-Robin fairness algorithm, which avoids the domination of the bus by high-priority masters and exclusion of low-priority masters. The PCI Arbiter is capable of Parking the current Master to stay on last master in absence of other requests. The support of the non-PCI clients presents special challenges to the arbitration scheme.

The PCI Arbiter runs independently. The programmability consists of a Soft Reset, which allows to reset the PCI Arbiter, and one status bit to detect the Broken Master condition. and a corresponding enable bit for the generation of a CPU interrupt for the Broken Master condition. All these register bits are located in registers of the PCI Controller.

In case of broken master detection the external PCI REQ# will be dis-connected internally and will be re-connected after external deassertion of PCI REQ# or by software (Softreset) or by Hardreset. After broken master detection (bus idle for 16 clocks) the arbiter will ignore any PCI FRAME# assertion.

The PCI Arbiter does not support preemption of the internal masters XIPCI or SCPCI. The internal master is granted until the transaction has been completed. The Latency Timer (LT) cannot terminate any transfer.

10.6 Application Information

This section provides example usage of some of the features of the PCI module.

10.6.1 XL bus Initiated Transaction Mapping

The use of the PCI Configuration Address Register along with the initiator window registers provide many possibilities for PCI command and address generation. Table 10-15 shows how the PCI Controller accepts read and write requests from a XLB bus master and decodes them to different address ranges resulting in the generation of memory, I/O, configuration, interrupt acknowledge and special cycles on the PCI bus. The Window Registers are defined in Section 10.3.2.6, *Initiator Window 1 Base/Translation Address Register PCIIW1BTAR(RW) —MBAR + 0x0D74* through Section 10.3.2.8, *Initiator Window Configuration Register PCIIWCR(RW) —MBAR + 0x0D80*.

Table 10-15. Transaction Mapping: XLB -> PCI

XL bus Transaction (XLB Slave Interface)	Cache Line Size Register= 8	Initiator Register Settings				PCI Transaction Controller (XLB Initiator Interface) -> PCI Target
		Initiator Window Configuration bits		Configuration Address Register		
		IO/M#	PRC	En	device number == b1_1111	
Single-Beat 1 -> 8 byte Read	x	0	b00	x	x	Memory Read
Burst Read (32 bytes)	x	0	b00	x	x	Memory Read
Single-Beat 1 -> 8 byte Read	x	0	b01	x	x	Memory Read
Burst Read	false	0	b01	x	x	Memory Read
Burst Read	true	0	b01	x	x	Memory Read Line
Single-Beat 1 -> 8 byte Read	x	0	b10	x	x	Memory Read Multiple
Burst Read	x	0	b10	x	x	Memory Read Multiple
Single-Beat 1 -> 8 byte, or Burst Write	x	0	x	x	x	Memory Write
Single-Beat 1 -> 4 byte Read	x	1	x	0	x	I/O Read
Single-Beat 1 -> 4 byte Write	x	1	x	0	x	I/O Write
Single-Beat 1 -> 4 byte Read	x	1	x	1	false	Configuration Read
Single-Beat 1 -> 4 byte Write	x	1	x	1	false	Configuration Write
Single-Beat 1 -> 4 byte Read	x	1	x	1	true	Interrupt acknowledge

Table 10-15. Transaction Mapping: XLB -> PCI (continued)

XL bus Transaction (XLB Slave Interface)	Cache Line Size Register= 8	Initiator Register Settings				PCI Transaction Controller (XLB Initiator Interface) -> PCI Target
		Initiator Window Configuration bits		Configuration Address Register		
		IO/M#	PRC	En	device number == b1_1111	
Single-Beat 1 -> 4 byte Write	x	1	x	1	true	Special Cycle

Note:

1. Dual Address Cycles and Memory Write and Invalidate Commands are not supported
2. x means “don’t care”

10.6.2 Address Maps

The address mapping in MPC5200 system is setup by software through a number of base address registers. (Section 3.2, *Internal Register Memory Map* for more detail). The internal CPU writes the base address value to module base address register MBAR. MBAR holds the base address for the 256 Kbyte space allocated to internal registers.

10.6.2.1 Address Translation

10.6.2.1.1 Inbound Address Translation

The MPC5200-as-Target occupies 2 memory target address windows on the PCI bus. The location is determined by the values programmed to BAR0 and BAR1 of the PCI Type 00h Configuration space. These inbound memory window sizes are fixed to one 256 Kbyte window (BAR0) and one 1 Gbyte window (BAR1).

PCI inbound address translation allows address translation to any space in the MPC5200 space (4 Gbyte of address space). The target base address translation registers TBATR0 and TBATR1 specify the location of the inbound memory window. These registers are described in Section Section 10.4.3, *Configuration Interface*. Address translation occurs for all enabled inbound transactions. If the enable bit of the Target Base Address Translation Registers is cleared, MPC5200 aborts all PCI memory transactions to that base address window.

Note, the PCI configuring master can program BAR0 to overlap BAR1. The default address translation value is TBATR0 in that case. It is not recommended to program overlapping BAR0 and BAR1 or overlapping TBATR0 and TBATR1. An overlap of TBATRs can cause data write-over of BAR0 data.

The Initiator Window Base Address Registers are used to decode XL bus addresses for PCI bus transactions. The base address and base address mask values define the upper byte of address to decode. The XL bus address space in MPC5200 dedicated to PCI transactions can be mapped to two 16-Mbyte or larger address spaces in MPC5200. In normal operation, software should not program either Target Address Window Translation Register to address Initiator Window space. In that event, MPC5200-as-Target transaction would propagate through MPC5200’s internal bus and request PCI bus access as the PCI Initiator. The PCI arbiter could see the PCI bus as busy (target read transaction in progress) and only a time-out would free the PCI bus.

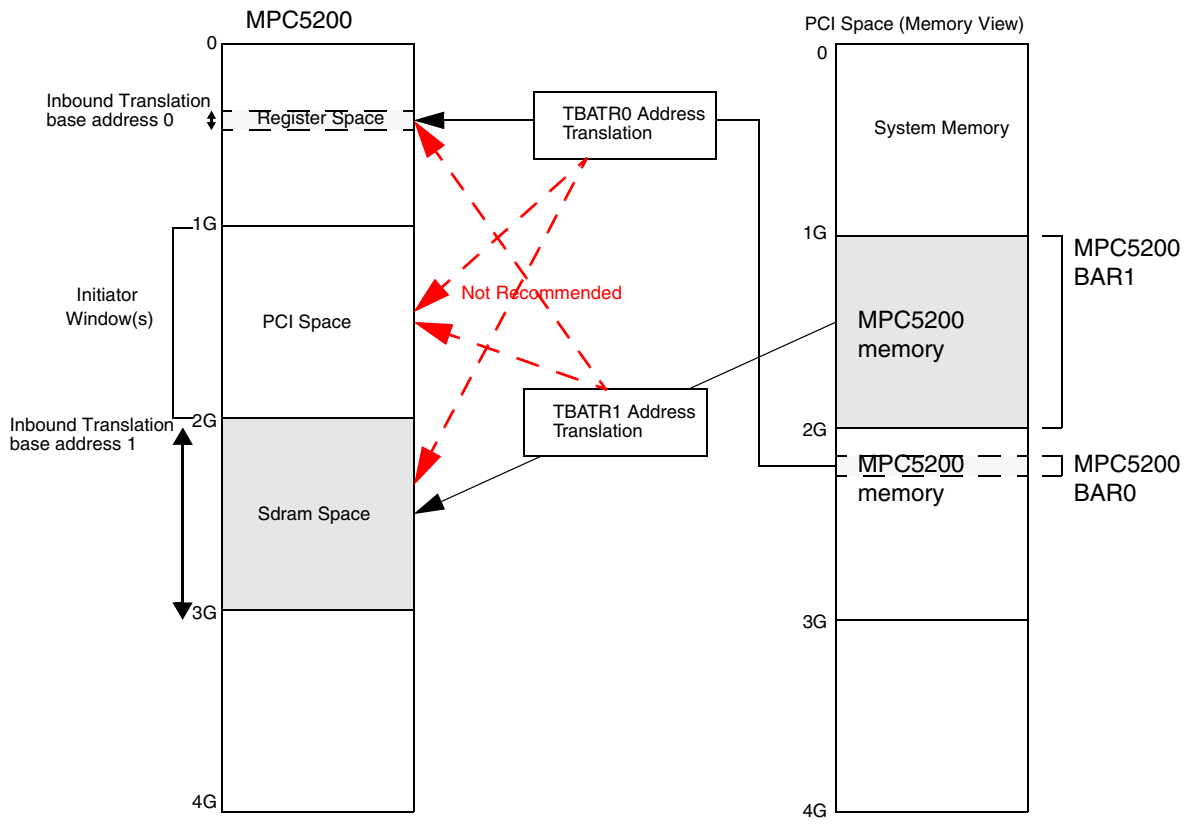


Figure 10-8. Inbound Address Map

10.6.2.1.2 Outbound Address Translation

Figure 10-9 shows example XLB Initiator Window configurations. Overlapping the inbound memory window (MPC5200 Memory) and the outbound translation window is not supported and can cause unpredictable behavior.

This figure doesn't show configuration mechanism.

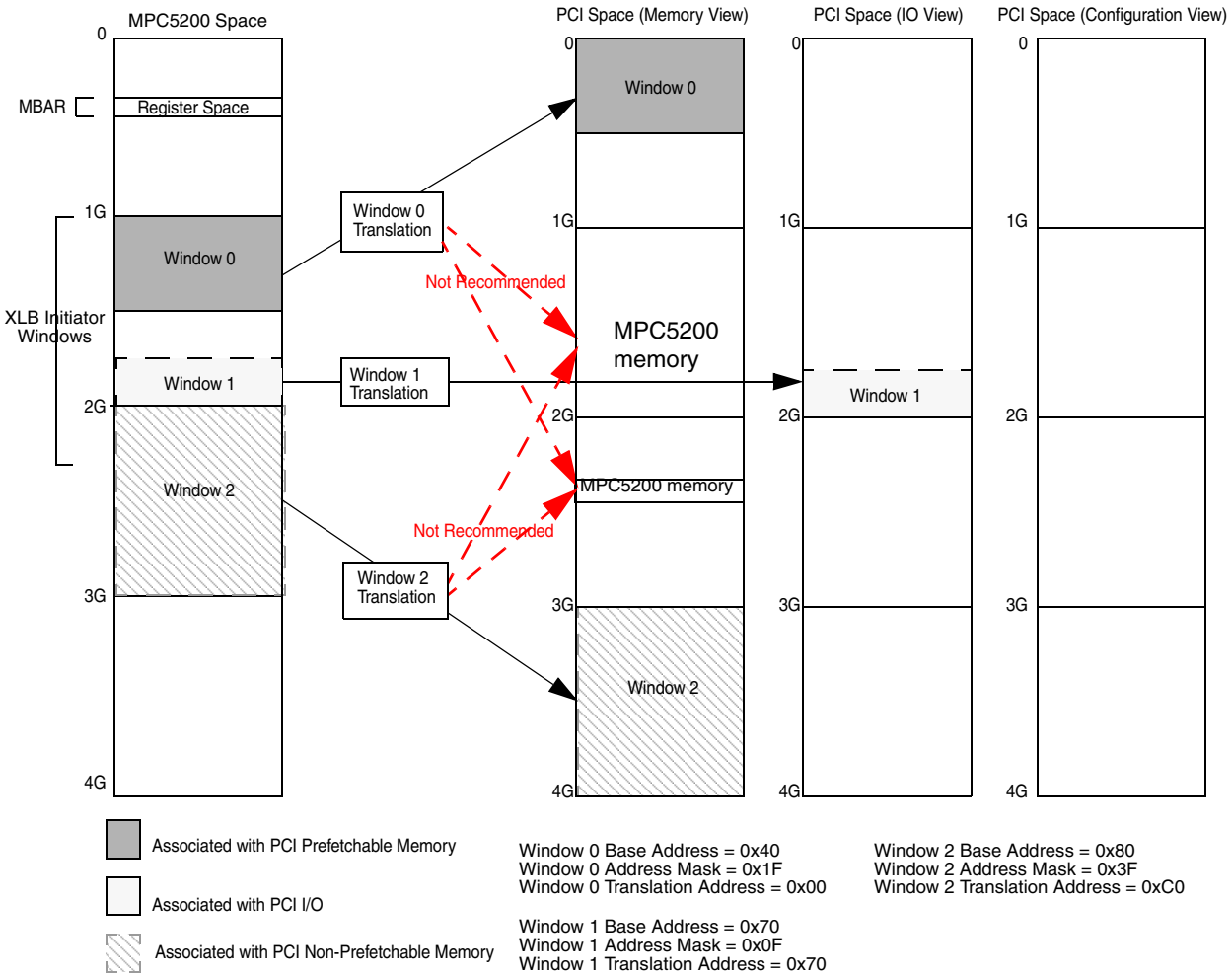


Figure 10-9. Outbound Address Map

10.6.2.1.3 Base Address Register Overview

Table 10-15 shows the available accessibility for all PCI associated base address and translation address registers in MPC5200.

Base Address Register	Register Function	PCI Bus Configuration Access	Processor Access	Any XL bus Master Access
BAR0	PCI Base Address Register 0 (256 Kbyte)	X	X	X
BAR1	PCI Base Address Register 1 (1 Gbyte)	X	X	X
TBATR0	Target Base Address Translation Register 0 (256Kbyte)		X	X
TBATR1	Target Base Address Translation Register 0 (1 Gbyte)		X	X
IMWBAR	Initiator Window Base/Translation Address Registers		X	X

10.6.3 XL bus Arbitration Priority

When the XLB Arbiter Master Priority Register (Section 16.2.11, *Arbiter Master Priority Register (R/W)—MBAR + 0x1F68*) is set to any configuration except all-master fair-share (all masters have the same priority), live lock can occur on the shared PCI bus and the XLB, which results in system-wide live lock.

The only resolution that guarantees that this live lock scenario will not occur is to set all the XLB Arbiter master priorities to be equal. Additionally, it is usually preferable that all master priorities are not set to zero, as this can generate an interrupt by the XLB Arbiter, if enabled.

Chapter 11
ATA Controller

11.1 Overview

The following sections are contained in this document:

- Section 11.2, *BestComm Key Features*
  - Section 11.3, *ATA Register Interface*, includes:
  - Section 11.3.1, *ATA Host Registers—MBAR + 0x3A00*
  - Section 11.3.2, *ATA FIFO Registers—MBAR + 0x3A00*
  - Section 11.3.3, *ATA Drive Registers—MBAR + 0x3A00*
- Section 11.4, *ATA Host Controller Operation*
- Section 11.5, *Signals and Connections*
- Section 11.6, *ATA Interface Description*
- Section 11.7, *ATA Bus Background*
- Section 11.8, *ATA RESET/Power-Up*
- Section 11.9, *ATA I/O Cable Specifications*

The Advanced Technology Attachment (ATA) Controller provides full functional compatibility with ATA-4 documentation, supporting Ultra-33. For more ATA Standards information, refer to "*American National Standard for Information Technology—AT Attachment with Packet Interface Extension (ATA/ATAPI-4)*".

A dedicated MPC5200 pin for ATA reset is *not* provided. An appropriate signal on the board should be routed to the reset input on the ATA connector. If ATA reset is tied to HRESET or SRESET on MPC5200 pins, they are asserted and internally held low for an appropriate period of time to satisfy ATA reset. An MPC5200 GPIO may be used to drive ATA reset independently if special software control is needed.

Figure 11-1 shows the ATA Controller Interface.

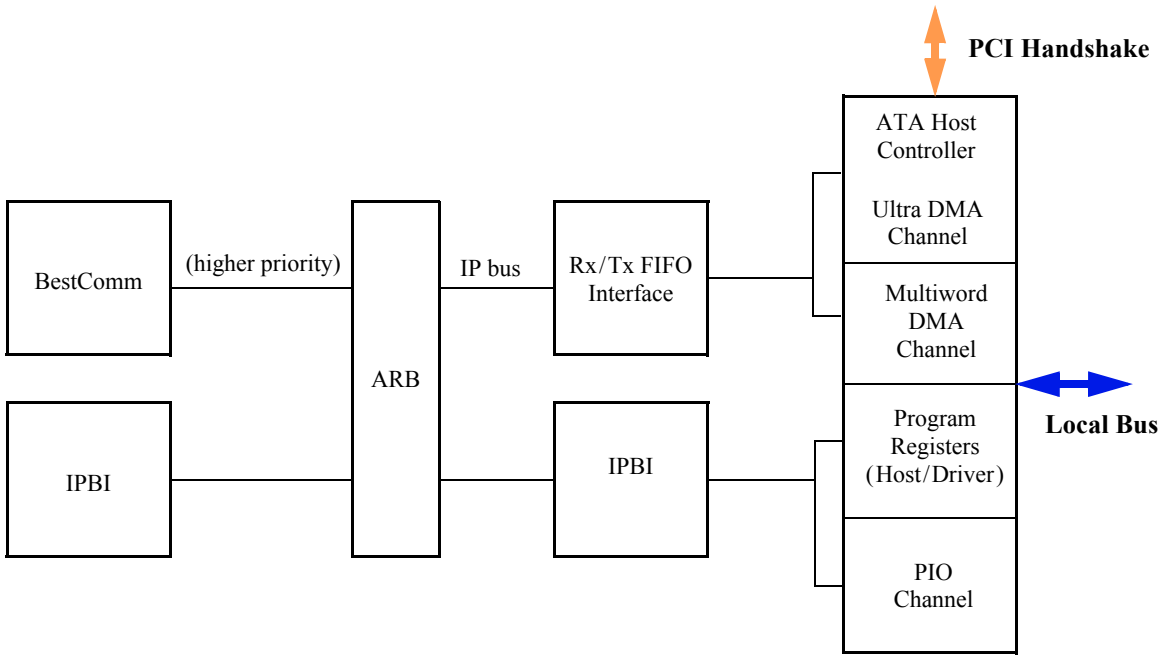


Figure 11-1. ATA Controller Interface

11.2 BestComm Key Features

11.2.1 BestComm Read

1. microprocessor sets up descriptors in BC RAM and initiates a transfer.
2. BC hits on an ATA command FIFO space and writes a command (ATA drive register address, transfer size) into FIFO.
3. ATA Controller reads data from the drive and puts data in FIFO.

4. As FIFO fills, BC is interrupted and moves data from FIFO to an internal destination.

11.2.2 BestComm Write

1. microprocessor sets up descriptors in BC RAM and initiates a transfer.
2. BC hits on an ATA command FIFO space and writes a command (ATA drive register address, transfer size) into FIFO.
3. BC reads data from internal source and puts data in FIFO
4. ATA Controller transfers data from FIFO and writes to drive.

NOTE

Any DMA transfer, where source and destination are both on the local bus, requires internal BC SRAM buffering.

11.3 ATA Register Interface

The IPBI module contains all software-programmable ATA Controller registers and the IPB glue logic needed to read and write these registers. The IPBI registers are listed below. Unless otherwise noted, each register is written and read from the same address.

11.3.1 ATA Host Registers—MBAR + 0x3A00

ATA is controlled by 10 32-bit registers. These registers are located at an offset from MBAR of 0x3A00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3A00 + register address**

Hyperlinks to the ATA Host registers are provided below:

- ATA Host Configuration Register (0x3A00)
- ATA Host Status Register (0x3A04)
- ATA PIO Timing 1 Register (0x3A08)
- ATA PIO Timing 2 Register (0x3A0C)
- ATA Multiword DMA Timing 1 Register (0x3A10)
- ATA Multiword DMA Timing 2 Register (0x3A14)
- ATA Ultra DMA Timing 1 Register (0x3A18)
- ATA Ultra DMA Timing 2 Register (0x3A1C)
- ATA Ultra DMA Timing 3 Register (0x3A20)
- ATA Ultra DMA Timing 4 Register (0x3A24)
- ATA Ultra DMA Timing 5 Register (0x3A28)

11.3.1.1 ATA Host Configuration Register—MBAR + 0x3A00

Table 11-1. ATA Host Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMR	FR	Reserved				IE	IORDY	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	SMR	State Machine Reset—bit resets ATA state machine to IDLE state for PIO, DMA and UDMA read/write.
1	FR	FIFO Reset—bit can be used to reset FIFO when bit 0 of this register is set to reset the ATA state machine. During normal ATA transaction, FIFO can be reset by setting ATA Drive Command Register FR bit (see Table 11-29.)
2:5	—	Reserved

Bits	Name	Description
6	IE	Enables drive interrupt to pass to CPU in PIO modes.
7	IORDY	Set by software when the drive supports IORDY. Required for PIO mode 3 and above.
16:31	—	Reserved

11.3.1.2 ATA Host Status Register—MBAR + 0x3A04

Table 11-2. ATA Host Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TIP	UREP	Reserved				RERR	WERR	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	TIP	Transaction in Progress—indicator bit MUST be polled by software before PIO access. System bus (XL bus) locks up if PIO access is attempted while this bit is set. This bit is read-only.
1	UREP	UDMA Read Extended Pause—bit sets when drive stops strobing for an extended period without initiating burst termination by negating DMARQ, during an UDMA read burst. Software may initiate an Ultra DMA read burst termination, in this case by setting ATA Drive Device Command Register HUT bit (see Table 11-29.).
2:5	—	Reserved
6	RERR	Read Error—An un-implemented register read.
7	WERR	Write Error—An un-implemented register write.
8:31	—	Reserved

11.3.1.3 ATA PIO Timing 1 Register—MBAR + 0x3A08

Table 11-3. ATA PIO Timing 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	pio_t0								pio_t2_8							
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 lsb
R	pio_t2_16								Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	pio_t0	PIO cycle time count value is based on system clock operating frequency.
8:15	pio_t2_8	PIO read/write pulse width for 8-bit transfers. Count value is based on system clock operating frequency.
16:23	pio_t2_16	PIO read/write pulse width for 16-bit transfers. Count value is based on system clock operating frequency.
24:31	—	Reserved

11.3.1.4 ATA PIO Timing 2 Register—MBAR + 0x3A0C

Table 11-4. ATA PIO Timing 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	pio_t4								pio_t1							
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 lsb
R	pio_ta								Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	pio_t4	PIO write (DIOW) data hold time. Count value is based on system clock operating frequency.
8:15	pio_t1	Address valid to DIOR/DIOW setup. Count value is based on system clock operating frequency.
16:23	pio_ta	IORDY setup time. Count value is based on system clock operating frequency.
24:31	—	Reserved

11.3.1.5 ATA Multiword DMA Timing 1 Register—MBAR + 0x3A10

Table 11-5. ATA Multiword DMA Timing 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	dma_t0								dma_td							
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 lsb
R	dma_tk								dma_tm							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	dma_t0	Multiword DMA cycle time. Count value is based on system clock operating frequency.
8:15	dma_td	Multiword DMA read/write (DIOR/DIOW) asserted pulse width. Count value is based on system clock operating frequency.
16:23	dma_tk	Multiword DMA read/write (DIOR/DIOW) negated pulse width. Count value is based on system clock operating frequency.
24:31	dma_tm	$\overline{CS}$[0], $\overline{CS}$[1] valid to DIOR/DIOW. Count value is based on system clock operating frequency.

11.3.1.6 ATA Multiword DMA Timing 2 Register—MBAR + 0x3A14

Table 11-6. ATA Multiword DMA Timing 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	dma_th								dma_tj							
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 lsb
R	dma_tn								Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	dma_th	Multiword DMA write (DIOW) data hold time. Count value is based on system clock operating frequency.
8:15	dma_tj	Multiword DMA read/write (DIOR/DIOW) asserted pulse width. Count value is based on system clock operating frequency.
16:23	dma_tn	$\overline{CS}$[0], $\overline{CS}$[1] hold. Count value is based on system clock operating frequency.
24:31	—	Reserved

11.3.1.7 ATA Ultra DMA Timing 1 Register—MBAR + 0x3A18

Table 11-7. ATA Ultra DMA Timing 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	udma_t2cyc								udma_tcyc							
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 lsb
R	udma_tds								udma_tdh							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	udma_t2cyc	Ultra DMA sustained average two cycle time. Count value is based on system clock operating frequency.
8:15	udma_tcyc	Ultra DMA strobe edge to strobe edge cycle time. Count value is based on system clock operating frequency.
16:23	udma_tds	Ultra DMA read data setup time. Count value is based on system clock operating frequency.
24:31	udma_tdh	Ultra DMA read data hold time. Count value is based on system clock operating frequency.

11.3.1.8 ATA Ultra DMA Timing 2 Register—MBAR + 0x3A1C

Table 11-8. ATA Ultra DMA Timing 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	udma_tdvs								udma_tdvh							
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 lsb
R	udma_tfs								udma_tli							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	udma_tdvs	Ultra DMA write data setup time. Count value is based on system clock operating frequency.
8:15	udma_tdvh	Ultra DMA write data hold time. Count value is based on system clock operating frequency.
16:23	udma_tfs	First strobe time during the initiation of ultra DMA data transfer. Count value is based on system clock operating frequency.
24:31	udma_tli	Limited interlock time with a defined maximum, when drive or host are waiting for response from each other. Count value is based on system clock operating frequency.

11.3.1.9 ATA Ultra DMA Timing 3 Register—MBAR + 0x3A20

Table 11-9. ATA Ultra DMA Timing 3 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	udma_tmli								udma_taz							
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 lsb
R	udma_tenv								udma_tsri							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	udma_tmli	Limited interlock time with a defined minimum, when drive or host are waiting for response from each other. Count value is based on system clock operating frequency.
8:15	udma_taz	Maximum time allowed for output drivers to release from being driven. Count value is based on system clock operating frequency.
16:23	udma_tenv	Envelope time from DMACK to STOP and HDMARDY during data-out burst initiation. Count value is based on system clock operating frequency.
24:31	udma_tsr	Strobe to DMARDY time. If DMARDY is negated before this long after strobe edge the recipient receives no more than one additional data word. Count value is based on system clock operating frequency.

11.3.1.10 ATA Ultra DMA Timing 4 Register—MBAR + 0x3A24

Table 11-10. ATA Ultra DMA Timing 4 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	udma_tss								udma_trfs							
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 lsb
R	udma_trp								udma_tac							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	udma_tss	Time from strobe edge to negation of DMARQ (when drive terminates burst) or assertion of STOP (when host terminates burst). Count value is based on system clock operating frequency.
8:15	udma_trfs	Ready-to-final-strobe time. No strobe edges are sent this long after negation of DMARDY. Count value is based on system clock operating frequency.
16:23	udma_trp	Ready-to-pause time. The time that recipient waits to initiate pause after negating DMARDY. Count value is based on system clock operating frequency.
24:31	udma_tack	Setup and hold times for DMACK before negation or assertion. Count value is based on system clock operating frequency.

11.3.1.11 ATA Ultra DMA Timing 5 Register—MBAR + 0x3A28

Table 11-11. ATA Ultra DMA Timing 5 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	udma_tzah								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	udma_tzah	Minimum delay time required for output drivers to assert or negate from release state. Count value is based on system clock operating frequency.
8:31	—	Reserved

11.3.1.12 ATA Share Count Register—MBAR + 0x3A2C

Table 11-12. ata_shre_cnt

	msb0	1	2	3	3	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								ata_share_cnt							
W																
RESET:	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	—	Reserved
8:15	ata_share_cnt	This 8-bit value controls the length of the “time slot” assigned to ATA transactions when PCI arbiter provides a grant to the ATA device. This is in IPB clocks. The arbiter will maintain the grant to ATA for (at least) the ata_share_cnt value. When this value has expired, ATA may be interrupted (paused) by the arbiter, to service other pending requests for the AD bus. Default value at reset is 128 **Note:** The maximal allowed setting is 0xFE.
16:31	—	Reserved

11.3.2 ATA FIFO Registers—MBAR + 0x3A00

ATA uses a single FIFO that changes direction based on the Rx/Tx mode. Software controls direction change and flushes FIFO before changing directions. FIFO memory is 512Bytes (Four 8 x 128 memories).

ATA FIFO is controlled by 32-bit registers. These registers are located at an offset from MBAR of 0x3a00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3A00 + register address**

Hyperlinks to the ATA FIFO registers are provided below:

- ATA Rx/Tx FIFO Data Word Register (0x3A3C)
- ATA Rx/Tx FIFO Status Register (0x3A40)
- ATA Rx/Tx FIFO Control Register (0x3A44)
- ATA Rx/Tx FIFO Alarm Register (0x3A48)
- ATA Rx/Tx FIFO Read Pointer Register (0x3A4C)
- ATA Rx/Tx FIFO Write Pointer Register (0x3A50)

11.3.2.1 ATA Rx/Tx FIFO Data Word Register—MBAR + 0x3A3C

Table 11-13. ATA Rx/Tx FIFO Data Word Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_Data_Word															
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 lsb
R	FIFO_Data_Word															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:31	FIFO_Data_Word	The FIFO data port. Reading from this location “pops” data from the FIFO, writing “pushes” data into the FIFO. During normal operation the BestComm Controller pushes data here. **Note: NOTE:** ONLY full long-word access is allowed. If all byte enables are not asserted when accessing this location, a FIFO error flag is generated.

11.3.2.2 ATA Rx/Tx FIFO Status Register—MBAR + 0x3A40

Table 11-14. ATA Rx/Tx FIFO Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									Err	UF	OF	Full	HI	LO	Emty
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:8	—	Reserved
9	Err	Error—flag bit is essentially the logical "OR" of other flag bits and can be polled for detection of any FIFO error. After clearing the offending condition, writing 1 to this bit clears flag.

Bits	Name	Description
10	UF	UnderFlow—flag indicates read pointer has surpassed the write pointer. FIFO was read beyond empty. Resetting FIFO clears this condition; writing 1 to this bit clears flag.
11	OF	OverFlow—flag indicates write pointer surpassed read pointer. FIFO was written beyond full. Resetting FIFO clears this condition; writing 1 to this bit clears flag.
12	Full	FIFO full—this is NOT a sticky bit or error condition. Full indication tracks with FIFO state.
13	HI	High—FIFO requests attention, because high level alarm is asserted. To clear this condition, FIFO must be read to a level below the setting in granularity bits.
14	LO	Low—FIFO requests attention, because Low level alarm is asserted. To clear this condition, FIFO must be written to a level in which the space remaining is less than the granularity bit setting.
15	Emty	FIFO empty—this is NOT a sticky bit or error condition. Full indication tracks with FIFO state.
16:31	—	Reserved

11.3.2.3 ATA Rx/Tx FIFO Control Register—MBAR + 0x3A44

Table 11-15. ATA Rx/Tx FIFO Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		WFR	Reserved		GR			Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:1	—	Reserved
2	WFR	Write End of Frame (EOF) This bis should remain low.
3:4	—	Reserved
5:7	GR	Granularity—bits control high “watermark” point at which FIFO negates Alarm condition (i.e., request for data). It represents the number of free bytes times 4. 000 = FIFO waits to become completely full before stopping data request. 001 = FIFO stops data request when only one long word of space remains.
8:31	—	Reserved

11.3.2.4 ATA Rx/Tx FIFO Alarm Register—MBAR + 0x3A48

Table 11-16. ATA Rx/Tx FIFO Alarm Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				Alarm											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	—	Reserved
20:31	Alarm	User writes these bits to set low level “watermark”, which is the point where FIFO asserts request for BestComm Controller data filling. Value is in bytes. For example, with Alarm = 32, alarm condition occurs when FIFO contains 32 Bytes or less. Once asserted, alarm does not negate until high level mark is reached, as specified by FIFO control register granularity bits.

11.3.2.5 ATA Rx/Tx FIFO Read Pointer Register—MBAR + 0x3A4C

Table 11-17. ATA Rx/Tx FIFO Read Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				ReadPtr											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	—	Reserved
20:31	ReadPtr	Value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. Value represents the Read address presented to the FIFO RAM.

11.3.2.6 ATA Rx/Tx FIFO Write Pointer Register—MBAR + 0x3A50

Table 11-18. ATA Rx/Tx FIFO Write Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				WritePtr											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:19	—	Reserved
20:31	WritePtr	Value is maintained by FIFO hardware and is NOT normally written. It can be adjusted in special cases, but this disrupts data flow integrity. Value represents the Read address presented to the FIFO RAM.

11.3.3 ATA Drive Registers—MBAR + 0x3A00

The ATA drive registers are physically located inside the drive controller on the ATA disk drive. The MPC5200 ATA Host Controller provides access to these registers using the chip selects and address bits.

ATA Drive is controlled by 32-bit registers. These registers are located at an offset from MBAR of 0x3a00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3A00 + register address**

Hyperlinks to the ATA Drive registers are provided below:

- ATA Drive Device Control Register (0x3A5C), write-only
- ATA Drive Alternate Status Register (0x3A5C), read-only
- ATA Drive Data Register (0x3A60), R/W
- ATA Drive Features Register (0x3A64), write-only
- ATA Drive Error Register (0x3A64), read-only
- ATA Drive Sector Count Register (0x3A68), R/W
- ATA Drive Sector Number Register (0x3A6C), R/W
- ATA Drive Cylinder Low Register (0x3A70), R/W
- ATA Drive Cylinder High Register (0x3A74), R/W
- ATA Drive Device/Head Register (0x3A78), R/W
- ATA Drive Device Command Register (0x3A7C), write-only
- ATA Drive Device Status Register, (0x3A7C) read-only

11.3.3.1 ATA Drive Device Control Register—MBAR + 0x3A5C

Table 11-19. ATA Drive Device Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							Reserved								
W						SRST	nIEN									
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:4	—	Reserved
5	SRST	Software Reset—Host controlled software reset bit. Drive executes software reset protocol when bit is set to 1 by host.
6	nIEN	Interrupt Enable—Host controlled interrupt enable. INTRQ is enabled when this bit is cleared to 0. **Note: NOTE:** For MPC5200 ATA Host Controller, enabling INTRQ is mandatory for DMA/UDMA data transfer modes.
7:31	—	Reserved

11.3.3.2 ATA Drive Alternate Status Register—MBAR + 0x3A5C

Table 11-20. ATA Drive Alternate Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	DRDY	Reserved		DRQ	Rsvd		ERR	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	BSY	Drive Busy—Transactions internal to drive are in progress. Host must wait.
1	DRDY	Drive Ready
2:3	—	Reserved
4	DRQ	Set to 1 indicates drive is ready to transfer a word of data.
5:6	—	Reserved
7	ERR	Indicates an error during the execution of the previous command.
8:31	—	Reserved

11.3.3.3 ATA Drive Data Register—MBAR + 0x3A60

Table 11-21. ATA Drive Data Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data H								Data L							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Upper byte of drive data (read/write)
8:15	Data	Lower byte of drive data (read/write)
16:31	—	Reserved

11.3.3.4 ATA Drive Features Register—MBAR + 0x3A64

Table 11-22. ATA Drive Features Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									Reserved							
W	Data															
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Register content is command dependent. Contents become command parameters when the ATA drive command register is written.
8:31	—	Reserved

11.3.3.5 ATA Drive Error Register—MBAR + 0x3A64

Table 11-23. ATA Drive Error Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data					ABRT	Data		Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:4	Data	Register content is command dependent. Contents become command parameters when the ATA drive command register is written. Register content is valid when BSY and DRQ bits are set to 0 and ERR bit is set to 1 in the ATA drive status register. Register content is not valid when drive is in sleep mode.
5	ABRT	Bit is set to 1 to indicate requested command has been aborted, because command code or a command parameter is invalid or some other error occurred.
0:7	Data	Register content is command dependent. Contents become command parameters when the ATA drive command register is written. Register content is valid when BSY and DRQ bits are set to 0 and ERR bit is set to 1 in the ATA drive status register. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.6 ATA Drive Sector Count Register—MBAR + 0x3A68

Table 11-24. ATA Drive Sector Count Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Bit content is command dependent. For most read/write commands, this register indicates the total number of sectors requested for transfer. Register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.7 ATA Drive Sector Number Register—MBAR + 0x3A6C

Table 11-25. ATA Drive Sector Number Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Bit content is command dependent. For most commands, this register indicates the data transfer starting sector number for when CHS addressing is enabled. This register indicates part of the LBA address when the LBA addressing is enabled. Register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.8 ATA Drive Cylinder Low Register—MBAR + 0x3A70

Table 11-26. ATA Drive Cylinder Low Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Bit content is command dependent. For most commands, this register indicates the data transfer starting sector number for when CHS addressing is enabled. This register indicates part of the LBA address when the LBA addressing is enabled. Register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If this register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.9 ATA Drive Cylinder High Register—MBAR + 0x3A74

Table 11-27. ATA Drive Cylinder High Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data								Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Bit content is command dependent. For most commands, this register indicates the data transfer starting sector number for when CHS addressing is enabled. This register indicates part of the LBA address when the LBA addressing is enabled. This register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If this register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.10 ATA Drive Device/Head Register—MBAR + 0x3A78

Table 11-28. ATA Drive Device/Head Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	Data	Rsvd	DEV	Data				Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	—	Reserved
1	Data	Bit is command dependent. In LBA addressing mode, this bit is set to 1 to indicate LBA addressing is chosen for data transfer.
2	—	Reserved
3	—	Reserved
4:7	Data	Bit content is command dependent. For most commands, this register indicates the data transfer starting sector number for when CHS addressing is enabled. This register indicates part of the LBA address when the LBA addressing is enabled. This register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If this register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
8:31	—	Reserved

11.3.3.11 ATA Drive Device Command Register—MBAR + 0x3A7C

Table 11-29. ATA Drive Device Command Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									Rsvd	HUT	FR	FE	IE	UDMA	READ	WRITE
W	Data															
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Data	Register contains the command code sent to the drive. When this register is written, command execution begins immediately. Writing this register clears any pending interrupt condition.
8	—	Reserved
9	HUT	Host UDMA burst Terminate—Software can terminate UDMA burst prematurely by setting this bit. Bits 15 through 10 are unaffected and retain previous values.
10	FR	FIFO Reset—Hardware resets FIFO when the direction is switched from Tx to Rx. No hardware reset is done for Rx to Tx switch. Software must verify FIFO is empty before filling it for Tx. When bit 10 is set, FIFO is being reset and bits 15, 14, 13, 12, 11, 9 and 8 are invalid.
11	FE	Enable FIFO flush in Rx mode—For all commands except DEVICE RESET, this register is written only when the ATA drive status register bits BSY and DRQ equal 0 and DMACK is not asserted. If this register is written when BSY or DRQ bits are set to 1, the result is indeterminate except for the DEVICE RESET command. Register content is not valid when drive is in sleep mode.
12	IE	Enables drive interrupt to pass to CPU in DMA/UDMA modes. Software writes to this register as follows: • FE (bit 11) and IE (bit 12) • Clear IE and set FE if SDMA task loop count is the same as the data transfer requested from the drive. The following is a typical sequence if the BestComm task loop is a larger count than data request programmed for the drive: 1. Start transaction with IE set and FE cleared. 2. Repeat 1 until task loop count expires. 3. Start last transaction with IE clear and FE set. • Controller issues flush at end. • Task loop completes and interrupts CPU. • CPU responds to SDMA interrupt instead of drive interrupt. • UDMA (bit 13)—Set when UDMA protocol is selected for data transfer, cleared for DMA protocol. • READ (bit 14)—Set when read command for DMA/UDMA protocols is written to drive command register, cleared otherwise. • WRITE (bit 15)—Set when write command for DMA/UDMA protocols is written to drive command register, cleared otherwise. **MANDATORY—Be Aware:** Drive interrupt must be enabled by clearing bit 1 of drive control register for DMA/UDMA mode transfers.
13	UDAMA	Bit is set when UDMA protocol is selected, cleared when multiword DMA protocol is selected.
14	READ	Bit is set when READ DMA command is issued.
15	WRITE	Bit is set when WRITE DMA command is issued.
16:31	—	Reserved

11.3.3.12 ATA Drive Device Status Register—MBAR + 0x3A7C

Table 11-30. ATA Drive Device Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	DRDY	Data		DRQ	Reserved		ERR	Rsvd	HUT	FR	FE	IE	UDMA	Read	Write
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	BSY	Indicates drive is busy processing a command.
1	DRDY	Indicates drive is ready to accept executable commands.
2:3	Data	Command dependent—Register is written only when ATA drive status register bits BSY and DRQ equal 0 and $\overline{\text{DMACK}}$ is not asserted. If this register is written when BSY and DRQ bits are set to 1, the result is indeterminate. Register content is not valid when drive is in sleep mode.
4	DRQ	Indicates drive is ready to transfer a data word.
5:6	—	Reserved
7	ERR	Set to 1 indicates ATA drive error register bits are valid.
8	—	Reserved
9	HUT	Host UDMA burst Terminate—Software can terminate UDMA burst prematurely by setting this bit. Bits 15 through 10 are unaffected and retain previous values.
10	FR	FIFO Reset—Hardware resets FIFO when the direction is switched from Tx to Rx. No hardware reset is done for Rx to Tx switch. Software must verify FIFO is empty before filling it for Tx. When bit 10 is set, FIFO is being reset and bits 15, 14, 13, 12, 11, 9 and 8 are invalid.
11	FE	Enable FIFO flush in Rx mode—For all commands except DEVICE RESET, this register is written only when the ATA drive status register bits BSY and DRQ equal 0 and DMACK is not asserted. If this register is written when BSY or DRQ bits are set to 1, the result is indeterminate except for the DEVICE RESET command. Register content is not valid when drive is in sleep mode.

Bits	Name	Description
12	IE	Enables drive interrupt to pass to CPU in DMA/UDMA modes. Software writes to this register as follows: • FE (bit 11) and IE (bit 12) • Clear IE and set FE if SDMA task loop count is the same as the data transfer requested from the drive. The following is a typical sequence if the SDMA task loop is a larger count than data request programmed for the drive: 1. Start transaction with IE set and FE cleared. 2. Repeat 1 until task loop count expires. 3. Start last transaction with IE clear and FE set. • Controller issues flush at end. • Task loop completes and interrupts CPU. • CPU responds to BestComm interrupt instead of drive interrupt. • UDMA (bit 13)—Set when UDMA protocol is selected for data transfer, cleared for DMA protocol. • READ (bit 14)—Set when read command for DMA/UDMA protocols is written to drive command register, cleared otherwise. • WRITE (bit 15)—Set when write command for DMA/UDMA protocols is written to drive command register, cleared otherwise. **MANDATORY—Be Aware:** Drive interrupt must be enabled by clearing bit 1 of drive control register for DMA/UDMA mode transfers.
13	UDAMA	Bit is set when UDMA protocol is selected, cleared when multiword DMA protocol is selected.
14	READ	Bit is set when READ DMA command is issued.
15	WRITE	Bit is set when WRITE DMA command is issued.
16:31	—	Reserved

11.4 ATA Host Controller Operation

With the asynchronous ATA interface, an interface must be implemented that meets the timing specifications, given an input clock from the processor that is not fixed among all applications. The challenge is to meet the minimum ATA specifications while minimizing wasted time. Time is wasted because of differences between the minimum specification and the number of clock-cycles, multiplied by the clock-cycle period. This indicates the counter compare value depends on:

- the data transfer mode
- the clock frequency driving the ATA state machine (IPB clock)
- the minimum data transfer mode cycle-time passed in the INDENTIFY block from the drive to the ATA Host Controller

Software requirements for setting up the Host Controller are as follows:

1. Write into ata_config register to enable (ata_config[7] == 1) support for IORDY for PIO modes 3 and 4.
2. Software determines ATA mode timing based on the operating clock frequency

$$\text{Count} = \frac{\text{ATA\_mode\_timing\_spec} + \text{ipbi\_clock\_period} - 1}{\text{clock\_period}}$$

This rounds up to the smallest integer number of clock counts that meet the minimum specification.

In the case of counters that control duration of a read strobe (pio_t2_8, pio_t2_16 and dma_td), the added transceiver propagation delay must be taken into account so the read data meets setup time to the rising edge of the strobe. Therefore:

$$\text{Count} = \frac{\text{ATA\_mode\_timing\_spec} + 2 \times \text{XCVR\_PROP\_DLY} + \text{clock\_period} - 1}{\text{clock\_period}}$$

udma_t2cyc is another special case. Unlike the name implies, this register does not control 2 UDMA timing cycles. Rather, it controls how long the host continues to accept data after it has de-asserted HDMARDY–. According to the ATA-4 specification—if tSR is met, the host should accept 0–1 more data words, or if tSR is exceeded, 0–2 more data words. A safe value to ensure the host accepts these data words after HDMARDY– de-asserts is:

$$\text{Count} = \frac{4 + \text{t2CYC\_sec[mode]} + \text{clock\_period} - 1}{\text{clock\_period}}$$

1. Write the calculated count in the timing registers provided in the ATA host register memory map.
2. Write ATA drive registers per ATA-4 specification using Host Controller register memory map to the setup drive for desired operation.
3. Read/Write to unimplemented registers or read of a write-only or vice versa errors set flag bits in the ATA Host Controller status register. The status register is cleared by writing 1 to the flag bit set to indicate an error.
4. Write ata_dma_mode register to indicate UDMA/DMA READ/WRITE operations for UDMA/DMA data transfer modes.
5. Initiate and complete data transfers according to protocols described in ATA-4 specification.

ATA host hardware does data transfers per chosen protocol. Hardware also maintains proper handshaking with the MPC5200 system.

The ATA state machine is a combination of several small state machines. The data transfers is initiated by the software. The software chooses the mode of operation and sets up needed registers in the ATA Host Controller IPBI module.

The ATA drive registers are also set up by the software through ATA IPBI module using PIO mode. The ATA drive command and control block registers are mapped into ATA Host Controller register memory map.

The software writes a command to be executed in the ATA drive command register. The command code is decoded by the drive electronics. The software, at the same time indicates to the host if UDMA/DMA protocol is used for READ/WRITE of the data. This is done by setting proper bits in the ata_dma_mode register in the ATA IPBI module.

11.4.1 PIO State Machine

In the ATA-4 spec, 16 timing characteristics must be met for a PIO data or register access:

- 9 are driven by the ATA drive controller—2 (t1 and ta) are counted by the Host Controller for checking/latching purposes.
- 7 are driven by the ATA Host Controller

To simplify Host Controller design, the following implementation is used:

- Counter—The counter used to count this timing spec (pio_<name>_counter). All non-zero counters count down from an initial value to 1 (end)
- Start from—Where this counter is initialized.
- Activity at end—What activity to perform when counter reaches 1
- Dependencies—When counter reaches 0, what signals must be checked before counter is finished (cleared to 0)

Table 11-31. PIO Timing Requirements

Counter	Start from	Activity at end	Dependencies
t0	t1	go to IDLE	t2=0, t2i=0, t4=0
t11	N/A (Use t1 instead)	—	—
t2	t1	Latch Read_Data	IORDY_reg=1
t2i	t2	—	—
t32	N/A (Use t2 instead)	—	—
t4	t3	write_enable=0	—
		address_enable=0	—
t5	N/A (Timing controlled by drive controller)	—	—
t6	N/A (Timing controlled by drive controller)	—	—
t6z	N/A (Timing controlled by drive controller)	—	—

Table 11-31. PIO Timing Requirements (continued)

Counter	Start from	Activity at end	Dependencies
t93	N/A (Use t4 instead)	—	—
tA	t1	Check IORDY	IORDY=1
tB	N/A (Timing controlled by drive controller)	—	—
tC	N/A (Timing controlled by drive controller)	—	—

Note:

1. Since t1 and t1 are both minimum specs, and t1 <= t1 for PIO modes 0–2, and t1 >= t1 for PIO modes 3–4, t1 is used to count both, by loading in an initial value that depends on the PIO mode being used. This is the responsibility of software.
2. Since t3 (WDATA setup time) is a minimum, and t3 <= t2 for all PIO modes, t2 is used to determine when to drive Write_Data on DD.
3. Since t4 and t9 are both minimum specs, and t4 >= t9 for all PIO modes, t4 is used to count from DIOR/DIOW negate to $\overline{CS}$[1]FX/$\overline{CS}$[3]FX/ADDR negate.

If ATA drive address space is hit by microprocessor, the ATA IPBI module generates:

- a signal to enable the PIO mode state machine
- a wait state to the IPBI module to hold off any further IPBI module access

The PIO state machine indicates transfer is in progress to the IPBI module. This extends the transfer wait to the IPBI module until the PIO transaction is complete.

11.4.2 DMA State Machine

The interface between the ATA Controller DMA channel and the rest of the system is through a standard Type 1 BestComm FIFO interface. When this interface is fully defined, the design specifics may be detailed. Table 11-32 shows the timing requirements specified in the ATA-4 spec for multiword DMA data transfers.

Table 11-32. Multiword DMA Timing Requirements

Counter	Start from	Activity at end	Dependencies
TM	START (Negate CS0, CS1, set DMA_In_Progress flag)	Assert DMACK, Assert DIOR/DIOW, Write Data ready	DMARQ asserted by drive
TE	N/A (Timing controlled by drive controller)	—	—
TD	TM	Negate DIOR/DIOW, Latch Read Data/Drive Write Data	DMARQ=1
TK	TD	Assert DIOR/DIOW	DMARQ=1
TH	TD	Ready for new write data	DMARQ=1
T0	TD	Begin next cycle	DMARQ=1
		Start TJ, Start TN	DMARQ=0
TJ	T0	Negate DMACK, Go to Idle	DMARQ Negated, DMACK asserted, T0=0
TN	T0	Clear DMA_In_Progress flag. Allow CS0, CS1 to be driven	DMARQ Negated, DMACK asserted, T0=0

11.4.2.1 Software Requirements

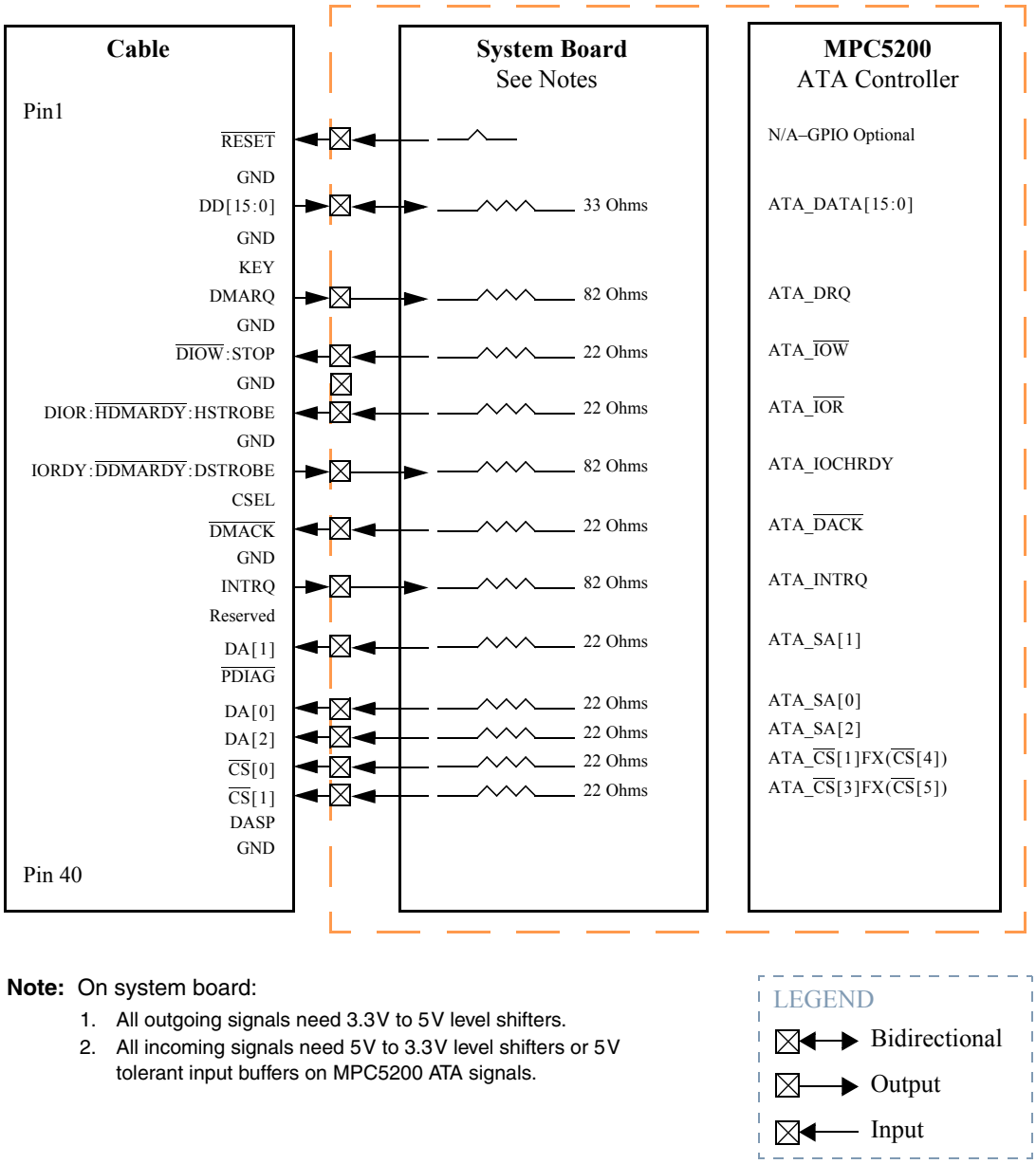
Software calculates the appropriate values of TD and TK based on information reported for the cycle time (T0) in the drive's IDENTIFY DEVICE data and the operating clock frequency. Cycle time (T0) must be greater than the sum of TD and TK.

11.5 Signals and Connections

Table 11-33. MPC5200 External Signals

Signal	I/O	Description
DATA[15:0]	I/O	Data—16-bit Data Bus (DD pins on ATA cable).
SA[2:0]	O	Address—3-bit address, when combined with the two chip-selects, CS1FX and CS3FX, is used to address Control and Command Block Registers in an ATA drive controller (DA2, DA1 and DA0 on ATA cable, respectively).
$\overline{CS}$[1]FX	O	Chip select connected to $\overline{CS}$[0] on ATA cable.
$\overline{CS}$[3]FX	O	Chip select connected to $\overline{CS}$[1] on ATA cable.
Signal	**I/O**	**Description**
IOW	O	I/O Write—Active low signal that denotes a WRITE transaction (DIOW on ATA cable).
IOR	O	I/O Read—Active low signal that denotes a READ transaction (DIOR on ATA cable).
DACK	O	DMA Acknowledge (DMACK on ATA cable).
INTRQ	O	ATA interrupt.
ATA_ISOLATION	O	ATA Write Enable to allow sharing of the ATA DD bus with PCI Bus.
IOCHRDY	I	I/O Channel Ready (IORDY pin on ATA cable)
DRQ	I	DMA Request (DMARQ pin on ATA cable)
RESET	NC1	Reset—Handled at the board level

Note:
1. NC=No Connection



Note: On system board:

1. All outgoing signals need 3.3V to 5V level shifters.
2. All incoming signals need 5V to 3.3V level shifters or 5V tolerant input buffers on MPC5200 ATA signals.

Figure 11-2. Connections—Controller Cable, System Board, MPC5200

11.6 ATA Interface Description

Table 11-34. ATA Controller External Connections

Pin#	Cable	I/O	System Board	I/O	MPC5200
1	RESET	O	RESET:Reset	—	N/A—GPIO optional
2	GND	—	—	—	—
3–18	DD[15:0] 3,5,7,9,11,13,15,17→DD[7:0] 18,16,14,12,10,8,6,4→DD[15:8]	I/O	DD[0:15]	I/O	ATA_DATA[15:0]
19	GND	—	—	—	—

Table 11-34. ATA Controller External Connections (continued)

Pin#	Cable	I/O	System Board	I/O	MPC5200
20	KEY	—	No Signal:Alignment key	—	—
2	DMARQ	I	DMARQ:DMA Request	I	ATA_DRQ
22	GND	—	—	—	—
23	$\overline{\text{DIOW}}$:STOP	O	DIOW	O	ATA_$\overline{\text{IOW}}$
24	GND	—	—	—	—
25	$\overline{\text{DIOR}}$:$\overline{\text{HDMARDY}}$:HSTROBE	O	DIOR	O	ATA_$\overline{\text{IOR}}$
26	GND	—	—	—	—
27	IORDY:$\overline{\text{DDMARDY}}$:DSTROBE	I	IORDY	I	ATA_IOCHRDY
28	CSEL	—	NC	—	—
29	DMACK	O	DMACK	O	ATA_$\overline{\text{DACK}}$
30	GND	—	—	—	—
31	INTRQ	I	INTRQ	I	ATA_INTRQ
32	Reserved	—	—	—	—
33	DA[1]	O	DA[1]:Address Bus Bit1	O	ATA_SA[1]
34	PDIAG	—	NC	—	—
35	DA[0]	O	DA[0]:Address Bus Bit0	O	ATA_SA[0]
36	DA[2]	O	DA[2]:Address Bus Bit2	O	ATA_SA[2]
37	CS[0]	O	$\overline{\text{CS}}$[1]$\overline{\text{FX}}$:Chip Select 0	O	ATA_$\overline{\text{CS}}$[1]$\overline{\text{FX}}$($\overline{\text{CS}}$[4])
38	CS[1]	O	$\overline{\text{CS}}$[3]$\overline{\text{FX}}$:Chip Select 1	O	ATA_$\overline{\text{CS}}$[3]$\overline{\text{FX}}$($\overline{\text{CS}}$[5])
39	DASP	—	NC	—	—
40	GND	—	—	—	—

NOTE

MPC5200 provides the ATA_ISOLATION output signal. This signal is shared with the A22 output of the LocalPlus Most/Graphics mode.

The ATA_ISOLATION is not a signal defined by the ATA Standard. It is provided to support an external ATA transceiver. ATA_ISOLATION is an active high signal to control external transceiver devices and to ‘isolate’ the ATA bus from the LocalPlus (shared) bus.

It can force the transceiver direction "MPC5200 -> disk drive". Only during an ATA read is this signal allowed to go low, forcing tranceiver direction "disk drive ->MPC5200".

The ATA_ISOLATION should be connected to the Direction input of the transceiver.

- High = Write to drive
- Low = Read from drive

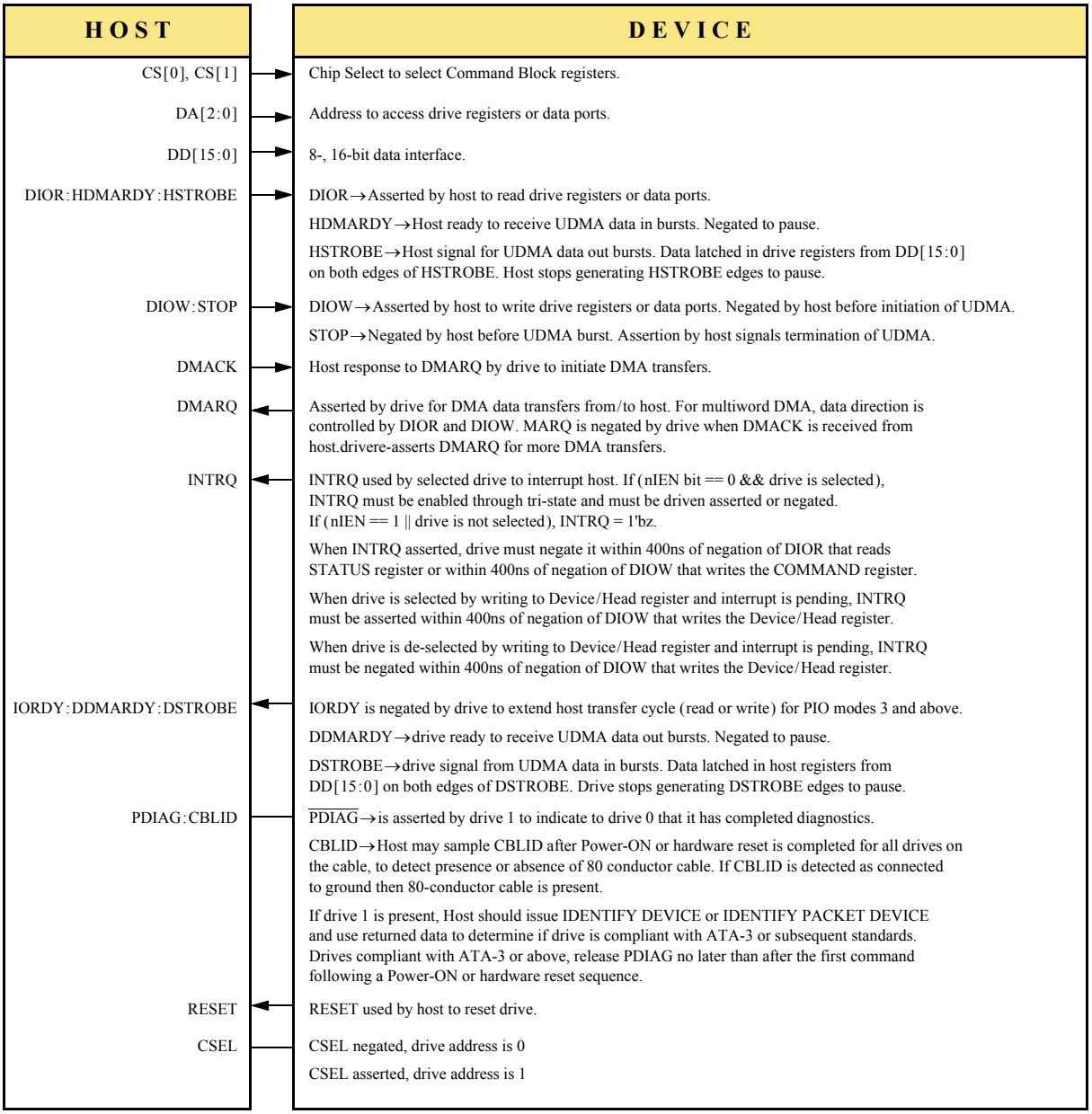


Figure 11-3. Pin Description—ATA Interface

11.7 ATA Bus Background

11.7.1 Terminology

The most popular interface used in modern hard disks is the Integrated Drive Electronics (IDE) interface, also known by various other names such as: ATA, EIDE, ATA-2, Fast ATA, Ultra ATA, etc.

- Western Digital® used the term IDE when they first integrated the drive controller logic board on the disk drive.
- Quantum® and Seagate® used the term ATA (Advanced Technology Attachment) or AT-Attachment, because it has a 16-bit data interface like original AT machines.

ATA is the interface name adopted by the American National Standards Institute (ANSI). Thus far, ANSI has published ATA, ATA-2, ATA-3 and ATA-4 interfaces. More work is underway for ATA-5 and future extensions of the ATA interface. Table 11-35 summarizes the different ATA standards.

MPC5200 is compliant with the latest officially published ANSI ATA-4 interface.

Table 11-35. ATA Standards

Interface Standard	Standard Type	PIO Modes	DMA Modes	Special Features or Enhancements introduced Relative to IDE/ATA
IDE/ATA	ANSI	0,1,2	Single word—0,1,2 Multiword—0	—
ATA-2	ANSI	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2	Block transfers, logical block addressing, improved identify drive command
FAST ATA	Marketing	0,1,2,3	Single word—0,1,2 Multiword 0,1	Same as ATA-2
Fast ATA-2	Marketing	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2	Same as ATA-2
ATA-3	Unofficial	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2	Same as ATA-2, plus improved reliability, SMART
Ultra ATA	Unofficial	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2,3	Same as ATA-3
ATAPI	ANSI	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2	Support for non-hard-disk devices CD-ROM, Tape drives, etc.
EIDE	Marketing	0,1,2,3,4	Single word—0,1,2 Multiword—0,1,2	Same as ATA-2, plus ATAPI and dual host adapters
ATA-4	ANSI	0,1,2,3,4	Multiword—0,1,2 Ultra DMA—0,1,2	Same as ATA-3, Single word DMA retired

11.7.2 ATA Modes

Table 11-36. ATA Physical Level Modes

Mode	Cycle Time (ns)	Transfer Rate (MB/s)	Standard
PIO mode 0	600	3.3	ATA
PIO mode 1	383	5.2	ATA
PIO mode 2	240	8.3	ATA
PIO mode 3	180	11.1	ATA-2 (IORDY required)
PIO mode 4	120	16.7	ATA-2 (IORDY required)
DMA mode 0 (Multiword)	480	4.2	ATA
DMA mode 1 (Multiword)	150	13.3	ATA-2
DMA mode 2 (Multiword)	120	16.7	ATA-2
Ultra DMA mode 0	114	16.7	ATA-4
Ultra DMA mode 1	75	25	ATA-4
Ultra DMA mode 2	55	33	ATA-4

11.7.3 ATA Addressing

In the ATA interface, there are two aspects of addressing that are present: register addressing and sector addressing. These are discussed in the next sections.

11.7.3.1 ATA Register Addressing

The address used to reference an ATA drive register. This is the actual address ($\overline{CS}[1]\overline{FX}$, $\overline{CS}[3]\overline{FX}$, DA[2:0]) present on the physical ATA interface. Table 11-37 gives details.

Table 11-37. ATA Register Address/Chip Select Decoding

Address						Function	
System Address	CS[1]FX	CS[3]FX	DA[2]	DA[1]	DA[0]	READ ($\overline{DIOR}$)	WRITE ($\overline{DIOW}$)
						Control Block Registers	
—	1	1	x	x	x	Data bus high impedance	Not used
03F0–03F3	1	0	0	x	x	Data bus high impedance	Not used
03F4–03F5	1	0	1	0	x	Data bus high impedance	Not used
03F6	1	0	1	1	0	Alternate status	Device control
03F7	1	0	1	1	1	Obsolete	Not used
						Command Block Registers	
01F0	0	1	0	0	0	Data	Data
01F1	0	1	0	0	1	Error register	Features
01F2	0	1	0	1	0	Sector count	Sector count
01F3	0	1	0	1	1	Sector number	Sector number
01F3	0	1	0	1	1	LBA bits 0–7[1]	LBA bits 0–7[1]
01F4	0	1	1	0	0	Cylinder low	Cylinder low
01F4	0	1	1	0	0	LBA bits 8–15[1]	LBA bits 8–15[1]
01F5	0	1	1	0	1	Cylinder high	Cylinder high
01F5	0	1	1	0	1	LBA bits 16–23[1]	LBA bits 16–23[1]
01F6	0	1	1	1	0	Drive/head	Drive/head
01F6	0	1	1	1	0	LBA bits 24–27[1]	LBA bits 24–27[1]
01F7	0	1	1	1	1	Status	Command
—	0	0	x	x	x	Invalid address	Invalid address

Note:
1. LBA mode register mapping—system addresses are for a single channel, accommodating two drives only.

11.7.3.2 Drive Interrupt

A pending drive interrupt is cleared by the following actions:

- Read of status (not the alternate status) register
- Write to command register

11.7.3.3 Sector Addressing

Sector addressing is the address used to reference data on the drive. It is the address used by the low-level drivers to access a particular piece of data and to place it into one or more ATA registers as part of a command block. To understand the data addressing, it is necessary to understand the physical organization of data in a drive, as presented in Figure 11-1. Each drive contains a number of disks, each with one or two heads (one head per surface). Each disk is divided into concentric tracks that are then divided into a number of sectors. A sector is the smallest unit of data that can be written or read by a drive. The collections of tracks that can be accessed by the heads at a single position is called a cylinder. Therefore, a sector can be uniquely identified by a sector number, a head number and a cylinder number. From this addressing scheme there are two ways to address an individual sector: physical addressing and logical block addressing, which are described in the next two sections.

Notes

1. LBA mode is only available in ATA-2 or later specifications.
2. A block mode exists (not to be confused with logical block addressing), in which sectors are grouped into a unit, called a block, for purposes of data transfer. The number of sectors is set with SET MULTIPLE MODE command and is used by the READ MULTIPLE and WRITE MULTIPLE commands. When specifying sectors within a block, either CHS or LBA mode may be used.

11.7.3.4 Physical/Logical Addressing Modes

Addressing is done by referencing the sector, head and cylinder for a particular sector. Using a physical addressing mode, there are two mappings available:

- Natural—Sector, head and cylinder numbers represent actual physical sectors, heads and cylinders on the drive.
- Logical—Sector, head and cylinder numbers map to different physical sectors, heads and cylinders on the drive.

Most modern hard disks usually have 2, 3 or 4 platters. All platters are connected together on a common spindle to spin as a single assembly. Each platter has two surfaces and two heads to access each surface. The platter is a collection of concentric circles called tracks, to store data. Each track is subdivided into sectors. Each sector can hold 540 Bytes of information, with 512 Bytes being used for data and 28 Bytes being used for error correction code (ECC). A set of tracks under each head at the same track position is called a cylinder. So to get to the disk read/write data point, a cylinder address, a head address and a sector address is needed. Hence the basic addressing mode is called cylinder head sector (CHS) addressing.

In this mode, the address is written into the ATA registers as follows:

- Cylinder→{Cylinder High (0x01F5), Cylinder Low (0x01F4)}
- Head→Drive/Head (0x01F6)
- Sector→Sector Number (0x01F3)

To most efficiently use the drive for data storage, the physical geometry is translated into logical geometry by the hard disk manufacturers. The BIOS or overlay software from the disk manufacturer translates the logical geometry to physical geometry to get to the physical location of the data written/read on/from the disk.

The CHS method is limited to 1024 cylinders, 16 heads and 63 sectors. This limits the hard disk recognition to a maximum of 504 MBytes. This limit is increased for larger disks by enhancing the CHS translation. BIOS limits cylinder size to 1024 (10 bits allocated), but allows the number of heads to be 256 (8 bits allocated). Therefore, a 3.1 GByte hard disk with 6136 cylinders and 16 heads is translated by dividing the cylinders by 8 (6136 ÷ 8 = 767). The number of heads is then multiplied by the same number (16 x 8 = 128). This fits well within the limits set by the BIOS and a larger disk is recognized for its true size (767 x 128 x 63 x 512 = 3.1 GBytes).

Another form of addressing is called logical block addressing (LBA). This uses 28 bits in the ATA standard to address a particular sector on a hard disk. A sum total of sectors on a drive is available and each unique sector is addressed using LBA.

Mapping from physical organization to logical block numbers is done using the following formula:

$$LBA \rightarrow (Cylinder\# \times HeadCount + Head\#) \times SectorCount + Sector\# - 1$$

In this mode, the address is written in the ATA Registers as follows:

LBA→{LBA[0:7](0x01F3), LBA[8:15](0x01F4), LBA[16:23](0x01F5), LBA[24:27](0x01F6)}

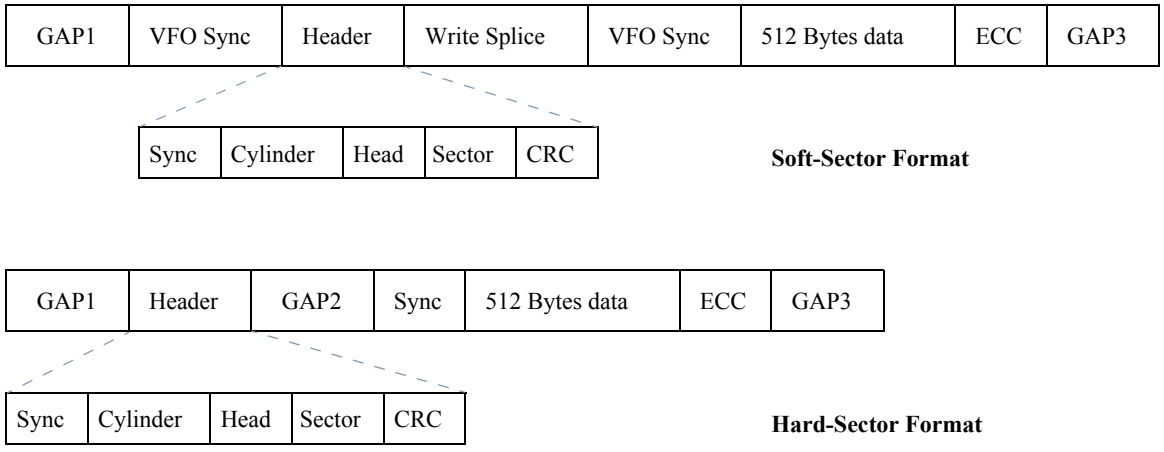


Figure 11-4. ATA Sector Format

11.7.4 ATA Transactions

ATA Transactions are divided into three types:

- PIO Mode
- Multiword DMA
- Ultra DMA

11.7.4.1 PIO Mode Transactions

PIO mode transactions are the simplest transaction available on the ATA interface. They essentially consist of single word accesses across the ATA interface. There are currently 6 PIO modes available, which are summarized in Table 11-36. Timing and sequence information are given in the MPC5200 datasheet.

Three classes of ATA commands use PIO Mode:

- Class 1—PIO Read
- Class 2—PIO Write
- Class Non-Data Command

11.7.4.1.1 Class 1—PIO Read

Figure 11-5 shows the PIO Read process.

— PIO Single sector read [identify drive, read buffer, read sector(s)]

— Interrupt is generated after each sector is read into the sector buffer:

1. HOST: Write to ATA control/command block registers to setup for data read.
2. HOST: Write to ATA command register to execute read command.
3. HOST: Poll drive to see if it is ready.
4. DRIVE: Read sector from physical medium to sector buffer.
5. DRIVE: Interrupt HOST when done.
6. HOST: Read ATA control/command block registers to get status
7. DRIVE: Clear interrupt after reading status register.
8. HOST: Read ATA data register 256 times to get all 512 Bytes from sector buffer.
9. Repeat steps 4–8 for multiple sectors.

— PIO Block mode read [read multiple]

— Interrupt is generated after each block is read into sector buffer:

1. HOST: Write to ATA control/command block registers to setup for data read.
2. HOST: Write to ATA command register to execute read command.
3. HOST: Poll drive to see if it is ready.
4. DRIVE: Read block of sectors from physical medium to sector buffer.
5. DRIVE: Interrupt HOST when done.
6. HOST: Read ATA control/command block registers to get status.
7. DRIVE: Clear interrupt after reading status register.
8. HOST: Read ATA data register to get all sectors from sector buffer.

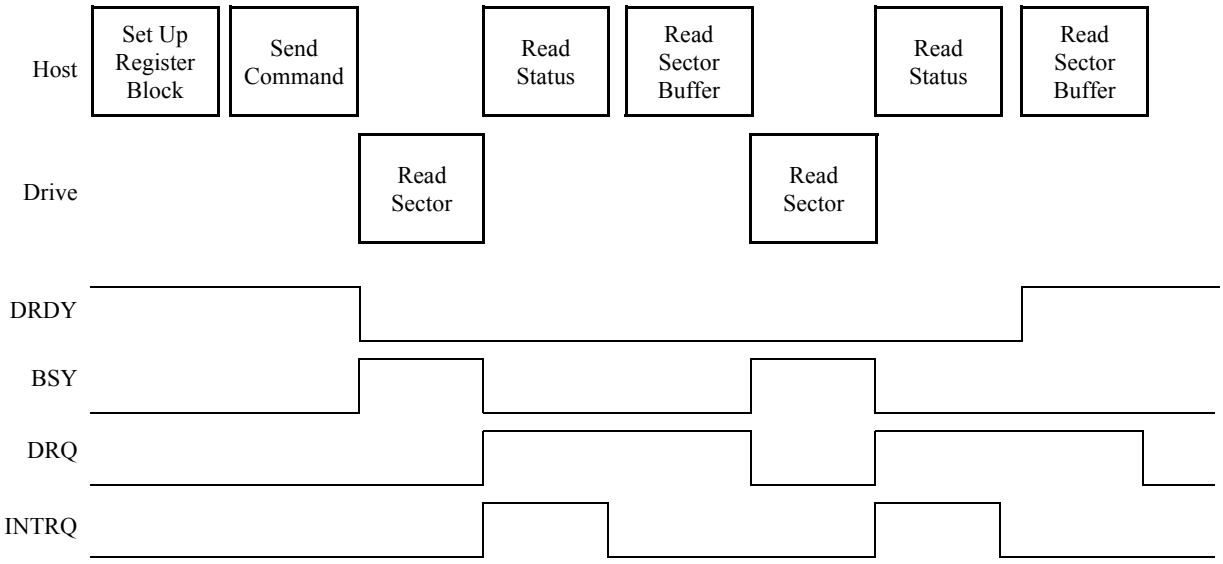


Figure 11-5. Timing Diagram—PIO Read Command (Class 1)

11.7.4.1.2 Class 2—PIO Write

The PIO single sector write command [format, write buffer, write sector(s)] is as follows:

1. HOST: Write to ATA control/command block registers to setup for data write.
2. HOST: Write to ATA command register to execute write command.
3. HOST: Poll drive to see if it is ready.
4. HOST: Write ATA data register 256 times to get all 512 Bytes into sector buffer.
5. DRIVE: When sector buffer is filled, write sector to physical medium.
6. DRIVE: Interrupt HOST when done.
7. HOST: Read ATA control/command block registers to get status.
8. DRIVE: Clear interrupt after reading status register.
9. Repeat steps 4–8 for multiple sector writes.

The PIO block mode write command (write multiple) is as follows:

1. HOST: Write to ATA control/command block registers to set up for data write.
2. HOST: Write to ATA command register to execute write command.
3. HOST: Poll drive to see if it is ready.
4. HOST: Write ATA data register 256 times to get all sectors into sector buffer.
5. DRIVE: When sector buffer is filled, write sector to physical medium.
6. DRIVE: Interrupt HOST when done.
7. HOST: Read ATA control/command block registers to get status.
8. DRIVE: Clear interrupt after reading status register.

Figure 11-6 shows the PIO Write process.

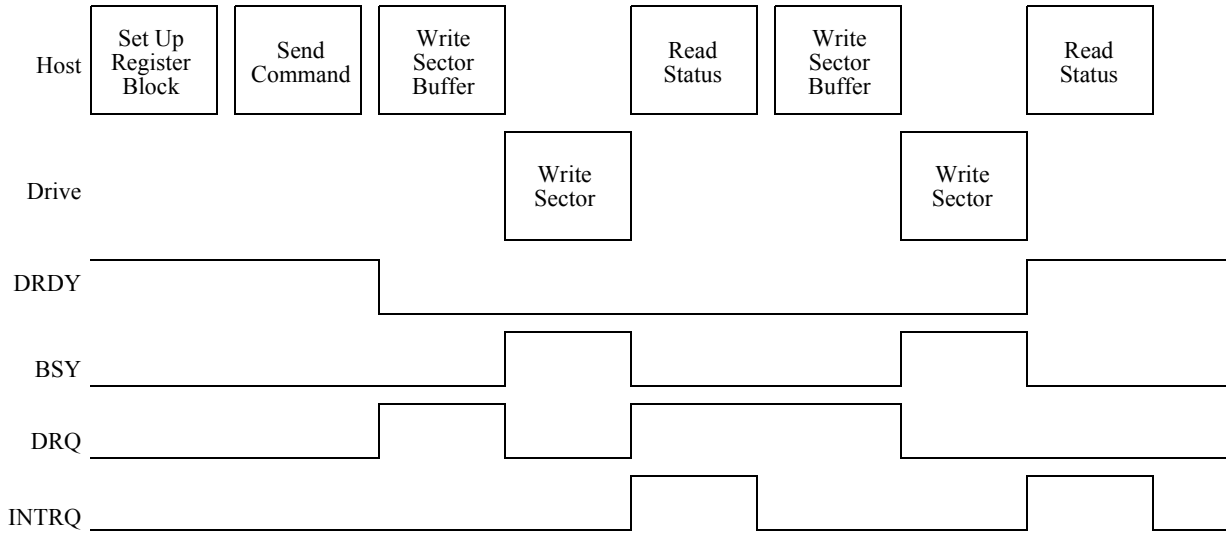


Figure 11-6. Timing Diagram—PIO Write Command (Class 2)

11.7.4.1.3 Class 3—Non-Data Command

The Non-Data Command is as follows:

1. HOST: Write to ATA control/command block registers to setup for data read.
2. HOST: Write to ATA command register to execute read command.
3. DRIVE: Execute command.

Figure 11-7 shows the Non-Data Command.

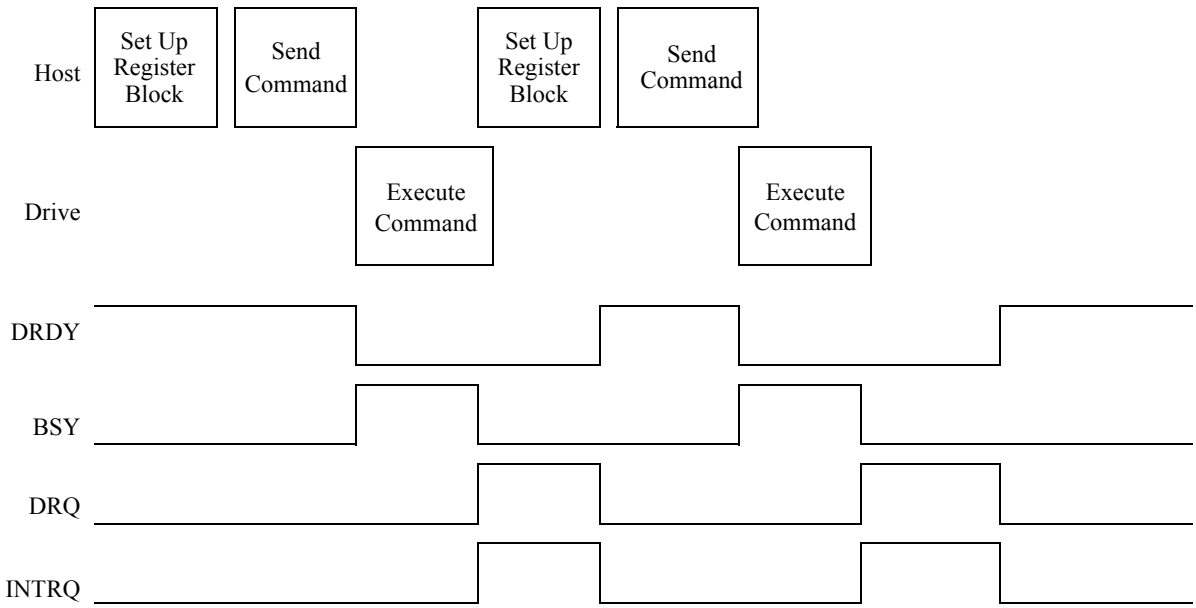


Figure 11-7. Timing Diagram—Non-Data Command (Class 3)

11.7.4.2 DMA Protocol

The DMA protocol has the following commands:

- READ DMA
- WRITE DMA

The Host selects the multiword DMA protocol as follows:

1. Write 00100b to upper 5 bits ([7:3]) of sector count register to select multiword DMA protocol. Write desired mode value to lower 3 bits ([2:0]) of sector count register to set multiword DMA transfer mode (mode 0=000b, mode 1=001b, etc.).
2. Write sub-command code 0x03 to features register to set transfer mode, based on value in sector count register.

3. Write command code 0xEF to command register to execute SET FEATURES command. This sets the data transfer protocol to multiword DMA with desired mode.

Data transfers into DMA differ from a PIO transfer in that:

- Data is transferred using the DMA channel.
- A single interrupt is issued at command completion.

The Host initializes the DMA channel prior to issuing DMA mode commands. The drive asserts an interrupt when data transfer is complete. The DMA command protocol is as follows:

1. HOST: Read status or alternate status register until BSY and DRQ are both 0. (ATA-4, 41, 48).
2. HOST: Write device/head register with appropriate DEV bit value to select drive. (ATA-4, 45).
3. HOST: Wait 400 ns, read status or alternate status register until BSY & DRQ are set to 0. The required drive is then assured to be selected.
4. HOST: Write required command parameters to the features, sector count, sector number, cylinder high, cylinder low, and device/head registers. (ATA-4, chapter 7).
5. HOST: Write command code to command register for drive to start processing command using parameters from the command block registers. (ATA-4, 41).
6. DRIVE: If no drive error exists, set BSY=1 and begin processing command.
7. HOST: Wait 400ns, read status or alternate status register to ensure valid contents.
8. DRIVE: Set BSY=1 or BSY=0 && DRQ=1.
9. DRIVE: Assert DMARQ when ready, transfer data per multiword DMA timing or ultra DMA protocol.
10. HOST: Assert DMACK, negate $\overline{CS}[0]$ and $\overline{CS}[1]$ when ready to transfer data per multiword DMA timing or ultra DMA protocol. Transfers are 16-bit wide from the data port. DMA data out (drive→host) transfers are processed by a series of reads to the data port. Each read transfers the data that follows the previous read. DMA in data (host→drive) transfers are processed by a series of writes to this port. Each write transfers the data that follows the previous write. Results are indeterminate if data port is written during a DMA data out or data port is read during a DMA data in transfers.
11. DRIVE: Negate DMARQ when transfer is complete.
12. DRIVE: Set error status in error register if error exists.
13. DRIVE: Clear BSY and DRQ.
14. DRIVE: Assert INTRQ if Host has enabled nIEN (set to 0) in command control register. This register is written by the host to enable interrupt from the drive by clearing nIEN bit to 0. INTRQ is in a high impedance state if nIEN bit is set to 1.

When host sets command control register bit SRST to 1, software can reset selected drive. However, the command control register must be written while DMACK is not asserted. Bit 0 must be cleared to 0.

1. HOST: To clear pending interrupt, read status register (regardless of nIEN status).
2. DRIVE: If enabled by nIEN (nIEN = 0), negate INTRQ.
3. DMA command completes.

Table 11-38. DMA Command Parameters

DMA Command	Command Code	Parameters Used (Registers)				
		Features	Sector Count	Sector Number/LBA	Cylinder HI/LO/LBA	Device/Head/LBA
READ DMA	C8h	Yes	Yes	Yes	Yes	D/H Both
WRITE DMA	CAh	Yes	Yes	Yes	Yes	D/H Both

Figure 11-8 shows the DMA command protocol flow diagram.

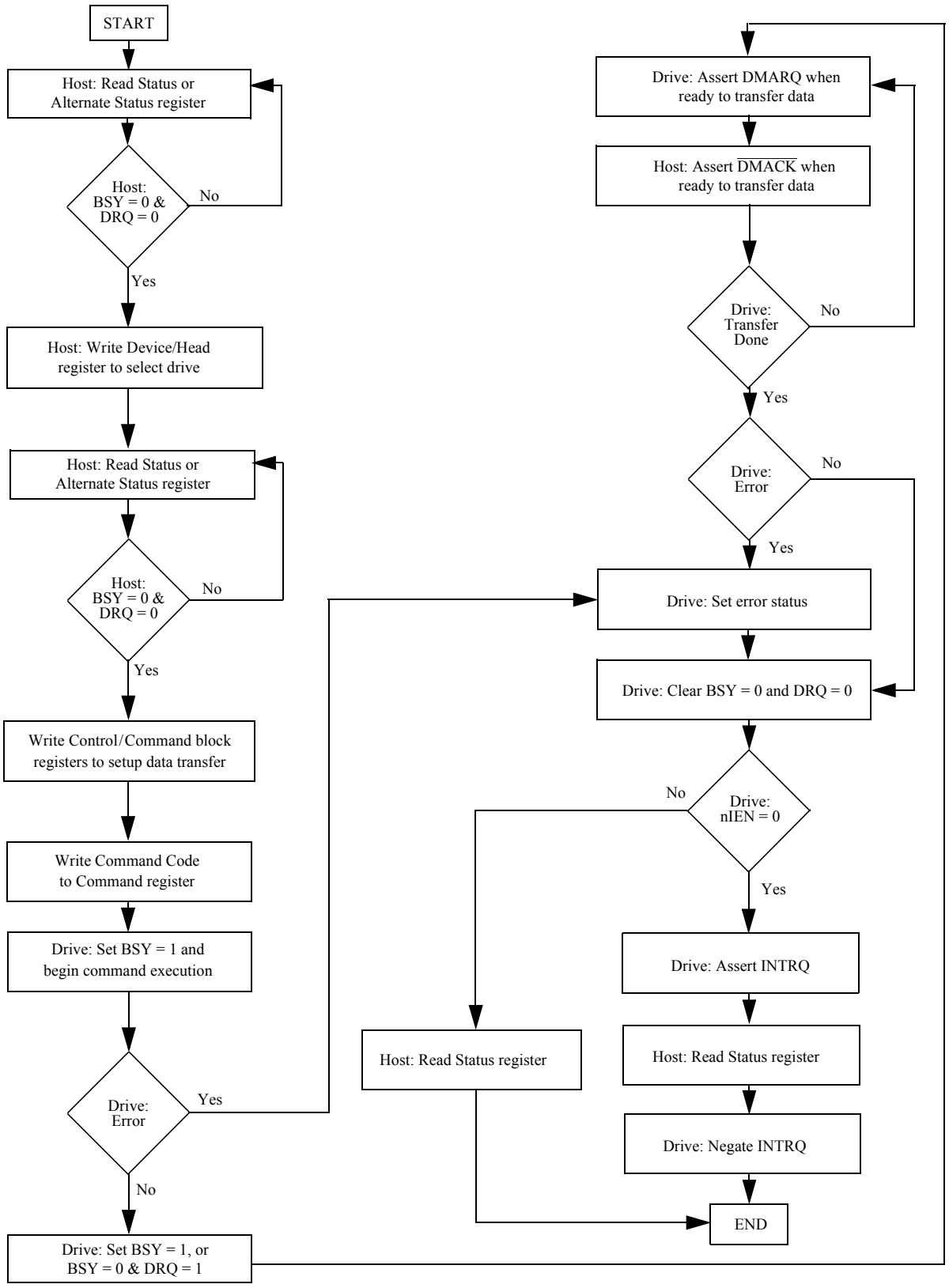


Figure 11-8. Flow Diagram—DMA Command Protocol

11.7.4.3 Multiword DMA Transactions

Multiword DMA transactions differ from PIO mode transactions in three ways:

1. Data transfers are done using a drive DMA and a host DMA (optional).
2. Handshaking is done with DMARQ and DMACK, no address is necessary.
3. Interrupts do not occur after every sector for multi-sector transfers

11.7.4.3.1 Class 4—DMA Command

Figure 11-9 shows the DMA timing diagram. The DMA command (Read DMA, Write DMA) is as follows:

1. HOST: Set up HOST DMA (in ATA Host Controller or system DMA).
2. HOST: Write to ATA control/command block registers to setup drive DMA.
3. HOST: Write to ATA control/command block registers to set up data read/write.
4. HOST: Write to ATA command register to execute the read/write command.
5. DRIVE: Assert DMARQ.
6. HOST: When DMARQ is asserted, assert DMACK.
7. DRIVE: Read sector from physical medium to sector buffer.
8. DRIVE: Transfer data to HOST using DMA handshaking.
9. Repeat steps 7–8 as needed for multiple sectors.
10. DRIVE: De-assert DMARQ.
11. HOST: De-assert DMACK.
12. DRIVE: Interrupt HOST.
13. HOST: Stop HOST DMA.
14. HOST: Read ATA control/command block registers to get status.
15. DRIVE: Clear interrupt after reading status register.

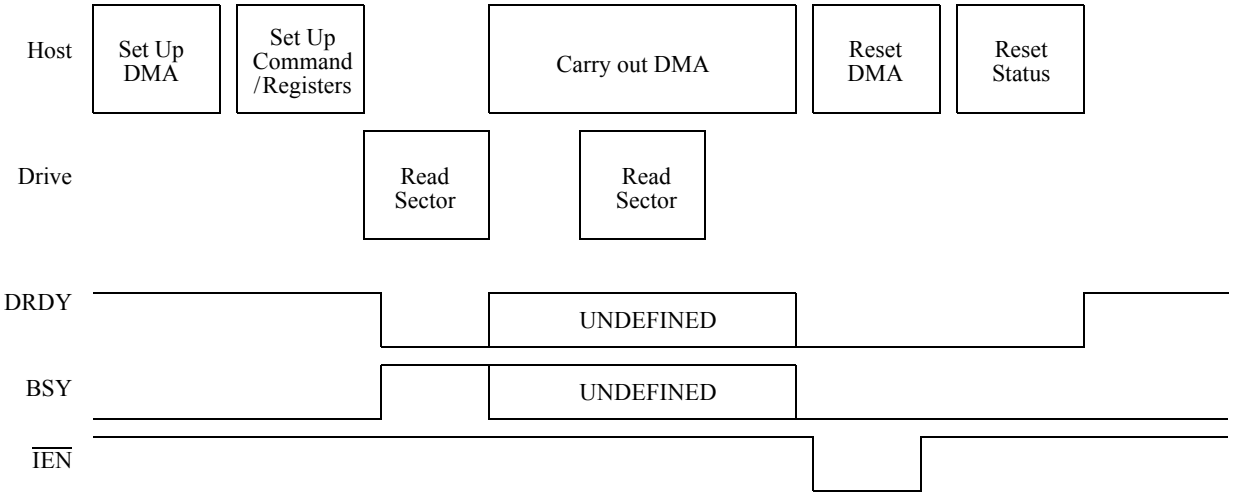


Figure 11-9. Timing Diagram—DMA Command (Class 4)

11.7.4.4 Ultra DMA Protocol

The Ultra DMA protocol has the following commands:

- READ DMA
- WRITE DMA

The host selects the Ultra DMA protocol as follows:

- Write 01000b to upper 5 bits ([7:3]) of sector count register to select ultra DMA protocol. Write desired mode value to lower 3 bits ([2:0]) of sector count register to set ultra DMA transfer mode (mode 0=000b, mode 1=001b, etc.).
- Write sub-command code 03h to features register to set transfer mode based on value in sector count register.
- Write command code EFh to command register to execute SET FEATURES command, which sets the data transfer protocol to ultra DMA with desired mode.

When enabled, the ultra DMA protocol is used instead of the multiword DMA protocol.

NOTE

Ultra DMA mode 2 (UDMA2) requires that the ipb_clk speed is at least 66 MHz.

Table 11-39 lists the redefined ultra DMA protocol signal lines. These lines provide new functions during the ultra DMA mode. At termination of an ultra DMA burst, the host negates DMACK and the lines revert to the definitions used for non-ultra DMA transfers.

Table 11-39. Redefinition of Signal Lines for Ultra DMA Protocol

Non-Ultra DMA modes	Ultra DMA Modes	Description
DIOR	HDMARDY	Host DMA ready during Ultra DMA data in bursts
	HSTROBE	Host data strobe during Ultra DMA data out bursts
IORDY	DDMARDY	Drive DMA ready during Ultra DMA data out bursts
	DSTROBE	Drive data strobe during Ultra DMA data in bursts
DIOW	STOP	Host stop ultra DMA bursts

Both the host and drive do a CRC function during an ultra DMA burst:

- The host sends CRC data to the drive.
- The drive does a CRC data comparison.

If the CRC comparison fails, the error register ERR bit is set. The drive always reports the first error that occurs.

11.8 ATA RESET/Power-Up

11.8.1 Hardware Reset

The host asserts $\overline{\text{RESET}}$ for a minimum of 25 µs after power has stabilized within system specified tolerance. A signal assertion less than 20 ns is not recognized by the drive.

The host should not do the following:

- set the device control register bit SRST to 1 to enable the drive for software reset
- issue a DEVICE RESET command while the status register BSY bit is set to 1.

NOTE

Hardware reset is a board requirement, not an MPC5200 function unless GPIO is used.

11.8.2 Software Reset

The host sets the device control register bit SRST to 1. Any subsequent setting and clearing of the SRST bit must be at least 5 µs apart.

Figure 11-10 shows the Reset timing diagram. Table 11-40 gives timing characteristics.

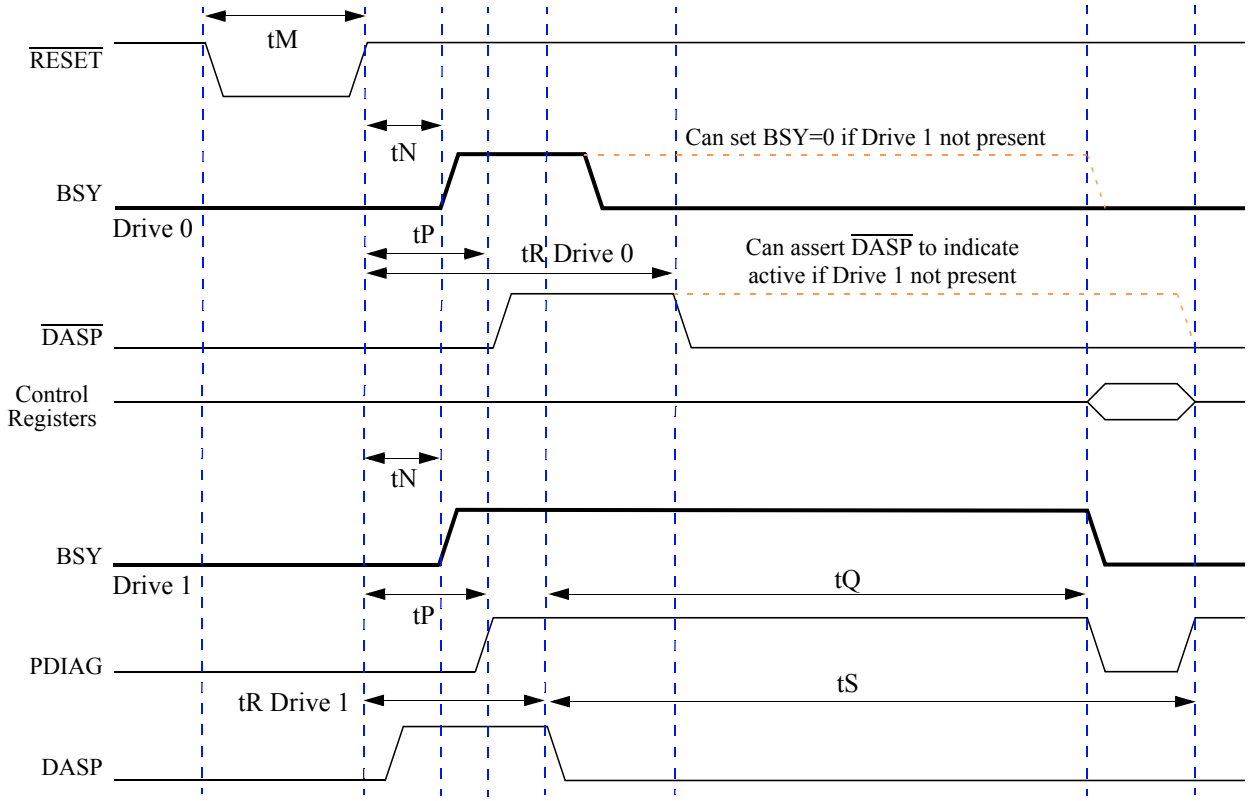


Figure 11-10. Timing Diagram—Reset Timing

Table 11-40. Reset Timing Characteristics

Name	PIO Timing Parameter	Min/Max	Timing
tM	Reset pulse width	Min	25µs
tN	Reset negated to BSY active setup	Max	400ns
tP	Reset negated to DASP inactive setup	Max	1ms
tQ	DASP active to PDIAG active setup	Max	30s
tR	Drive 0—Reset negated to DASP active setup	Max	450ms
	Drive 1—Reset negated to DASP active setup	Max	400ms
tS	DASP active to PDIAG inactive setup	Max	30.5s

11.9 ATA I/O Cable Specifications

For reference, the standard ATA cable specifications affects stem integrity and should not exceed 18 inches or 0.46 m. Total cable capacitance should not exceed 35 pF.

Notes

Chapter 12
Universal Serial Bus (USB)

12.1 Overview

The following sections are contained in this document:

- Section 12.2, *Data Transfer Types*
- Section 12.4, *Host Control (HC) Operational Registers*, includes:
  - Section 12.4.2, *Control and Status Partition—MBAR + 0x1000*
  - Section 12.4.3, *Memory Pointer Partition—MBAR + 0x1018*
  - Section 12.4.4, *Frame Counter Partition—MBAR + 0x1034*
  - Section 12.4.5, *Root Hub Partition—MBAR + 0x1048*

The Universal Serial Bus (USB) is an external bus standard that supports data transfer rates of 12Mbps. Figure 12-1 shows the four main areas of a USB system, which are:

- Client software/USB driver—software implemented
- Host Controller Driver (HCD)—software implemented
- Host Controller (HC)—hardware implemented
- USB device—hardware implemented

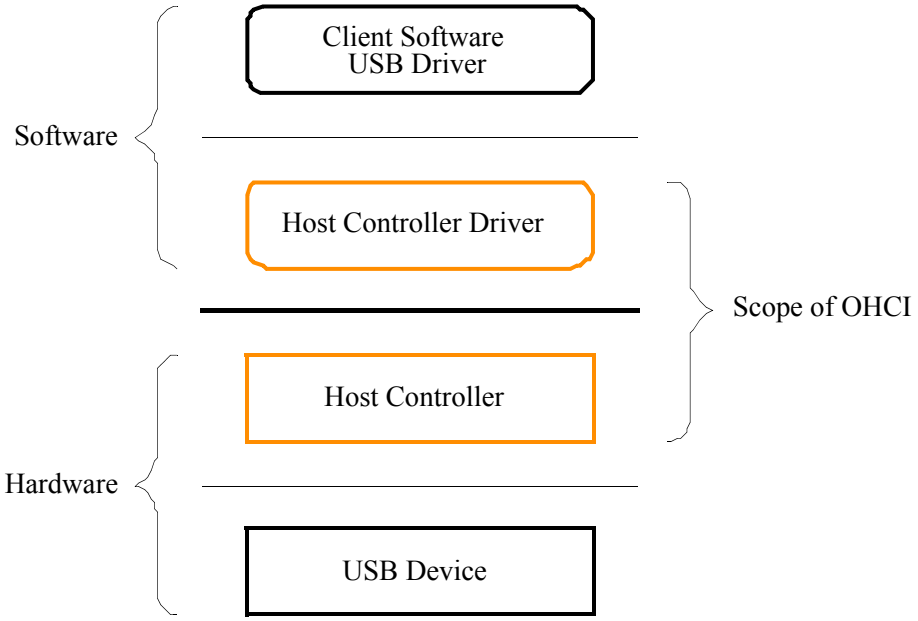


Figure 12-1. USB Focus Areas

The Open Host Controller Interface (OHCI) is a register-level description of a HC for the Universal Serial Bus (USB). OHCI specifies the interface between and the fundamental HCD operation and the HC.

The HCD and HC work in tandem to transfer data between client software and a USB device. Data is translated from shared-memory data structures at the client software end, to USB signal protocols at the USB device end, and vice-versa.

12.2 Data Transfer Types

Four data transfer types are defined in the USB. Each type is optimized to match the service requirements between client software and the USB device. These types are:

- **Interrupt Transfers**—Small data transfers used to communicate information from the USB device to the client software. The HCD polls the USB device by issuing tokens to the device at a periodic interval sufficient for the requirements of the device.
- **Isochronous Transfers**—Periodic data transfers with a constant data rate. Data transfers are correlated in time between the sender and receiver.
- **Control Transfers**—Non-periodic data transfers used to communicate configuration/command/status type information between client software and the USB device.

- **Bulk Transfers**—Non-periodic data transfers used to communicate large amounts of information between client software and the USB device.

In OpenHCI the data transfer types are classified into two categories: periodic and nonperiodic. Periodic transfers are interrupt and isochronous since they are scheduled to run at periodic intervals. Non-periodic transfers are control and bulk since they are not scheduled to run at any specific time, but rather on a time-available basis.

12.3 Host Controller Interface

12.3.1 Communication Channels

There are two communication channels between the HC and HCD.

1. The first channel uses a set of operational registers located on the HC. The HC is the target for all communication on this channel. The operational registers contain control, status, and list pointer registers. Within the operational register set is a pointer to a location in shared memory named the HC Communications Area (HCCA).
2. The HCCA is the second communication channel. The HC is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt endpoint descriptor lists, the head pointer to the done queue, and status information associated with start-of-frame processing.

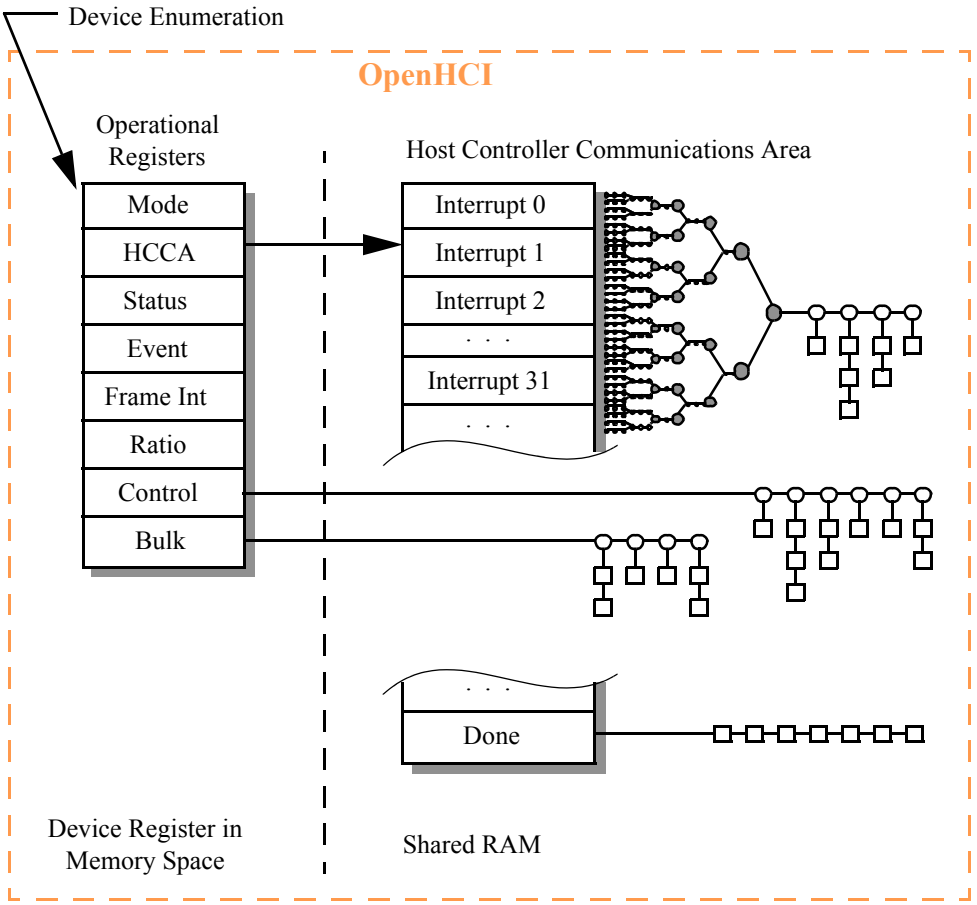


Figure 12-2. Communication Channels

12.3.2 Data Structures

The basic building blocks for communication across the interface are the endpoint descriptor (ED) and transfer descriptor (TD).

The HCD assigns an endpoint descriptor to each endpoint in the system. The endpoint descriptor contains the information necessary for the HC to communicate with the endpoint. The fields include the maximum packet size, the endpoint address, the speed of the endpoint, and the direction of data flow. Endpoint descriptors are linked in a list.

A queue of transfer descriptors is linked to the endpoint descriptor for the specific endpoint. The transfer descriptor contains the information necessary to describe the data packets to be transferred. The fields include data toggle information, shared memory buffer location, and

completion status codes. Each transfer descriptor contains information that describes one or more data packets. The data buffer for each transfer descriptor ranges in size from 0 to 8192 Bytes with a maximum of one physical page crossing. Transfer descriptors are linked in a queue; the first one queued is the first one processed.

Each data transfer type has its own linked list of endpoint descriptors to be processed. Figure 12-3 shows the data structure relationship.

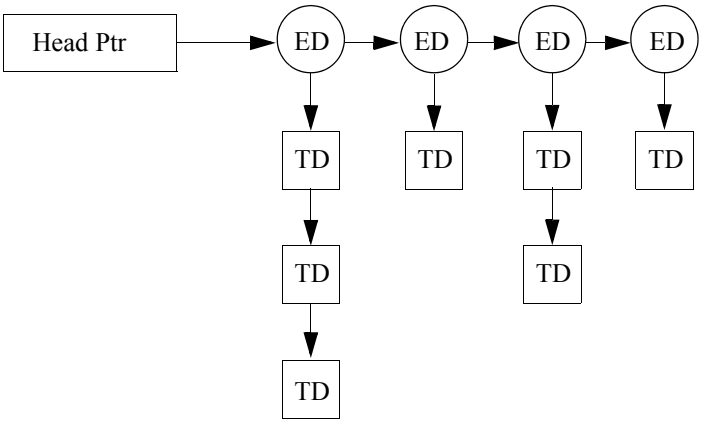


Figure 12-3. Typical List Structure

The head pointers to the bulk and control endpoint descriptor lists are maintained within the operational registers in the HC. The HCD initializes these pointers prior to the HC gaining access to them. Should these pointers need to be updated, the HCD may need to stop the HC from processing the specific list, update the pointer, then re-enable the HC.

The head pointers to the interrupt endpoint descriptor lists are maintained within the HCCA. There is no separate head pointer for isochronous transfers. The first isochronous endpoint descriptor simply links to the last interrupt endpoint descriptor. There are 32 interrupt head pointers. The head pointer used for a particular frame is determined by using the last five bits of the frame counter as an offset into the interrupt array within the HCCA.

The interrupt endpoint descriptors are organized into a tree structure with the head pointers being the leaf nodes. The desired interrupt endpoint polling rate is achieved by scheduling the endpoint descriptor at the appropriate depth in the tree. The higher the polling rate, the closer to the root of the tree the endpoint descriptor is placed. Figure 12-4 shows the interrupt endpoint structure. The Interrupt endpoint descriptor placeholder indicates where zero or more endpoint descriptors may be queued. The numbers on the left are the index into the HCCA interrupt head pointer array.

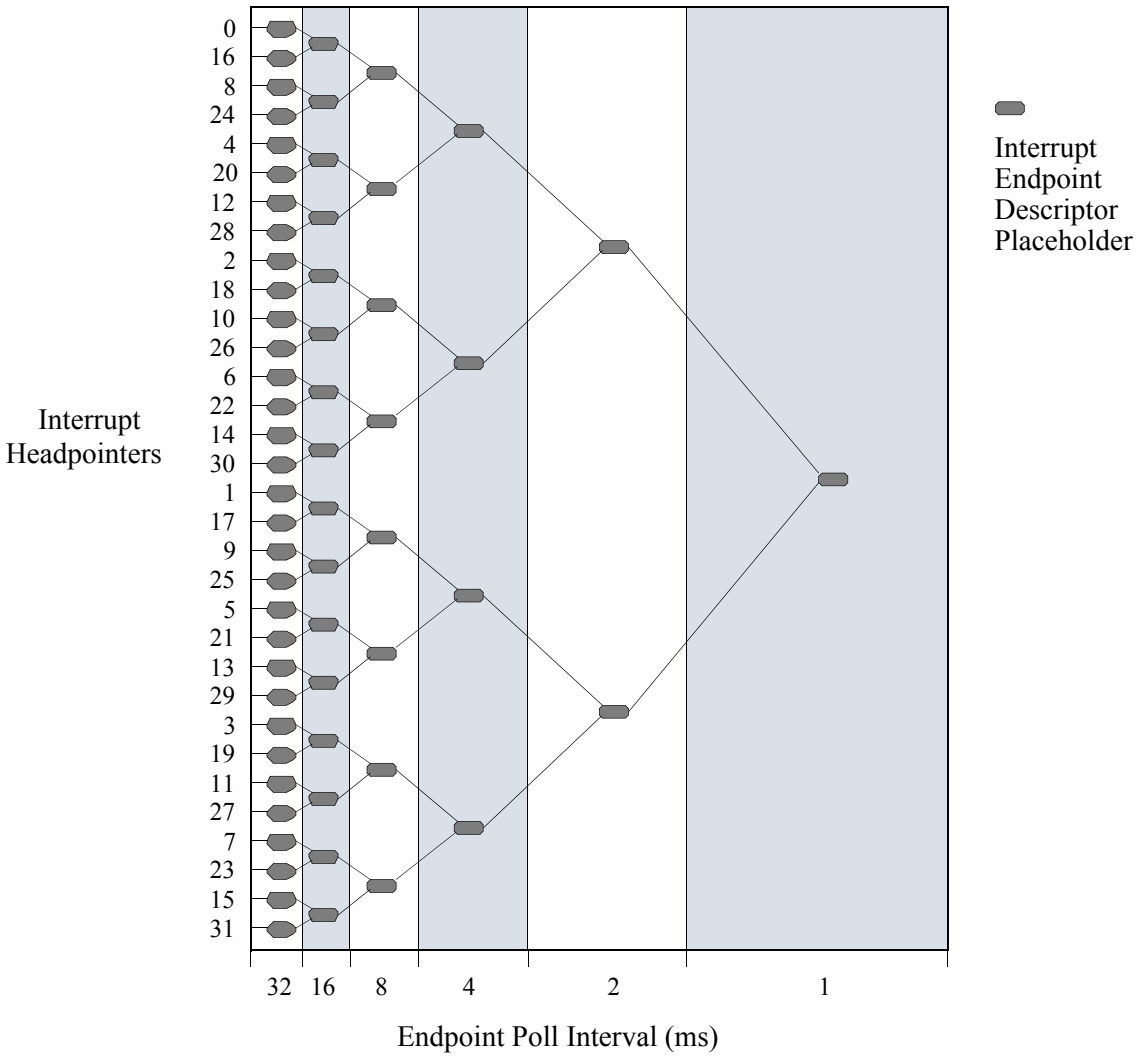


Figure 12-4. Interrupt ED Structure

Figure 12-5 shows a sample interrupt endpoint schedule. The schedule shows:

- two endpoint descriptors at a 1ms poll interval
- two endpoint descriptors at a 2ms poll interval
- one endpoint descriptor at a 4ms poll interval
- two endpoint descriptors at an 8ms poll interval
- two endpoint descriptors at a 16ms poll interval
- two endpoint descriptors at a 32ms poll interval.

NOTE

Unused interrupt endpoint placeholders are bypassed and the link is connected to the next available endpoint in the hierarchy.

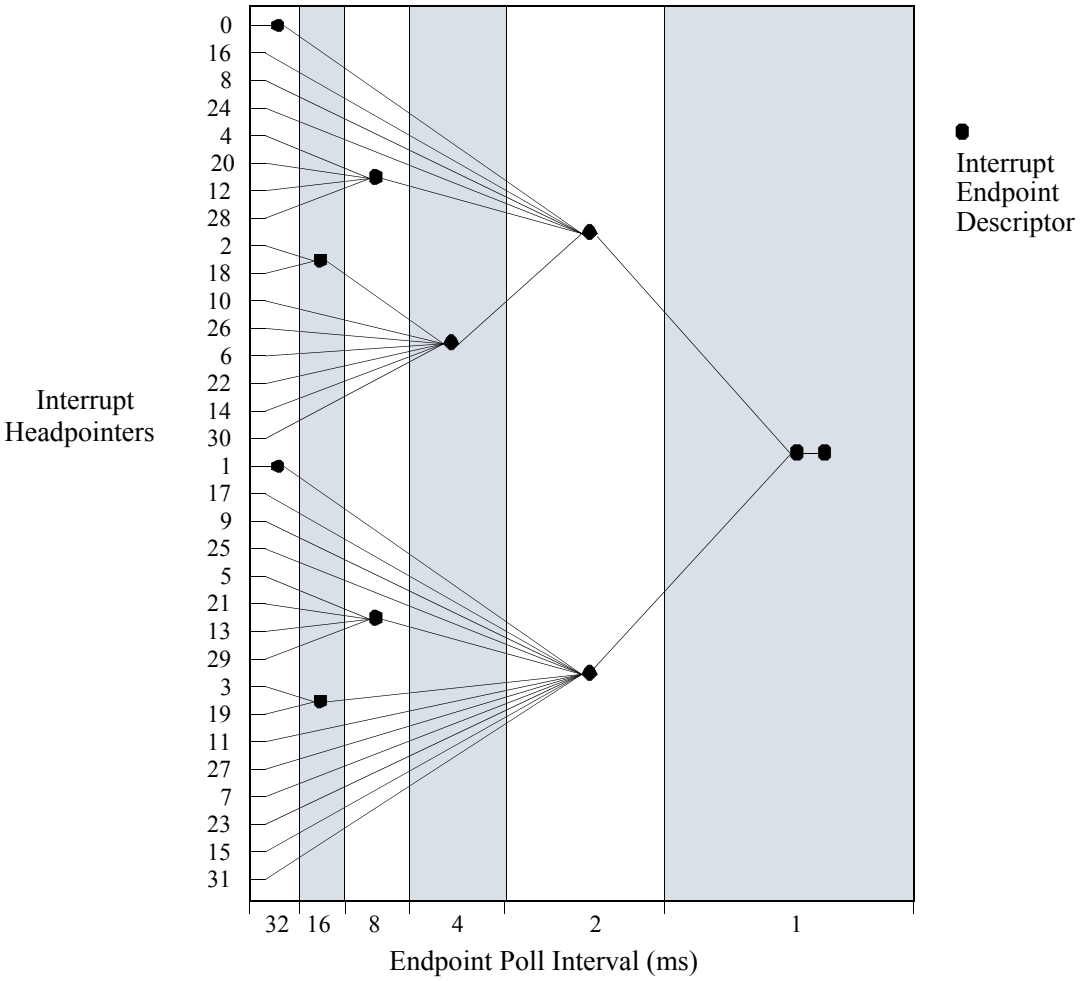


Figure 12-5. Sample Interrupt Endpoint Schedule

12.4 Host Control (HC) Operational Registers

Host Control contains a set of on-chip operational registers which are mapped into a non-cacheable portion of the system addressable space. These registers are used by the HCD. According to the function of these registers, they are divided into four partitions, specifically for control and status, memory pointer, frame counter and root hub. All of the registers should be read and written as D-words.

Reserved bits may be allocated in future releases of this specification. To ensure interoperability, the HCD that does not use a reserved field should not assume the reserved field contains 0. In addition, HCD should always preserve the reserved field value(s).

When a R/$\overline{W}$ register is modified, the HCD should first read the register and modify the bits desired. Then, HCD should write the register with the reserved bits still containing the read value. Alternatively, HCD can maintain an in-memory copy of previously written values that can be modified and written to the HC register. When a write to the set/clear register is written, bits written to reserved fields should be 0.

12.4.1 Programming Note

Programmers should observe the following notes:

1. The Clock Distribution Module (CDM) (0x0210) must be initialized before you can access any USB registers. If this register is not initialized, every USB register access will cause a machine check interrupt.
   For Example: If the SYS_XTAL_IN frequency is 33 MHz and the RST_CFG6 pin is low (multiplier 16), than the four phase divide ratios must be set to 0x5, fractional counter divide ration of f_{system}/11. 33 MHz * 16 / 11 = 48 MHz (USB frequency)
2. The GPS Port Configuration Register—MBAR + 0x0B00 (0xB00) must be initialized to communicate over the muxed USB port. It configures USB for Differential or SE0 mode, the port to be used for USB2 and if the IrDA/USB 48MHz clock is generated internally or externally.

12.4.2 Control and Status Partition—MBAR + 0x1000

This HC partition uses 6 32-bit registers. These registers are located at an offset from MBAR of 0x1000. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x1000 + register address**

The following registers are available:

- USB HC Revision Register (0x1000)
- USB HC Control Register (0x1004)
- USB HC Command Status Register (0x1008)
- USB HC Interrupt Status Register (0x100C)
- USB HC Interrupt Enable Register (0x1010)
- USB HC Interrupt Disable Register (0x1014)

12.4.2.1 USB HC Revision Register—MBAR + 0x1000

Table 12-1. USB HC Revision Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved								REV							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	Name	Description
0:23	–	Reserved
24:31	REV	Revision—a read-only field containing the BCD representation of the HCI specification version implemented by this HC. For example, a value of 11h corresponds to version 1.1. All HC implementations compliant with this specification have a value of 10h.

12.4.2.2 USB HC Control Register—MBAR + 0x1004

The HC Control register defines HC operating modes. Except for HostController FunctionalState and RemoteWakeUpConnected, most fields in this register are modified only by the HCD.

Table 12-2. USB HC Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:20	—	Reserved
21	RWE	RemoteWakeUpEnable—HCD uses bit to enable or disable the remote WakeUp feature on detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote WakeUp is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.
22	RWC	RemoteWakeUpConnected—bit indicates whether HC supports remote WakeUp signaling. If remote WakeUp is supported and used by the system it is the responsibility of system firmware to set this bit during POST. HC clears bit on a hardware reset, but does not alter it on a software reset. Host system remote WakeUp signaling is host-bus-specific and not described in this specification.
23	IR	InterruptRouting—bit determines routing of interrupts generated by events registered in HcInterruptStatus. • If clear, all interrupts are routed to the normal host bus interrupt mechanism. • If set, interrupts are routed to the System Management Interrupt. HCD clears this bit on a hardware reset, but does not alter it on a software reset. HCD uses this bit as a tag to indicate HC ownership.
24:25	HCFS	HostControllerFunctionalState—a USB field: 00=USBRESET 01=USBRESUME 10=USBOPERATIONAL 11=USBSUSPEND Transition to USBOPERATIONAL from another state causes SOF generation to begin 1ms later. HCD may determine if HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC, only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. A hardware reset also resets the Root Hub and asserts subsequent reset signaling to downstream ports.
26	BLE	BulkListEnable—setting bit enables Bulk list processing in next Frame. • If cleared by HCD, Bulk list processing does not occur after next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. • If HcBulkCurrentED points to an ED to be removed, HCD advances pointer by updating HcBulkCurrentED before re-enabling list processing.
27	CLE	ControlListEnable—setting bit enables Control list processing in next Frame. • If cleared by HCD, Control list processing does not occur after next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. • If HcControlCurrentED points to an ED to be removed, HCD advances pointer by updating HcControlCurrentED before re-enabling list processing.
28	IE	IsochronousEnable—HCD uses bit to enable/disable isochronous EDs processing. While processing the periodic list in a Frame, HC checks bit status when it finds an Isochronous ED (F=1). • If set (enabled), HC continues processing the EDs. • If cleared (disabled), HC halts periodic list processing, which now contains only isochronous EDs, and begins processing Bulk/Control lists. Setting this bit is guaranteed to take effect in the next Frame, not the current Frame.

Bits	Name	Description
29	PLE	PeriodicListEnable—setting bit enables periodic list processing in next Frame. If cleared by HCD, periodic list processing does not occur after the next SOF. HC checks this bit prior to starting list processing.
30:31	CBSR	ControlBulkServiceRatio—field specifies the service ratio between Control and Bulk EDs. Before processing non-periodic lists, HC compares the ratio specified with its internal count on how many non-empty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. When crossing the frame boundary, the internal count is retained. In case of reset, HCD is responsible for restoring this value. CBSR=Number of Control EDs Over Bulk EDs Served 0=1:1 1=2:1 2=3:1 3=4:1

12.4.2.3 USB HC Command Status Register—MBAR + 0x1008

HC uses the HC Command Status register to receive (Rx) commands issued by HCD. It reflects the current HC status. To HCD, it appears to be a write-to-set register. HC ensures bits written as 1 are set in the register, while bits written as 0 remain unchanged in the register. HCD may issue multiple distinct commands to HC without concern for corrupting previously issued commands. HCD has normal read access to all bits.

The SchedulingOverrunCount field indicates the number of frames in which HC detects scheduling overrun errors. This occurs when the Periodic list does not complete before EOF. When a scheduling overrun error is detected, HC increments the counter and sets SchedulingOverrun field in HcInterruptStatus register.

Table 12-3. USB HC Command Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved														SOC	
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 lsb
R	Reserved												OCR	BLF	CLF	HCR
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:13	—	Reserved
14:15	SOC	SchedulingOverrunCount—bits are incremented on each scheduling overrun error. SOC is initialized to 00 and wraps at 11. SOC increments when a scheduling overrun is detected, even if SchedulingOverrun in HcInterruptStatus has already been set. HCD uses SOC to monitor any persistent scheduling problems.
16:27	—	Reserved
28	OCR	OwnershipChangeRequest—OS HCD sets this bit to request an HC change of control. When set, HC sets the OwnershipChange field in HcInterruptStatus. After changeover, this bit is cleared and remains clear until the next OS HCD request.

Bits	Name	Description
29	BLF	BulkListFilled—bit indicates whether there are Bulk List TDs. HCD sets this bit when it adds a TD to a Bulk List ED. When HC begins processing the Bulk List head, it checks BF. • If BLF is 0, HC does not start Bulk List processing. • If BLF is 1, HC starts Bulk List processing and sets BF to 0. • If HC finds a Bulk List TD, HC sets BLF to 1, causing Bulk List processing to continue. • If HC does not find a Bulk List TD and HCD does not set BLF, then BLF remains 0 when HC completes processing and Bulk List processing stops.
30	CLF	ControlListFilled—bit indicates whether there are Control List TDs. HCD sets this bit when it adds a TD to a Control List ED. When HC begins processing the Control List head, it checks CLF. • If CLF is 0, HC does not start Control List processing. • If CF is 1, HC starts Control List processing and sets CLF to 0. • If HC finds a Control List TD, CLF is set to 1, causing Control List processing to continue. • If HC does not find a Control List TD and HCD does not set CLF, then CLF remains 0 when HC completes processing and Control List processing stops.
31	HCR	HostControllerReset—HCD sets bit to initiate a software reset of HC. Regardless of the HC functional state, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise. For example, HcControl Interrupt Routing field and no Host bus access is allowed. On completion of the reset operation, HC clears this bit. Completion must be within 10ms. When set, this bit should not cause a root hub reset and no subsequent reset signaling should be asserted to downstream ports.

12.4.2.4 USB HC Interrupt Status Register —MBAR + 0x100C

This register provides status on various events that cause hardware interrupts. When an event occurs, HC sets the corresponding register bit. When a bit is set, a hardware interrupt is generated, if the interrupt is enabled in the HcInterruptEnable register and the MasterInterruptEnable bit is set. HCD may clear specific bits in this register by writing 1 to bit positions to be cleared. HCD may not set any of these bits. HC never clears the bit.

Table 12-4. USB HC Interrupt Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	OC	Reserved													
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 lsb
R	Reserved									RHSC	FNO	UE	RD	SF	WDH	SO
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	—	Reserved
1	OC	OwnershipChange—HC sets this bit when HCD sets the HcCommandStatus OwnershipChangeRequest field. This event, when unmasked, always generate an immediate System Management Interrupt (SMI). When the SMI pin is not implemented, the OC bit is tied to 0.
2:24	—	Reserved

Bits	Name	Description
25	RHSC	RootHubStatusChange—bit is set when HcRhStatus content or content of any HcRhPortStatus[Number of Downstream Port] changes.
26	FNO	FrameNumberOverflow—bit is set when HcFmNumber msb (bit 15) changes value (from 0 to 1, or from 1 to 0) and after HccaFrameNumber is updated.
27	UE	UnrecoverableError—bit is set when HC detects a system error not related to USB. HC should not proceed with processing or signaling prior to the system error being corrected. HCD clears this bit after HC is reset.
28	RD	ResumeDetected—bit is set when HC detects a USB device asserting a resume signal. It is the transition from no resume signaling to resume signaling that causes this bit to be set. This bit is not set when HCD sets the USBRESUME state.
29	SF	StartofFrame—bit is set by HC at each start of a frame and after updating the HccaFrameNumber. HC also generates an SOF token at the same time.
30	WDH	WritebackDoneHead—bit is set immediately after HC writes HcDoneHead to HccaDoneHead. Further HccaDoneHead updates do not occur until this bit is cleared. HCD should only clear this bit after saving HccaDoneHead contents.
31	SO	SchedulingOverrun—bit is set when USB schedule for the current Frame overruns and after an HccaFrameNumber update. A scheduling overrun also causes the HcCommandStatus SOC to increment.

12.4.2.5 USB HC Interrupt Enable Register—MBAR + 0x1010

Each enable bit in the HC Interrupt Enable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When:

1. a bit is set in the HcInterruptStatus register, and
2. the corresponding bit is set in the HcInterruptEnable register, and
3. the MasterInterruptEnable bit is set, then
4. a hardware interrupt is requested on the host bus.

Writing 1 to a bit in this register sets the corresponding bit, whereas writing 0 to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Table 12-5. USB HC Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIE	OC	Reserved													
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 lsb
R	Reserved									RHSC	FNO	UE	RD	SF	WDH	SO
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	MIE	Master Interrupt Enable—used by HCD. 0 written to this bit is ignored by HC. 1 written to this bit enables interrupt generation, due to events specified in other bits of this register.
1	OC	Ownership Change Ignore Enable interrupt generation due to ownership change
2:24	—	Reserved
25	RHSC	Root Hub Status Change Ignore Enable interrupt generation due to root hub status change.
26	FNO	Frame Number Overflow Ignore Enable interrupt generation due to frame number overflow.
27	UE	Unrecoverable Error Ignore Enable interrupt generation due to unrecoverable error.
28	RD	Resume Detected Ignore Enable interrupt generation due to resume detect.
29	SF	Start of Frame Ignore Enable interrupt generation due to start of frame.
30	WDH	Writeback Done Head Ignore Enable interrupt generation due to HcDoneHead writeback.
31	SO	Scheduling Overrun Ignore Enable interrupt generation due to scheduling overrun.

12.4.2.6 USB HC Interrupt Disable Register—MBAR + 0x1014

Each disable bit in the HC Interrupt Disable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register is coupled with the HcInterruptEnable register. Thus, writing a ‘1’ to a bit in this register clears the corresponding bit in the HcInterruptEnable register, whereas writing a ‘0’ to a bit in this register leaves the corresponding bit in the HcInterruptEnable register unchanged. On read, the current value of the HcInterruptEnable register is returned.

Table 12-6. USB HC Interrupt Disable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIE	OC	Reserved													
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 lsb
R	Reserved									RHSC	FNO	UE	RD	SF	WDH	SO
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	MIE	Master Interrupt Enable—bit is set after a hardware or software reset. 0 written to this bit is ignored by HC. 1 written to this bit disables interrupt generation, due to events specified in other bits of this register.
1	OC	Ownership Change Ignore Disable interrupt generation due to Ownership Change
2:24	—	Reserved
25	RHSC	Root Hub Status Change Ignore Disable interrupt generation due to root hub status change.
26	FNO	Frame Number Overflow Ignore Disable interrupt generation due to frame number overflow.
27	UE	Unrecoverable Error Ignore Disable interrupt generation due to unrecoverable error.
28	RD	Resume Detected Ignore Disable interrupt generation due to resume detect.
29	SF	Start of Frame Ignore Disable interrupt generation due to start of frame.
30	WDH	Writeback Done Head Ignore Disable interrupt generation due to HcDoneHead writeback.
31	SO	Scheduling Overrun Ignore Disable interrupt generation due to scheduling overrun.

12.4.3 Memory Pointer Partition—MBAR + 0x1018

This HC partition uses 7 32-bit registers. These registers are located at an offset from MBAR of 0x1018. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x1018 + register address**

The following registers are available:

- USB HC HCCA Register (0x1018)
- USB HC Period Current Endpoint Descriptor Register (0x101C)
- USB HC Control Head Endpoint Descriptor Register (0x1020)
- USB HC Control Current Endpoint Descriptor Register (0x1024)
- USB HC Bulk Head Endpoint Descriptor Register (0x1028)

- USB HC Bulk Current Endpint Descriptor Register (0x102C)
- USB HC Done Head Register (0x1030)

12.4.3.1 USB HC HCCA Register—MBAR + 0x1018

The HC HCCA register contains the physical address of the Host Controller Communication Area. HCD determines alignment restrictions by writing all 1s to HcHCCA and reading the HcHCCA content. Alignment is evaluated by examining the number of 0s in the lower order bits. Minimum alignment is 256 Bytes. Bits 0 through 7 must always return 0 when read. This area holds control structures and the interrupt table, which are accessed by both the HC and HCD.

Table 12-7. USB HC HCCA Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HCCA															
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 lsb
R	HCCA								Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:23	HCCA	Host Controller Communication Area—base address.
24:31	—	Reserved

12.4.3.2 USB HC Period Current Endpoint Descriptor Register —MBAR + 0x101C

The HC Period Current Endpoint Descriptor (ED) register contains the physical address of the current isochronous or interrupt endpoint descriptor.

Table 12-8. USB HC Period Current Endpoint Descriptor Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PCED															
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 lsb
R	PCED												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	PCED	PeriodCurrentED—HC uses this field to point to the head of one of the Periodic lists, which is processed in the current Frame. HC updates register content after a periodic ED is processed. HCD may read the content in determining which ED is currently being processed at the time of reading.
28:31	—	Reserved

12.4.3.3 USB HC Control Head Endpoint Descriptor Register —MBAR + 0x1020

The HC Control Head Endpoint Descriptor register contains the physical address of the first endpoint descriptor of the Control list.

Table 12-9. USB HC Control Head Endpoint Descriptor Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHED															
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 lsb
R	CHED												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	CHED	ControlHeadED—HC traverses the control list starting with the HcControlHeadED pointer. Content is loaded from HCCA during HC initialization.
28:31	—	Reserved

12.4.3.4 USB HC Control Current Endpoint Descriptor Register —MBAR + 0x1024

The HC Control Current Endpoint Descriptor register contains the physical address of the current control list endpoint descriptor.

Table 12-10. USB HC Control Current Endpoint Descriptor Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CCED															
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 lsb
R	CCED												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	CCED	ControlCurrentED—pointer is advanced to next ED after serving the present one. HC continues processing the list from where it left off in the last frame. When it reaches the control list end, HC checks the HcCommandStatus ControlListFilled. • If set, CCED copies HcControlHeadED content to HcControlCurrentED and clears bit. • If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, this is set to 0 to indicate the end of the Control List.
28:31	—	Reserved

12.4.3.5 USB HC Bulk Head Endpoint Descriptor Register—MBAR + 0x1028

The HC Head Endpoint Descriptor register contains the physical address of the first bulk list endpoint descriptor.

Table 12-11. USB HC Bulk Head Endpoint Descriptor Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BHED															
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 lsb
R	BHED												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	BHED	BulkHeadED—HC traverses the Bulk List starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the HC initialization.
28:31	—	Reserved

12.4.3.6 USB HC Bulk Current Endpoint Descriptor Register—MBAR + 0x102C

The HC Bulk Current Endpoint Descriptor register contains the physical address of the current endpoint of the bulk list. The bulk list is served in a round-robin fashion, therefore endpoints are ordered according to their insertion into the list.

Table 12-12. USB HC Bulk Current Endpint Descriptor Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BCED															
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 lsb
R	BCED												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	BHED	BulkCurrentED—advances to the next ED after HC has served the present ED. HC continues processing the list from where it left off in the last Frame. When it reaches the end of the Bulk List, HC checks the HcCommandStatus BulkListFilled. • If set, BHED copies HcBulkHeadED content to HcBulkCurrentED and clears bit. • If not set, it does nothing. HCD is only allowed to modify this register when HcControl BulkListEnable is cleared. When set, HCD only reads the instantaneous value of this register. This is initially set to 0 to indicate the end of the Bulk List.
28:31	—	Reserved

12.4.3.7 USB HC Done Head Register—MBAR + 0x1030

The HC Done Head register contains the physical address of the last completed transfer descriptor that was added to the done queue. In normal operation, HCD does not need to read this register as its content is periodically written to the HCCA.

Table 12-13. USB HC Done Head Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DH															
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 lsb
R	DH												Reserved			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:27	DH	DoneHead—When a TD is complete, HC writes the HcDoneHead content to the TD NextTD field. HC then overwrites the HcDoneHead content with the TD address. This is set to 0 when HC writes the register content to HCCA. HcInterruptStatus WritebackDoneHead is also set.
28:31	—	Reserved

12.4.4 Frame Counter Partition—MBAR + 0x1034

This HC partition uses 5 32-bit registers. These registers are located at an offset from MBAR of 0x1034. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x1034 + register address**

The following registers are available:

- USB HC Frame Interval Register (0x1034)
- USB HC Frame Remaining Register (0x1038)
- USB HC Frame Number Register (0x103C)
- USB HC Periodic Start Register (0x1040)
- USB HC LS Threshold Register (0x1044)

12.4.4.1 USB HC Frame Interval Register—MBAR + 0x1034

The HC Frame Interval register contains a 14-bit value that indicates:

- the bit-time interval in a Frame. For example, between two consecutive SOFs.
- a 15-bit value that indicates the full speed maximum packet size the HC may transmit or receive without causing scheduling overruns.

HCD may carry out minor adjustment on the frame interval by writing a new value over the present one at each SOF. This provides the programmability necessary for the HC to synchronize with an external clocking resource and to adjust any unknown local clock offset.

Table 12-14. USB HC Frame Interval Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIT	FSMPS														
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 lsb
R	Reserved	FI														
W																
RESET:	0	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1

Bits	Name	Description
0	FIT	FrameIntervalToggle—HCD toggles this bit when it loads a new value to the frame interval.
1:15	FSMPS	FSLargestDataPacket—specifies a value that is loaded into the largest data packet counter at the beginning of each frame. The counter value represents the largest amount of data in bits that the HC can send or received in a single transaction at any given time without causing scheduling overrun. HCD calculates this field value.
16:17	—	Reserved
18:31	FI	FrameInterval—specifies the bit-time interval between two consecutive SOFs. Nominally, this value is set to 11,999. HCD should store the field's current value before resetting HC. Setting the HcCommandStatus HostControllerReset field causes the HC to reset this field to its nominal value. HCD may choose to restore the stored value when the reset sequence completes.

12.4.4.2 USB HC Frame Remaining Register—MBAR + 0x1038

This register is a 14-bit count-down counter containing the remaining current Frame bit-time.

Table 12-15. USB HC Frame Remaining Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRT	Reserved														
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 lsb
R	Reserved		FR													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	FRT	FrameRemainingToggle—bit is loaded from the HcFmInterval FrameIntervalToggle field when FrameRemaining reaches 0. HCD uses this bit for synchronization between FrameInterval and FrameRemaining.
1:17	—	Reserved
18:31	FR	FrameRemaining—is a counter that is decremented at each bit-time. When it reaches 0, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit-time boundary. When entering the USBOPERATIONAL state, HC reloads the content with the HcFmInterval FrameInterval and uses the updated value from the next SOF.

12.4.4.3 USB HC Frame Number Register—MBAR + 0x103C

The HC Frame Number register is a 16-bit counter. It provides a timing reference among events happening in the HC and HCD. The HC driver may use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Table 12-16. USB HC Frame Number Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	FN															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
16:31	FN	FrameNumber—is incremented when HcFmRemaining is re-loaded. FN rolls over to 0 after ffff. When entering the USBOPERATIONAL state, this is automatically incremented. Content is written to HCCA after HC has incremented the FN at each frame boundary and sent a SOF, but before HC reads the first ED in that frame. After writing to HCCA, HC sets the HcInterruptStatus StartofFrame.
0:15	—	Reserved

12.4.4.4 USB HC Periodic Start Register—MBAR + 0x1040

This register has a 14-bit programmable value that determines when is the earliest time HC should start processing the periodic list.

Table 12-17. USB HC Periodic Start Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved		PS													
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:17	—	Reserved
18:31	PS	PeriodicStart—field is cleared after a hardware reset. PS is then set by HCD during HC initialization. PS value is calculated roughly as 10% off from HcFmInterval. A typical value is 3E67. When HcFmRemaining reaches the value specified, processing of periodic lists has priority over Control/Bulk processing. HC then starts processing the Interrupt list after completing the current Control or Bulk transaction in progress.

12.4.4.5 USB HC LS Threshold Register—MBAR + 0x1044

This register contains an 11-bit value used by the HC to determine whether to commit to the transfer of a maximum 8-Byte LS packet before EOF. Neither the HC nor HCD are allowed to change this value.

Table 12-18. USB HC LS Threshold Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved				LST											
W																
RESET:	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0

Bits	Name	Description
0:19	—	Reserved
20:31	LST	LSThreshold—field contains a value which is compared to the FrameRemaining field prior to initiating a low speed transaction. The transaction is started only if FrameRemaining is greater than or equal to this field. HCD calculates this value with the consideration of transmission and setup overhead.

12.4.5 Root Hub Partition—MBAR + 0x1048

This HC partition uses 5 32-bit registers. These registers are located at an offset from MBAR of 0x1048. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x1048 + register address**

The following registers are available:

- USB HC Rh Descriptor A Register (0x1048)
- USB HC Rh Descriptor B Register (0x104C)
- USB HC Rh Status Register (0x1050)
- USB HC Rh Port1 Status Register (0x1054)
- USB HC Rh Port2 Status Register (0x1058)

All registers included in this partition are dedicated to the USB root hub, which is an integral part of the HC though still a functionally separate entity. HCD emulates USBD access to the root hub via a register interface. HCD maintains many USB-defined hub features which are not required to be supported in hardware. For example, the hub's device, configuration, interface, and endpoint descriptors are maintained only in the HCD and some class descriptor static fields. HCD also maintains and decodes the root hub device address as well as other trivial operations better suited to software than hardware.

The root hub register interface is otherwise developed to maintain similarity of bit organization and operation to typical hubs which are found in the system. Each register is read and written as a D-word. These registers are only written during initialization to correspond with the system implementation.

- HcRhDescriptorA and HcRhDescriptorB registers should be implemented such that they are writeable regardless of the HC USB state.
- HcRhStatus and HcRhPortStatus must be writeable during the USBOPERATIONAL state.

NOTE

IS denotes an implementation-specific reset value for that field.

12.4.5.1 USB HC Rh Descriptor A Register—MBAR + 0x1048

This register is the first of two registers describing the root hub characteristics. Reset values are implementation-specific. The HCD emulates the following hub class descriptor fields:

- descriptor length (11)
- descriptor type (TBD)
- hub controller current (0)

All other fields are located in the HcRhDescriptorA and HcRhDescriptorB registers.

Table 12-19. USB HC Rh Descriptor A Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	POTPGT								Reserved							
W																
RESET:	0	0	1	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 lsb
R	Reserved			NOCP	OCPM	DT	NPS	PSM	NDP							
W																
RESET:	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0

Bits	Name	Description
0:7	POTPGT	Power On To Power Good Time—specifies the duration HCD must wait before accessing a Root Hub powered-on port. POTPGT is implementation-specific. The time unit is 2ms. Duration is calculated as POTPGT x 2ms.
8:18	—	Reserved
19	NOCP	No Over Current Protection—describes how the Root Hub port overcurrent status is reported. When NOCP is cleared, OCPM specifies global or per-port reporting. 0 = Overcurrent status is reported collectively for all downstream ports. 1 = No overcurrent protection supported.
20	OCPM	Over Current Protection Mode—describes how the Root Hub port overcurrent status is reported. At reset, OCPM should reflect the same mode as Power Switching Mode. OCPM is valid only if No Over Current Protection is cleared. 0 = Overcurrent status is reported collectively for all downstream ports. 1 = Overcurrent status is reported on a per-port basis.
21	DT	Device Type—specifies Root Hub is not a compound device. Root Hub is not permitted to be a compound device. DT should always read/write 0.
22	NPS	No Power Switching—specifies whether power switching is supported or ports are always powered. NPS is implementation specific. When this bit is cleared, PSM specifies global or per-port switching. 0 = Ports are power switched. 1 = Ports are always powered on when HC is powered on.
23	PSM	Power Switching Mode—specifies how the root hub port power switching is controlled. PSM is implementation-specific and is only valid if the No Power Switching field is cleared. 0 = All ports are powered at the same time. 1 = Each port is powered individually. This mode lets port power be controlled by either the global switch or per-port switching. • If Port Power Control Mask bit is set, port responds only to port power commands (Set/Clear Port Power). • If port mask is cleared, port is controlled only by the global power switch (Set/Clear Global Power).
24:31	NDP	Number Downstream Ports—specifies the number of downstream ports supported by the Root Hub. NDP is implementation-specific. • Minimum number of ports is 1. • Maximum number of ports (supported by OpenHCI) is 15.

12.4.5.2 USB HC Rh Descriptor B Register—MBAR + 0x104C

This register is the second of two registers describing the Root Hub characteristics. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Table 12-20. USB HC Rh Descriptor B Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPCM															
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 lsb
R	DR															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	PPCM	PortPowerControlMask—each bit indicates whether a port is affected by a global power control command when PSM is set. • When set, port power state is only affected by per-port power control (Set/ClearPortPower). • When cleared, port is controlled by the global power switch (Set/ClearGlobalPower). If device is configured to Global Switching Mode (PSM=0), this field is not valid. bit 0—Reserved bit 1—Ganged-power mask on Port #1 bit 2—Ganged-power mask on Port #2 … bit15—Ganged-power mask on Port #15
16:31	DR	NDeviceRemovable—each bit is dedicated to a Root Hub port. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0—Reserved bit 1—Device attached to Port #1 bit 2—Device attached to Port #2 … bit15—Device attached to Port #15

12.4.5.3 USB HC Rh Status Register—MBAR + 0x1050

This register is divided into two parts. The lower word of a D-word represents the hub status field; the upper word represents the hub status change field. Reserved bits should always be written 0.

Table 12-21. USB HC Rh Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRWE	Reserved													OCIC	LPSC
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 lsb
R	DRWE	Reserved													OCI	LPS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	CRWE	Clear Remote Wake Up Enable (write) • Writing 1 clears DRWE. • Writing 0 has no effect.
1:13	—	Reserved
14	OCIC	Over Current Indicator Change—is set by hardware when a change occurs to the OCI field of this register. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect.
15	LPSC	Local Power Status Change (read)—Root Hub does not support the local power status feature. Thus, this bit is always read as 0. Set Global Power (write) • In global power mode (PSM=0), LPSC is written to 1 to turn on power to all ports (clear Port Power Status). • In per-port power mode, LPSC sets Port Power Status only on ports whose PPCM bit is not set. Writing 0 has no effect.
16	DRWE	Device Remote Wake Up Enable (write)—enables a Connect Status Change bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the Resume Detected interrupt. 0 = Connect Status Change is not a remote Wake Up event. 1 = Connect Status Change is a remote Wake Up event. Set Remote Wake Up Enable (read). 1 = Sets DRWE. 0 = Has no effect.
17:29	—	Reserved
30	OCI	Over Current Indicator—reports overcurrent conditions when global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implemented this bit is always 0.
31	LPS	Local Power Status—Root Hub does not support the local power status feature. This bit is always read as 0 (write) Clear Global Power. In global power mode (PSM=0), bit is written to 1 to turn off power to all ports (clear Port Power Status). In per-port power mode, bit clears Port Power Status only on ports whose PPCM bit is not set. Writing 0 has no effect.

12.4.5.4 USB HC Rh Port1 Status Register—MBAR + 0x1054

This register is controls and reports port events on a per-port basis. The Number of Downstream Ports (NDP) represents the number of Hc Rh Port Status registers that are implemented in hardware. The lower word is used to reflect the port status; the upper word reflects the status change bits. MPC5200 has NDP = 2, therefore, HcRhPort1Status (MBAR + 1054) and HcRhPort2Status (MBAR + 1058).

Some status bits are implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change is postponed until the transaction completes. Reserved bits should always be written 0.

Table 12-22. USB HC Rh Port1 Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved											PRSC	OCIC	PSSC	PESC	CSC
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 lsb
R	Reserved						LSDA	PPS	Reserved			PRS	POCI	PSS	PES	CCS
W																
RESET:	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bits	Name	Description
0:10	—	Reserved
11	PRSC	Port Reset Status Change—bit is set at the end of the 10 ms port reset signal. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = Port reset not complete 1 = Port reset complete
12	OCIC	Port Over Current Indicator Change—bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when Root Hub changes the Port Over Current Indicator bit. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = No change in POCI 1 = POCI has changed
13	PSSC	Port Suspend Status Change—bit is set when the full resume sequence completes. Sequence includes a 20 s resume pulse, LS EOP, and 3 ms resychronization delay. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. This bit is also cleared when Reset Status Change is set. 0 = Resume not complete 1 = Resume complete
14	PESC	Port Enable Status Change—bit is set when hardware events cause the PES bit to be cleared. Changes from HCD writes do not set this bit. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = No change in PES 1 = Change in PES

Bits	Name	Description
15	CSC	ConnectStatusChange—bit is set whenever a connect or disconnect event occurs. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. If CCS is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. 0 = No change in CCS 1 = Change in CCS If the DeviceRemovable[NDP] bit is set, this bit is set only after a Root Hub reset to notify the system that the device is attached.
16:21	—	Reserved
22	LSDA	LowSpeedDeviceAttached (read)—bit indicates the speed of the device attached to this port. 0 = Full speed device attached 1 = Low speed device attached This field is valid only when CurrentConnectStatus is set. ClearPortPower (write) • Writing 1 causes HCD to clear the PortPowerStatus bit. • Writing 0 has no effect.
23	PPS	PortPowerStatus (read)—bit reflects the port power status, regardless of the type of power switching implemented. If an overcurrent condition is detected, this bit is cleared. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode and PortPortControlMask[NDP]. In global switching mode (PSM=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PSM=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. If port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. 0 = Port power is off 1 = Port power is on SetPortPower (write) • Writing causes HCD to set the PortPowerStatus bit. • Writing 0 has no effect. If power switching is not supported, this bit always reads ‘1b’.
24:26	—	Reserved
27	PRS	PortResetStatus (read)—When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. 0 = Port reset signal is not active 1 = Port reset signal is active SetPortReset (write) • Writing 1 causes HCD to set port reset signaling. • Writing 0 has no effect. If CurrentConnectStatus is cleared, a write does not set PortResetStatus. Instead, it sets ConnectStatusChange. This notifies the driver that an attempt was made to reset a disconnected port.

Bits	Name	Description
28	POCI	PortOverCurrentIndicator (read)—bit is only valid when root hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal 0 = No overcurrent condition. 1 = Overcurrent condition detected. ClearSuspendStatus (write) • Writing 1 causes HCD to initiate a resume. • Writing 0 has no effect. A resume is initiated only if PSS is set.
29	PSS	PortSuspendStatus (read)—bit indicates port is suspended or in resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CCS. This bit is cleared when: • PortResetStatusChange is set at the end of the port reset, or • when HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. 0 = Port is not suspended 1 = Port is suspended SetPortSuspend (write) • Writing 1 causes HCD to set PSS bit. • Writing 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PSS. Instead it sets ConnectStatusChange. This notifies the driver an attempt was made to suspend a disconnected port.

Bits	Name	Description
30	PES	PortEnableStatus (read)—indicates whether the port is enabled or disabled. The Root Hub may clear this bit when the following conditions are detected: • an overcurrent condition • disconnect event • switched-off power • operational bus error (such as babble) This change causes PESC to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. PES cannot be set when CurrentConnectStatus is cleared. If not already set, PES is set at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled 1 = port is enabled SetPortEnable (write)—HCD sets PES by writing 1. Writing 0 has no effect. If CCS is cleared, this write does not set PES, but instead sets CSC. This notifies the driver that an attempt was made to enable a disconnected port.
31	CCS	CurrentConnectStatus (read)—reflects current state of downstream port. 0 = No device connected 1 = Device connected ClearPortEnable (write)—HCD writes 1 to this bit to clear PortEnableStatus bit. Writing 0 has no effect. CCS is not affected by any write. **Note:** This bit is always read '1b' when the attached device is non-removable (DeviceRemoveable[NDP]).

12.4.5.5 USB HC Rh Port2 Status Register—MBAR + 0x1058

This register is controls and reports port events on a per-port basis. The Number of Downstream Ports (NDP) represents the number of HcRhPortStatus registers that are implemented in hardware. The lower word is used to reflect the port status; the upper word reflects the status change bits. MPC5200 has NDP = 2, therefore, HcRhPort1Status (MBAR + 1054) and HcRhPort2Status (MBAR + 1058).

Some status bits are implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change is postponed until the transaction completes. Reserved bits should always be written 0.

Table 12-23. USB HC Rh Port2 Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved											PRSC	OCIC	PSSC	PESC	CSC
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 lsb
R	Reserved						LSDA	PPS	Reserved			PRS	POCI	PSS	PES	CCS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:10	—	Reserved
11	PRSC	PortResetStatusChange—bit is set at the end of the 10ms port reset signal. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = Port reset not complete 1 = Port reset complete
12	OCIC	PortOverCurrentIndicatorChange—bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when Root Hub changes the PortOverCurrentIndicator bit. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = No change in POCI 1 = POCI has changed
13	PSSC	PortSuspendStatusChange—bit is set when the full resume sequence completes. Sequence includes a 20s resume pulse, LS EOP, and 3ms resychronization delay. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. This bit is also cleared when ResetStatusChange is set. 0 = Resume not complete 1 = Resume complete
14	PESC	PortEnableStatusChange—bit is set when hardware events cause the PES bit to be cleared. Changes from HCD writes do not set this bit. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. 0 = No change in PES 1 = Change in PES
15	CSC	ConnectStatusChange—bit is set whenever a connect or disconnect event occurs. • Writing 1 causes HCD to clear this bit. • Writing 0 has no effect. If CCS is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. 0 = No change in CCS 1 = Change in CCS If the DeviceRemovable[NDP] bit is set, this bit is set only after a Root Hub reset to notify the system that the device is attached.
16:21	—	Reserved
22	LSDA	LowSpeedDeviceAttached (read)—bit indicates the speed of the device attached to this port. 0 = Full speed device attached 1 = Low speed device attached This field is valid only when CurrentConnectStatus is set. ClearPortPower (write) • Writing 1 causes HCD to clear the PortPowerStatus bit. • Writing 0 has no effect.

Bits	Name	Description
23	PPS	PortPowerStatus (read)—bit reflects the port power status, regardless of the type of power switching implemented. If an overcurrent condition is detected, this bit is cleared. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode and PortPortControlMask[NDP]. In global switching mode (PSM=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PSM=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. If port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. 0 = Port power is off 1 = Port power is on SetPortPower (write) • Writing causes HCD to set the PortPowerStatus bit. • Writing 0 has no effect. If power switching is not supported, this bit always reads ‘1b’.
24:26	—	Reserved
27	PRS	PortResetStatus (read)—When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. 0 = Port reset signal is not active 1 = Port reset signal is active SetPortReset (write) • Writing 1 causes HCD to set port reset signaling. • Writing 0 has no effect. If CurrentConnectStatus is cleared, a write does not set PortResetStatus. Instead, it sets ConnectStatusChange. This notifies the driver that an attempt was made to reset a disconnected port.
28	POCI	PortOverCurrentIndicator (read)—bit is only valid when root hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal 0 = No overcurrent condition. 1 = Overcurrent condition detected. ClearSuspendStatus (write) • Writing 1 causes HCD to initiate a resume. • Writing 0 has no effect. A resume is initiated only if PSS is set.

Bits	Name	Description
29	PSS	PortSuspendStatus (read)—bit indicates port is suspended or in resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CCS. This bit is cleared when: • PortResetStatusChange is set at the end of the port reset, or • when HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. 0 = Port is not suspended 1 = Port is suspended SetPortSuspend (write) • Writing 1 causes HCD to set PSS bit. • Writing 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PSS. Instead it sets ConnectStatusChange. This notifies the driver an attempt was made to suspend a disconnected port.
30	PES	PortEnableStatus (read)—indicates whether the port is enabled or disabled. The Root Hub may clear this bit when the following conditions are detected: • an overcurrent condition • disconnect event • switched-off power • operational bus error (such as babble) This change causes PESC to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. PES cannot be set when CurrentConnectStatus is cleared. If not already set, PES is set at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled 1 = port is enabled SetPortEnable (write)—HCD sets PES by writing 1. Writing 0 has no effect. If CCS is cleared, this write does not set PES, but instead sets CSC. This notifies the driver that an attempt was made to enable a disconnected port.
31	CCS	CurrentConnectStatus (read)—reflects current state of downstream port. 0 = No device connected 1 = Device connected ClearPortEnable (write)—HCD writes 1 to this bit to clear PortEnableStatus bit. Writing 0 has no effect. CCS is not affected by any write. **Note:** This bit is always read ‘1b’ when the attached device is non-removable (DeviceRemoveable[NDP]).

Notes

Chapter 13
BestComm

13.1 Overview

The following sections are contained in this document:

- Section 13.2, *BestComm Functional Description*
- Section 13.12, *BestComm DMA Registers—MBAR + 0x1200*
- Section 13.13, *On-Chip SRAM*

BestComm provides an efficient, integrated approach to gathering and manipulating data sets from a broad range of communication interfaces. BestComm consists of:

- BestComm (based on the SmartDMA [SDMA] module), with interfaces to:
  - peripherals using the CommBus,
  - the processor using an IP bus,
  - the system main SDRAM via the processor bus interface (XLB),
- a defined set of communication-oriented peripherals,
- local buffer memory,
- standard bus interfaces.

The Direct Memory Access controller (DMA) module provides a flexible and efficient means to move blocks of data within the system. The DMA controller reduces the workload on the microprocessor, allowing it to continue execution of system software. The DMA microcode engine is tailored to efficiently transfer data across the internal bus architecture to memory and peripheral devices.

The DMA controller processes microcode tasks that are stored in local memory (SRAM 16 kBytes). A task is a sequence of instructions, referred to as *descriptors*, that specifies a series of data movements or manipulations. The DMA controller steps through the descriptors and executes the specified function in a similar fashion to a CPU executing a program.

For the MPC5200, BestComm consists of SDMA and the following peripheral interfaces:

- 10/100 Fast Ethernet Controller (FEC)
- I^2C
- PCI
- ATA
- LocalPlus
- Peripheral Serial Controller (implementing a different mix of functionalities such as SPI, UART, CODEC 8-16-32 bits, AC97 controller, I2S, IrDA controller)

Many of the peripherals' port pins serve multiple functions, allowing flexibility in optimizing the system to meet a specific set of integration requirements. For a description of the pin multiplexing scheme and supported functions, refer to Chapter 2, *Signal Descriptions*.

Other peripheral functions are included in MPC5200, but are not directly supported by BestComm. These peripherals include:

- A separate Serial Peripheral Interface (SPI), which:
  - supports a 6.25 MHz rate as a master
  - supports a 12.5 MHz rate as a slave
- USB Host/Hub controller
- MSCAN controller
- General Purposes Timers

13.2 BestComm Functional Description

The BestComm I/O subsystem consists of the following:

- a BestComm DMA Controller
- an on-chip 16 kBytes SRAM
- a set of peripheral interface modules with DMA controllable:
  - transmit (Tx)
  - receive (Rx)

The BestComm unit provides an interrupt control and data movement interface. The Interface is on a separate peripheral bus to several on-chip peripheral functions. This independent control of data movement leaves the G2_LE core free to concentrate on higher level activities, which increases overall system performance.

BestComm DMA can control data movement on the following peripherals and interfaces:

- PCI bus
- ATA Controller
- Ethernet
- PSC
- I^2C
- IrDA
- LP bus interface

BestComm DMA performs general purpose DMA transfers. Most data transactions are between the peripheral/interface (typically a FIFO) and the system SDRAM.

BestComm allows up to 16 tasks to run simultaneously under the control of up to 32 DMA hardware requestors, user selectable from a possible 64 DMA request sources.

A hardware logic unit capable of basic logic operations (boolean arbitrary operations, shift, byte swap) plus some precoded CRC (CRC-16, CRC-CCITT, CRC-32, Internet Checksum) is also integrated in the SDMA engine.

BestComm uses internal buffers to prefetch reads and post writes such that bursting is used whenever possible. This optimizes both internal and external bus activity.

Speculative reads from system SDRAM may also be enabled to increase performance.

FIFO interfaces are implemented between the DMA and each peripheral/interface. As FIFOs are filled or emptied, automatic requests are made to the DMA unit. Based on programmable water mark levels (called ALARM and GRANULARITY level), the DMA unit moves data to and from the FIFOs. This method insures uninterrupted data movement at the given peripheral/interface rate.

13.3 Features summary

- A programmatic, deterministic capability for managing bus resources while servicing many data streams with individual latency and processing requirements.
- Single cycle access of peripheral and memory data.
- Support for up to 16 simultaneously enabled tasks (channels).
- Support for up to 32 separate DMA requestors at a time, user selectable from a possible 64 DMA request sources.
- Support for operations with up to 12 sources, or 11 sources and 1 destination.
- Simultaneous 32-bit reads and writes.
- Checksum generation.
- Endian conversion.
- Chaining/Scatter-gather capability.
- Support for packet-based I/O protocols (limitation might be dictated by performance when too much control is implemented within the task).

13.4 Descriptors

The DMA controller interprets a series of descriptors that specifies a sequence of data movements and manipulations. A collection of these descriptors is much like a program. The two types of descriptors are Loop Control Descriptors (LCDs) and Data Routing Descriptors (DRDs). These descriptors allow a “for”-loop programming style for the SDMA engine.

The LCDs specify the index variables (memory pointers, byte counters, etc.) along with the termination and increment values, while the DRDs specify the nature of the operation to perform.

13.5 Tasks

A task is a microcode program that embodies a desired function. An example could be to gather an ethernet frame, store it in memory and interrupt the processor when done. The multi-channel DMA supports sixteen simultaneously enabled tasks. By dynamically swapping task pointers in the task table, an unlimited number of tasks could be supported.

13.6 Memory Map/ Register Definitions

Memory organization is described in the register array pointed to by the Task Base Address Register (TaskBAR).

The TaskBAR identifies a location for a table of pointers to multi-channel DMA tasks (Task TABLE or Entry Table).

Each task has an entry (8 long words) that contains information about the microcode’s location (start address and stop address) in memory as well as pointers to the variable table to be used in the task, the Function Descriptor Table for the logic functions used within the task, the

Context Save area used during task switch/swap and some specific flags to enable performance affecting modes such as speculative reads, prefetch enable, readline and combined write.

A task's code should always be loaded into SRAM as the SDMA engine can fetch its descriptors from this internal memory with one cycle access per instruction. It is not recommended to place the code in SDRAM as there will then be a few overhead clocks which are needed to load the SDMA instruction unit.

13.7 Task Table (Entry Table)

The Task Table (or Entry Table) is a memory region containing pointers to each SDMA task. A Task Table Base Address Register (taskBAR) sets the location of the Task Table itself. Each entry in the Task Table contains pointers to the task's first descriptor, last descriptor, Variable Table, and other task-specific information.

13.8 Task Descriptor Table

Each Task Descriptor Table is a memory region containing the descriptors that comprise the task. The pointers in the Task Table define the beginning and end of each Task Descriptor Table.

13.9 Variable Table

Each task has a private 32-word Variable Table, where a word is four bytes (32 bits). According to the application requirements, the user initializes some of the words in the Variable Table as follows. The first 24 words are for pointers, counter values and initial data. The DMA Engine manipulates these variables as it executes loops. The next 8 words hold words-aligned, two-byte ("short word" or 16 bit word) increment variables.

13.10 Function Descriptor Table

An area of 256 bytes divided in 4 groups of 64 bytes. Each group can represent a set of 16 different Logic Functions belonging to a single execution unit. Every function is encoded with a single word (32 bits).

The implemented SDMA engine uses only one out of four potential Execution Units, execution unit 3, so all the functions needed by the task will be encoded in the third group (starting at offset 0xC0 from the start address of the Function Descriptor Table). The other words are reserved and must be written to '0' to maintain memory alignment.

For space optimization, tasks which use the same logic functions could share a single Function Descriptor Table avoiding the redundancy of re-writing the same table many times in SRAM.

13.11 Context Save Area

This is an area allocated for each task to allow the SDMA engine to save vital data (such as index values, etc.) during a task switch operation to allow later restoration.

The context save area should never be used or modified by the user as it is managed directly by the SDMA engine.

13.12 BestComm DMA Registers—MBAR+0x1200

A register overview is provided in Section 3.2, *Internal Register Memory Map*.

Hyperlinks to the BestComm DMA registers are provided below:

13.12.1 SDMA Task Bar Register—MBAR + 0x1200

sdma_task_bar_register

Table 13-1. SDMA Task Bar Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	taskBar															
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 lsb
R	taskBar															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	taskBar	TaskBAR is the pointer to the base address of the Task Table (Entry Table)

13.12.2 SDMA Current Pointer Register—MBAR + 0x1204

Table 13-2. SDMA Current Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CurrentPointer															
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 lsb
R	CurrentPointer															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	currentPointer	CurrentPointer contains the address of the currently executing DMA descriptor.

13.12.3 SDMA End Pointer Register—MBAR + 0x1208

Table 13-3. SDMA End Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EndPointer															
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 lsb
R	EndPointer															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	endPointer	EndPointer contains the address of the last descriptor in the currently executing SDMA task.

13.12.4 SDMA Variable Pointer Register—MBAR + 0x120C

Table 13-4. SDMA Variable Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VariablePointer															
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 lsb
R	VariablePointer															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	variablePointer	VariablePointer contains the starting address of the variable table for the currently executing task.

13.12.5 SDMA Interrupt Vector, PTD Control Register—MBAR + 0x1210

Table 13-5. SDMA Interrupt Vector, PTD Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Vector A[7:6]				INA[3:0]				Vector B[7:6]				INB[3:0]			
W																
RESET:	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	T/I	TEA	HE	Reserved												PE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IntVect1	The Interrupt Vector register is used during interrupt acknowledge read cycles. The high order four bits are programmed by the user, and the low order four bits are decoded from either the current task number or execution unit. If any task interrupts are asserted, Interrupt Vector 1 is driven during the interrupt acknowledge cycle. If the task interrupts are negated and the execution unit interrupts are asserted, Interrupt Vector 2 is driven during the interrupt acknowledge cycle. The registers are set to the uninitialized vector $0F by system reset. The interrupt A number is prioritized with IPR[15] the highest and IPR[0] the lowest. If all interrupt mask bits are set, then INA[3:0] = 1111 is read from this location. The interrupt B number is prioritized with the dbgInterrupt as the highest and euInterrupt[0] the lowest. If all interrupt mask bits are set, then INB[3:0] = 1111 is read from this location.
8:15	IntVect2	See above
16	T/I	T/I: Task/Iniator priority. Set to ‘1’ to switch to “TASK priority” control; set to ‘0’ to revert to INITIATOR (Requestor) Priority mode. The priority level of either the TASK or the initiator is set in the register IPR0 through IPR31
17	TEA	TEA: If set to ‘1’ a TEA received by BestComm will be ignored and the task will NOT be halted. TEA indication can still trigger an interrupt if the proper mask bit is cleared in the Interrupt Mask Register and the TEA status bit plus the TASK number of the task which received the TEA are still updated in the Interrupt Pending Register.
18	HE	HE = 1; allows smartDMA higher task number same request priority to block current task, and allow arbitration. HE = 0; disables higher task number from blocking. This bit is cleared by reset.
19:30	—	Reserved
31	PE	Prefetch Disable: set to ‘1’ to disable prefetch. Set to ‘0’ to enable prefetch on CommBus

13.12.6 SDMA Interrupt Pending Register—MBAR + 0x1214

Table 13-6. SDMA Interrupt Pending Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	Rsvd		TEA	Etn[3:0]				EU[7:0]							
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 lsb
R	TASK[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	DBG	Debug
1:2	—	Reserved
3	TEA	A TEA has been received by the currently running task. The corresponding task number is written in the Error Task Number field
4:7	Etn[3:0]	Error Task Number: when a TEA is received by the currently executing task its corresponding number is indicated here . If the TEA bit of the PtdControl register is set then the task will **not** be halted. If the TEA Msk bit in the Mask register is set then **no** interrupt to the core will be generated.
8:15	EU[7-0]	Execution Unit: only EU3 is valid for MPC5200
16:31	TASK[15:0]	Each bit corresponds to an interrupt source defined by the task number or execution unit. This register contains a registered copy of the interrupt signal that the interrupting source generates. The corresponding bit in the register reflects the state of the interrupt signal even if the corresponding mask bit is set. An interrupt is masked by setting the corresponding bit in the IntMask register. A bit is cleared by writing 1 to that bit location. Writing 0 has no effect. At system reset, all bits are initialized to logic 0. 0 = The corresponding interrupt source is **not** pending. 1 = The corresponding interrupt source is pending.

13.12.7 SDMA Interrupt Mask Register—MBAR + 0x1218

Table 13-7. SDMA Interrupt Mask Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	Reserved		TEA Msk	Reserved				EU[7:0]							
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	TASK[15:0]															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0	DBG	Debug: set to ‘1’ to mask the “debug” interrupt (see the SDMA Debug Control Register)
1:2	—	Reserved
3	TEA Msk	TEA Mask: set to ‘1’ to mask the TEA. If set to ‘1’ and a TEA is received in the currently executing Task an interrupt is generated.
4:7	—	Reserved

Bit	Name	Description
8:15	EU[x]	Execution Unit: Only EU3 is present in MPC5200
16:31	TASK[15:0]	Each bit corresponds to an interrupt source defined by the task number or execution unit. An interrupt is masked by setting the corresponding bit. At system reset, all bits are initialized to logic 1. 0 = The corresponding interrupt source is **not** masked. 1 = The corresponding interrupt source is masked (no interrupt is generated).

13.12.8 SDMA Task Control 0 Register—MBAR + 0x121C
SDMA Task Control 1 Register—MBAR + 0x121E

Table 13-8. SDMA Tas k Control 0 Register
SDMA Task Control 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	En	Val	Alw Init	IN[4:0]					Auto Start	High En	Hold	Rsvd	AS [3:0]			
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 lsb
R	TCR1 (same as for TCR0)															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	EN	Each of the sixteen tasks has an associated task control register. Only one register is shown. At system reset, all bits are initialized to logic zeros. Enable - Task Enable 0 = Disabled 1 = Enabled This bit can be set or cleared by the programmer at any time when a task is enabled or disabled. This bit is also set by the PTD logic if the auto-restart bit is set and the task completes.
1	Val	Valid - Initiator Number is Valid 0 = Initiator is not valid 1 = Initiator is valid This bit is set by the engine logic when it obtains the requestor value from the first DRD that is parsed. This bit is cleared by the logic when the task completes. At system reset, this bit is cleared.
2	Alw Init	Always Init - Decode of the always initiator 0 = The always initiator is not being used 1 = The always initiator is being used This bit is a status bit only and is set and cleared by writing the initiator number into the Task Control Register.

Bit	Name	Description
3:7	IN[4:0]	InitNum[4:0] - Initiator number from task descriptor These bits are registered when the SDMA engine has parsed the first DRD to obtain the requestor number. These bits are cleared by system reset. These bits can be written by the programmer when the Hold Init Num bit is set or being set and the task is not enabled. At system reset, these bits are cleared.
8	Auto Start	Auto-Start - Task Start 0 = Task will not restart within program control 1 = Task will restart at end of task automatically. This bit can be set or cleared by the programmer at any time. This bit is also cleared if the SDMA engine encounters an error in the task. At system reset, this bit is cleared.
9	High En	High-Enable - High Priority Task Enable 0 = Normal task enable control 1 = High priority task enable control This bit can be set or cleared by the programmer at any time. This bit enables the SDMA to give priority to the enabled task function over running a task. At system reset, this bit is cleared.
10	Hold	Hold Init Num- Hold initiator number 0 = Allow the SDMA engine to update initiator number for task 1 = Keep current initiator number. This bit allows the initiator number to be set by the programmer and held for the complete task. The SDMA can not overwrite the programmed initiator except for the use of the always initiator which is contained in a separate control bit.
11	—	Reserved
12-15	AS[3:0]	ASNum[3:0] - Auto-Start Task Number These four bits contain the task number which will be auto-started when the Auto-Start control bit is set. At system reset, these bits are cleared.
16:31	TCR1	Task control register for task 1. Same bit layout as for TCR0

13.12.9 SDMA Task Control 2 Register—MBAR + 0x1220
SDMA Task Control 3 Register—MBAR + 0x1222

Table 13-9. SDMA Task Control 2 Register
SDMA Task Control 3 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR2															
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 lsb
R	TCR3															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCR2	Task control register for task 2. Same bit layout as for TCR0
16:31	TCR3	Task control register for task 3. Same bit layout as for TCR0

13.12.10 SDMA Task Control 4 Register—MBAR + 0x1224
SDMA Task Control 5 Register—MBAR + 0x1226

Table 13-10. SDMA Task Control 4 Register
SDMA Task Control 5 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR4															
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 lsb
R	TCR5															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCR4	Task control register for task 4. Same bit layout as for TCR0
16:31	TCR5	Task control register for task 5. Same bit layout as for TCR0

13.12.11 SDMA Task Control 6 Register—MBAR + 0x1228
SDMA Task Control 7 Register—MBAR + 0x122A

Table 13-11. SDMA Task Control 6 Register
SDMA Task Control 7 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR6															
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 lsb
R	TCR7															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCR6	Task control register for task 6. Same bit layout as for TCR0
16:31	TCR7	Task control register for task 7. Same bit layout as for TCR0

13.12.12 SDMA Task Control 8 Register—MBAR + 0x122C
SDMA Task Control 9 Register—MBAR + 0x122E

Table 13-12. SDMA Task Control 8 Register
SDMA Task Control 9 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR8															
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 lsb
R	TCR9															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCR8	Task control register for task 8. Same bit layout as for TCR0
16:31	TCR9	Task control register for task 9. Same bit layout as for TCR0

13.12.13 SDMA Task Control A Register—MBAR + 0x1230
SDMA Task Control B Register—MBAR + 0x1232

Table 13-13. SDMA Task Control A Register
SDMA Task Control B Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCRA															
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 lsb
R	TCRB															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCRA	Task control register for task 10. Same bit layout as for TCR0
16:31	TCRB	Task control register for task 11. Same bit layout as for TCR0

13.12.14 SDMA Task Control C Register—MBAR + 0x1234
SDMA Task Control D Register—MBAR + 0x1236

Table 13-14. SDMA Task Control C Register
SDMA Task Control D Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCRC															
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 lsb
R	TCRD															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCRC	Task control register for task 12. Same bit layout as for TCR0
16:31	TCRD	Task control register for task 13. Same bit layout as for TCR0

13.12.15 SDMA Task Control E Register—MBAR + 0x1238
SDMA Task Control F Register—MBAR + 0x123C

Table 13-15. SDMA Task Control E Register
SDMA Task Control F Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCRE															
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 lsb
R	TCRF															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCRE	Task control register for task 14. Same bit layout as for TCR0
16:31	TCRF	Task control register for task 15. Same bit layout as for TCR0

13.12.16 SDMA Initiator Priority 0 Register—MBAR + 0x123C
SDMA Initiator Priority 1 Register—MBAR + 0x123D
SDMA Initiator Priority 2 Register—MBAR + 0x123E
SDMA Initiator Priority 3 Register—MBAR + 0x123F

Table 13-16. SDMA Initiator Priority 0 Register
SDMA Initiator Priority 1 Register
SDMA Initiator Priority 2 Register
SDMA Initiator Priority 3 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR0 Hold	Reserved				Prior [2:0]			IPR1							
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 lsb
R	IPR2								IPR3							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
Each of the thirty-two initiators has an associated priority level. Only one register is shown. All bits are set to ‘0 at reset.		
0	IPR0 Hold	Hold - Keep current priority of initiator 0 = Allow higher priority initiator to block current initiator 1 = Hold current initiator priority level This bit can be set or cleared by the programmer at any time. This bit allows the current initiator to hold priority until the initiator has negated or the task has finished. When this bit is cleared, an initiator with a higher priority will block the current initiator and force arbitration. At system reset, this bit is cleared.
1:4	—	Reserved
5:7	Prior[2:0]	InitPrior[2:0] - Initiator/Task priority level. These bits can be set by the programmer at any time. These bits control the priority of the requestor/task which will be serviced next depending on the setting of the T/I bit in the PtdControl register. The highest priority is level 7. The lower priority is level 0.. If more than one initiator/task contains the same priority then the order of the task within the Task table (task 7 highest to task 0 lowest) will set the priority.
8:15	IPR1	Initiator Priority register for initiator 1 (or Task1 if PtdControl[16]=1). Same bit layout as IPR0
16:23	IPR2	Initiator Priority register for initiator 2.(or Task2 if PtdControl[16]=1) Same bit layout as IPR0
24:31	IPR3	Initiator Priority register for initiator 3.(or Task3 if PtdControl[16]=1) Same bit layout as IPR0

13.12.17 SDMA Initiator Priority 4 Register—MBAR + 0x1240
SDMA Initiator Priority 5 Register—MBAR + 0x1241
SDMA Initiator Priority 6 Register—MBAR + 0x1242
SDMA Initiator Priority 7 Register—MBAR + 0x1243

Table 13-17. SDMA Initiator Priority 4 Register
SDMA Initiator Priority 5 Register
SDMA Initiator Priority 6 Register
SDMA Initiator Priority 7 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR4								IPR5							
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 lsb
R	IPR6								IPR7							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR4	Initiator Priority register for initiator 4 (or Task4 if PtdControl[16]=1) Same bit layout as IPR0
8:15	IPR5	Initiator Priority register for initiator 5 (or Task5 if PtdControl[16]=1) Same bit layout as IPR0
16:23	IPR6	Initiator Priority register for initiator 6 (or Task6 if PtdControl[16]=1) Same bit layout as IPR0
24:31	IPR7	Initiator Priority register for initiator 7 (or Task7 if PtdControl[16]=1) Same bit layout as IPR0

13.12.18 SDMA Initiator Priority 8 Register—MBAR + 0x1244
SDMA Initiator Priority 9 Register—MBAR + 0x1245
SDMA Initiator Priority 10 Register—MBAR + 0x1246
SDMA Initiator Priority 11 Register—MBAR + 0x1247

Table 13-18. SDMA Initiator Priority 8 Register
SDMA Initiator Priority 9 Register
SDMA Initiator Priority 10 Register
SDMA Initiator Priority 11 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR8								IPR9							
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 lsb
R	IPR10								IPR11							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR8	Initiator Priority register for initiator 8 (or Task8 if PtdControl[16]=1) Same bit layout as IPR0
8:15	IPR9	Initiator Priority register for initiator 9 (or Task9 if PtdControl[16]=1) Same bit layout as IPR0
16:23	IPR10	Initiator Priority register for initiator 10 (or Task10 if PtdControl[16]=1) Same bit layout as IPR0
24:31	IPR11	Initiator Priority register for initiator 11 (or Task11 if PtdControl[16]=1) Same bit layout as IPR0

13.12.19 SDMA Initiator Priority 12 Register—MBAR + 0x1248
SDMA Initiator Priority 13 Register—MBAR + 0x1249
SDMA Initiator Priority 14 Register—MBAR + 0x124A
SDMA Initiator Priority 15 Register—MBAR + 0x124B

Table 13-19. SDMA Initiator Priority 12 Register
SDMA Initiator Priority 13 Register
SDMA Initiator Priority 14 Register
SDMA Initiator Priority 15 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR12								IPR13							
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 lsb
R	IPR14								IPR15							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR12	Initiator Priority register for initiator 12 (or Task12 if PtdControl[16]=1) Same bit layout as IPR0
8:15	IPR13	Initiator Priority register for initiator 13 (or Task13 if PtdControl[16]=1) Same bit layout as IPR0

Bit	Name	Description
16:23	IPR14	Initiator Priority register for initiator 14 (or Task14 if PtdControl[16]=1) Same bit layout as IPR0
24:31	IPR15	Initiator Priority register for initiator 15 (or Task15 if PtdControl[16]=1) Same bit layout as IPR0

13.12.20 SDMA Initiator Priority 16 Register—MBAR + 0x124C
SDMA Initiator Priority 17 Register—MBAR + 0x124D
SDMA Initiator Priority 18 Register—MBAR + 0x124E
SDMA Initiator Priority 19 Register—MBAR + 0x124F

Table 13-20. SDMA Initiator Priority 16 Register
SDMA Initiator Priority 17 Register
SDMA Initiator Priority 18 Register
SDMA Initiator Priority 19 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR16								IPR17							
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 lsb
R	IPR18								IPR19							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR16	Initiator Priority register for initiator 16. Same bit layout as IPR0
8:15	IPR17	Initiator Priority register for initiator 17. Same bit layout as IPR0
16:23	IPR18	Initiator Priority register for initiator 18. Same bit layout as IPR0
24:31	IPR19	Initiator Priority register for initiator 19. Same bit layout as IPR0

13.12.21 SDMA Initiator Priority 20 Register—MBAR + 0x1250
SDMA Initiator Priority 21 Register—MBAR + 0x1251
SDMA Initiator Priority 22 Register—MBAR + 0x1252
SDMA Initiator Priority 23 Register—MBAR + 0x1253

Table 13-21. SDMA Initiator Priority 20 Register
SDMA Initiator Priority 21 Register
SDMA Initiator Priority 22 Register
SDMA Initiator Priority 23 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR20								IPR21							
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 lsb
R	IPR22								IPR23							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR20	Initiator Priority register for initiator 20. Same bit layout as IPR0
8:15	IPR21	Initiator Priority register for initiator 21. Same bit layout as IPR0
16:23	IPR22	Initiator Priority register for initiator 22. Same bit layout as IPR0
24:31	IPR23	Initiator Priority register for initiator 23. Same bit layout as IPR0

13.12.22 SDMA Initiator Priority 24 Register—MBAR + 0x1254
SDMA Initiator Priority 25 Register—MBAR + 0x1255
SDMA Initiator Priority 26 Register—MBAR + 0x1256
SDMA Initiator Priority 27 Register—MBAR + 0x1257

Table 13-22. SDMA Initiator Priority 24 Register
SDMA Initiator Priority 25 Register
SDMA Initiator Priority 26 Register
SDMA Initiator Priority 27 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR24								IPR25							
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 lsb
R	IPR26								IPR27							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR24	Initiator Priority register for initiator 24. Same bit layout as IPR0
8:15	IPR25	Initiator Priority register for initiator 25. Same bit layout as IPR0
16:23	IPR26	Initiator Priority register for initiator 26. Same bit layout as IPR0
24:31	IPR27	Initiator Priority register for initiator 27. Same bit layout as IPR0

13.12.23 SDMA Initiator Priority 28 Register—MBAR + 0x1258
SDMA Initiator Priority 29 Register—MBAR + 0x1259
SDMA Initiator Priority 30 Register—MBAR + 0x125A
SDMA Initiator Priority 31 Register—MBAR + 0x125B

Table 13-23. SDMA Initiator Priority 28 Register
SDMA Initiator Priority 29 Register
SDMA Initiator Priority 30 Register
SDMA Initiator Priority 31 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPR28								IPR29							
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 lsb
R	IPR30								IPR31							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR28	Initiator Priority register for initiator 28. Same bit layout as IPR0
8:15	IPR29	Initiator Priority register for initiator 29. Same bit layout as IPR0

Bit	Name	Description
16:23	IPR30	Initiator Priority register for initiator 30. Same bit layout as IPR0
24:31	IPR31	Initiator Priority register for initiator 31. Same bit layout as IPR0

13.12.24 SDMA Requestor MuxControl—MBAR + 0x125C

Table 13-24. SDMA Request MuxControl

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Req31		Req30		Req29		Req28		Req27		Req26		Req25		Req24	
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 lsb
R	Req23		Req22		Req21		Req20		Req19		Req18		Req17		Req16	
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	Req31	00: Requestor (RESERVED) 01-10: No active requestor 11: Always Requestor 31
2:3	Req30	00: Requestor (RESERVED) 01-10: No active requestor 11: Always Requestor 30
4:5	Req29	00: Requestor (RESERVED) 01-10: No active requestor 11: Always Requestor 29
6:7	Req28	00: Requestor (RESERVED) 01-10: No active requestor 11: Always Requestor 28
8:9	Req27	00: Requestor (RESERVED) 01-10: No active requestor 11: Always Requestor 27
10:11	Req26	00: Requestor IrDA TX (PSC_6) 01-10: No active requestor 11: Always Requestor 26
12:13	Req25	00: Requestor IrDA RX (PSC_6) 01-10: No active requestor 11: Always Requestor 25

Bit	Name	Description
14:15	Req24	00: Requestor I2C1_TX 01-10: No active requestor 11: Always Requestor 24
16:17	Req23	00: Requestor I2C1_RX 01-10: No active requestor 11: Always Requestor 23
18:19	Req22	00: Requestor I2C2_TX 01-10: No active requestor 11: Always Requestor 22
20:21	Req21	00: Requestor I2C2_RX 01-10: No active requestor 11: Always Requestor 21
22:23	Req20	00: Requestor PSC4_TX 01-10: No active requestor 11: Always Requestor 20
24:25	Req19	00: Requestor PSC4_RX 01-10: No active requestor 11: Always Requestor 19
26:27	Req18	00: Requestor PSC5_TX 01-10: No active requestor 11: Always Requestor 18
28:29	Req17	00: Requestor PSC5_RX 01-10: No active requestor 11: Always Requestor 17
30:31	Req16	00: Requestor LP 01-10: No active requestor 11: Always Requestor 16

The remaining 16 Requestors are fixed as follows:

Table 13-25. FIxed REquestors Table

REQUESTORS	Peripheral
REQ15	(RESERVED)
REQ14	PSC1_TX
REQ13	PSC1_RX
REQ12	PSC2_TX
REQ11	PSC2_RX
REQ10	PSC3_TX
REQ9	PSC3_RX
REQ8	PCI TX
REQ7	PCI RX
REQ6	ATA TX

Table 13-25. FIxed REquestors Table (continued)

REQUESTORS	Peripheral
REQ5	ATA RX
REQ4	FEC TX
REQ3	FEC RX
REQ2	(RESERVED)
REQ1	(RESERVED)
REQ0	ALWAYS

13.12.25 SDMA task Size0—MBAR + 0x1260
SDMA task Size 1—MBAR + 0x1264

Table 13-26. SDMA task Size 0/1

Bits	0,4,8,12, 16,20,24,28	1,5,9,13, 17,21,25,29	2,6,10,14, 18,22,26,30	3,7,1115 19,23,27,31
R	srcSize[1]	srcSize[0]	dstSize[1]	dstSize[0]
W				
RESET:	At reset all Bits are set to 0			

Bit	Name	Description
	srcSize[1:0]	Each of the 16 tasks can be programmed to use the source and destination sizes contained in one of the Task Size Registers. The task size information is used by the SDMA module to determine the source and destination transfer size of the operands. When the size contained the task descriptor is set to 2'b11 then the size field from the Task Size Control register is selected. srcSize[1:0] - source size 00 - Word (32 bit) 01 - Byte 10 - Word 11 - Word destSize[1:0] - destination size 00 - Word (32 bit) 01 - Byte 10 - Word 11 - Word

13.12.26 SDMA task 0 & task Size 1 map

Table 13-27. SDMA task Size Map

Offset	Register Name	Byte 0	Byte 1	Byte 2	Byte 3	Access
0x1260	task Size 0	TS[0:1]	TS[2:3]	TS[4:5]	TS[5:7]	R/W
0x1264	task Size 1	TS[8:9]	TS[10:11]	TS[12:13]	TS[14:15]	R/W

Bit	Name	Description
		See Table 13-26 for details. Each task has 4 bits allocated (2 for source and 2 for destination Size)

13.12.27 SDMA Reserved Register 1—MBAR + 0x1268

Table 13-28. SDMA Reserved Register 4

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	res1															
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 lsb
R	res1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	res1	Reserved

13.12.28 SDMA Reserved Register 2—MBAR + 0x126C

Table 13-29. SDMA Reserved Register 2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	res2															
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 lsb
R	res2															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	res2	Reserved

13.12.29 SDMA Debug Module Comparator 1, Value1 Register—MBAR + 0x1270

Table 13-30. SDMA Debug Module Comparator 1, Value1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Value1															
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 lsb
R	Value1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	Value1	Debug Module Comparator 1 Value.

13.12.30 SDMA Debug Module Comparator 2, Value2 Register—MBAR + 0x1274

Table 13-31. SDMA Debug Module Comparator 2, Value2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Value2															
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 lsb
R	Value2															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	Value2	Debug Module Comparator 2 Value.

13.12.31 SDMA Debug Module Control Register—MBAR + 0x1278

Table 13-32. SDMA Debug Module Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Block Tasks															
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 lsb
R	AA	B	Comparator Type 1		Comparator Type 2				and/ or	EU breakpoints				E	I	B
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	Block Tasks	Specify for each of tasks 15-0, whether to block that task with detection of a breakpoint (bit 0 halts TASK 15, bit 1 halts TASK 14, etc) 0 Do not block task 1 Block the task
16	AA	AutoArm—specifies whether or not the triggered bit dbgStatusReg[16] will be automatically reset to 0 following the saving of context for a breakpoint. This bit is set to 0 at reset. 0 Triggered bit will not be automatically reset 1 Triggered bit will be automatically reset
17	B	Breakpoint—This bit specifies whether or not to take a breakpoint. This bit is set to 0 at reset. 0 Disable breakpoints 1 Enable breakpoints
18:20	Comparator Type 1	Comparator 1 type—These bits specify the type of data that has been loaded into comparator 1; refer to Table 13-33 for the bit encoding.
21:23	Comparators Type 2	Comparator 2 type—These bits specify the type of data that has been loaded into comparator 2; refer to Table 13-34 for the bit encoding.
24	and / or	AND/OR—This specifies what type of operation is to be used with the comparators. This bit is set to 0 at reset. 0 Indicates an OR'ing of the comparators 1 Indicates an AND'ing of the comparators
25:28	EU breakpoints	euBreakpoint: These bits indicate that a breakpoint has occurred in one of the four execution units. Each execution unit has one bit dedicated to it. A 1 in any of these bits indicates that the associated execution unit has issued breakpoint. These bits are sticky and must be overwritten to continue. These bits are cleared to zero at reset. See Table 13-35 for the bit encoding. MPC5200 has integrated only EU3
29	E	Enable interrupt breakpoint. 0 Do not enable external breakpoint to cause a halt condition 1 Allow external breakpoint to cause a halt condition
30	I	Enable internal breakpoint 0 Do not enable internal breakpoint to cause a halt condition 1 Allow external breakpoint to cause a halt condition
31	EB	Master External Breakpoint (this bit must be always set to allow any kind of breakpoint to halt the task) 0 Disable external breakpoint 1 Enable external breakpoint

Table 13-33. Comparator 1 Type Bit Encoding

Encoding	Comparator Type 1
000	uninitialized
001	write address
010	read address
011	current pointer

Table 13-33. Comparator 1 Type Bit Encoding (continued)

Encoding	Comparator Type 1
100	task #
101	reserved
110	reserved
111	reserved

Table 13-34. Comparator 2 Type Bit Encoding

Encoding	Comparator Type 1
000	uninitialized
001	write address
010	read address
011	current pointer
100	task #
101	counter value
110	reserved
111	reserved

The reserved encodings are set to 0 indicating an uninitialized state.

Table 13-35. EU Breakpoint encoding

	EU3	EU2	EU1	EU0
Reset	0	0	0	0

It must be noted that even if a breakpoint is issued at a specific address the SDMA engine will halt ONLY at a “data aligned” boundary (for instance, if the task moves 32 bits of data per transaction and a breakpoint is set at address 0x02 then the task will be halted at offset 0x04).

13.12.32 SDMA Debug Module Status Register—MBAR + 0x127C

Table 13-36. SDMA Debug Module Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved													I	E	T
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 lsb
R	dbgStatusReg[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:12	Reserved	Reserved
13	I	Interrupt—This bit indicates whether or not an interrupt has been taken. This bit is set to 0 at reset. It can be written by the user or the SDMA engine. 0 No Interrupt 1 Interrupt taken
14	E	External Breakpoint—This bit indicates detection of an external breakpoint. Status bit is sticky and requires a one (1) to be written to it to clear it. The writing of a zero (0) to this bit has no effect. This bit is set to zero (0) at reset. 0 No external breakpoint detected 1 External breakpoint detected
15	T	Triggered (dbgStatusReg[16])—This bit indicates that a SmartDMA breakpoint has occurred with the current settings. Status bit is sticky and requires a one (1) to be written to it to clear it. The writing of a zero (0) to this bit has no effect. This bit is set to zero (0) at reset. 0 Armed or normal operation 1 Triggered or debug mode
16:31	dbgStatusReg[15:0]	dbgTaskBlock (dbgStatusReg[15:0])—Each bit corresponds to one of the 16 task numbers. The value of the register bit reflects the debug state of the task number. A bit is cleared by writing a one to that bit location; writing a zero (0) has no effect. At system reset, all bits are initialized to logic zeros (0). 0 Unblocked or normal operation 1 Blocked, task has been blocked due to a breakpoint

13.13 On-Chip SRAM

MPC5200 contains 16KBytes of on-chip SRAM. This memory is directly accessible by the BestComm DMA unit. It is used primarily as storage for task table and buffer descriptors used by BestComm DMA to move peripheral data to and from SDRAM or other locations. These descriptors must be downloaded to the SRAM at boot.

This SRAM resides in the MPC5200 internal register space and is also accessible by the processor core. As such it can be used for other purposes, such as scratch pad storage. The 16kBytes SRAM starts at location MBAR + 0x8000.

13.14 Programming Model

The SDMA engine expects the programmer to initialize several things in memory including the Task Table and the Variable Table(s). These are described and illustrated in the following sub-sections. The various descriptors used in each task are also described below.

13.14.1 Task Table

The programmer must initialize the *taskBAR register* in the IPB Interface Module (offset 0x1200). The Task Table (sometime also referred to as Entry Table), whose format is shown in Figure 13-1, should reside at the address specified by *taskBAR*.

The Task Table base address must be aligned to a 512-byte boundary. There are sixteen tasks, each of which has its own unique Task Descriptor Table (TDT) start pointer, TDT end pointer, Variable Table pointer, control information, and status information. The TDT start pointer is a 32-bit value that points to the first descriptor, an LCD, of that particular task. The remaining descriptors (LCDs and DRDs) should consecutively follow the first one in memory, except in special branching cases. The TDT end pointer is a 32-bit value that points to the last descriptor, which must be a DRD, of that particular task.

The 32-bit Variable Table pointer points to the top of the 32-word (128 byte) memory space where this task's Variable Table resides. The Variable Table format is explained later in more detail.

The control information is located in the fourth word of each task's Task Table information as shown in Figure 13-1. Bits 0 through 23 contain the base address for this task's function descriptors. Control bits 24 through 31 are for precise increment, not resetting the error code, whether to pack data, integer mode, complex data mode, to enable speculative reads and whether bursting is allowed on reads and writes.

The fifth and sixth word of the task table are reserved.

The seventh word is a pointer to the Context Save Area where important data is saved and later restored in case of a task switch.

The last word is used by the SDMA engine in conjunction with Literal Initialization of LCD (to save variable usage). The user should not modify the values stored there.

	0	4 5	14 15	19 20	31
Task 0	Task Descriptor Start Pointer				
	Task Descriptor End Pointer				
	Variable Table Pointer				
	Function Descriptor Base Address				RSV PI E P I SPR CW RL
	Reserved				
	Reserved				
	Base Address for Context Save Space				
	Literal Base 0	Reserved	Literal Base 1	Reserved	
	.....				
Task n	Task Descriptor Start Pointer				
	Task Descriptor End Pointer				
	Variable Table Pointer				
	Function Descriptor Base Address				RSV PI E P I SPR CW RL
	Reserved				
	Reserved				
	Base Address for Context Save Space				

Note: For each task, the start pointer, end pointer, and variable table pointer are 32-bit values. For the task control bits, bits 0 through 23 are for the Function Descriptor Base Address, and bits 24 through 31 are RSV = Reserved, PI = Precise Increment, E = do not reset error code if ‘1’, P = Pack data if ‘1’, I = Integer mode if ‘1’ (else fractional), SPR = speculative enable, CW = Combined Write Enable if ‘1’, and RL = Read Line Buffer Enable if ‘1’

speculative Reads if ‘1’

Figure 13-1. Task Table

Table 13-37. Behavior of Task Table Control Bits

Control Function	Value	Meaning
Precise Increment	0	Increments are allowed at any time the SDMA can do it
	1	Only increment at the end of an iteration
No Error Code Reset	0	Reserved
	1	Reserved
Pack	0	Do not pack data
	1	Pack data
Integer Mode	0	Fractional data representation
	1	Integer data representation
Speculative Reads	0	Disabled
	1	Enabled
Combined Write Enable	0	Do not enable combined writes
	1	Enable combined writes
Read Line Buffer Enable	0	Do not enable line reads
	1	Enable line reads

13.14.1.1 Integer Mode

This input signal is only valid if the pack signal is negated (set to ‘0’). This signal indicates if the SDMA engine should operate in integer mode or fractional mode. During integer mode, the engine sign-extends read data and the it reads the write size amount of data starting from the MSB position and drives it to the proper destination byte lanes as indicated by the write address.

During fractional mode, the engine zero-extends read data and the ADS reads the write size amount of data starting from the LSB position and drives it to the proper write byte lanes as indicated by the write address.

13.14.1.2 Pack

This input signal indicates that packing or unpacking of data should occur if the read size does not equal the write size. The pack signal has precedence over the integerMode signal.

This signal indicates to the SmartDMA that it should pack data when the source size does not match the destination size. When this signal is asserted, the SmartDMA should pack data, and the *integerMode* signal is ignored. Otherwise, the SmartDMA should not pack data. Packing data refers to the case where the SmartDMA will wait for a full word of data before passing the data to one of the memory interfaces.

13.14.2 Variable Table

Table 13-38 shows the Variable Table format to which each task must adhere. The Variable Table pointer that is located in the Task Table in Figure 13-1 points to the first location in this 32-word (128-byte) memory space.

If restoring, and the Variable Table has been modified, then the new Variable Table pointer is located at the end of the context save space for the corresponding task. The Variable Table for each task must be aligned to a 16-byte boundary (to aid in address calculation). Before executing a particular task, that task’s Variable Table must be initialized with the appropriate data. Specifically, any constants, initial values, and increment values must be written to the Variable Table before executing the corresponding task.

Variables may be loaded into words 0 through 23.Increment values 0 through 7 may be loaded into words 24 through 31, respectively.

Any of the eight increment values may be used as normal Loop-Index Variables or Constants if they are not needed as increment values. All of these variables and increment values may be used to initialize loop-index registers. However, only variables 0 through 31 may be written by the ADS. At this time, if variables 24 through 31 are written, it is assumed that these variables should be treated as normal Loop-Index Variables or Constants and not as increment values. Also, note that Variable Tables may overlap if sharing the last eight variables with another task’s Variable Table is desired. In addition, if a task does not use the last 16 variables, another Variable Table could start immediately after that task’s increment values, so as to not waste memory.

Table 13-38. Variable Table per Task

#	Hex Offset	Contents	Comments
0	00	Loop-Index Variable or Constant 0	These twenty-four words (32 bits) are used for constant operands to the EUs, for initialization values, or for a place to write results straight to a variable in this table. These are typically preloaded by the CPU unless you are writing directly to a variable.
1	04	Loop-Index Variable or Constant 1	
2	08	Loop-Index Variable or Constant 2	
3	0c	Loop-Index Variable or Constant 3	
4	10	Loop-Index Variable or Constant 4	
5	14	Loop-Index Variable or Constant 5	
6	18	Loop-Index Variable or Constant 6	
7	1c	Loop-Index Variable or Constant 7	
8	20	Loop-Index Variable or Constant 8	
9	24	Loop-Index Variable or Constant 9	
10	28	Loop-Index Variable or Constant 10	
11	2c	Loop-Index Variable or Constant 11	
12	30	Loop-Index Variable or Constant 12	
13	34	Loop-Index Variable or Constant 13	
14	38	Loop-Index Variable or Constant 14	
15	3c	Loop-Index Variable or Constant 15	
16	40	Loop-Index Variable or Constant 16	
17	44	Loop-Index Variable or Constant 17	
18	48	Loop-Index Variable or Constant 18	
19	4c	Loop-Index Variable or Constant 19	
20	50	Loop-Index Variable or Constant 20	
21	54	Loop-Index Variable or Constant 21	
22	58	Loop-Index Variable or Constant 22	
23	5c	Loop-Index Variable or Constant 23	
24	60	Compare Type[31:29], Reserved[28:16], Increment Variable 0[15:0]	Variables 24 - 31 may be increment variables of the format shown to the left. Any of these variables may be used as normal Loop-Index Variables or Constants (like variables 0 - 23) instead.
25	64	Compare Type[31:29], Reserved[28:16], Increment Variable 1[15:0]	
26	68	Compare Type[31:29], Reserved[28:16], Increment Variable 2[15:0]	
27	6c	Compare Type[31:29], Reserved[28:16], Increment Variable 3[15:0]	
28	70	Compare Type[31:29], Reserved[28:16], Increment Variable 4[15:0]	
29	74	Compare Type[31:29], Reserved[28:16], Increment Variable 5[15:0]	
30	78	Compare Type[31:29], Reserved[28:16], Increment Variable 6[15:0]	
31	7c	Compare Type[31:29], Reserved[28:16], Increment Variable 7[15:0]	

When the user writes a program, or when the assembler converts the user's programs, the SDMA engine will use the initialization variables and constants that the user or processor should have loaded into the Variable Table. The initial index variables in the LCD tells the engine to allocate space for the resulting variables in the loop registers. The space will be allocated consecutively, so the user knows with which register each variable will be associated. This is important when the user's program tries to reference one of these previously allocated variables. Also, the eight increment variables in positions 24 through 31 of Table 13-38 are preloaded by the processor, as programmed by the user.

Notes

Chapter 14
Fast Ethernet Controller (FEC)

14.1 Overview

The fast Ethernet controller (FEC) is an ethernet MAC plus two 1 Kbyte FIFOs that work under the control of the processor and BestComm DMA engine to support 10/100 Mbps Ethernet/802.3 networks. Table 14-1 shows a block diagram.

A brief introduction and overview of the major functional blocks aid in understanding and programming the FEC.

The FEC is controlled by writing through the system interface (SIF) module into control registers located in each block. The control/status register (CSR) block provides global control and interrupt handling registers. User programming of the CSR is the primary focus of this chapter.

The RISC based controller provides the following functions:

- Initialization
- Address recognition for receive frames
- Random number generation for transmit collision backoff timer

The FIFO controller is the focal point of all data flow in the FEC. The FIFO is divided into a transmit and receive FIFO of 1Kbyte each. Transmit data flows from the CommBus into the transmit FIFO and through the transmit block to the physical layer device (PHY). Receive data flows from the PHY to the receive block and is pulled out of the FIFO by BestComm. BestComm data transfers are interrupt driven. Interrupt driven data movement from the processor is not supported.

The bus controller decides which block is to be the T-bus master for each cycle. All the blocks receive their control information over the T-bus and, for the most part, provide status information over this same internal bus.

The media independent interface (MII) block provides a serial channel for control/status communication with the external physical layer device (transceiver or PHY). The serial channel consists of the MDC (clock) and MDIO (bidirectional data I/O) lines of the MII interface.

The transmit and receive blocks provide the ethernet MAC functionality (with some assistance from the microcode). Internal to these blocks are clock domain boundaries between the system clock and the network clocks supplied by the PHY.

The management information base (MIB) block maintains the counters for a variety of network events and statistics. The counters support the RMON (RFC 1757) ethernet statistics group and some of the IEEE 802.3 counters.

The FEC supports several standard MAC-PHY interfaces to connect to an external ethernet transceiver. One is the 10/100 Mbps MII interface. Another is the 10-Mbps only 7-Wire interface, which uses a subset of the MII pins.

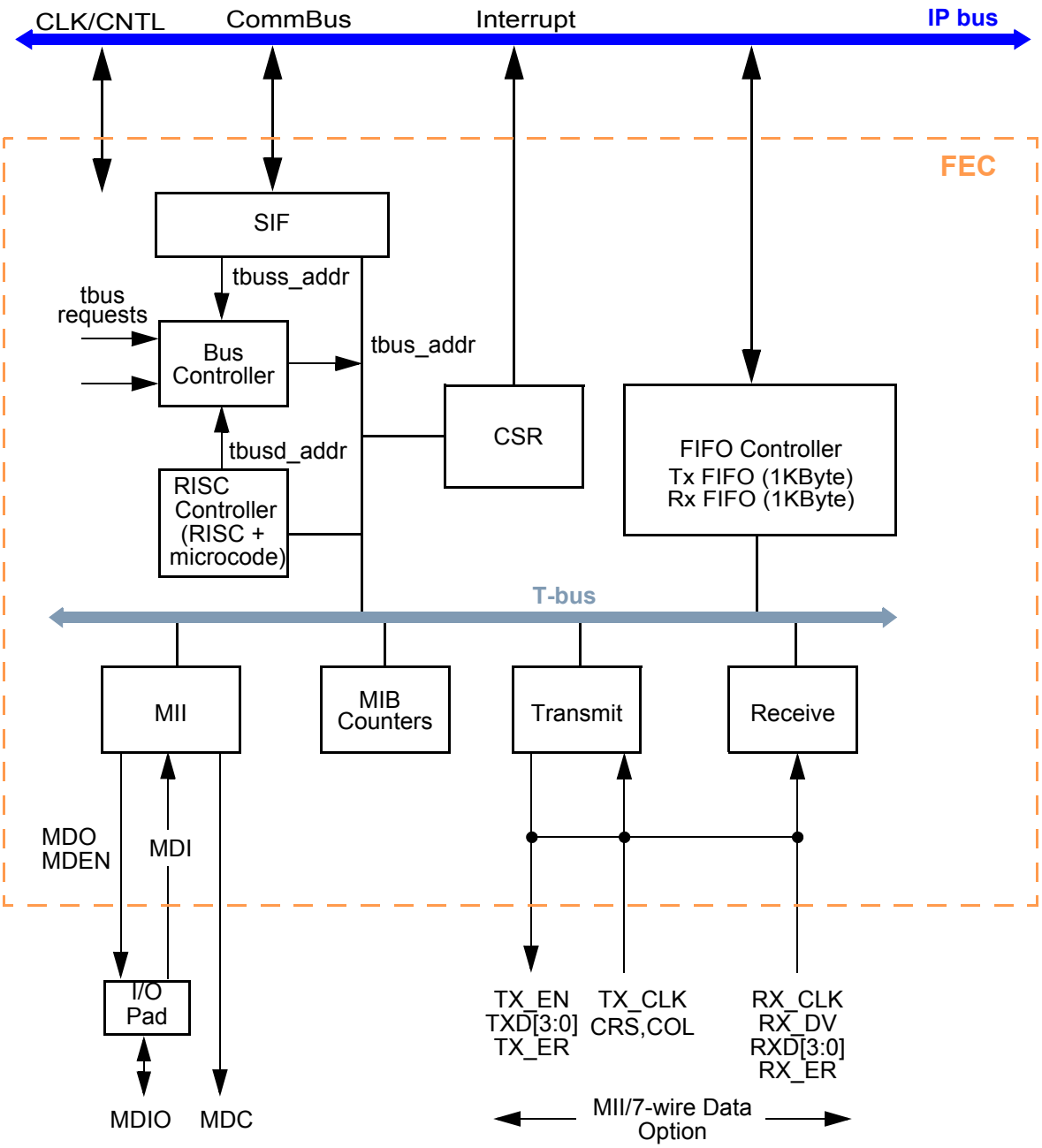


Figure 14-1. Block Diagram—FEC

14.1.1 Features

The FEC incorporates several features/design goals that are key to its use:

- Support for different ethernet physical interfaces:
  - 100 Mbps IEEE 802.3 MII
  - 10 Mbps IEEE 802.3 MII
  - 10 Mbps 7-wire interface (industry standard)
- IEEE 802.3 full-duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum system clock rate of 50 MHz.
- Support for half-duplex operation (100 Mbps throughput) with a minimum system clock rate of 25 MHz.
- Large (1 Kbyte) on-chip transmit and receive FIFOs to support a variety of bus latencies.
- Retransmission from transmit FIFO following a collision (no processor bus utilization).

- Automatic internal flushing of the Rx FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization).
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

14.2 Modes of Operation

The primary operational modes are described in this section.

14.2.1 Full- and Half-Duplex Operation

This is determined by the X_CNTRL register FDEN bit. Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

Full-duplex flow control is an option that may be enabled in full-duplex mode.

14.2.2 10 Mbps and 100 Mbps MII Interface Operation

The MAC-PHY interface operates in MII mode by asserting the R_CNTRL register MII_MODE bit. MII is the media independent interface defined by the 802.3 standard for 10/100 Mbps operation.

Speed of operation is determined by the TX_CLK and RX_CLK pins, which are driven by the transceiver. The transceiver either auto-negotiates the speed or it may be controlled by software using the serial management interface (MDC/MDIO pins) to the transceiver.

14.2.3 10 Mbps 7-Wire Interface Operation

If the external transceiver supports 10 Mbps only and uses a 7-wire style interface then deassert the R_CNTRL register MII_MODE bit in the R_CNTRL register. This style of interface is not defined by the 802.3 standard, but instead is an industry standard.

14.2.4 Address Recognition Options

The options supported are promiscuous, broadcast reject, individual address hash or exact match and multicast hash match. Refer to the R_CNTRL register for address recognition programming.

14.2.5 Internal Loopback

Internal loopback mode is selected using the R_CNTRL register LOOP bit.

14.3 I/O Signal Overview

This section defines the FEC-to-chip pin I/O. The FEC network interface supports multiple options. One is the MII option that requires 18 I/O pins and supports both data and an out-of-band serial management interface to the PHY (transceiver) device. The MII option supports both 10 and 100 Mbps ethernet rates. The second is referred to as the 7-wire interface and supports only 10 Mbps ethernet data. The 7-wire interface uses a subset of the MII signals.

Table 14-1 shows the network interface signals and lists 18 signals, all of which are used for the 10/100 MII interface.

NOTE

The MDIO pin is bidirectional and corresponds to the FEC block MDI, MDO and MDIO pins. The 7-wire interface option uses a subset of these signals.

Table 14-1. Signal Properties

Signal Name	Chip Pin	Function	Reset State
tx_en	ETH0	MII—transmit data valid output 7-wire—transmit data valid output	0
tdata[0]	ETH1	MII—transmit data bit 0 output 7-wire—transmit data output	

Table 14-1. Signal Properties (continued)

Signal Name	Chip Pin	Function	Reset State
tdata[1]	ETH2	MII—transmit data bit 1 output	
tdata[2]	ETH3	MII—transmit data bit 2 output	
tdata[3]	ETH4	MII—transmit data bit 3 output	
tx_er	ETH5	MII—transmit error output	0
mdc	ETH6	MII—management clock output	0
mdi mdo md_en	ETH7	MII—management data bidirect	Hi-Z (input)
rx_dv	ETH8	MII—Rx data valid input 7-wire—rena input	
rx_clk	ETH9	MII—Rx clock input 7-wire—Rx clock input	
col	ETH10	MII—collision input 10 Mbps 7-wire—collision input	
tx_clk	ETH11	MII—transmit clock input 7-wire—transmit clock input	
rdata[0]	ETH12	MII—Rx data bit 0 input 7-wire—Rx data input	
rdata[1]	ETH13	MII—Rx data bit 1 input	
rdata[2]	ETH14	MII—Rx data bit 2 input	
rdata[3]	ETH15	MII—Rx data bit 3 input	
rx_er	ETH16	MII—Rx error input	
crs	ETH17	MII—carrier sense input	

14.3.1 Detailed Signal Descriptions

14.3.1.1 MII Ethernet MAC-PHY Interface

This section gives a detailed description of the Media-Independent Interface (MII). An overview of the MII is presented followed by a description of the MII signals. Two different types of MII frames are described. A brief MII management function overview is given.

The MII interface has 18 signals. Tx and Rx functions require 7 signals each:

- 4 data signals
- 1 delimiter
- 1 error
- 1 clock

Media status is indicated by 2 signals:

- 1 signal indicates a carrier is present.
- 1 signal indicates a collision occurred.

Management interface is provided by 2 signals.

MII signals are described below.

Tx_CLK . . . . . A continuous clock that provides a timing reference for Tx_EN, TxD, and Tx_ER. The frequency of Tx_CLK is 25% of the transmit data rate, ± 100 ppm. Duty cycle shall be 35%-65% inclusive.

Rx_CLK . . . . . A continuous clock that provides a timing reference for Rx_DV, RxD, and Rx_ER. The frequency of Rx_CLK is 25% of the Rx data rate, with a duty cycle between 35% and 65%.

Tx_EN . . . . . Assertion of this signals indicates valid nibbles are being presented on the MII. This signal is asserted with the first nibble of preamble and is negated prior to the first Tx_CLK following the final nibble of the frame.

TxD . . . . . TxD[0:3] represent a nibble of data when Tx_EN is asserted and have no meaning when Tx_EN is de-asserted. Table 14-2 summarizes the permissible encoding of TxD.

Tx_ER . . . . . Assertion of this signal for one or more clock cycles while Tx_EN is asserted causes PHY to transmit one or more illegal symbols. Asserting Tx_ER has no affect when operating at 10 Mbps or when Tx_EN is de-asserted This signal transitions synchronously with respect to Tx_CLK.

Rx_DV . . . . . When this signal is asserted, PHY is indicating a valid nibble is present on the MII. This signal remains asserted from the first recovered nibble of the frame through the last nibble. Assertion of Rx_DV must start no later than the SFD, and exclude any EOF.

RxD . . . . . RxD[0:3] represents a nibble of data to be transferred from the PHY to the MAC when Rx_DV is asserted. A completely formed SFD must be passed across the MII. When Rx_DV is not asserted, RxD has no meaning. There is an exception to this which is explained later. Table 14-3 summarizes the permissible encoding of RXD.

Rx_ER . . . . . When Rx_ER and Rx_DV are asserted, the PHY has detected an error in the current frame. When Rx_DV is not asserted, Rx_ER shall have no affect. This signal transitions synchronously with Rx_CLK

CRS . . . . . Signal is asserted when Tx or Rx medium is not idle. If a collision occurs, CRS remains asserted through the duration of the collision. This signal is not required to transition synchronously with Tx_CLK or Rx_CLK.

COL . . . . . Signal is asserted on a collision detection, and remains asserted while the collision persists. The signal behavior is not specified when in full-duplex mode. This signal is not required to transition synchronously with Tx_CLK or Rx_CLK.

MDC . . . . . Signal provides a timing reference to the PHY for data transfers on the MDIO signal. MDC is aperiodic, and has no maximum high or low times. The minimum high and low times is 160 ns, with the minimum period being 400 ns.

MDIO . . . . . Signal transfers control/status information between the PHY and MAC. It transitions synchronously to MDC. The MDIO pin is a bidirectional pin. The internal FEC signals that connect to this pad are: MDI (data in), MDO (data out), and MD_EN (direction control, high for output).

Table 14-2 lists the interpretation of possible encodings for Tx_EN and Tx_ER

Table 14-2. MII: Valid Encoding of TxD, Tx_EN and Tx_ER

TX_EN	TX_ER	TXD	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000 through 1111	Reserved
1	0	0000 through 1111	Normal data transmission
1	1	0000 through 1111	Transmit error propagation

A false carrier condition occurs if PHY detects a bad start-of-stream delimiter. This condition signals MII by asserting Rx_ER and placing 1110 on RxD. Rx_DV must also be de-asserted. Valid Rx_DV, Rx_ER and RxD[3:0] encodings are shown in Table 14-3.

Table 14-3. MII: Valid Encoding of RxD, Rx_ER and Rx_DV

RX_DV	RX_ER	RXD	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Normal inter-frame
0	1	0001 through 1101	Reserved
0	1	1110	False Carrier
0	1	1111	Reserved
1	0	0000 through 1111	Normal data reception
1	1	0000 through 1111	Data reception with errors

14.3.1.2 MII Management Frame Structure

A transceiver management frame transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the following format:

<preamble><st><op><phyad><regad><ta><data><idle>

Table 14-4. MMI Format Definitions

Name	Description
<preamble>	Optional—consists of a sequence of 32 continuous logic 1s.
<st>	Start of frame—indicated by a <01> pattern.
<op>	Operation code: Read instruction is <10> Write instruction is <01>
<phyad>	A 5-bit field that lists up to 32 PHYs be addressed. The first address bit transmitted is the msb of the address.
<regad>	A 5-bit field that lets 32 registers be addressed within each PHY. The first register bit transmitted is the msb of the address.
<ta>	A 2-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	Data field is 16 bits wide. Data bit 15 is first bit transmitted and received.
<idle>	During idle condition, MDIO is in the high impedance state.

14.3.1.2.1 MII Management Register Set

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in Table 14-5.

Table 14-5. MII Management Register Set

Register Address	Register Name	Basic/Extended
0	Control	B
1	Status	B
2:3	PHY Identifier	E
4	Auto-Negotiation Advertisement	E
5	AN Link Partner Ability	E
6	AN Expansion	E
7	AN Next Page Transmit	E
8:15	Reserved	E
16:31	Vendor Specific	E

14.4 FEC Memory Map and Registers

The FEC device is programmed by a combination of control/status registers (CSRs) and BestComm task loops. Since the FEC software model is BestComm-based, there is no similarity with existing CPM-based products' coding.

The CSRs are used for mode control, interrupts and extraction of status information. BestComm tasks are used to pass data buffers and related buffer or frame information between the hardware and software.

All access via microprocessor to and from the registers must be 32-bit accesses. There is no support for accesses other than 32-bit. All access via BestComm to and from the registers may be byte, word or longword (32-bit) accesses. However, name based register access is 32-bit aligned.

14.4.1 Top Level Module Memory Map

The FEC implementation requires a 2K Byte memory map space. This is divided into two sections of 512 Bytes and an additional 1KBytes of reserved space. The first 512 Bytes is used for Control and Status Registers. The second contains event/statistic counters held in the MIB block. Table 14-6 defines the top level memory map.

Table 14-6. Module Memory Map

Address	Function
000–1FF	Control/Status Registers
200–3FF	MIB Block Counters, see Table 14-7
400–7FF	Reserved

14.4.2 Control and Status (CSR) Memory Map

TABLE 1. CSR Counters

Address	Mnemonic	Name
000	FEC_ID	FEC_ID register
004	IEVENT	Interrupt Event Register
008	IMASK	Interrupt Enable Register
00C		Reserved
010	R_DES_ACTIVE	Receive Ring Updated Flag
014	X_DES_ACTIVE	Transmit Ring Updated Flag
018-020		Reserved
024	ECNTRL	Ethernet Control Register
028-03C		Reserved
040	MII_DATA	MII Data Register
044	MII_SPEED	MII Speed Register
04C-060		Reserved
064	MIB_CONTROL	MIB Control/Status Register
068-080		Reserved
084	R_CNTRL	Receive Control register
088	R_HASH	Receive Hash
08C-0C0		Reserved
0C4	X_CNTRL	Transmit Control Register
0C8-0E0		Reserved
0E4	PADDR1	Physical Address Low
0E8	PADDR2	Physical Address High+ Type Field
0EC	OP_PAUSE	Opcode + Pause Duration
0F0-114		Reserved
118	IADDR1	Upper 32 bits of individual Hash Table
11C	IADDR2	Lower 32 bits of individual Hash Table

TABLE 1. CSR Counters

Address	Mnemonic	Name
120	GADDR1	Upper 32 bits of group Hash Table
124	GADDR2	Lower 32 bits of group Hash Table
128-140		Reserved
144	X_WMRK	Transmit FIFO Watermark
148-180		Reserved
184	RFIFO_DATA	Receive FIFO Data
188	RFIFO_STATUS	Receive FIFO Status
18C	RFIFO_CONTROL	Receive FIFO Control
190	RFIFO_LRF_PTR	Receive FIFO Last Read Frame Pointer
194	RFIFO_LWF_PTR	Receive FIFO Last Write Frame Pointer
198	RFIFO_ALARM	Receive FIFO Alarm Pointer
19C	RFIFO_RDPTR	Receive FIFO Read Pointer
1A0	RFIFO_WRPTR	Receive FIFO Write Pointer
1A4	TFIFO_DATA	Transmit FIFO Data
1A8	TFIFO_STATUS	Transmit FIFO Status
1AC	TFIFO_CONTROL	Transmit FIFO Control
1B0	TFIFO_LRF_PTR	Transmit FIFO Last Read Frame Pointer
1B4	TFIFO_LWF_PTR	Transmit FIFO Last Write Frame Pointer
1B8	TFIFO_ALARM	Transmit FIFO Alarm Pointer
1BC	TFIFO_RDPTR	Transmit FIFO Read Pointer
1C0	TFIFO_WRPTR	Transmit FIFO Write Pointer
1C4	RESET_CNTRL	Reset Control
1C8	XMIT_FSM	Transmit FSM
1CC-1FF		

14.4.3 MIB Block Counters Memory Map

Table 14-7 defines the MIB Counters memory map, which defines the MIB RAM space locations where hardware-maintained counters reside. These fall in the 3200-33FF address range. Counters are divided into two groups.

1. **RMON counters**—are included, which cover ethernet statistics counters defined in RFC 1757. In addition to ethernet statistics group counters, a counter is included to count truncated frames, as FEC only supports frame lengths up to 2047 Bytes. RMON counters are implemented independently for Tx and Rx, to ensure accurate network statistics when operating in full duplex mode.
2. **IEEE counters**—are included, which support the Mandatory and Recommended counter packages defined in Section 5 of ANSI/IEEE Standard 802.3 (1998 edition). FEC supports IEEE Basic Package objects, but does not require MIB block counters. In addition, some recommended package objects supported do not require MIB counters. Counters for Tx and Rx full duplex flow control frames are included.

Table 14-7. MIB Counters

Address	Mnemonic	Description
200	RMON_T_DROP	Count of Frames Not Correctly Counted
204	RMON_T_PACKETS	RMON Tx Packet Count
208	RMON_T_BC_PKT	RMON Tx Broadcast Packets
20C	RMON_T_MC_PKT	RMON Tx Multicast Packets
210	RMON_T_CRC_ALIGN	RMON Tx Packets with CRC/Align error
214	RMON_T_UNDERSIZE	RMON Tx Packets less than 64 bytes, good CRC
218	RMON_T_OVERSIZE	RMON Tx Packets greater than MAX_FL bytes, good CRC
21C	RMON_T_FRAG	RMON Tx Packets less than 64 bytes, bad CRC
220	RMON_T_JAB	RMONTxPackets greater than MAX_FL bytes, bad CRC
224	RMON_T_COL	RMON Tx collision count
228	RMON_T_P64	RMON Tx 64 Byte packets
22C	RMON_T_P65TO127	RMON Tx 65 to 127 Byte packets
230	RMON_T_P128TO255	RMON Tx 128 to 255 Byte packets
234	RMON_T_P256TO511	RMON Tx 256 to 511 Byte packets
238	RMON_T_P512TO1023	RMON Tx 512 to 1023 Byte packets
23C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047 Byte packets
240	RMON_T_P_GTE2048	RMON Tx packets with greater than 2048 Bytes
244	RMON_T_OCTETS	RMON Tx Octets
248	IEEE_T_DROP	Count of Frames Not Counted Correctly
24C	IEEE_T_FRAME_OK	Frames Transmitted OK
250	IEEE_T_1COL	Frames Transmitted with Single Collision
254	IEEE_T_MCOL	Frames Transmitted with Multiple Collisions
258	IEEE_T_DEF	Frames Transmitted after Deferral Delay
25c	IEEE_T_LCOL	Frames Transmitted with Late Collision
260	IEEE_T_EXCOL	Frames Transmitted with Excessive Collisions
264	IEEE_T_MACERR	Frames Transmitted with Tx FIFO Underrun
268	IEEE_T_CSERR	Frames Transmitted with Carrier Sense Error
26C	IEEE_T_SQE	Frames Transmitted with SQE Error
270	T_FDXFC	Flow Control Pause Frames Transmitted
274	IEEE_T_OCTETS_OK	Octet Count for Frames Transmitted w/o Error
278–27C	rsvd	Reserved
280	RMON_R_DROP	Count of frames Not Counted Correctly
284	RMON_R_PACKETS	RMON Rx Packet Count
288	RMON_R_BC_PKT	RMON Rx Broadcast Packets
28C	RMON_R_MC_PKT	RMON Rx Multicast Packets

Table 14-7. MIB Counters (continued)

Address	Mnemonic	Description
290	RMON_R_CRC_ALIGN	RMON Rx Packets with CRC/Align error
294	RMON_R_UNDERSIZE	RMON Rx Packets less than 64 Bytes, good CRC
298	RMON_R_OVERSIZE	RMON Rx Packets greater than MAX_FL bytes, good CRC
29C	RMON_R_FRAG	RMON Rx Packets less than 64 Bytes, bad CRC
2A0	RMON_R_JAB	RMONRxPackets greater than MAX_FL bytes, bad CRC
2A4	RMON_R_RESVD_0	Reserved
2A8	RMON_R_P64	RMON Rx 64 Byte packets
2AC	RMON_R_P65TO127	RMON Rx 65 to 127 Byte packets
2B0	RMON_R_P128TO255	RMON Rx 128 to 255 Byte packets
2B4	RMON_R_P256TO511	RMON Rx 256 to 511 Byte packets
2B8	RMON_R_P512TO1023	RMON Rx 512 to 1023 Byte packets
2BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047 Byte packets
2C0	RMON_R_P_GTE2048	RMON Rx packets with greater than 2048 Bytes
2C4	RMON_R_OCTETS	RMON Rx Octets
2C8	IEEE_R_DROP	Count of frames not counted correctly
2CC	IEEE_R_FRAME_OK	Frames received OK
2D0	IEEE_R_CRC	Frames received with CRC error
2D4	IEEE_R_ALIGN	Frames received with alignment error
2D8	IEEE_R_MACERR	Rx FIFO overflow count
2DC	R_FDXFC	Flow Control Pause frames received
2E0	IEEE_R_OCTETS_OK	Octet count for frames received without error
2E4–2FC	rsvd	Reserved
300–3FF	rsvd	Reserved

14.5 FEC Registers—MBAR + 0x3000

The FEC uses 37 32-bit registers. These registers are located at an offset from MBAR of 0x3000. Register addresses are relative to this offset. Therefore, the actual register address is **MBAR + 0x3000 + register address**

Hyperlinks to the FEC registers are provided below:

14.5.1 FEC ID Register—MBAR + 0x3000

The read-only FEC ID register (FEC_ID) identifies the FEC block and revision.

Table 14-8. FEC ID Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEC_ID															
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 lsb
R	Reserved					DMA	FIFO	Rsvd	FEC_REV							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:15	FEC_ID	Value identifying the FEC 000 = Unique identifier for FEC
16:20	—	Reserved
21	DMA	DMA function is included in the FEC 0 = FEC does not include DMA (BestComm is the DMA engine)
22	FIFO	FIFO function included in the FEC 1 = FEC does include a FIFO
24:31	FEC_REV	Value identifies the FEC revision 00 = Initial revision

14.5.2 FEC Interrupt Event Register—MBAR + 0x3004

When an event occurs that sets a bit in the IEVENT register, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. The IEVENT register bit is cleared if 1 is written to that bit position. A 0 write has no effect. A hardware reset clears this register.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are:

- GRA
- TFINT
- MII

Interrupts resulting from errors/problems detected in the network or transceiver are:

- HBERR
- BABR
- BABT
- LATE_COL
- COL_RETRY_LIM

Interrupts resulting from internal errors are:

- XFIFO_UN
- XFIFO_ERROR
- RFIFO_ERROR

Some error interrupts are independently counted in the MIB block counters. Software may choose to mask these interrupts, since the errors are visible to network management via the MIB counters.

- HBERR – IEEE_T_SQE
- BABR – RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT – RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL – IEEE_T_LCOL
- COL_RETRY_LIM – IEEE_T_EXCOL
- XFIFO_UN – IEEE_T_MACERR

Table 14-9. FEC Interrupt Event Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HBERR	BABR	BABT	GRA	TFINT	Reserved			MII	Rsvd	LATE_COL	COL_ RETRY_LIM	XFIFO_UN	XFIFO_ ERROR	RFIFO_ ERROR	Rsvd
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	HBERR	Heartbeat Error— interrupt bit indicates HBC is set in the X_CNTRL register and COL input was not asserted within the heartbeat window following a transmission.
1	BABR	Babbling Receive Error—bit indicates frame was received with a length in excess of R_CNTRL.MAX_FL bytes.
2	BABT	Babbling Transmit Error—bit indicates transmitted frame length exceeded R_CNTRL.MAX_FL bytes. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation does not occur.
3	GRA	Graceful Stop Complete—interrupt bit is asserted for one of three reasons. 1 = A graceful stop initiated by setting X_CNTRL.GTS bit is complete. 2 = A graceful stop initiated by setting X_CNTRL.FC_PAUSE bit is complete. 3 = A graceful stop initiated by reception of a valid full duplex flow control “pause” frame is complete. Refer to “Full Duplex Flow Control” section of the Ethernet Operation chapter. A "graceful stop" means the transmitter is put into a pause state after completion of the frame currently being transmitted.
4	TFINT	Transmit frame interrupt. This bit indicates that a frame has been transmitted.
5	—	Reserved
6	—	Reserved
7	—	Reserved
8	MII	MII Interrupt—bit indicates MII completed the data transfer requested.
9	—	Reserved
10	LATE_COL	Bit indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC. Remainder of the frame is discarded.
11	COL_RETRY_LIM	Collision Retry Limit—bit indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame begins. Only occurs in half-duplex mode.
12	XFIFO_UN	Transmit FIFO Underrun—bit indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and remainder of frame is discarded.
13	XFIFO_ERROR	Transmit FIFO Error—indicates an error occurred within the forest green version transmit FIFO. When XFIFO_ERROR bit is set, ECNTRL.ETHER_EN is cleared, halting FEC frame processing. When this occurs, software must ensure both the FIFO Controller and BestComm are soft-reset.
14	RFIFO_ERROR	Receive FIFO Error—indicates error occurred within the forest green version RX FIFO. When RFIFO_ERROR bit is set, ECNTRL.ETHER_EN is cleared, halting FEC frame processing. When this occurs, software must ensure both the FIFO Controller and BestComm are soft-reset.
15:31	—	Reserved.

14.5.3 FEC Interrupt Enable Register—MBAR + 0x3008

The IMASK register provides control over the interrupt events allowed to generate an interrupt. All implemented bits in this CSR are R/$\overline{W}$. This register is cleared by a hardware reset. If corresponding bits in both the IEVENT and IMASK registers are set, the interrupt is signalled to the CPU. The interrupt signal remains asserted until 1 is written to the IEVENT bit (write 1 to clear) or a 0 is written to the IMASK bit.

Table 14-10. FEC Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBEEN	BREN	BTEN	GRAEN	TFIEN	Reserved			MIIEN	Rsvd	LCEN	CRLEN	XFUNEN	XFERREN	RFERREN	Rsvd
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	HBEEN	Heartbeat Error Interrupt Enable
1	BREN	Babbling Receiver Interrupt Enable
2	BTEN	Babbling Transmitter Interrupt Enable
3	GRAEN	Graceful Stop Interrupt Enable
4	TFIEN	Transmit Frame Interrupt Enable
5	—	Reserved
6	—	Reserved
7	—	Reserved
8	MIIEN	MII Interrupt Enable
9	—	Reserved
10	LCEN	Late Collision Enable
11	CRLEN	Late Collision Enable
12	XFUNEN	Transmit FIFO Underrun Enable
13	XFERREN	Transmit FIFO Error Enable
14	RFERREN	Receive FIFO Error Enable
15:31	—	Reserved

14.5.4 FEC Rx Descriptor Active Register—MBAR + 0x3010

The FEC descriptor active register is a command register which should be written by the user to indicate that the receive descriptor ring has been updated (empty receive buffers have been produced by the driver with the E bit set).

Whenever the register is written the R_DES_ACTIVE bit is set. This is independent of the data actually written by the user. When set, the FEC will poll the receive descriptor ring and process receive frames (provided ETHER_EN is also set). Once the FEC polls a receive descriptor whose ownership bit is not set, then the FEC will clear the R_DES_ACTIVE bit and cease receive descriptor ring polling until the user sets the bit again, signifying additional descriptors have been placed into the receive descriptor ring.

The R_DES_ACTIVE bit is cleared at reset and by the clearing of ETHER_EN.

Table 14-11. FEC Rx Descriptor Active Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							R_DES_ACTIVE	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	R_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device whenever no additional “ready” descriptors remain in the receive ring.
8:31	—	Reserved

14.5.5 FEC Tx Descriptor Active Register—MBAR + 0x3014

The FEC descriptor active register is a command register which should be written by the user to indicate that the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the R bit set in the buffer descriptor).

Whenever the register is written the X_DES_ACTIVE bit is set. This is independent of the data actually written by the user. When set, the FEC will poll the transmit descriptor ring and process transmit frames (provided ETHER_EN is also set). Once the FEC polls a transmit descriptor whose ownership bit is not set, then the FEC will clear the X_DES_ACTIVE bit and cease transmit descriptor ring polling until the sets the bit again, signifying additional descriptors have been placed into the transmit descriptor ring.

The X_DES_ACTIVE bit is cleared at reset and by the clearing of ETHER_EN.

Table 14-12. FEC Tx Descriptor Active Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							X_DES_ACTIVE	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:6	—	Reserved
7	X_DES_ACTIVE	Set to one when this register is written, regardless of the value written. Cleared by the FEC device whenever no additional “ready” descriptors remain in the transmit ring.
8:31	—	Reserved

14.5.6 FEC Ethernet Control Register—MBAR + 0x3024

The ECNTRL register is a read/write user register that can enable/disable the FEC. Some fields may be altered by hardware.

Table 14-13. FEC Ethernet Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TAG0	TAG1	TAG2	TAG3	Rsvd	TESTMD	Reserved									
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 lsb
R	Reserved													FEC_OE	ETHER_EN	RESET
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	TAG[0:3]	This field allows programming and reading the TBUS tag bits. This field is used for debug/test only, and is implemented in two separate 4-bit registers. The “tags_in” register is written to when a sky blue write to this register takes place. This field (tags_in) resets to 1111. During a write cycle to any FEC register other than ECNTRL the tags_in value is driven onto the tbus data bus tag field. During a read cycle the tbus tag field bits is latched and saved in the “tags_out” register. When the ECNTRL register is read the value from “tags_out” shows in the TAG field.
4	—	Reserved
5	TESTMD	Test Mode—used for manufacturing test only. TESTMD resets to 0. This bit forces the bus controller to ignore all bus requests except the one from the SIF.
6:28	—	Reserved
29	FEC_OE	FEC Output Enable—It is a spare bit and has no affect on internal operation.
30	ETHER_EN	Ethernet Enable—When this bit is set, FEC is enabled and Rx/Tx can occur. When bit is cleared, Rx stops immediately; Tx stops after a bad CRC is appended to any frame currently being transmitted. The ETHER_EN bit is altered by hardware under the following conditions: • If ECNTRL.RESET is written to 1 by software, ETHER_EN is cleared. • If error conditions causing the IEVENT.EBERR, XFIFO_ERROR or RFIFO_ERROR bits to set occur ETHER_EN is cleared.
31	RESET	Ethernet Controller Reset—When this bit is set, the equivalent of a hardware reset is done, but it is local to the FEC. ETHER_EN is cleared and all other FEC registers take their reset values. Also, any Tx/Rx currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately 8 clock cycles after RESET is written with 1.

14.5.7 FEC MII Management Frame Register—MBAR + 0x3040

This MII_DATA register does not reset to a defined value. The MII_DATA register is used to communicate with the attached MII compatible PHY device(s), providing read/write access to the MII registers.

Writing to the MII_DATA register causes a management frame to be sourced unless the MII_SPEED register has been programmed to 0. When writing to MII_DATA when MII_SPEED = 0, if the MII_SPEED register is then written to a non-zero value, an MII frame is generated with the data previously written to the MII_DATA register. This let MII_DATA and MII_SPEED be programmed in either order if MII_SPEED is currently 0.

Table 14-14. FEC MII Management Frame Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ST		OP		PA					RA					TA	
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	DATA															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit does not reset to a defined value.

Bits	Name	Description
0:1	ST	Start of Frame Delimiter—bits must be programmed to 01 for a valid MII management frame.
2:3	OP	Operation Code—field must be programmed to 10 (read) or 01 (write) to generate a valid MII management frame. • A value of 11 causes a “read” frame operation. • A value of 00 causes a “write” frame operation. However, these frames are not MII compliant.
4:8	PA	PHY Address—specifies 1 of up to 32 attached PHY devices.
9:13	RA	Register Address—specifies 1 of up to 32 registers within the specified PHY device.
14:15	TA	TurnAround—must be programmed to 10 to generate a valid MII management frame.
16:31	DATA	Management Frame Data—used for data written to or read from PHY register.

To do a read or write operation, the MII management interface writes to the MII_DATA register. To generate a valid read or write management frame:

- the ST field must be written with a 01
- the OP field must be written with either:
  - 01 (management register write frame), or
  - 10 (management register read frame), and
- the TA field must be written with a 10

If other patterns are written to these fields, a frame is generated, but it does not comply to the IEEE 802.3 MII definition:

- OP field = 1x produces a “read” frame operation, while
- OP field = 0x produces a “write” frame operation.

To generate an IEEE 802.3 compliant MII management interface write frame (write to a PHY register), the user must write the following to the MII_DATA register:

{01 01 PHYAD REGAD 10 DATA}

Writing this pattern causes control logic to shift out the data in the MII_DATA register following a preamble generated by the control state machine. During this time, the MII_DATA register contents are altered as the contents are serially shifted, and is unpredictable if read by the

user. When the write management frame operation is complete, the MII_DATAIO_COMPL interrupt is generated. At this time the MII_DATA register contents match the original value written.

To generate an MII Management Interface read frame (read a PHY register) the user must write the following to the MII_DATA register (DATA field content is "don't care"):

{01 10 PHYAD REGAD 10 XXXX}

Writing this pattern causes control logic to shift out data in the MII_DATA register following a preamble generated by the control state machine. During this time, the MII_DATA register contents are altered as the contents are serially shifted, and is unpredictable if read by the user. When the read management frame operation is complete, the MII_DATAIO_COMPL interrupt is generated. At this time the MII_DATA register contents matches the original value written, except for the DATA field whose contents have been replaced by the value read from the PHY register.

If the MII_DATA register is written while frame generation is in progress, frame contents are altered. Software should use the MII_STATUS register and/or the MII_DATAIO_COMPL interrupt to avoid writing to the MII_DATA register while frame generation is in process.

14.5.8 FEC MII Speed Control Register—MBAR + 0x3044

The MII_SPEED register provides MII clock (MDC pin) frequency control. This allows dropping the MII management frame preamble and provides observability (intended for manufacturing test) of an internal counter used in generating an MDC clock signal.

Table 14-15. FEC MII Speed Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved								DIS_PREAMBLE	MII_SPEED						Rsvd
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:23	—	Reserved
24	DIS_PREAMBLE	Asserting this bit causes preamble (32 1s) to not be prepended to the MII management frame. The MII standard allows the preamble to be dropped, if not required by the attached PHY device(s).
25:30	MII_SPEED	Controls the frequency of the MII management interface clock (MDC) relative to ipb_clk. A 0 value in this field “turns off” the MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of 1/(MII_SPEED*2) of the ipb_clk frequency. The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE MII characteristic. The MII_SPEED must be set to a non-zero value in order to source a read or write management frame. After the management frame is complete, the MII_SPEED register may optionally be set to 0 to turn off the MDC. The MDC generated has a 50% duty cycle except when MII_SPEED is changed during operation (change takes affect following either a rising or falling edge of MDC). If the ipb_clk is 25MHz, programming MII_SPEED field to 0x5 results in a MDC frequency of 25MHz * 1/(5*2) = 2.5 MHz. Table 14-16 shows MII_SPEED optimum values as a function of the ipb_clk frequency.
31	—	Reserved

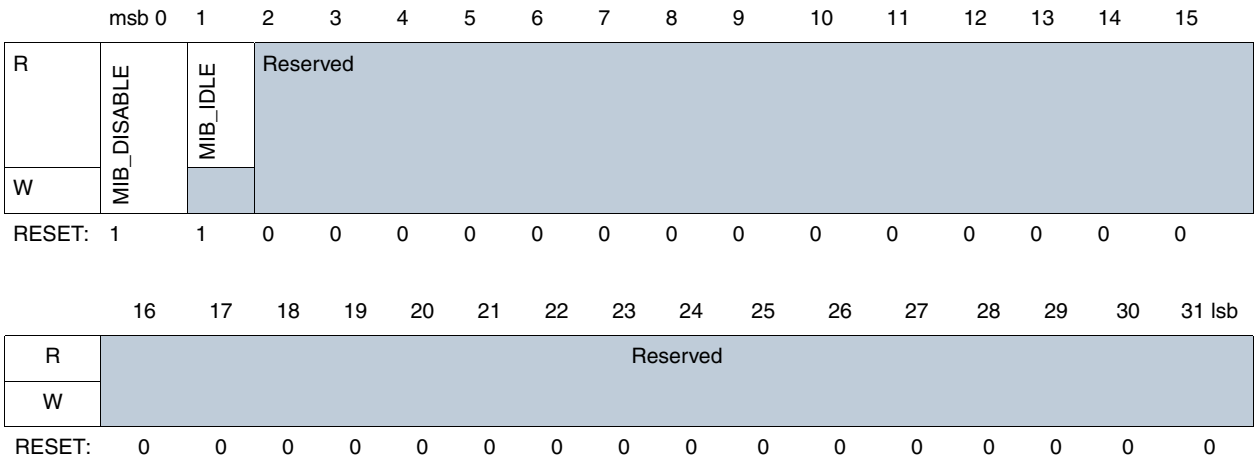
Table 14-16. Programming Examples for MII_SPEED Register

ipb_clk Frequency	MII_SPEED (Field in Register)	MDC Frequency
25MHz	$5	2.5MHz
33MHz	$7	2.36MHz
40MHz	$8	2.5MHz
50MHz	$A	2.5MHz

14.5.9 FEC MIB Control Register—MBAR + 0x3064

The MIB_CONTROL register is a read/write register used to provide control of and to observe the state of the MIB block. This register is accessed by user software if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM the user should disable the MIB block, clear all MIB RAM locations, then enable the MIB block. The MIB_DISABLE bit is reset to 1.

Table 14-17. FEC MIB Control Register



Bits	Name	Description
0	MIB_DISABLE	A read/write control bit. If set, MIB logic halts and MIB counters do not update.
1	MIB_IDLE	A read-only status bit. If set, MIB block is not currently updating MIB counters.
2:31	—	Reserved

14.5.10 FEC Receive Control Register—MBAR + 0x3084

The R_CNTRL register is user programmable. It controls the operational mode of the receive block and should be written only when ETHER_EN = 0 (initialization time).

Table 14-18. FEC Receive Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					MAX_FL										
W																
RESET:	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved										FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Name	Description
0:4	—	Reserved
5:15	MAX_FL	Maximum Frame Length—User R/W field. Resets to decimal 1518. The length is measured starting at DA and includes CRC at End Of Frame (EOF). Tx frames longer than MAX_FL causes the BABT interrupt to occur. Rx Frames longer than MAX_FL causes BABR interrupt to occur and sets the EOF buffer descriptor LG bit. The recommended user programmed default value is 1518, or if VLAN Tags are supported, 1522.
16:25	—	Reserved
26	FCE	Flow Control Enable—If asserted, the receiver detects PAUSE frames. On PAUSE frame detection, transmitter stops transmitting data frames for a given duration.
27	BC_REJ	Broadcast Frame Reject—If asserted, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected, unless PROM bit is set. If both BC_REJ and PROM = 1, frames with broadcast DA are accepted and M (MISS) bit is set in the Rx buffer descriptor.
28	PROM	Promiscuous mode—All frames are accepted regardless of address matching.
29	MII_MODE	Selects External Interface Mode—controls the interface mode for Tx/Rx blocks. • Setting bit to 1 selects MII mode. • Setting bit to 0 selects 7wire mode (used only for serial 10Mbps).

Bits	Name	Description
30	DRT	Disable Receive on Transmit 0 = Rx path operates independently of Tx (use for full-duplex or to monitor Tx activity in half-duplex mode). 1 = Disable frames reception while transmitting (normally used for half-duplex mode).
31	LOOP	Internal Loopback—If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The system clock is substituted for TX_CLK when LOOP is asserted. DRT must be set to 0 when asserting LOOP.

14.5.11 FEC Hash Register—MBAR + 0x3088

The read-only R_HASH register provides address recognition information from the Rx block about the frame currently being received. These bits provide information used in the address recognition subroutine.

Table 14-19. FEC Hash Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FCE_DC	MULTI CAST	HASH						Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	FCE_DC	This is a read-only view of the R_CNTRL register FCE bit.
1	MULTICAST	Set if current Rx frame contained a multi-cast destination address, indicating DA LSB was set. Cleared if current Rx frame does not correspond to a multi-cast address.
2:7	HASH	Corresponds to “hash” value of current Rx frame’s destination address. Hash value is a 6-bit field extracted from least significant portion of CRC register.
8:31	—	Reserved

14.5.12 FEC Tx Control Register—MBAR + 0x30C4

This X_CNTRL register is read/write and is written to configure the transmit block. This register is cleared at system reset. Bits 29:30 should be modified only when ETHER_EN = 0.

Table 14-20. FEC Tx Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved											RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:26	—	Reserved
27	RFC_PAUSE	This read-only status bit is asserted when a full-duplex flow control pause frame is received. The transmitter is paused for the duration defined in this pause frame. Bit automatically clears when the pause duration is complete.
28	TFC_PAUSE	Assert to transmit a PAUSE frame. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, the INTR_EVENT register GRA interrupt is asserted. With transmission of data frames stopped, the MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. **Note:** If the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, MAC may still transmit a MAC Control PAUSE frame.
29	FDEN	Full Duplex Enable—If set, frames are transmitted independent of Carrier Sense and Collision inputs. This bit should only be modified when ETHER_EN is deasserted.
30	HBC	Heartbeat Control—If set, the heartbeat check is done following End Of Transmission (EOT) and the Event Status Register HB bit is set if the collision input does not assert within the heartbeat window. This bit should only be modified when ETHER_EN is deasserted.
31	GTS	Graceful Transmit Stop—When this bit is set, the MAC stops transmission after any frame that is currently being transmitted is complete and the INTR_EVENT register GRA interrupt is asserted. If frame transmission is not currently underway, the GRA interrupt is immediately asserted. Once transmission completes, a “restart” can be done by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS = 1, transmission stops after the collision. The frame is transmitted again once GTS is cleared. **Note:** Old frames may exist in the transmit FIFO and be transmitted when GTS is reasserted. To avoid this, deassert ETHER_EN after the GRA interrupt.

14.5.13 FEC Physical Address Low Register—MBAR + 0x30E4

The PADDR1 register is written by the user. This register contains the lower 32 bits (Bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in Bytes 0:3 of the 6-Byte source address field when transmitting PAUSE frames. This register is not reset and must be initialized.

Table 14-21. FEC Physical Address Low Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	PADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:31	PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8) and 3 (bits 7:0) of the 6-byte individual address used for an exact match, and the Source Address field in PAUSE frames.

14.5.14 FEC Physical Address High Register—MBAR + 0x30E8

The PADDR2 register is written by the user. This register contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in Bytes 4 and 5 of the 6-Byte source address field when transmitting PAUSE frames. Bits 16:31 of XMIT.PADDR2 contain a constant type field (hex 8808) used for transmission of PAUSE frames. This register is not reset and bits 0:15 must be initialized.

Table 14-22. FEC Physical Address High Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR2															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	TYPE															
W																
RESET:	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:15	PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for an exact match, and the Source Address field in PAUSE frames.
16:31	TYPE	These 16 bits are a constant value, hex 8808.

14.5.15 FEC Opcode/Pause Duration Register—MBAR + 0x30EC

The OP_PAUSE register is read/write accessible. This register contains the 16-bit opcode, and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, hex 0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. This register is not reset and bits 16:31 must be initialized.

Table 14-23. FEC Opcode/Pause Duration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPCODE															
W																
RESET:	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 lsb
R	PAUSE_DUR															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:15	OPCODE	Opcode field used in PAUSE frames. Bits are a constant value, hex 0001.
16:31	PAUSE_DUR	Pause Duration field used in PAUSE frames.

14.5.16 FEC Descriptor Individual Address 1 Registe—MBAR + 0x3118

The IADDR1 register is written by the user. This register contains the upper 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized.

Table 14-24. FEC Descriptor Individual Address 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	IADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:31	IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. • Bit 31 contains hash index bit 63. • Bit 0 contains hash index bit 32.

14.5.17 FEC Descriptor Individual Address 2 Register—MBAR + 0x311C

The IADDR2 register is written by the user. This register contains the lower 32 bits of the 64-bit individual address hash table used in the address recognition process to check for possible match with the DA field of receive frames with an individual DA. This register is not reset and must be initialized.

Table 14-25. FEC Descriptor Individual Address 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IADDR2															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	IADDR2															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:31	IADDR2	The lower 32bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. • Bit 31 contains hash index bit 31. • Bit 0 contains hash index bit 0.

14.5.18 FEC Descriptor Group Address 1 Register—MBAR + 0x3120

The GADDR1 register is written by the user. This register contains the upper 32bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized.

Table 14-26. FEC Descriptor Group Address 1 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	GADDR1															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:31	GADDR1	The GADDR1 register contains the upper 32bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. • Bit 31 contains hash index bit 63. • Bit 0 contains hash index bit 32.

14.5.19 FEC Descriptor Group Address 2 Register—MBAR + 0x3124

The GADDR2 register is written by the user. The GADDR2 register contains the lower 32bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. This register must be initialized.

Table 14-27. FEC Descriptor Group Address 2 Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GADDR2															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	GADDR2															
W																
RESET:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Note: X: Bit is not reset and must be initialized.

Bits	Name	Description
0:31	GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. • Bit 31 contains hash index bit 31. • Bit 0 contains hash index bit 0.

14.5.20 FEC Tx FIFO Watermark Register—MBAR + 0x3144

The X_WMRK register is a user programmable 4-bit read/write register that controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This lets the user minimize transmit latency (X_WMRK = 0000) or allows for larger bus access latency (X_WMRK = 1111) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The X_WMRK register resets to 0.

NOTE

This register value may need to be customized by software for specific FEC applications to be compatible with specific FIFO/system bus access latency requirements.

Table 14-28. FEC Tx FIFO Watermark Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved												X_WMRK			
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:28	—	Reserved
28:31	X_WMRK	Transmit FIFO Watermark—Frame transmission begins: • If the number of bytes selected by this field are written into the transmit FIFO, or • if an EOF is written to the FIFO, or • if the FIFO is full before the selected number of bytes are written. Options are: 0000 = 64 Bytes written to FIFO. 0001 = 128 Bytes written to FIFO. 0010 = 192 Bytes written to FIFO. 0011 = 256 Bytes written to FIFO. 0100 = 320 Bytes written to FIFO. 0101 = 384 Bytes written to FIFO. 0110 = 448 Bytes written to FIFO. 0111 = 512 Bytes written to FIFO. 1000 = 576 Bytes written to FIFO. 1001 = 640 Bytes written to FIFO. 1010 = 704 Bytes written to FIFO. 1011 = 768 Bytes written to FIFO. 1100 = 832 Bytes written to FIFO. 1101 = 896 Bytes written to FIFO. 1110 = 960 Bytes written to FIFO. 1111 = 1024 Bytes written to FIFO.

14.6 FIFO Interface

The programming interface to the FIFO allows access to Data, Status, Control, Last Write Pointer, Last Read Pointer, Alarm, Read and Write Pointers for Transmit and Receive configurations. The FIFO can be accessed by byte, word, or longword, but all accesses must be aligned with the most significant byte (big endian) of the data port. BestComm supports byte, word or longword accesses. The processor supports longword access only. All register name access is longword aligned

Table 14-29. FIFO Interface Register Map

Address	byte0	byte1	byte2	byte3	Description
0x184	Data	Data	Data	Data	Receive FIFO Data
0x188	Stat	Stat			Receive FIFO Status
0x18C	Ctl				Receive FIFO Control
0x190			LRF	LRF	Receive Last Read Frame Pointer
0x194			LWF	LWF	Receive Last Write Frame Pointer
0x198			Alarm	Alarm	Receive (High/Low) Alarm Pointer
0x19C			Read	Read	Receive FIFO Read Pointer
0x1A0			Write	Write	Receive FIFO Write Pointer
0x1A4	Data	Data	Data	Data	Transmit FIFO Data
0x1A8	Stat	Stat			Transmit FIFO Status
0x1AC	Ctl				Transmit FIFO Control
0x1B0			LRF	LRF	Transmit Last Read Frame Pointer

Table 14-29. FIFO Interface Register Map (continued)

Address	byte0	byte1	byte2	byte3	Description
0x1B4			LWF	LWF	Transmit Last Write Frame Pointer
0x1B8			Alarm	Alarm	Transmit (High/Low) Alarm Pointer
0x1BC			Read	Read	Transmit FIFO Read Pointer
0x1C0			Write	Write	Transmit FIFO Write Pointer

14.6.1 FEC Rx FIFO Data Register—MBAR + 0x3184

14.7 FEC Tx FIFO Data Register—MBAR + 0x31A4

The RFIFO_DATA and TFIFO_DATA registers are the main interface port for the Transmit and Receive FIFO. Data which is to be buffered in the FIFO, or has been buffered in the FIFO, is accessed through this register.

14.7.1 FEC Rx FIFO Status Register—MBAR + 0x3188

14.8 FEC Tx FIFO Status Register—MBAR + 0x31A8

The RFIFO_STATUS and TFIFO_STATUS registers contain bits which provide information about the status of the FIFO controller. The bits marked sticky are cleared by writing a ‘1’ to their positions.

Table 14-30. FEC Rx FIFO Status Register
FEC Tx FIFO Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				Frame[0:3]				Rsvd	Error	UF	OF	FR	Full	Alarm	Empty
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bits	Name	Description
0:3	—	Reserved
4:7	Frame[0:3]	Frame Indicator – READ ONLY This bus provides a frame status indicator for non-DMA applications. Frame[0] = A frame boundary has occurred on the [31:24] byte of the data bus. Frame[1] = A frame boundary has occurred on the [23:16] byte of the data bus. Frame[2] = A frame boundary has occurred on the [15:8] byte of the data bus. Frame[3] = A frame boundary has occurred on the [7:0] byte of the data bus.
8	---	Reserved
9	Error	FIFO Error – Sticky, Write To Clear. This bit signifies that an error has occurred in the FIFO controller. Errors can be caused by underflow, overflow,or pointers being out of bounds. This bit will remain set until this bit of the FIFO status register has been written with a 1.

Bits	Name	Description
10	UF	UF FIFO Underflow – Sticky, Write To Clear This bit signifies the read pointer has surpassed the write pointer. This bit will remain set until this bit of the FIFO status register has been written with a 1.
11	OF	OF FIFO Overflow – Sticky, Write To Clear This bit signifies the write pointer has surpassed the read pointer. This bit will remain set until this bit of the FIFO status register has been written with a 1.
12	FR	FR Frame Ready – Read Only The FIFO has requested attention because there is framed data ready. All complete frames must be read from the FIFO to clear this alarm. This alarm will only be asserted while in frame mode.
13	Full	Full Alarm – Read Only The FIFO has requested attention because it is full. The FIFO must be read to clear this alarm.
14	Alarm	FIFO Alarm – Read Only The FIFO has requested attention because it has determined an alarm condition. The specific alarm condition detected is dependent upon the FIFO direction (Transmit or Receive); if it is a Transmit FIFO, then the FIFO alarm output pin provides indication of a low level, asserting when there is less than alarm bytes of data remaining in the FIFO, and deasserting when there are less than 4* granularity free bytes remaining. When the FIFO is configured to Receive, the FIFO alarm provides high level indication, asserting when there are less than alarm bytes free in the FIFO, and deasserting when there are less than granularity bytes of data remaining. This signal can be cleared by reading or writing (as appropriate) the FIFO, or manipulating the FIFO pointers.
15	Empty	Empty – Read Only The FIFO has requested attention because it is empty. The FIFO must be written to clear this alarm.
16:31	---	Reserved

14.8.1 FEC Rx FIFO Control Register—MBAR + 0x318C
FEC Tx FIFO Control Register—MBAR + 0x31AC

The RFIFO_CONTROL and TFIFO_CONTROL registers provide programmability of many FIFO behaviors, from last transfer granularity to frame operation. Last transfer granularity allows the user to control when the FIFO controller stops requesting data transfers through the FIFO alarm. When the alarm is configured as a Receive FIFO, the granularity value is the GR[2:0] value. When the alarm is configured as a Transmit FIFO, the granularity value is four times the GR[2:0] value, or the pipeline depth. The frame bit of the control register provides a capability to enable and control the FIFO controller's ability to view data on a packetized basis. The FIFO controller also has the programmable capability to not request attention after it has received a complete frame until Ethernet has reported completion of transmission. Frame mode supersedes the FIFO granularity bits, through the assertion of a hardware signal to BestComm.

Table 14-31. FEC Rx FIFO Control Register
FEC Tx FIFO Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	WFR[1:0]		COMP	FRAME	GR[2:0]			Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	—	Reserved
1:2	WFR[1:0]	Write Frame 01 = the FIFO controller assumes the next write to its data port is the next to last write. 10 = the FIFO controller assumes the next write to its data port is status / control information.
3	COMP	COMP Re-enable Requests on Frame Transmission Completion. When this bit is set, the FIFO controller will not request attention between receiving the last data of the frame from the BestComm until the peripheral acknowledges transmission of the frame.
4	FRAME	Frame Mode Enable. When this bit is set, the FIFO controller monitors frame done information from the peripheral or BestComm. Setting this bit also enables the other frame control bits in this register, as well as other frame functions. This bit must be set to use frame functions.
5:7	GR[2:0]	Last Transfer Granularity. These bits define the deassertion point for the “high” service request and also define the deassertion point for the “low” service request. A “high” service request is deasserted when there are less than GR[2:0] data bytes remaining in the FIFO. A “low” service request is deasserted when there are less than (4 * GR[2:0]) free bytes remaining in the FIFO.

14.8.2 FEC Rx FIFO Last Read Frame Pointer Register—MBAR + 0x3190
FEC Tx FIFO Last Read Frame Pointer Register—MBAR + 0x31B0

The RFIFO_LRF_PTR and TFIFO_LRF_PTR are a FIFO-maintained pointer which indicates the location of the start of the most recently read frame, or the start of the frame currently in transmission. The LRFP updates on FIFO read data accesses to a frame boundary. The LRFP can be read and written for debug purposes. For the frame retransmit function, the LRFP indicates which point to begin retransmission of the data frame. The LRFP carries validity information, however, there are no safeguards to prevent retransmitting data which has been overwritten. When FRAME is not set, then this pointer has no meaning.

Table 14-32. FEC Rx FIFO Last Read Frame Pointer Register
FEC Tx FIFO Last Read Frame Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved						LRFP[9:0]									
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:21	—	Reserved
22:31	LRFP[9:0]	LRFP Last Read Frame Pointer. This pointer indicates the start of the last data frame read from the FIFO by the peripheral.

14.8.3 FEC Rx FIFO Last Write Frame Pointer Register—MBAR + 0x3194
FEC Tx FIFO Last Write Frame Pointer Register—MBAR + 0x31B4

The RFIFO_LWF_PTR and TFIFO_LWF_PTR are a FIFO maintained pointer which indicates the location of the start of the last frame written into the FIFO. The LWFP updates on FIFO write data accesses which create a frame boundary, whether that be by setting the writeFrameCtl control bit, or by feeding a frame bit in on the appropriate bus. The LWFP can be read and written for debug purposes. For the frame discard function, the LWFP divides the valid data region of the FIFO (the area in-between the read and write pointers) into framed and unframed data. Data between the LWFP and write pointer constitutes an incomplete frame, while data between the read pointer and the LWFP has been received as whole frames. When FRAME is not set, then this pointer has no meaning.

Table 14-33. FEC Rx FIFO Last Write Frame Pointer Register
FEC Tx FIFO Last Write Frame Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved						LRFP[9:0]									
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:21	—	Reserved
22:31	LWFP[9:0]	LRFP Last WriteFrame Pointer. This pointer indicates the start of the last data frame written into the FIFO by the peripheral.

14.8.4 FEC Rx FIFO Alarm Pointer Register—MBAR + 0x3198
FEC Tx FIFO Alarm Pointer Register—MBAR + 0x31B8

RFIFO_ALARM and TFIFO_ALARM include pointer which provide high/low level alarm information to the user integration logic and the BestComm interface. A low level alarm reports lack of data; a high level alarm reports lack of space. The alarm pointer is interpreted depending on the state of the FIFO transmit input pin: if FIFO transmit = ‘1’, then the alarm is represented in terms of data bytes, if FIFO Transmit = ‘0’, the alarm is represented in terms of free bytes. This programmable alarm can warn the system when the FIFO is almost full of data (FIFO Transmit = ‘0’), or when the FIFO is almost out of data (FIFO Transmit = ‘1’). This register is programmed to the upper limit for the number of bytes in the FIFO of data, when FIFO transmit is negated, or space, when FIFO transmit is asserted, before an internal alarm is set. Any time the amount of data or space in the FIFO is above the indicated amount, the alarm will be set. The alarm is cleared when there is less data or space than is defined as the FIFO granularity or pipeline depth. The number of bits in the alarm pointer register will vary with the address space of the FIFO memory, and the alarm pointer is initialized to zero.

Table 14-34. FEC Rx FIFO Alarm Pointer Register
FEC Tx FIFO Alarm Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved						Alarm[9:0]									
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:21	—	Reserved
22:31	Alarm[9:0]	Alarm Pointer. This pointer indicates the point at (or below) which to assert the FIFO alarm signal. This value is compared with data or free bytes, depending upon the state of FIFO Transmit (FIFO Transmit = ‘1’, alarm measures data bytes).

14.8.5 FEC Rx FIFO Read Pointer Register—MBAR + 0x319C
FEC Tx FIFO Read Pointer Register—MBAR + 0x31BC

The RFIFO_RDPTR and TFIFO_RDPTR are a FIFO-maintained pointer which point to the next FIFO location to be read. The read pointer can be both read and written.

Table 14-35. FEC Rx FIFO Read Pointer Register
FEC Tx FIFO Read Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved						READ[9:0]									
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:21	—	Reserved
22:31	READ[9:0]	Read Pointer. This pointer indicates the next location to be read by the FIFO controller.

14.8.6 FEC Rx FIFO Write Pointer Register—MBAR + 0x31A0
FEC Tx FIFO Writer Pointer Register—MBAR + 0x31C0

The RFIFO_WRPTR and TFIFO_WRPTR are a FIFO-maintained pointer which point to the next FIFO location to be written. The write pointer can be both read and written.

Table 14-36. FEC Rx FIFO Write Pointer Register
FEC Tx FIFO Write Pointer Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
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 lsb
R W	Reserved						WRITE[9:0]									
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:21	—	Reserved
22:31	WRITE[9:0]	WRITE Pointer. This pointer indicates the next location to be written by the FIFO controller.

14.8.7 FEC Reset Control Register—MBAR + 0x31C4

The RESET_CNTRL register allows reset of the FIFO controllers.

Table 14-37. FEC Reset Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved						RCTL[1]	RCTL[0]	Reserved							
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 lsb
R W	Reserved															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-1.

Bits	Name	Description
0:5	—	Reserved
6	RCTL[1]	0 = Do not Reset FIFO controllers. 1 = Reset FIFO controllers.

Table 1-1.

Bits	Name	Description
7	RCTL[0]	0 = Disable fec_enable as a reset to FIFO controllers. 1 = Enable fec_enable as a reset to FIFO controllers.
8:31	---	Reserved

14.8.8 FEC Transmit FSM Register—MBAR + 0x31C8

The transmit finite state machine register (XMIT_FSM) controls operation of appending CRC. Typical use is enabled and CRC appended.

Table 14-38. FEC Transmit FSM Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						XFSM[1]	XFSM[0]	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:5	—	Reserved
6	XFSM[1]	0 = Do not append CRC. 1 = Append CRC (typical use).
7	XFSM[0]	0 = Disable CRC FSM. 1 = Enable CRC FSM (typical use is enabled).
8:31	---	Reserved

14.9 Initialization Sequence

This section describes which registers are hardware reset, which are reset by the FEC, and what locations the user must initialize prior to enabling the FEC.

14.9.1 Hardware Controlled Initialization

Some registers in the FEC are reset by internal logic. Specifically those registers are control logic that generate interrupts, cause outputs to be asserted and in general, configuration control bits.

Other registers are reset when the ETHER_EN bit is not asserted (i.e., cleared). To halt operation ETHER_EN is deasserted by either a hard reset or by software. By deasserting ETHER_EN configuration control registers such as X_CNTRL and R_CNTRL are not reset, but the entire data path is reset.

Table 14-39 shows the effect deasserting ETHER_EN has on Ethernet MAC operation and registers.

Table 14-39. ETHER_EN De-Assertion Affect on FEC

Register/Machine	Reset Value
XMIT block	Transmission Aborted (bad CRC appended)
RECV block	Receive activity aborted
Tx/Rx FIFO	Reset control logic dependent on reset_cntrl

14.9.2 User Initialization (Prior to Asserting ETHER_EN)

The user needs to initialize portions of the FEC prior to setting the ETHER_EN bit. The exact values depend on the particular application; the sequence of writing the registers is not important. Ethernet MAC registers requiring initialization are defined in Table 14-40.

Table 14-40. User Initialization (Before ETHER_EN)

Description
Initialize IMASK
Clear IEVENT (write FFFF_FFFF)
X_WMRK (optional)
IADDR2/IADDR1
GADDR1/GADDR2
PADDR1/PADDR2
OP_PAUSE (only needed for FDX flow control)
R_CNTRL
X_CNTRL
MII_SPEED (optional)
Clear MIB_RAM (locations 200–2FC)

14.9.2.1 Microcontroller Initialization

In the FEC the descriptor control RISC initializes some registers after ETHER_EN is asserted. After the Microcontroller initialization sequence is complete, hardware is ready for operation.

Table 14-41 shows RISC initialization operations common to the FEC.

Table 14-41. Microcontroller Initialization (FEC)

Description
Initialize BackOff random number seed
Activate Receiver
Activate Transmit

14.9.3 Frame Control/Status Words

In the FEC transmit frame control words and receive frame status words cross the following the end of frame data. These words are marked with a type value of 10 and have the following formats.

14.9.3.1 Receive Frame Status Word

Table 14-2 below defines the format for the receive frame status word.

Table 14-42. Receive Frame Status Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	1 (Last)	0	0	M	BC	MC	LG	NO	0	CR	OV	TR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	FRAME_LENGTH										

Bits 31-28, 26-25, 19 and 15-11—Reserved

L—Last in Frame, written by the FEC

The buffer is not the last in a frame.

The buffer is the last in a frame.

BC—Will be set if the DA is broadcast (FF-FF-FF-FF-FF-FF)

MC—Will be set if the DA is multicast and not BC

LG—Rx Frame Length Violation, written by the FEC.

A frame length greater than R_CNTRL.MAX_FL was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.

NO—Rx Non-octet Aligned Frame, written by the FEC.

A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set the CR bit will not be set.

CR—Rx CRC Error, written by the FEC.

This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.

OV—Overrun, written by the FEC.

A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and will be zero. This bit is valid only if the L-bit is set.

TR—Rx Frame Truncated

Will be set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set the frame should be discarded and the other error bits should be ignored as they may be incorrect.

FRAME_LENGTH— Length of Received Frame

14.9.3.2 Transmit Frame Control Word

The only requirement for this control word is to have the TC and ABC bits valid. The TC bit defines whether the transmit block should append the CRC (TC = 1) or not (TC = 0) for the current frame. The ABC bit defines whether the transmit block should append a bad CRC (ABC = 1), independent of the TC value. Refer to Table 14-43 below for the format of the transmit frame control word.

Table 14-43. Transmit Frame Control Word Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					TC	ABC									

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

Bits 31-27, 24-0—Reserved

TC—Transmit CRC, written by user

0 = End transmission immediately after the last data byte.

1 = Transmit the CRC sequence after the last data byte.

ABC—Append Bad CRC, written by user

0 = No affect

1 = Transmit the CRC sequence inverted after the last data bye (regardless of TC value).

14.9.4 Network Interface Options

The FEC supports both an MII interface for 10/100 Mbps Ethernet and a 7-wire serial interface for 10 Mbps Ethernet. The interface mode is selected by the MII_MODE bit in the R_CNTRL register. In MII mode (R_CNTRL.MII_MODE = 1), there are 18 signals defined by the 802.3 standard and supported by the FEC. These are shown in Table 14-1:

The 7-Wire serial interface (R_CNTRL.MII_MODE = 0) operates in what is generally referred to as the “AMD” mode. FEC Frame Transmission

The Ethernet transmitter is designed to work with almost no intervention from software. Once ETHER_EN is asserted and data appears in the transmit FIFO the Ethernet MAC is able to transmit onto the network.

When the transmit FIFO fills to the watermark (defined by the X_WMRK register), the MAC transmit logic will assert TX_EN and start transmitting the preamble sequence, the start frame delimiter, and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (carrier sense is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, then the transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive).

If a collision occurs during transmission of the frame (half-duplex mode), the ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data has been transmitted, the FCS (32-bit CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC will be appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the ethernet controller writes the frame status information to the MIB block. Short frames are automatically padded by the transmit logic (if the TC bit in the transmit buffer descriptor for the end of frame buffer = 1).

The FEC frame interrupts may be generated as determined by the settings in the IMASK register.

Transmit error interrupts are HBERR, BABT, LATE_COL, COL_RETRY_LIM, XFIFO_UN and XFIFO_ERROR. If the transmit frame length exceeds MAX_FL bytes the BABT interrupt will be asserted, however the entire frame will be transmitted (no truncation).

To pause transmission, set the GTS (Graceful Transmit Stop) bit in the X_CNTRL register. When the GTS is set the FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame either finishes or terminates with a collision. After the transmitter has stopped the GRA (Graceful Stop Complete) interrupt is asserted. If GTS is cleared, the FEC resumes transmission with the next frame.

The ethernet controller transmits bytes least significant bit first.

14.9.5 FEC Frame Reception

The FEC receiver is designed to work with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking and maximum frame length checking.

When the driver enables the FEC receiver by asserting ETHER_EN it will immediately start processing receive frames. When RX_DV asserts, the receiver will first check for a valid PA/SFD header. If the PA/SFD is valid it will be stripped and the frame will be processed by the receiver. If a valid PA/SFD is not found the frame will be ignored.

In 7-wire serial mode, the first 16 bit times of RX_D0 following assertion of RX_DV (RENA) are ignored. Following the first 16 bit times the data sequence is checked for alternating 1s and 0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame have been received, the FEC performs address recognition on the frame.

Once a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, the receive FIFO is signalled that the frame is “accepted” and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to “reject” the frame. Thus, no collision fragments are presented to the user except late collisions, which indicate serious LAN problems.

During reception, the ethernet controller checks for various error conditions and once the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, SH, CR, OV and TR status bits, and the frame length.

The ethernet controller receives serial data LSB first.

14.9.6 Ethernet Address Recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast) or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames is illustrated in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure14-4 illustrates the address recognition decisions made by the receive block, while Figure 14-5 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (R_CNTRL.BC_REJ) is deasserted, then the frame will be accepted unconditionally as shown in Figure 14-4. Otherwise, if the DA in not a broadcast address the microcontroller runs the address recognition subroutine as shown in Figure 14-5.

If the DA is a group (multicast) address and flow control is disabled the microcontroller will perform a group hash table lookup using the 64-entry hash table programmed in GADDR1 and GADDR2. If a hash match occurs AR_HM_B (address recognition hash match bar) is set to 0 and the receiver accepts the frame. If flow control is enabled the microcontroller will do an exact address match check between the DA and the designated PAUSE DA in registers XMIT.FDXFC_DA1 and XMIT.FDXFC_DA2. In the case where a PAUSE DA exact match occurs AR_EM_B (address recognition exact match bar) is set to 0. If the receive block determines that the received frame is a valid PAUSE frame the frame will be rejected. Note the receiver will detect a PAUSE frame with the DA field set to either the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that the user programs in the PADDR1 and PADDR2 registers. If an exact match occurs AR_EM_B is set to 0; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers IADDR1 and IADDR2. In the case of an individual hash match AR_HM_B is set to 0. Again, the receiver will accept or reject the frame based on PAUSE frame detection, shown in Figure 14-4.

If neither a hash match (group or individual) nor an exact match (group or individual) occur both AR_HM_B and AR_EM_B are set to 1. In this case, if promiscuous mode is enabled (R_CNTRL.PROM = 1), then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected and the MISS bit will be cleared.

Similarly, if the DA is a broadcast address, broadcast reject (R_CNTRL.BC_REJ) is asserted and promiscuous mode is enabled. Then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected and the MISS bit will be cleared.

In general, when a frame is rejected it is flushed from the FIFO.

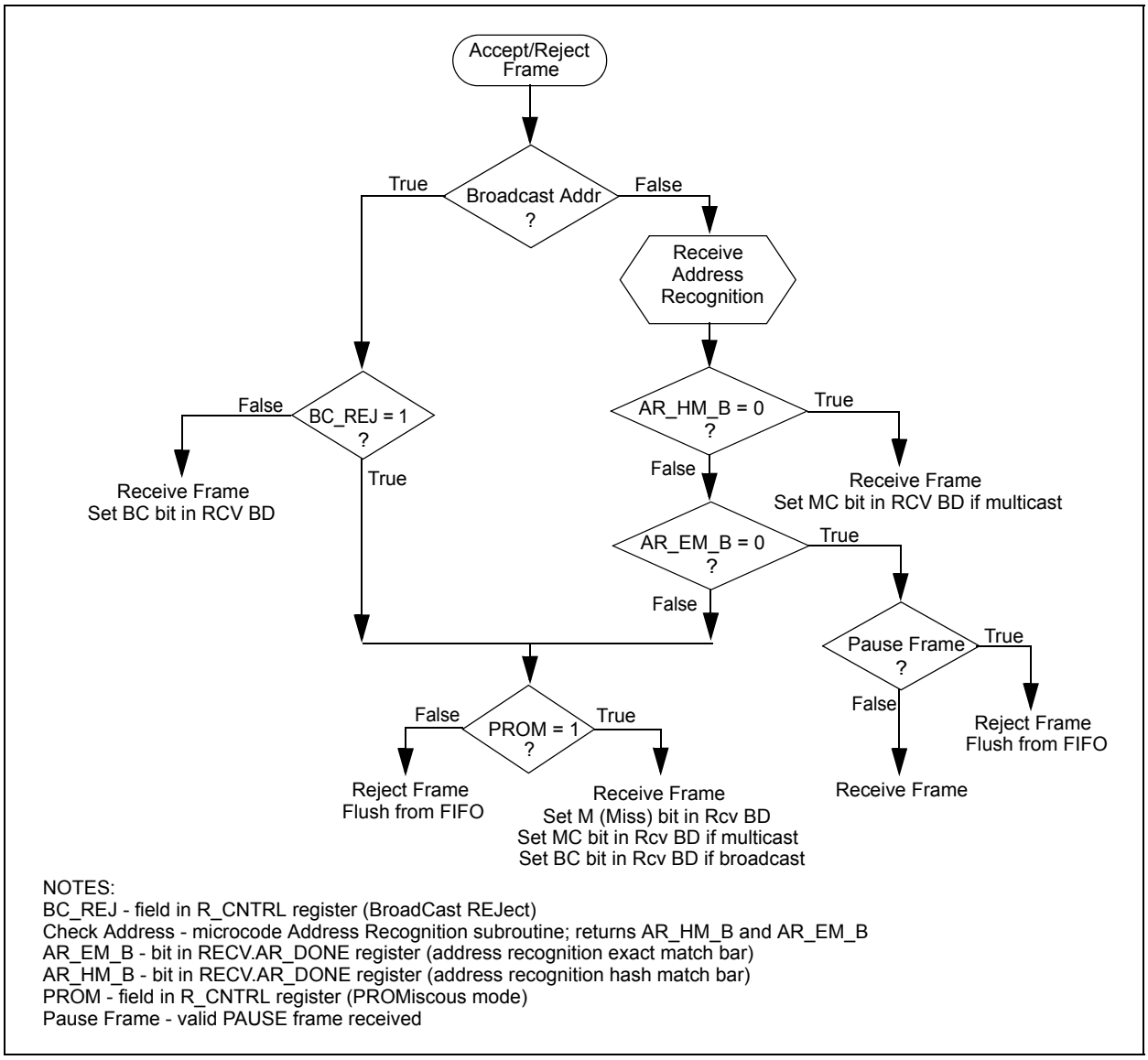


Figure 14-2. Ethernet Address Recognition - receive block decisions

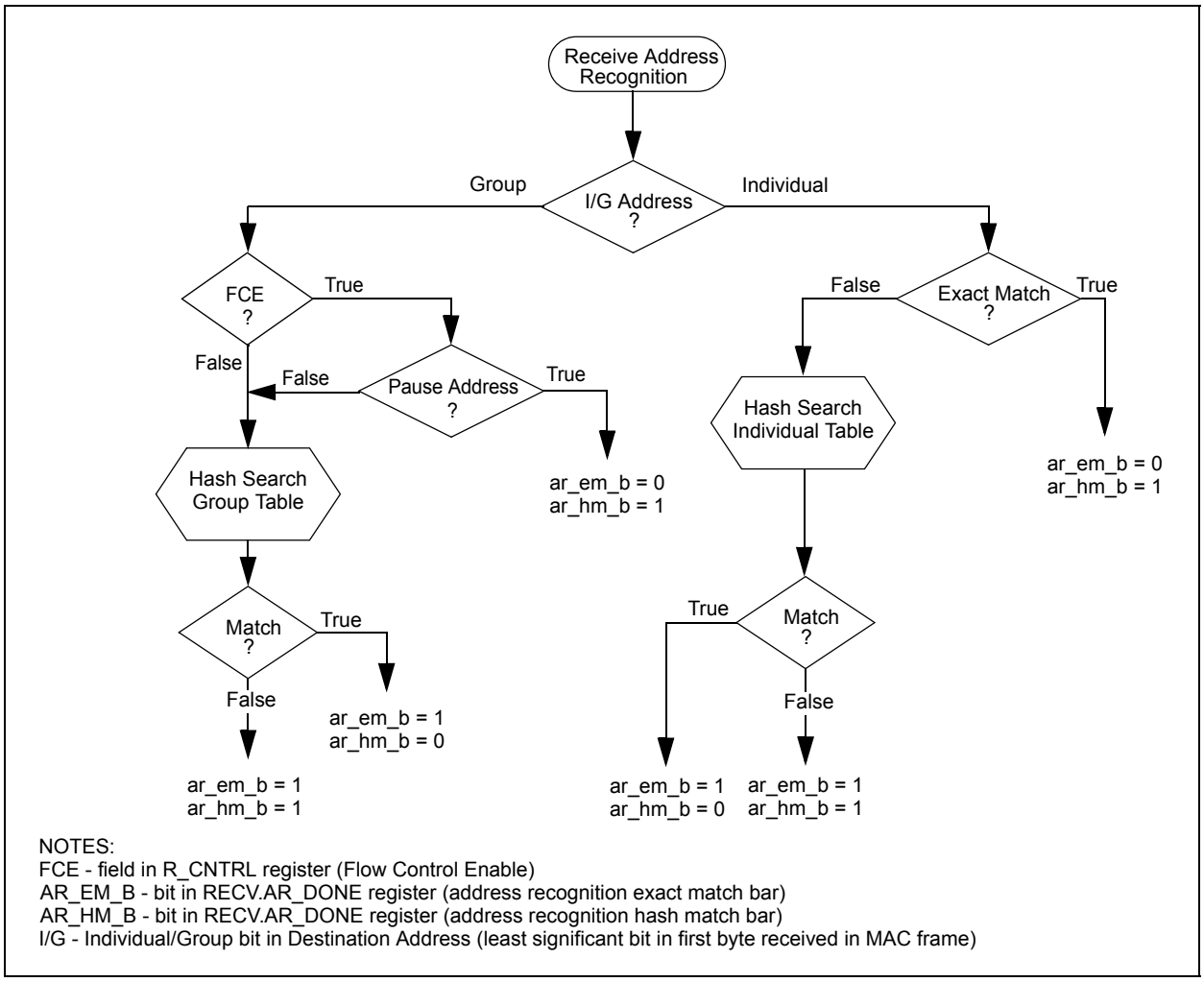


Figure 14-3. Ethernet Address Recognition - microcode decisions

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits which are represented by 64 bits stored in GADDR1,2 (group address hash match) or IADDR1,2 (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the 6 most significant bits of the CRC-encoded result to generate a number between 0 and 63. The MSB of the CRC result selects GADDR1 (MSB = 1) or GADDR2 (MSB = 0). The least significant 5 bits of the hash result select the bit within the selected register. If the CRC generator selects a bit that is set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truely contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The hash table registers must be initialized by the user. The user may compute the hash for a particular address in software. The CRC32 polynomial to use in computing the hash is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

A table of example Destination Addresses and corresponding hash values is included below for reference.

Table 14-44. Destination Address to 6-Bit Hash

48-bit DA	6-bit hash (in hex)	hash decimal value
65:ff:ff:ff:ff:ff	0x0	0
55:ff:ff:ff:ff:ff	0x1	1
15:ff:ff:ff:ff:ff	0x2	2
35:ff:ff:ff:ff:ff	0x3	3
b5:ff:ff:ff:ff:ff	0x4	4
95:ff:ff:ff:ff:ff	0x5	5
d5:ff:ff:ff:ff:ff	0x6	6
f5:ff:ff:ff:ff:ff	0x7	7
db:ff:ff:ff:ff:ff	0x8	8
fb:ff:ff:ff:ff:ff	0x9	9
bb:ff:ff:ff:ff:ff	0xa	10
8b:ff:ff:ff:ff:ff	0xb	11
0b:ff:ff:ff:ff:ff	0xc	12
3b:ff:ff:ff:ff:ff	0xd	13
7b:ff:ff:ff:ff:ff	0xe	14
5b:ff:ff:ff:ff:ff	0xf	15
27:ff:ff:ff:ff:ff	0x10	16
07:ff:ff:ff:ff:ff	0x11	17
57:ff:ff:ff:ff:ff	0x12	18
77:ff:ff:ff:ff:ff	0x13	19
f7:ff:ff:ff:ff:ff	0x14	20
c7:ff:ff:ff:ff:ff	0x15	21
97:ff:ff:ff:ff:ff	0x16	22
a7:ff:ff:ff:ff:ff	0x17	23
99:ff:ff:ff:ff:ff	0x18	24
b9:ff:ff:ff:ff:ff	0x19	25
f9:ff:ff:ff:ff:ff	0x1a	26
c9:ff:ff:ff:ff:ff	0x1b	27
59:ff:ff:ff:ff:ff	0x1c	28
79:ff:ff:ff:ff:ff	0x1d	29
29:ff:ff:ff:ff:ff	0x1e	30
19:ff:ff:ff:ff:ff	0x1f	31
d1:ff:ff:ff:ff:ff	0x20	32
f1:ff:ff:ff:ff:ff	0x21	33
b1:ff:ff:ff:ff:ff	0x22	34

Table 14-44. Destination Address to 6-Bit Hash (continued)

48-bit DA	6-bit hash (in hex)	hash decimal value
91:ff:ff:ff:ff:ff	0x23	35
11:ff:ff:ff:ff:ff	0x24	36
31:ff:ff:ff:ff:ff	0x25	37
71:ff:ff:ff:ff:ff	0x26	38
51:ff:ff:ff:ff:ff	0x27	39
7f:ff:ff:ff:ff:ff	0x28	40
4f:ff:ff:ff:ff:ff	0x29	41
1f:ff:ff:ff:ff:ff	0x2a	42
3f:ff:ff:ff:ff:ff	0x2b	43
bf:ff:ff:ff:ff:ff	0x2c	44
9f:ff:ff:ff:ff:ff	0x2d	45
df:ff:ff:ff:ff:ff	0x2e	46
ef:ff:ff:ff:ff:ff	0x2f	47
93:ff:ff:ff:ff:ff	0x30	48
b3:ff:ff:ff:ff:ff	0x31	49
f3:ff:ff:ff:ff:ff	0x32	50
d3:ff:ff:ff:ff:ff	0x33	51
53:ff:ff:ff:ff:ff	0x34	52
73:ff:ff:ff:ff:ff	0x35	53
23:ff:ff:ff:ff:ff	0x36	54
13:ff:ff:ff:ff:ff	0x37	55
3d:ff:ff:ff:ff:ff	0x38	56
0d:ff:ff:ff:ff:ff	0x39	57
5d:ff:ff:ff:ff:ff	0x3a	58
7d:ff:ff:ff:ff:ff	0x3b	59
fd:ff:ff:ff:ff:ff	0x3c	60
dd:ff:ff:ff:ff:ff	0x3d	61
9d:ff:ff:ff:ff:ff	0x3e	62
bd:ff:ff:ff:ff:ff	0x3f	63

14.9.7 Full-Duplex Flow Control

Full-duplex flow control allows the user to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable pause frame detection, the FEC must operate in full-duplex mode (X_CNTRL.FDEN asserted) and flow control enable (R_CNTRL.FCE) must be asserted. The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications as shown in the table below. In addition, the receive status associated with the frame should indicate that the frame is valid

Table 14-45. PAUSE Frame Field Specification

48-bit destination address	0180_c200_0001 or Physical ADDRESS
48-bit Source Address	any
16-bit type	8808
16-bit opcode	0001
16-bit PAUSE duration	0000 to ffff

Pause frame detection is performed by the receiver and microcontroller modules. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, graceful transmit stop is asserted by the FEC internally. When transmission has paused, the GRA (Graceful Stop complete) interrupt is asserted and the pause timer begins to increment. Note that the pause timer makes use of the transmit backoff timer hardware which is used for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time until PAUSE_DURATION slot times have expired. On PAUSE_DURATION expiration, graceful transmit stop is deasserted allowing MAC data frame transmission to resume. Note that the receive flow control pause (X_CNTRL.RFC_PAUSE) status bit is asserted while the transmitter is paused due to reception of a pause frame.

To transmit a pause frame the FEC must operate in full-duplex mode and the user must assert flow control pause (X_CNTRL.TFC_PAUSE). On assertion of transmit flow control pause (X_CNTRL.TFC_PAUSE) the transmitter asserts graceful transmit stop internally. When the transmission of data frames stops the GRA (Graceful Stop complete) interrupt asserts. Following GRA assertion the Pause frame is transmitted. On completion of pause frame transmission flow control pause (X_CNTRL.TFC_PAUSE) and graceful transmit stop are deasserted internally.

During pause frame transmission the transmit hardware places data into the transmit data stream from the registers shown in the table below.

Table 14-46. Transmit Pause Frame Registers

PAUSE FRame fields	FEC register	Register Contents
48-bit destination address	{FDXFC_DA1[0:31], FDXFC_DA2[0:15]}	0180_c200_0001
48-bit Source Address	{PADDR1[0:31], PADDR2[0:15]}	physical address
16-bit type	PADDR2[16:31]	8808
16-bit opcode	OP_PAUSE[0:15]	0001
16-bit PAUSE duration	OP_PAUSE[16:31]	0000 to ffff

The user must specify the desired pause duration in the OP_PAUSE register.

Note that when the transmitter is paused due to receiver/microcontroller pause frame detection, transmit flow control pause (X_CNTRL.TFC_PAUSE) still may be asserted and will cause the transmission of a single pause frame. In this case the GRA interrupt will not be asserted.

14.9.8 Inter-Packet Gap Time

The minimum inter packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times it will be ignored and a collision will occur.

The receiver receives back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times the following frame may be discarded by the receiver.

14.9.9 Collision Handling

If a collision occurs during frame transmission the ethernet controller will continue the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 1’s. If the collision occurs during the preamble sequence the JAM pattern will be sent after the end of the preamble sequence.

If a collision occurs within 64 byte times the retry process is initiated. The transmitter waits a random number of slot times. A slot time is 512 bit times. If a collision occurs after 64 byte times no retransmission is performed and the end of frame buffer is closed with an LC error indication.

14.9.10 Internal and External Loopback

Both internal and external loopback are supported by the ethernet controller. In loopback mode both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of the LOOP and DRT bits in the R_CNTRL register and the FDEN bit in the X_CNTRL register.

For both internal and external loopback set FDEN = 1.

For internal loopback set LOOP = 1 and DRT = 0. TX_EN and TX_ER will not assert during internal loopback. During internal loopback the transmit/receive data rate is higher than in normal operation because the internal system clock is used by the transmit and receive blocks instead of the clocks from the external transceiver. This will cause an increase in the required system bus bandwidth for transmit and receive data being transferred to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underrun and receive FIFO overflow.

For external loopback set LOOP = 0, DRT = 0 and configure the external transceiver for loopback.

14.9.11 Ethernet Error-Handling Procedure

The ethernet controller reports frame reception and transmission error conditions using the FEC BDs (receive), the IEVENT register and the MIB block counters.

14.9.11.1 Transmission Errors

Transmitter Underrun

- If this error occurs the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed. The UN bit is set in the X_STATUS register. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.
- The XFIFO_UN interrupt will be asserted if enabled in the IMASK register.

Carrier Sense Lost During Frame Transmission

- When this error occurs and no collision is detected in the frame the FEC sets the CSL bit in X_STATUS register. The frame is transmitted normally. No retries are performed as a result of this error.
- No interrupt is generated as a result of this error.

Retransmission Attempts Limit Expired

- When this error occurs the FEC terminates transmission. All remaining buffers for that frame are then flushed and closed and the RL bit is set in the X_STATUS register. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.
- The COL_RETRY_LIM interrupt will be asserted if enabled in the IMASK register.

Late Collision

- When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are then flushed and closed and the LC bit is set in the X_STATUS register. The FEC will then continue to the next transmit buffer descriptor and begin transmitting the next frame.
- The LATE_COL interrupt will be asserted if enabled in the IMASK register.

Heartbeat

- Some transceivers have a self-test feature called “heartbeat” or “signal quality error.” To signify a good self-test the transceiver indicates a collision to the FEC within 20 clocks after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.
- If the HBC bit is set in the X_CNTRL register and the heartbeat condition is not detected by the **FEC** after a frame transmission a heartbeat error occurs. When this error occurs the FEC closes the buffer, sets the HB bit in the X_STATUS register and generates the HBERR interrupt if it is enabled.

14.9.11.2 Reception Errors

Overrun Error

- If the receive block has data to put into the receive FIFO and the receive FIFO is full, the FEC sets the OV bit in the receive status word. All subsequent data in the frame will be discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit isset. This frame must be discarded by the driver.

Non-Octet Error (Dribbling Bits)

- The Ethernet controller handles up to seven dribbling bits when the receive frame terminates nonoctet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, then the frame nonoctet aligned (NO) error is reported in the Receive Frame Status Word . If there is no CRC error, then no error is reported.

CRC Error

- When a CRC error occurs with no dribble bits, the FEC closes the buffer and sets the CR bit in the Receive Frame Status Word. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

Frame Length Violation

- When the receive frame length exceeds MAX_FL bytes the BABR interrupt will be generated and the LG bit in the end of frame Receive Frame Status Word will be set. The frame is not truncated (truncation occurs if the frame length exceeds 2047 bytes).

Truncation

- When the receive frame length exceeds 2047 bytes the frame is truncated and the TR bit is set in the receive BD.

Notes

Chapter 15
Programmable Serial Controllers (PSC)

15.1 Overview

The following sections are contained in this document:

- Section 15.2, *PSC Registers—MBAR + 0x2000, 0x2200, 0x2400, 0x2600, 0x2800, 0x2C00*
- Section 15.3, *PSC Operation Modes*

The MPC5200 has 6 independent Programmable Serial Controllers (PSCs)

- PSC1 = MBAR + 0x2000
- PSC2 = MBAR + 0x2200
- PSC3 = MBAR + 0x2400
- PSC4 = MBAR + 0x2600
- PSC5 = MBAR + 0x2800
- PSC6 = MBAR + 0x2C00

Each PSC can be clocked by an internal clock source or an external clock source. In addition, each PSC module interfaces directly to the CPU and consists of the following:

- Serial Communication Channel
- Programmable Transmit (Tx) Receive (Rx) Clock Generation
- Internal Channel Control Logic
- Interrupt Control Logic

15.1.1 PSC Functions Overview

The MPC5200 provides 6 independent PSC modules, all configuration registers and internal functions are equal for each PSC. Because of pin out limitations, not all functions are available for all PSCs on every port. Table 15-1 shows, which PSC supports which mode.

In this section the name Codec mode is used as a collective term for the “normal soft Modem” the SPI and the I2S mode and the name IrDA is a common term for the Infrared interfaces SIR, MIR and FIR. The PSC can also be used to interface externally to an AC97 device, or to a device that supports a UART protocol. Figure 15-1. shows on overview about the supported functions.

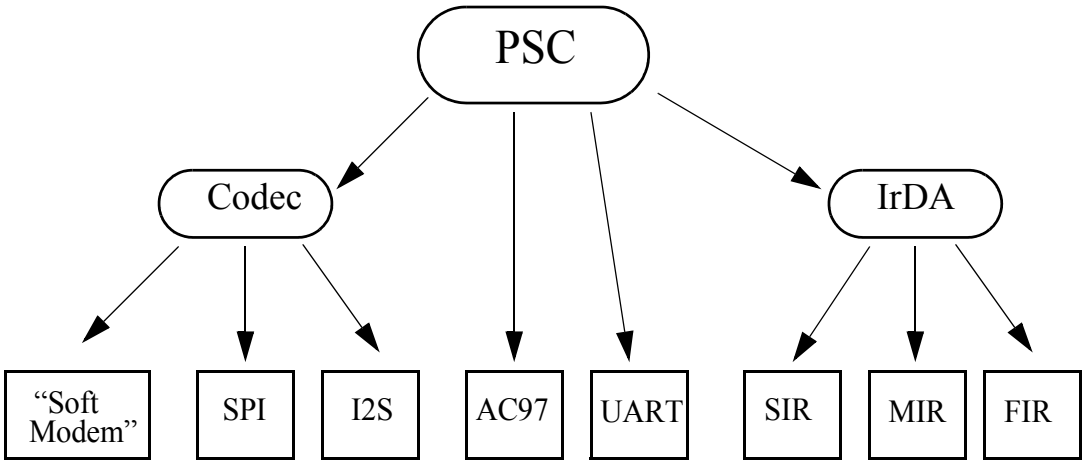


Figure 15-1. PSC Functions Overview

Table 15-1. PSC Functions Overview

	PSC1	PSC2	PSC3	PSC4	PSC5	PSC6
UART	yes	yes	yes	yes	yes	yes
Modem / SPI / I2S	yes	yes	yes	no	no	yes
Mclk Generation output	yes	yes	yes	no	no	no
AC97	yes	yes	no	no	no	no
IrDA	no	no	no	no	no	yes
Cell Phone	master	slave	slave	no	no	slave

Figure 15-2 shows a simplified PSC block diagram.

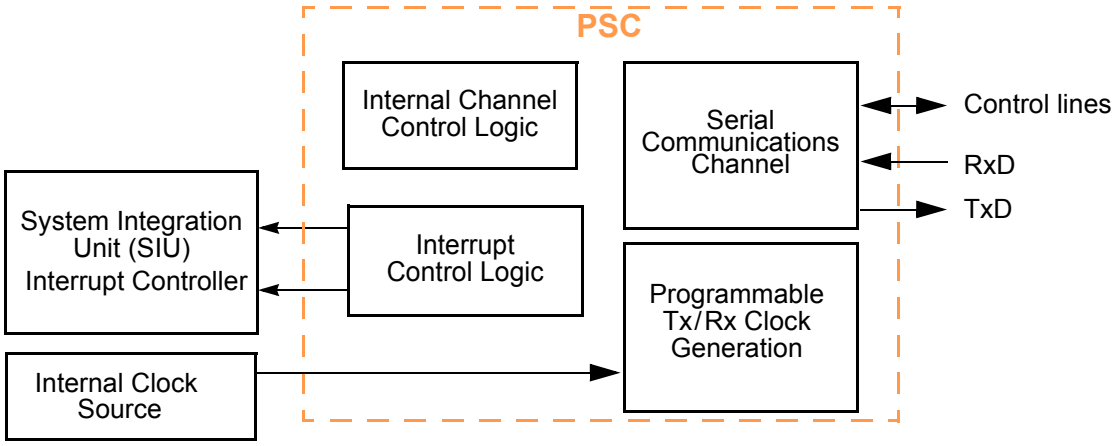


Figure 15-2. Simplified Block Diagram

The PSC can interface to a Codec through a serial port consisting of Tx and Rx serial data and serial bit-clock and frame signals. Digital sample data is transferred to and from the Codec through the serial port. The PSCs support Codec mode with 8, 16, 24 and 32 bit data width and the SPI and I2S operation modes. A Codec chip provides a data conversion interface for high-speed Modem designs meeting a high range of standards, such as ITU-T V.34 and PCM. In addition the PSC provide an Mclk for the external Codec, eliminating the need for an external crystal for the external device. For more information about the Codec mode see section: Section 15.3.2, *PSC in Codec Mode*.

AC97 defines an architecture for audio-intensive personal computer applications such as gaming, authoring, and high-resolution music and video playback. An external AC97 analog device performs mixing, analog processing, and sample-rate DAC and ADC. PSC1 and PSC2 can interface to the AC97 device through a serial port consisting of Tx and Rx serial data, a serial bit-clock input, and a frame sync output generated by the PSC from the serial bit-clock. An MPC5200 General-Purpose I/O (GPIO) is used to reset the AC97 device. The PSC transfers digital sample data as well as control/status information to and from the AC97 device through the serial port. For more information about the AC97 mode see section: Section 15.3.3, *PSC in AC97 Mode*.

When programmed as a UART the PSC serial communication channel provides a full-duplex asynchronous receiver and transmitter deriving an operating frequency from an internal clock. The transmitter converts parallel data from the CPU to a serial bit-stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (TxD). The receiver converts serial data from the channel receiver serial data input (RxD) to parallel format, checks for start, stop, and parity bits, or line break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be poll-driven or interrupt-driven. For more information about the UART mode see section: Section 15.3.1, *PSC in UART Mode*.

15.1.2 Features

General Features:

- 512-byte receiver (Rx) FIFO
- 512-byte transmitter (Tx) FIFO
- Each channel is programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic Walk-up mode for multidrop applications
- 6 maskable interrupt conditions
- PSC Tx and Rx FIFOs can be programmed to interrupt either the BestComm or the CPU when they require filling or emptying, respectively.

PSC UART mode:

- Each is clocked by an internal clock source (IPB clock), eliminating the need for an external crystal
- Full-duplex asynchronous/synchronous receiver/transmitter channel
- Programmable data format:
  - five to eight data bits plus parity
  - Odd, even, no parity, or force parity
  - One, one-and-a-half, or two STOP bits
- Parity, framing, and overrun error detection

- False-start bit detection
- Line-break detection and generation
- Detection of breaks originating in the middle of a character
- Start/end break interrupt/status

PSC Codec mode:

- Programmable to interface to an 8, 16, 24 or 32bit Codec for “soft modem” support
- Support master mode, driving clock and frame sync signals
- Support slave mode, receiving clock and the frame sync from the external Codec
- Supports full duplex SPI interface
- Supports I2S interface
- No parity error, framing error, or line break detection in Codec mode
- Ability to generate a master clock (Mclk) for an external Codec device, independent from the mode (master or slave)
- Programmable width of the frame sync signal
- Frame sync and bit clock frequencies are independently programmable
- Frame sync and bit clock polarity are programmable
- Support “digital cell phone” interface

AC97 mode:

- PSC1 and PSC2 support an AC97 interface

IrDA SIR mode:

- Baud rate: 2400 to 115200 bps
- Selectable pulse width: either 3/16 bit duration or 1.6 µs

IrDA MIR mode:

- Baud rate: 0.576 Mbps to 1.152 Mbps

IrDA FIR mode:

- Baud rate: 4 Mbps

15.2 PSC Registers—MBAR + 0x2000, 0x2200, 0x2400, 0x2600, 0x2800, 0x2C00

The PSCs are located at an address as indicated below:

- PSC1 BASE = MBAR + 0x2000
- PSC2 BASE = MBAR + 0x2200
- PSC3 BASE = MBAR + 0x2400
- PSC4 BASE = MBAR + 0x2600
- PSC5 BASE = MBAR + 0x2800
- PSC6 BASE = MBAR + 0x2C00

Each PSC uses 42 registers. The register address is calculated as base address for the regarding PSC plus the offset value. Table 15-2 shows the list with all implemented registers and the associated offset value.

Table 15-2. PSC Memory Map

Offset	Register Name	Register width	Access
00	Mode Register 1 (0x00)—MR1	8	R/W
00	Mode Register 2 (0x00) — MR2	8	R/W
04	Status Register (0x04) — SR	16	R
04	Clock Select Register (0x04) — CSR	16	W
08	Command Register (0x08)—CR	8	R/W
0C	Rx Buffer Register (0x0C) — RB	32	R
0C	Tx Buffer Register (0x0C)—TB	32	W

Table 15-2. PSC Memory Map (continued)

10	Input Port Change Register (0x10) — IPCR	8	R
10	Auxiliary Control Register (0x10) — ACR	8	W
14	Interrupt Status Register (0x14) — ISR	16	R
14	Interrupt Mask Register (0x14)—IMR	16	W
18	Counter Timer Upper Register (0x18)—CTUR	8	W
1C	Counter Timer Lower Register (0x1C)—CTLR	8	W
20	Codec Clock Register (0x20)—CCR	16	R/W
30	Interrupt Vector Register (0x30)—IVR - Reserved	8	R/W
34	Input Port Register (0x34)—IP	8	R
38	Output Port 1 Bit Set (0x38)—OP1	8	W
3C	Output Port 0 Bit Set (0x3C)—OP0	8	W
40	Serial Interface Control Register (0x40)—SICR	32	R/W
44	Infrared Control 1 (0x44)—IRCR1	8	R/W
48	Infrared Control 2 (0x48)—IRCR2	8	R/W
4C	Infrared SIR Divide Register (0x4C)—IRSDR	8	R/W
50	Infrared MIR Divide Register (0x50)—IRMDR	8	R/W
54	Infrared FIR Divide Register (0x54)—IRFDR	8	R/W
58	Rx FIFO Number of Data (0x58)—RFNUM	16	R
5C	Tx FIFO Number of Data (0x5C)—TFNUM	16	R
60	Rx FIFO Data (0x60)—RFDATA	32	R/W
64	Rx FIFO Status (0x64)—RFSTAT	16	R/W
68	Rx FIFO Control (0x68)—RFCNTL	8	R/W
6E	Rx FIFO Alarm (0x6E)—RFALARM	16	R/W
72	Rx FIFO Read Pointer (0x72)—RFRPTR	16	R/W
76	Rx FIFO Write Pointer(0x76)—RFWPTR	16	R/W
7A	Rx FIFO Last Read Frame (0x7A)—RFLRFPTR - Reserved	16	R/W
7C	Rx FIFO Last Write Frame PTR (0x7C)—RFLWFPTR - Reserved	16	R/W
80	Tx FIFO Data (0x80)—TFDATA	32	R/W
84	Tx FIFO Status (0x84)—TFSTAT	16	R/W
88	Tx FIFO Control (0x88)—TFCNTL	8	R/W
8E	Tx FIFO Alarm (0x8E)—TFALARM	16	R/W
92	Tx FIFO Read Pointer (0x92)—TFRPTR	16	R/W
96	Tx FIFO Write Pointer (0x96)—TFWPTR	16	R/W
9A	Tx FIFO Last Read Frame (0x9A)—TFLRFPTR - Reserved	16	R/W
9C	Tx FIFO Last Write Frame PTR (0x9C)—TFLWFPTR - Reserved	16	R/W

PSC module operation is controlled by writing control bytes into the appropriate registers.

15.2.1 Mode Register 1 (0x00)—MR1

The Mode registers control configuration. MR1 can be read or written when the Mode register pointer points to it, at reset or after a reset Mode register pointer command using CR[MISC]. After MR1 is read or written, the pointer points to MR2.

Table 15-3. Mode Register 1 (0x00) for UART Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	RxRTS	RxIRQ/FFULL	Reserved	PM		PT	B/C	
W								
RESET:	0	0	1	0	0	0	0	0

Table 15-4. Mode Register 1 (0x00) for SIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	RxRTS	RxIRQ/FFULL	Reserved					
W								
RESET:	0	0	1	1	0	0	1	1

Table 15-5. Mode Register 1 (0x00) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved	RxIRQ/FFULL	Reserved					
W								
RESET:	0	0	1	1	0	0	1	1

Bit	Name	Description
0	RxRTS	**UART / SIR**—Receiver request-to-send—Allows $\overline{\text{RTS}}$ output to control the $\overline{\text{CTS}}$ input of the transmitting device to prevent receiver overrun. If both the receiver and transmitter are incorrectly programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control is disabled for both. Transmitter RTS control is configured in MR2[TxRTS]. Not used in Codec mode. 0 = Receiver has no effect on $\overline{\text{RTS}}$. 1 = When a valid start bit is received, $\overline{\text{RTS}}$ is negated if the PSC FIFO is full. $\overline{\text{RTS}}$ is reasserted when the FIFO has an empty position available. **other Modes**—Reserved
1	RxIRQ/ FFULL	Receiver interrupt select. 0 = RxRDY is the source that generates IRQ 1 = FFULL is the source that generates IRQ.
2	—	Reserved
3:4	PM	**UART**—Parity mode—Selects the parity or multidrop mode for the channel. The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. The value of PM affects PT, as shown Table 15-6. PM is not used in Codec mode. **other Modes**—Reserved

Bit	Name	Description
5	PT	**UART**—Parity Type—PM and PT together select parity type (PM = 0x) or determine whether a data or address character is transmitted (PM = 11). PT is not used in Codec mode. See Table 15-6. **other Modes**—Reserved
6:7	B/C	**UART**—Bits per Character—Select the number of data bits per character to be sent. The values shown do not include start, parity, or stop bits. B/C is not used in Codec mode. 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits **other Modes**—Reserved

Table 15-6. Parity Mode/Parity Type Definitions

PM	Parity Mode	Parity Type (PT=0)	Parity Type (PT=1)
00	With parity	Even parity	Odd parity
01	Force parity	Low parity	High parity
10	No parity	n/a	
11	Multidrop mode	Data character	Address character

15.2.2 Mode Register 2 (0x00) — MR2

MR2 can be read or written when the Mode register pointer points to it, which occurs after any access to MR1. An MR2 access does not update the mode register address.

Table 15-7. Mode Register 2 (0x00) for UART / SIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	CM		TxRTS	TxCTS	SB			
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-8. Mode Register 2 (0x00) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	CM		Reserved					
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	CM	Channel mode—Selects a channel mode.CM is used in both UART and Codec modes. 00 = Normal 01 = Automatic echo 10 = Local loop-back 11 = Remote loop-back
2	TxRTS	**UART / SIR**—Transmitter ready-to-send—Controls negation of $\overline{\text{RTS}}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{\text{RTS}}$ control is not permitted and disables $\overline{\text{RTS}}$ control for both. TxRTS is not used in Codec mode. 0 = The transmitter has no effect on $\overline{\text{RTS}}$. 1 = In applications where the transmitter is disabled after transmission completes, setting this bit automatically clears RTS line one bit-time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits. **other Modes**—Reserved
3	TxCTS	**UART / SIR**—Transmitter clear-to-send—If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. TxCTS is not used in Codec mode. 0 = $\overline{\text{CTS}}$ has no effect on the transmitter. 1 = Enables clear-to-send operation. The transmitter checks the state of $\overline{\text{CTS}}$ each time it is ready to send a character. If $\overline{\text{CTS}}$ is asserted, the character is sent If it is negated, the channel TxD remains in a high state and transmission is delayed until $\overline{\text{CTS}}$ is asserted. Changes in $\overline{\text{CTS}}$ as a character is being sent do not affect its transmission. **other Modes**—Reserved
4:7	SB	**UART**—Stop-Bit (length control)—Selects the stop bit length that is appended to the transmitted character. Stop-bit lengths of 9/16th to 2 bits are programmable for 6-, 8-bit characters. Lengths of 1 1/16th to 2 bits are programmable for 5-bit characters. In all cases, the receiver checks only for a high condition at the center of the first stop-bit position, that is, one bit-time after the last data bit or after the parity bit, if parity is enabled. If an external 1x clock is used for the transmitter, clearing bit 3 selects 1 stop bit and setting bit 3 selects 2 stop bits for transmission. Not used in Codec mode, see Table 15-9. **other Modes**—Reserved

Table 15-9. Stop-Bit Lengths

SB	5 Bits	6–8 Bits	SB	5 Bits	6–8 Bits	SB	5–8 Bits	SB	5–8 Bits
0000	1.063	0.563	0100	1.313	0.813	1000	1.563	1100	1.813
0001	1.125	0.625	0101	1.375	0.875	1001	1.625	1101	1.875
0010	1.188	0.688	0110	1.438	0.938	1010	1.688	1110	1.938
0011	1.250	0.750	0111	1.500	1.000	1011	1.750	1111	2.000

15.2.3 Status Register (0x04) — SR

The read-only SR register shows status of the transmitter, the receiver, and the FIFO.

Table 15-10. Status Register (0x04) for UART Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	RB	FE	PE	ORERR	TxEMP	TxRDY	FFULL	RxRDY	CDE	Reserved						
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-11. Status Register (0x04) for SIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	RB	FE	PE	ORERR	TxEMP	TxRDY	FFULL	RxRDY	Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-12. Status Register (0x04) for MIR / FIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	EOF	FHYERR	PE	ORERR	URERR	TxRDY	FFULL	RxRDY	DEOF	Reserved						
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-13. Status Register (0x04) for other Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved			ORERR	URERR	TxRDY	FFULL	RxRDY	Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RB/ EOF	**UART / SIR** —Received Break—detects breaks originating in middle of received character. Such a break must persist until the end of next detected character time. 0 = No break received. 1 = An all-0 character of the programmed length was received without a stop bit. RB is valid only when RxRDY = 1. Only a single FIFO position is occupied when a break is received. Further entries to FIFO are inhibited until RxD returns to high state for at least one-half bit-time, which equals two successive PSC clock edges. MIR / SIR—End of frame 0 = The next byte to be read from the RX-FIFO is not the last one of the frame. 1= The next byte to be read from the RX-FIFO is the last one of the frame. This bit is effective when RxRDY=1. **other Modes**—Reserved
1	FE/ PHYERR	**UART / SIR**—Framing Error— is not used (always 0) in Codec mode. 0 = No framing error occurred. 1 = No stop bit detected when corresponding FIFO data character received. Stop bit-check occurs in middle of first stop bit position. FE is valid only when RxRDY=1. MIR / FIR—Physical layer error 0 = No error 1 = In MIR mode, this denotes that the RX received an abort. In FIR mode, this denotes that there was a decode error. This bit can be cleared by the reset error status command in the CR. **other Modes**—Reserved
2	PE	**UART / SIR**—Parity Error—valid only if RxRDY = 1. PE is not used (always 0) in Codec mode. 0 = No parity error occurred. 1 = If MR1[PM]=0x (with parity or force parity), corresponding FIFO character was received with incorrect parity. If MR1[PM]=11 (multidrop), PE stores received A/D bit.\ **other Modes**—Reserved
3	ORERR	Overrun Error Indicates whether an overrun occurs. For purposes of overrun, FIFO full means all FIFO space is occupied; the Rx FIFO threshold is irrelevant to overrun. 0 = No overrun occurred. 1 = One or more characters in Rx data stream were lost. ORERR sets on receipt of a new character when FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the Rx shift register and its break detect, framing error status, and parity error, if any, are lost. ORERR is cleared by the RESET ERROR STATUS command in CR. Also see the note on the end of this table.

Bit	Name	Description
4	TxEMP/URERR	**UART / SIR**—Transmitter Empty 0 = Tx buffer not completely empty. Either a character is being shifted out, or Tx is disabled. Tx is enabled/disabled by programming CR [TC]. 1 = Tx has underrun (both the Tx holding register and Tx shift registers are empty). This bit sets after transmission of the last stop bit of a character, if there are no characters in the Tx holding register awaiting transmission. **other Modes**—Underrun error 0 = No error. 1 = Underrun error occurred, which means the number of Tx FIFO bytes is 0, the Tx shift register is empty, and a frame sync occurs. In other words, the time has come to transmit a new sample, but no sample is available in the Tx shift register. Unlike UART mode, TxEMP high indicates an error condition similar to the overrun condition (ORERR = 1), and as such it is now cleared the same way as ORERR, by a RESET ERROR STATUS command in the CR and not by a reset Tx command in the CR. Also see the note on the end of this table.
5	TxRDY	Transmitter Ready 0 - Tx FIFO contains a number of data bytes greater than the TFALARM register value, or the Tx is disabled. 1 - Tx FIFO is “almost empty” as defined by the TFALARM. TxRDY sets when the number of Tx FIFO bytes falls to, or below, the TFALARM value, due to data transfer from the Tx FIFO to the Tx shift register. Once set, TxRDY remains set until the number of empty bytes in the Tx FIFO falls to 4 times the granularity level specified in the TFCNTL register. In UART mode this bit only asserts if the Tx is enabled. Also see the note on the end of this table.
6	FFULL	Rx FIFO full 0 = The Rx FIFO is not “almost full” 1 = Rx FIFO is “almost full” as defined by the RFALARM. FFULL sets as soon as the number of bytes in the Rx FIFO exceeds the RFALARM value, due to the transfer of data from the Rx shift register to the Rx FIFO. Once set, FFULL remains set until the number of bytes in the Rx FIFO falls to the granularity level specified in the RFCNTL register. Also see the note on the end of this table.
7	RxRDY	Receiver Ready 0 = There is no data in the Rx FIFO. 1 = One or more characters were received and are waiting in the Rx buffer FIFO. Also see the note on the end of this table.
8	CDE	**UART**—DCD Error 0 = The $\overline{\text{DCD}}$ input is negated while receiving data. 1 = No error **other Modes**—Reserved
9:15	—	Reserved

NOTE

The FIFO related status bits ORERR, URERR, RxRDY, FFUL and TxRDY will be changed only if the peripheral (transmitter or receiver) access the FIFO. These bits reflect to the related bits in the ISR, therefore only the peripheral side can generate a FIFO access interrupt. The bits in the RFSTAT or TFSTAT register are also set. If the CPU side read from an empty FIFO or write to a full FIFO the status bits in the RFSTAT or TFSTAT register will be set, but the status bit in the SR register are unchanged. An access from the CPU side to the FIFO can’t generate an interrupt.

15.2.4 Clock Select Register (0x04) — CSR

The MPC5200 supports only one internal clock source for the UART / SIR mode. This internal clock source is the IPB clock divide by 32. After reset, the clock generation for UART/ SIR mode is disabled. Writing a valid value “0000 to 1110” to the RCS or TCS fields enable the clock generation. Writing “1110” or “1111” to this register will stop the UART / SIR clock generation.

Table 15-14. Clock Select Register (0x04) for UART / SIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R																
W	RCS				TCS				Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-15. Clock Select Register (0x04) for other Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	RCS	**UART / SIR** —Receiver Clock Enable 0000 -1101 = enable the clock generation 1110 -1111 = disable the clock generation **other Modes**—Reserved
4:7	TCS	**UART / SIR** —Transmitter Clock Enable 0000 -1101 = enable the clock generation 1110 -1111 = disable the clock generation **other Modes**—Reserved
8:15	—	Reserved

15.2.5 Command Register (0x08)—CR

The write-only command registers (CR), provide the commands to the PSC in all modes. Only multiple commands that do not conflict can be specified in a single write to a CR. For example, reset Tx and enable Tx cannot be specified in one command.

Table 15-16. Command Register (0x08) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W	Reserved	MISC			TC		RC	
RESET:	0	0	0	0	0	0	0	0

Bit	Value	Command	Description
0	—		Reserved

Bit	Value	Command	Description
1:3	000	no command	—
	001	reset mode register pointer	Causes MR register address to point to MR1.
	010	reset receiver	Immediately disables receiver, clears SR[FFULL,RxRDY], and re-initializes receiver FIFO pointer. No other registers are altered. Because it places the receiver in a known state, use this command instead of RECEIVER DISABLE when reconfigure the receiver.
	011	reset transmitter	In UART mode, immediately disables Tx and clears SR[TxEMP,TxRDY]. No other registers are altered. Because it places Tx in a known state, use this command instead of TRANSMITTER DISABLE when reconfigure transmitter. In Codec mode, URERR is not cleared by this soft reset. It is cleared the same way as the Rx overflow bit, by a RESET ERROR STATUS command.
	100	reset error status	In UART mode, clears SR[RB,FE,PE,ORERR]. In Codec mode, command clears ORERR and URERR.
	101	reset break change interrupt	Clears the delta break bit, ISR[DB]. Command has no effect in Codec mode.
	110	start break	Forces TxD low • If Tx is empty, break may be delayed up to one bit-time. • If Tx is active, break starts when character transmission completes. Break is delayed until any character in Tx shift register is sent. Any character in Tx holding register is sent after the break. Tx must be enabled for command to be accepted. This command ignores the $\overline{\text{CTS}}$ state and has no effect in Codec mode.
	111	stop break	Causes TxD to go high (mark) within two bit-times. Any characters in the Tx buffer are sent.

Bit	Value	Command	Description
4:5	00	no action taken	Causes Tx to stay in current mode. • If Tx is enabled, it remains enabled. • If Tx is disabled, it remains disabled.
	01	transmitter enable	Enables operation of Tx channels. SR[TxEMP,TxRDY] sets. If Tx is already enabled, this command has no effect. In UART mode, TxRDY and TxEMP bits in SR become asserted. In Codec mode: Tx FIFO can be loaded while Tx is disabled, unlike in UART mode. Therefore this command does not affect TxRDY or URERR behavior. It does not automatically set TxRDY and URERR. If no data is written to Tx FIFO, URERR sets at the first frame sync after Tx is enabled. In AC97 mode: URERR sets if Tx FIFO is empty, Tx is enabled, Rx detects a “Codec Ready” condition, and a frame sync occurs before samples are written to the Tx FIFO. **Note:** In Codec / AC97 mode it’s not possible to use the transmitter without the receiver. To transmit data also the receiver must be enabled!
	10	transmitter disable	Terminates Tx operation and clears SR[TxEMP,TxRDY]. • If a character is being sent when Tx is disabled, transmission completes before Tx becomes inactive. • If Tx is already disabled, the command has no effect. In UART mode, SR[TxEMP,TxRDY] are negated. In Codec mode SR[TxEMP] is negated. Tx does not clear SR[TxRDY] unless PSC is in remote loop-back or auto-echo mode. In Codec mode, unlike UART mode, the Tx FIFO may be loaded while Tx is disabled.
	11	—	Reserved, do not use.
6:7	00	no action taken	Causes receiver to stay in current mode. • If receiver is enabled, it remains enabled. • If receiver is disabled, it remains disabled.
	01	receiver enable	Enables receiver • If PSC module is not in multidrop mode (MR1[PM] ≠ 11), RECEIVER ENABLE command enables channel's receiver and forces it into a search-for-start-bit state. In multidrop mode the Rx continuously monitors the received data regardless of whether it is enabled or not. • If receiver is already enabled, this command has no effect.
	10	receiver disable	Immediately disables receiver. In UART mode any character being received is lost. The command does not affect receiver status bits or other control registers. • If the PSC module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. • If the receiver is already disabled, the command has no effect. In Codec mode, if the receiver is disabled while a character is being received, reception completes before the receiver becomes inactive.
	11	—	Reserved, do not use.
Note: This field selects a single command.			

15.2.6 Rx Buffer Register (0x0C) — RB

Data are read from the Rx FIFO by reading from this read-only register. The Rx FIFO size is 512 bytes. To read data from the RX FIFO you can also use the RFDATA register, see Section 15.2.27, *Rx FIFO Data (0x60)—RFDATA*.

Table 15-17. Rx Buffer Register (0x0C) for UART/SIR/MIR/FIR/Codec8/16/32

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
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 lsb
R	RB[16:31]															
W	Used by Tx Buffer															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-18. Rx Buffer Register (0x0C) for AC97

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
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 lsb
R	RB[16:19]				SOF	Reserved										
W	Used by Tx Buffer															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-19. Rx Buffer Register (0x0C) for Codec24

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
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 lsb
R	RB[16:23]								Reserved							
W	Used by Tx Buffer															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:19 (AC97) or 0:31 (other)	RB	**AC97** (0:19)—Received data—AC97 data must be read one complete sample at a time, where all samples except timeslot #0 are 20 bits. Timeslot #0 data is in bits 0:15. Bit 20 is 1 in the first sample of a new frame. Bit 20 contains the “Start of Frame Indicator” SOF: 0 = RB[0:19] is not the first sample in the frame. 1 = RB[0:15] is the first sample in a new frame. The number 0 slot is called the TAG slot. The bits [21:31] are reserved at this mode. **UART/SIR/MIR/FIR/Codec8** (0:31)—Received data—For these modes, data can be read 1, 2 or 4 bytes at a time. For one byte at a time, all bytes must be read from bits 0:7. For 2 bytes at a time, data must be read from bits 0:15. Lower-bit data was received before upper-bit data. **Codec16** (0:31)—Received data—For these modes, data can be read 2 or 4 bytes at a time. For 2 bytes at a time, data must be read from bits 0:15. Lower-bit data was received before upper-bit data. **Codec24** (0:23)—Received data—For these modes, data must be read 4 bytes at a time. The lower 24 bits contain the received data word. **Codec32** (0:31)—Received data—For these modes, data must be read 4 bytes at a time.

15.2.7 Tx Buffer Register (0x0C)—TB

Data is written to the Tx FIFO by writing to this write-only register. The Tx FIFO size is 512 bytes. To write data to the TX FIFO you can also use the TFDATA register, see Section 15.2.35, *Tx FIFO Data (0x80)—TFDATA*.

NOTE

Please note that the TX FIFO access via TB address will be blocked if the PSC was set to UART or SIR mode **and** the transmitter is disabled. The access via TFDATA will be never blocked.

Table 15-20. Tx Buffer Register (0x0C) for UART/SIR/MIR/FIR/Codec8/16/32 Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Used by Rx Buffer															
W	TB[0:15]															
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 lsb
R	Used by Rx Buffer															
W	TB[16:31]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-21. TX Buffer Register (0x0C) for AC97) Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Used by Rx Buffer															
W	TB[0:15]															
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 lsb
R	Used by Rx Buffer															
W	TB[16:19]				SOF	Reserved										
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-22. Tx Buffer Register (0x0c) for Codec24

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB[0:15]															
W	Used by Tx Buffer															
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 lsb
R	RB[16:23]								Reserved							
W	Used by Tx Buffer															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:19 (AC97) or 0:31 (other)	RB	**AC97** (0:19)—Transmit data—AC97 data must be written one complete sample at a time, where all samples except timeslot #0 are 20 bits. Timeslot #0 data is in bits 0:15. Bit 20 is 1 in the first sample of a new frame. Bit 20 contains the “Start of Frame Indicator” SOF: 0 = RB[0:19] is not the first sample in the frame. 1 = RB[0:15] is the first sample in a new frame. The number 0 slot is called the TAG slot. The bits [21:31] are reserved at this mode. **UART/SIR/MIR/FIR/Codec8** (0:31)—Transmit data—For these modes, data can be written 1, 2 or 4 bytes at a time. For one byte at a time, all bytes must be written to bits 0:7. For 2 bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data. **Codec16** (0:31)—Transmit data—For these modes, data can be written 2 or 4 bytes at a time. For 2 bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data. **Codec24** (0:23)—Transmit data—For these modes, data must be written 4 bytes at a time. The lower 24 bits contain the valid data word. **Codec32** (0:31)—Transmit data—For these modes, data must be written 4 bytes at a time.

15.2.8 Input Port Change Register (0x10) — IPCR

The read-only IPCR register shows the current state and change-of-state for the Modem control input port.

Table 15-23. Input Port Change Register (0x10) for UART/SIR/MIR/FIR Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved		D_DCD	D_CTS	Reserved		DCD	CTS
W	Used by ARC							
RESET:	0	0	0	0	0	0	0	0

	msb 0	1	2	3	4	5	6	7 lsb
R	SYNC	Reserved	D_DCD	D_CTS	Reserved		DCD	CTS
W	Used by ARC							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	SYNC	**Codec**—Sync detected. 0 = Has not detected sync. 1 = Detected sync (Frame = 1 in Codec Modes or Sync = 1 in AC97 mode) **other Modes**—Reserved
1	—	Reserved
2	D_DCD	Delta DCD. 0 = No change-of-state has occurred since the last time the CPU read the IPCR. A read of the IPCR also clears the ISR D_DCD bit. 1 = A change of state, lasting more than a certain time (1/16 or 1 bit duration determined by the CSR, CTUR and CTLR) has occurred at $\overline{DCD}$ input. When this bit is set, the ACR can be programmed to generate an interrupt to the processor.
3	D_CTS	Delta CTS. 0 = No change-of-state has occurred since the last time the CPU read the IPCR. A read of the IPCR also clears the ISR D_CTS bit. 1 = A change of state, lasting a certain time has occurred at $\overline{CTS}$ input. When this bit is set, the ACR can be programmed to generate an interrupt to the processor.
4:5	—	Reserved
6	DCD	Current state of $\overline{DCD}$ port. This input is double latched. 0 = The current state of the DCD input port is low. 1 = The current state of the DCD input port is high.
7	CTS	Current state of $\overline{CTS}$ port. This input is double latched. 0 = The current state of the $\overline{CTS}$ input port is low. 1 = The current state of the $\overline{CTS}$ input port is high.

15.2.9 Auxiliary Control Register (0x10) — ACR

The write-only ACR register controls Tx/Rx handshaking.

Table 15-24. PSC 1 Auxiliary Control Register (0x10) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Used by IPCR							
W	Reserved						IEC1	IEC0
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:5	—	Reserved
6	IEC1	Interrupt enable control for D_DCD. 0 = D_DCD has no effect on the IPC in the ISR. 1 = When the D_DCD becomes high, IPC bit in the ISR sets (causing an interrupt if mask is not set).
7	IEC0	Interrupt enable control for D_CTS. 0 = D_CTS has no effect on the IPC in the ISR. 1 = When the D_CTS becomes high, IPC bit in the ISR sets (causing an interrupt if mask is not set).

15.2.10 Interrupt Status Register (0x14) — ISR

The read-only ISR register provides status for all potential interrupt sources. Register contents is masked by the IMR.

- If an ISR flag sets and the corresponding IMR bit is also set, the internal interrupt output is asserted.
- If the corresponding IMR bit is cleared, the ISR bit state has no effect on the output.

Table 15-25. Interrupt Status Register (0x14) for UART / SIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved		ORERR	URERR	DB	RxRDY FFULL	TxRDY	Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-26. Interrupt Status Register (0x14) other Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved		ORERR	URERR	Reserved	RxRDY FFULL	TxRDY	DEOF	Reserved						
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	IPC	Input port change interrupt. 0 = No IPC event was occurred. 1 = An IPC event was occurred.
1:2	—	Reserved
3	ORERR	Overrun Error This bit is identical to the ORERR bit in the SR register. To clear this interrupt use the reset error status command in the CR register.
4	URERR	Underrun Error This bit is identical to the URERR bit in the SR register. To clear this interrupt use the reset error status command in the CR register.
5	DB	**UART / SIR**—Delta Break **other Modes**—Reserved
6	RxRDY FFULL	Rx FIFO over threshold. If MR1[1]=1, then this bit is identical to the FFULL bit in the SR register. If MR1[1]=0, then this bit is identical to the RxRDY bit in the SR register.
7	TxRDY	Transmitter ready - identical to the TxRDY bit in the SR register
8	DEOF	**MIR / FIR**—Detect End of Frame 0 = Rx did not receive an EOF after the last read SR command. 1 = Rx received the EOF in the frame. In this case, the interrupt and request can be asserted even if the Rx FIFO number is less than the threshold and MR1[1]=1.
9:15	—	Reserved

15.2.11 Interrupt Mask Register (0x14)—IMR

The write-only IMR register selects corresponding bits in the ISR that cause an interrupt.

- If one ISR bit is set and the corresponding IMR bit is also set, the internal interrupt output is asserted.
- If the corresponding bit in IMR is 0, the state of the ISR bit has no effect on the interrupt output. The IMR does not mask reading the ISR.

Table 15-27. Interrupt Mask Register (0x14) for UART / SIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W	IPC	Reserved		ORERR	URERR	DB	RxRDY FFULL	TxRDY	Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-28. Interrupt Mask Register (0x14) for other Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W	IPC	Reserved		ORERR	URERR	Reserved	RxRDY FFULL	TxRDY	DEOF	Reserved						
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	IPC	Input port change interrupt. 0 = IPC has no effect on the interrupt. 1 = Enable the interrupt for IPC in the ISR register.
1:2	—	Reserved
3	ORERR	Overrun Error 0 = ORERR has no effect on the interrupt. 1 = Enable the interrupt for ORERR
4	URERR	Underrun Error 0 = URERR has no effect on the interrupt. 1 = Enable the interrupt for URERR
5	DB	**UART / SIR** —Delta Break 0 = DB has no effect on the interrupt. 1 = Enable the interrupt for DB **other Modes**—Reserved
6	RxRDY FFULL	Rx FIFO over threshold 0 = RxRDY/FFULL has no effect on the interrupt. 1 = Enable the interrupt for RxRDY/FFULL.
7	TxRDY	Transmitter ready 0 = TxRDY has no effect on the interrupt. 1 = Enable the interrupt for TxRDY
8	DEOF	**MIR / FIR** Detect End of Frame 0 = DEOF has no effect on the interrupt. 1 = Enable the interrupt for DEOF.
9:15	—	Reserved

15.2.12 Counter Timer Upper Register (0x18)—CTUR

This write-only register holds the upper bytes of the preload value used by the timer to provide a given Baud rate. Reading from this register shows the current value of the Baud rate generation counter. For detailed description see next section, Section 15.2.13, *Counter Timer Lower Register (0x1C)—CTLR*.

Table 15-29. Counter Timer Upper Register (0x18) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W	CTUR[0:7]							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	CTUR	**Code**—Frame Sync Length, define the number of Bit clocks during the frame sync signal is active. Frame Sync Lenght = CTUR[0:7]+1 **UART/ SIR/ SPI** —Baud rate prescale value. See next section, Section 15.2.13, *Counter Timer Lower Register (0x1C)—CTLR* **Other**—Reserved

15.2.13 Counter Timer Lower Register (0x1C)—CTLR

This write-only register hold the lower bytes of the preload value used by the timer to provide a given Baud rate. Reading from this register shows the current value of the Baud rate generation counter.

Table 15-30. Counter Timer Lower Register (0x1C) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W	CTLR[0:7]							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	CTLR	**UART**—Baud rate prescale value. The Baud rate is calculated as: $\text{Baud rate} = \frac{\text{IPB clock frequency}}{\text{CT[0:15]} \times 32}$ where: CT[0:15] = {CTUR[0:7], CTLR[0:7]} The minimum CT value is 1; 0 denotes counter stop. **SIR**—Baud rate prescale value. The Baud rate is calculated as: $\text{Baud rate} = \frac{\text{IPB clock frequency}}{\text{CT[0:15]} \times 32}$ where: CT[0:15] = {CTUR[0:7], CTLR[0:7]} The minimum CT value is 1; 0 denotes counter stop. **SPI**—Delay After Transfer (DTL) $\text{DTL} = \frac{\text{CT[0:15]}}{\text{IPB clock frequency}}$ where: CT[0:15] = {CTUR[0:7], CTLR[0:7]} **Other**—Reserved

DTL — Length of Delay after Transfer

When the PSC is in SPI mode (SICR[SPI] = 1), the Counter Timer is used to determine the length of time that the PSC delays after each serial transfer, i. E. the length of time that SS stays high/inactive between consecutive transfers. This is a feature that exists in a QSPI. Delay after transfer can be used to ensure that the deselect time requirement (for peripherals having such a requirement) is met. Some peripherals must be deselected for a minimum period of time between consecutive serial transfers. A delay after transfer can be inserted between consecutive transfers to a given peripheral to ensure that its minimum deselect time requirement is met or to allow serial A/D converters to complete conversion before the next transfer is made.

15.2.14 Codec Clock Register (0x20)—CCR

This register defines the divider for the Frame and BitClk generation for Codec mode. This register value has only effect, if the GenClk bit in the PSC Control Register SICR was set to one. In UART, SIR and AC97 mode this register is reserved.

Table 15-31. Codec Clock Register (0x20)—CCR for Codec Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	FrameSyncDiv[0:7]								BitClkDiv[0:7]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 15-32. Codec Clock Register (0x20)—CCR for MIR/FIR Mode

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved								BitClkDiv[0:7]							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 15-33. Codec Clock Register (0x20)—CCR for other Modes

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	FrameSyncDiv	**Codec**—Frame Sync Divider Frame sync is generated internally by dividing down the Bit Clock frequency as follows: FrameSync frequency = BitClk / (FrameSyncDiv[0:7] +1) **Codec / SPI**—delay before SCK (DSCKL) When the PSC is in SPI mode (SICR[SPI] = 1), the FrameSyncDiv divider is used to determine the length of time the PSC delays after SS goes low/active before the first SCK transition of the serial transfer. This is a feature that exists in a QSPI. The following equation determines the actual delay before SCK: $\text{DSCKL delay} = \frac{\text{FrameSyncDiv[0:7]} + 1)}{\text{Mclk Frequency}}$ **other Modes**—Reserved **Note:** The value 0x00 stops this counter and disables the clock generator.
7:15	BitClkDiv	**Codec**—Bit Clock Divider Bit clock is generated internally by dividing down the Mclk frequency as follows: $\text{BitClk frequency} = \frac{\text{Mclk Frequency}}{\text{BitClkDiv[0:7]} + 1}$ **Codec SPI**—Baud Rate SCK is generated internally by dividing down the Mclk frequency as follows: $\text{SCK frequency} = \frac{\text{Mclk Frequency}}{\text{BitClkDiv[0:7]} + 1}$ **MIR / FIR**—Irda clock IrdaClk is generated internally by dividing down the Mclk frequency as follows: $\text{IrdaClk frequency} = \frac{\text{Mclk Frequency}}{\text{BitClkDiv[0:7]} + 1}$ **other Modes**—Reserved **Note:** The value 0x00 stops this counter and disables the clock generator.

The Mclk frequency is generated in the Clock Distribution Module (CDM) by dividing down the f_{system} frequency as follows:

$$Mclk = \frac{f_{system}}{MclkDiv\ [8:0] + 1}$$

There is a separate cdm_pscX_bitclk_config register in the CDM for each of PSC1,2,3 and 6, which are the PSCs available for use in Codec modes. These cdm_pscX_bitclk_config registers are further described in the CDM Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228* to Section 5.5.14, *PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234*.

15.2.15 Interrupt Vector Register (0x30)—IVR

This register is not used since the MPC5200 does not use interrupt vectors supplied by the peripherals.

Table 15-34. Interrupt Vector Register (0x30) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	IVR[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IVR	Interrupt Vector— Not applicable for MPC5200.

15.2.16 Input Port Register (0x34)—IP

This read-only IP register shows the current state of the input ports.

Table 15-35. Input Port Register (0x34) for UART/SIR/MIR/FIR Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved						DCD	CTS
W								
RESET:	1	1	1	1	1	1	0	0

Table 15-36. Input Port Register (0x34) for Codec Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved	TGL	Reserved				DCD	CTS
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-37. Input Port Register (0x34) for AC97 Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	LPWR	TGL	Reserved				DCD	CTS
W								
RESET:	1	1	1	1	1	1	0	0

Bit	Name	Description
0	LPWR	**AC97**—Low power mode in AC97 mode 0 = Codec is in low power mode. 1 = Usual operation. **other Modes**—Reserved
1	TGL	**AC97 / Codec**—Test usage. Toggle by frame sync. **other Modes**—Reserved
2:5	—	Reserved
6	DCD	Current state of the $\overline{\text{DCD}}$ input. 0 = $\overline{\text{DCD}}$ input is low. 1 = $\overline{\text{DCD}}$ input is high.
7	CTS	Current state of the $\overline{\text{CTS}}$ input 0 = Input port $\overline{\text{CTS}}$ is low. 1 = Input port $\overline{\text{CTS}}$ is high.

15.2.17 Output Port 1 Bit Set (0x38)—OP1

This is a write-only register. Output ports are asserted by writing to this register.

Table 15-38. Output Port 1 Bit Set Register (0x38) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W							RES	RTS
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:5	—	Reserved
6	RES	Assert RES output. 0 = No operation 1 = Asserts output port RES, (RES becomes 0)
7	RTS	**AC97**—Reserved **other Modes**—Assert RTS output. 0 = No operation 1 = Asserts output port $\overline{\text{RTS}}$, ($\overline{\text{RTS}}$ becomes 0)

15.2.18 Output Port 0 Bit Set (0x3C)—OP0

This is a write-only register. Output ports are negated by writing to this register.

Table 15-39. Output Port 0 Bit Set Register (0x3C) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W							RES	RTS
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:5	—	Reserved
6	RES	Assert RES output. 0 = No operation 1 = Negates output port RES, (RES becomes 1).
7	RTS	**AC97**—Reserved **other Modes**—Assert $\overline{\text{RTS}}$ output. 0 = No operation 1 = Negates output port $\overline{\text{RTS}}$, ($\overline{\text{RTS}}$ becomes 1).

15.2.19 Serial Interface Control Register (0x40)—SICR

This register sets the main operation mode.

Table 15-40. Serial Interface Control Register (0x40) for all Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	ACRB	AWR	DTS1	SHDIR	SIM[3:0]			
W								
RESET:	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R	GenClk	MultiWd	ClkPol	SyncPol	CellSlave	Cell2xClk	Reserved	
W								
RESET:	0	0	0	1	0	0	0	0

	16	17	18	19	20	21	22	23 msb
R	SPI	MSTR	CPOL	CPHA	UseEOF	Reserved		
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	ACRB	**AC97**—AC97 Cold Reset to the transceiver in PSC. This bit was prepared for backward compatibility with the MCF5407 USART. It is recommended to use OP1 and OP0 registers to set and to reset AC97 reset line. 0 = The transceiver recovers from low power mode in AC97. 1 = The transceiver stays in the current state. **other Modes**—Reserved
1	AWR	**AC97**—AC97 Warm Reset (to the PSC and off-chip AC97 Codec) 0 = AC97 warm reset is negated. $\overline{\text{RTS}}$ output functions normally as the AC97 frame sync. 1 = Force “1” on $\overline{\text{RTS}}$ output, which is used as the AC97 frame sync, and the PSC recovers from AC97 power down mode. **other Modes**—Reserved

Bit	Name	Description
2	DTS1	**Codec**—Delay of time slot #1. 0 = first bit of first time slot of a new frame starts at the rising edge of frame sync. 1 = first bit of first time slot of a new frame starts one bit clock cycle after the rising edge of frame sync. **other Modes**—Reserved
3	SHDIR	**Codec**—Shift Direction. 0 = msb first 1 = lsb first **other Modes**—Reserved
4:7	SIM[3:0]	PSC operation mode. **CAUTION**: When the operating mode change occurs, all Rx/Tx and error statuses are reset. Rx and Tx are disabled. 0000 = UART mode, $\overline{\text{DCD}}$ input ignored 1000 = UART mode, $\overline{\text{DCD}}$ input is effective x001 = Codec mode, 8-bit data x010 = Codec mode, 16-bit data x011 = AC97 mode 0100 = SIR model, $\overline{\text{DCD}}$ input ignored 1100 = SIR mode, $\overline{\text{DCD}}$ input is effective x101 = MIR mode x110 = FIR mode 0111 = Codec mode, 24-bit data 1111 = Codec mode, 32-bit data
8	GenClk	**Codec**—Generate Bit Clock and Frame Sync, not used to enable the SPI master mode, use the MSTR bit of the this register 0 = use bit clock and frame sync provided by external device 1 = use bit clock and frame sync generated internally from Mclk **MIR / FIR**—Generate Bit Clock and Frame Sync 0 = use for clock generation the external Clk from Pad IR_USB_CLK 1 = use for clock generation the internal Mclk **other Modes**—Reserved
9	MultiWd	**Codec**—Multi Word mode 0 = PSC send and receive only one data word per frame even if the frame length is greater than the word length. 1 = PSC send and receive more the one data word per frame, if the frame length is greater than the word length. The PSC send only complete data words. This bit is used to support the I2S mode. See Figure 15-14 **other Modes**—Reserved
10	ClkPol	**Codec**—Bit Clock Polarity 0 = data in is sampled on the falling edge of the BitClk and data out is shifted on the rising edge 1 = data in is sampled on the rising edge of the BitClk and data out is shifted on the falling edge **other Modes**—Reserved

Bit	Name	Description
11	SyncPol	**Codec**—Frame Sync Polarity 0 = Frame Sync is low true 1 = Frame Sync is high true **other Modes**—Reserved
12	CellSlave	**Codec**—Cell Phone Slave 0 = PSC is not a slave to PSC1 1 = PSC uses Bit Clock from PSC1 master as its Mclk
13	Cell2xClk	**Codec** —Cell Slave 2x Clock Frequency - takes effect only when bit 12 CellSlave = 1 0 = PSC Mclk frequency = Bit Clock from PSC1 master 1 = PSC Mclk frequency = 2x the Bit Clock from PSC1 master **other Modes**—Reserved
14:15	—	Reserved
16	SPI	**Codec**—SPI mode 0 = PSC does not behave like an SPI 1 = PSC behaves like an SPI **other Modes**—Reserved
17	MSTR	**Codec**—SPI Master mode - takes effect only when bit 16 SPI mode = 1 0 = PSC behaves as an SPI slave 1 = PSC behaves as an SPI master **other Modes**—Reserved
18	CPOL	**Codec**—SPI Clock Polarity - takes effect only when bit16 SPI mode = 1 This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values Active-low clocks selected; SCK idles high Active-high clocks selected; SCK idles low **other Modes**—Reserved
19	CPHA	**Codec**—SPI Clock Phase This bit is used to shift the SCK serial clock. 0 = The first SCK edge is issued one-half cycle into the data transfer 1 = The first SCK edge is issued at the beginning of the data transfer **other modes**—Reserved
20	UseEOF	**Codec**—Use End-of-Frame flag takes effect only when bit 16 SPI mode = 1 0 = either 1, 2 or 4 bytes are transferred while Slave Select (SS) is held low, as determined by Codec8, Codec16, Codec24 or Codec32 being selected by SICR[SIM] 1 = multiple bytes are transferred while maintaining SS low, up to and including the next byte read from the Tx FIFO that has its EOF flag set **other modes**—Reserved
21:23	—	Reserved

15.2.20 Infrared Control 1 (0x44)—IRCR1

This register controls the configuration in one of the IrDA modes (SIR/MIR/FIR).

Table 15-41. Infrared Control 1 (0x44) for SIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R W	Reserved					FD	Reserved	SPUL
RESET:	0	0	0	0	0	0	0	0

Table 15-42. Infrared Control 1 (0x44) for MIR/FIR Modes

	msb 0	1	2	3	4	5	6	7 lsb
R W	Reserved					FD	SIPEN	Reserved
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5	FD	**SIR / MIR / FIR**—Full duplex enable 0 = The receiver in IrDA mode is disabled while the TX is busy. 1= The receiver in IrDA mode is not disabled while the TX is busy. This bit should not be set in usual operations. In loop back channel mode, CM=10, this bit is automatically set. **other Modes**—Reserved
6	SIPEN	**MIR / FIR**—Send SIP enable after every frame 0 = SIP is sent only when the SIPREQ bit in the IRCR2 becomes high. 1 = The TX always send 1.6 µs SIP after the STO flag in order to inform slow speed devices that higher speed device is connecting. For more informations about the SIP pulse see also Figure 15-21.. **other Modes**—Reserved
7	SPUL	**SIR**—SIR pulse width 0 = SIR pulse width is 3/16 of the bit duration. 1 = SIR pulse width is 1.6 µs **other Modes**—Reserved

15.2.21 Infrared Control 2 (0x48)—IRCR2

This register controls the configuration in one of the IrDA modes (SIR/MIR/FIR).

Table 15-43. Infrared Control 2 (0x48) for MIR/FIR Modes

	msb 0	1	2	3	4	5	6	7 lsb
R W	Reserved					SIPREQ	ABORT	NXTEOF
RESET:	0	0	0	0	0	0	0	0

Table 15-44. Infrared Control 2 (0x48) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R W	Reserved							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5	SIPREQ	**MIR / FIR**—Request to send SIP 0 = No operation 1 = If the TX becomes idle state, the TX starts to send one SIP pulse. This bit keeps high until the TX finishes sending a SIP and becomes low automatically when the TX finishes sending a SIP. For more informations about the SIP pulse see also Figure 15-21.. **other Modes**—Reserved
6	ABORT	**MIR / FIR**— Abort output 0 = Stop sending abort sequence. 1 = While the TX is sending data or CRC, writing 1 to this bit causes the TX immediately start to output abort sequence (2 or more illegal symbol “0000” in FIR mode, or 7 or more consecutive 1 in MIR mode). Before the next frame is transmitted, this bit must be reset. **other Modes**—Reserved
7	NXTEOF	**SIR / MIR / FIR**— Next is the last byte. This bit works independent form the mode. For other modes which support the framing, like SPI, you can also use this bit to mark the last bit of frame. 0 = The next write data is not the last byte in a frame. 1 = The next write data is the last byte in the current frame. When the processor performs a write to the TB, an EOF mark is added to the data in the TX-FIFO memory. This bit is cleared after writing to the transmit buffer. This bit is usually set by IP bus write operation. During the CommBus transfer, the signal transmit done line indicate the end of frame, therefore this bit is not used by the CommBus write operation. **other Modes**—Reserved

15.2.22 Infrared SIR Divide Register (0x4C)—IRSDR

This register set the SIR pulse width. To set the SIR mode Baud rate see Section 15.2.12, *Counter Timer Upper Register (0x18)—CTUR*. This register is reserved in other modes.

Table 15-45. Infrared SIR Divide Register (0x48) for SIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	IRSTIM[0:7]							
W								
RESET:	0	0	1	1	0	1	1	0

Table 15-46. Infrared SIR Divide Register (0x48) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IRSTIM	**SIR**—Timer counter value for 1.6us pulse In SIR mode, this is used to make 1.6 μs pulse when SPUL in the IRCR1 is high and SIPREQ in the IRCR2 is high. This value should be set so that IPB clock period * IRSTIM = 1.6 μs Reset value is 54(0x36) and this is for 33 MHz bus clock. **other Modes**—Reserved

15.2.23 Infrared MIR Divide Register (0x50)—IRMDR

This register set the MIR mode Baud rate. This register is reserved in other modes.

Table 15-47. Infrared MIR Divide Register (0x50) for MIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	FREQ	M_FDIV						
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-48. Infrared MIR Divide Register (0x50) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W								
RESET:	0	0	0	0	0	0		0

Bit	Name	Description
0	FREQ	**MIR**—0.576 M bps mode. 0 = The Baud rate is 1.152 M bps. 1 = If the Baud rate is 0.576 Mbps, this bit should be set high in order to output 1. For more informations about the SIP pulse see also Figure 15-21.. **other Modes**—Reserved
1:7	M_FDIV	**MIR**—Clock divide ratio in MIR mode. The bit frequency is derived by: $f_{bit} = \frac{f_{IrdaClk}}{M\_FDIV + 1}$ This bit frequency should be 0.576 or 1.152 MHz. In order to send a quarter bit duration pulse and receive minimum pulse described in the IrDA spec, (M_FDIV + 1) should be a factor of 4 and larger than or equal to 8. Table 15-49 shows the selectable divide factor and the input clock frequency on IrdaClk port. For more informations about the frequency generation see also Figure 15-20,Section 15.2.14, *Codec Clock Register (0x20)—CCR* and Section 15.3.5, *PSC in MIR Mode*. **other Modes**—Reserved

Table 15-49. Frequency Selection in MIR Mode

M_FDIV[1:7]	Frequency of IrdaClk [MHz]	
	1.152 Mbps	0.576 Mbps
0x07	9.216	4.6080
0x0B	13.824	6.912
0x0F	18.432	9.216
0x13	23.040	11.520
0x17	27.648	13.824
...	...	...
0x7F	147.456	73.728

15.2.24 Infrared FIR Divide Register (0x54)—IRFDR

This register set the FIR mode Baud rate. This register is reserved in other modes.

Table 15-50. Infrared FIR Divide Register (0x54) for MIR Mode

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved				F_FDIV			
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-51. Infrared FIR Divide Register (0x54) for other Modes

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W								
RESET:	0	0	0	0	0	0		0

Bit	Name	Description
0:3	—	Reserved
4:7	F_FDIV	**FIR**—Clock divide ratio in FIR mode. The bit frequency is derived by: $$f_{bit} = \frac{f_{IrdaClk}}{F\_FDIV + 1}$$ This bit frequency should be 8 MHz. In order to receive the minimum pulse width described in the IrDA spec, (F_FDIV + 1) should be larger than or equal to 4. Table 15-52 shows several frequency selection. For more informations about the frequency generation see also Figure 15-20, Section 15.2.14, *Codec Clock Register (0x20)—CCR* and Section 15.3.6, *PSC in FIR Mode*. **other Modes**—Reserved

Table 15-52. Frequency Selection for FIR Mode

F_FDIV[3:0]	Frequency of IrdaClk [MHz]
0x3	32.0
0x4	40.0
0x5	48.0
0x6	56.0
0x7	64.0
0x8	72.0
0x9	80.0
0xA	88.0
...	...

15.2.25 Rx FIFO Number of Data (0x58)—RFNUM

Table 15-53. RX FIFO Number of DATA (0x58)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved							COUNT[8:0]								
W								Reserved								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	—	Reserved
7:15	COUNT	Number of data bytes in the Rx FIFO.

15.2.26 Tx FIFO Number of Data (0x5C)—TFNUM

Table 15-54. Tx FIFO Number of Data (0x5C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved							COUNT[8:0]								
W								Reserved								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	—	Reserved
7:15	COUNT	Number of data bytes in the Tx FIFO.

15.2.27 Rx FIFO Data (0x60)—RFDATA

Read - write register to access the internal RX FIFO Data register. Reads from this register reads out the receive data. In addition the register provides the possibility to fill the RX FIFO for debug issues. For more informations about the data format see Section 15.2.6, *Rx Buffer Register (0x0C) — RB*.

15.2.28 Rx FIFO Status (0x64)—RFSTAT

For additional informations about the FIFO related status bits see Section 15.2.3, *Status Register (0x04) — SR*.

NOTE

To make sure that the PSC never lost the data in the FIFO, the PSC controller avoid writing to a full FIFO or reading from an empty FIFO. Therefore the status bits in the FIFO STAT register never reports an ERROR, UF or OF state. The SR register reports these errors.

Table 15-55. Rx FIFO Status (0x64)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				Frame[3]	Frame[2]	Frame[1]	Frame[0]	Rsvd	Error	UF	OF	FR	FULL	ALARM	EMPTY
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:7	Frame[3:0]	Frame indicator. Not applicable to PSC FIFO's, since the PSCs do not recognize frame formats in the serial data stream.
8	—	Reserved
9	Error	FIFO error. A FIFO error has occurred due to either underflow, overflow, or read or write pointer out of bounds.This bit is cleared by writing a '1' to it.
10	UF	Underflow. The read pointer has surpassed the write pointer due to the FIFO having been read when it contained no data. This bit is cleared by writing a '1' to it.
11	OF	Overflow. The write pointer has surpassed the read pointer due to the FIFO having been written when it was already completely full of data. This bit is cleared by writing a '1' to it.
12	FR	Frame ready. Not applicable to PSC FIFO's, since the PSCs do not recognize frame formats in the serial data stream.
13	FULL	Full. The FIFO is completely full of data.
14	ALARM	The FIFO is requesting service from either BestComm or CPU. See Section 15.2.30, *Rx FIFO Alarm (0x6E)—RFALARM* for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

15.2.29 Rx FIFO Control (0x68)—RFCNTL

Table 15-56. Rx FIFO Control (0x68)

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved		WFR	COMP	FRAME	GR[2:0]		
W								
RESET:	0	0	0	0	1	0	0	1

Bit	Name	Description
0:1	—	Reserved
2	WFR	Write frame. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.
3	COMP	Re-enable requests on frame transmission completion. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.
4	FRAME	Frame mode enable. THIS BIT MUST BE CLEARED BY WRITING A '0' TO IT, since the PSCs do not recognize frame formats in the serial data stream.
5:7	GR[2:0]	Last transfer granularity. Amount of data remaining in the Rx FIFO at which the ALARM bit in the status register will go low/inactive. See Section 15.3.7, *PSC FIFO System* for details.

15.2.30 Rx FIFO Alarm (0x6E)—RFALARM

Table 15-57. Rx FIFO Alarm (0x6E)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				ALARM											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	ALARM	“Almost full” threshold level. Amount of empty space remaining in the Rx FIFO at which the ALARM bit in the status register goes high/active. See Section 15.3.7, *PSC FIFO System* for details.

15.2.31 Rx FIFO Read Pointer (0x72)—RFRPTR

Table 15-58. Rx FIFO Read Pointer (0x72)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				R_PTR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	R_PTR	Read pointer.This FIFO-maintained pointer points to the next FIFO location to be read.

15.2.32 Rx FIFO Write Pointer(0x76)—RFWPTR

Table 15-59. Rx FIFO Write Pointer (0x76)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				W_PTR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	W_PTR	Write pointer.This FIFO-maintained pointer points to the next FIFO location to be written to.

15.2.33 Rx FIFO Last Read Frame (0x7A)—RFLRFPTR

Table 15-60. Rx FIFO Last Read Frame (0x7A)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				LFP											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	LFP	Last Frame Pointer. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.

15.2.34 Rx FIFO Last Write Frame PTR (0x7C)—RFLWFPTR

Table 15-61. Rx FIFO Last Write Frame PTR (0x7C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				LFP											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	LFP	Last Frame Pointer. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.

15.2.35 Tx FIFO Data (0x80)—TFDATA

Read - write register to access the internal TX FIFO Data register. Write to this register write data to the transmit FIFO. Additional the register provide the possibility to read data back from the TX FIFO for debug issues. For more informations about the data format see Section 15.2.7, *Tx Buffer Register (0x0C)—TB*.

15.2.36 Tx FIFO Status (0x84)—TFSTAT

For additional informations about the FIFO related status bits see Section 15.2.3, *Status Register (0x04) — SR*.

NOTE

To make sure that the PSC never lost the data in the FIFO, the PSC controller avoid writing to a full FIFO or reading from an empty FIFO. Therefore the status bits in the FIFO STAT register never reports an ERROR, UF or OF state. The SR register reports these errors.

Table 15-62. Tx FIFO STAT (0x84)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				Frame[3]	Frame[2]	Frame[1]	Frame[0]	Rsvd	Error	UF	OF	FR	FULL	ALARM	EMPTY
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:7	Frame[3:0]	Frame indicator. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.
8	—	Reserved
9	Error	FIFO error. A FIFO error has occurred due to either underflow, overflow, or read or write pointer out of bounds.This bit is cleared by writing 1 to it.
10	UF	Underflow. The read pointer has surpassed the write pointer due to the FIFO having been read when it contained no data. This bit is cleared by writing 1 to it.
11	OF	Overflow. The write pointer has surpassed the read pointer due to the FIFO having been written when it was already completely full of data. This bit is cleared by writing 1 to it.
12	FR	Frame ready. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.
13	FULL	Full. The FIFO is completely full of data.

Bit	Name	Description
14	ALARM	The FIFO is requesting service from either BestComm or CPU. See Section 15.3.7, *PSC FIFO System* for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

15.2.37 Tx FIFO Control (0x88)—TFCNTL

Table 15-63. Tx FIFO Control (0x88)

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved		WFR	COMP	FRAME	GR[2:0]		
W								
RESET:	0	0	0	0	1	0	0	1

Bit	Name	Description
0:1	—	Reserved
2	WFR	Write frame. Not applicable for PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream.
3	COMP	Re-enable requests on frame transmission completion. Not applicable to PSC FIFO's, since the PSCs do not recognize frame formats in the serial data stream.
4	FRAME	Frame mode enable. THIS BIT MUST BE CLEARED BY WRITING A '0' TO IT, since the PSCs do not recognize frame formats in the serial data stream.
5:7	GR[2:0]	Last transfer granularity. Four times this value is the amount of data remaining in the FIFO at which the ALARM bit in the status register will go low/inactive. See Section 15.3.7, *PSC FIFO System* for details.

15.2.38 Tx FIFO Alarm (0x8E)—TFALARM

Table 15-64. Tx FIFO Alarm (0x8E)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				ALARM											
W																
RESET:	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Bit	Name	Description
0:3	—	Reserved
4:15	ALARM	"Almost empty" threshold level. Amount of data remaining in the Tx FIFO at which the ALARM bit in the status register goes high/active. See Section 15.3.7, *PSC FIFO System* for details

15.2.39 Tx FIFO Read Pointer (0x92)—TFRPTR

Table 15-65. Tx FIFO Read Pointer (0x92)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				R_PTR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	R_PTR	Read pointer.This FIFO-maintained pointer points to the next FIFO location to be read

15.2.40 Tx FIFO Write Pointer (0x96)—TFWPTR

Table 15-66. Tx FIFO Write Pointer (0x96)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				W_PTR											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	W_PTR	Write pointer.This FIFO-maintained pointer points to the next FIFO location to be written to

15.2.41 Tx FIFO Last Read Frame (0x9A)—TFLRFPTR

Table 15-67. Tx FIFO Last Read Frame PTR (0x9A)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				LFP											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	LFP	Last Frame Pointer. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream

15.2.42 Tx FIFO Last Write Frame PTR (0x9C)—TFLWFPTR

Table 15-68. Tx FIFO Last Write Frame PTR(0x9C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				LFP											
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:15	LFP	Last Frame Pointer. Not applicable to PSC FIFOs, since the PSCs do not recognize frame formats in the serial data stream

15.3 PSC Operation Modes

This chapter describe the different PSC operation modes including the pin muxing, the module configuration, signal definition and some programming examples. All PSC are independent and can be used at the same time in different modes. But not all PSCs support all modes, Table 15-69 shows an overview.

Table 15-69. PSC Modes Overview

	PSC1	PSC2	PSC3	PSC4	PSC5	PSC6
UART	yes	yes	yes	yes	yes	yes
Modem / SPI / I2S	yes	yes	yes	no	no	yes
Mclk Generation	yes	yes	yes	no	no	no
AC97	yes	yes	no	no	no	no
IRDA	no	no	no	no	no	yes
“Cell Phone”	master	slave	slave	no	no	slave

In this section the following abbreviations will used as:

Table 15-70. Clock Short Cuts

Short Cut	Description
f_{system}	Clock from the system PLL
Mclk	Clock from the Mclk divider, used as clock input for internal clock generation or as clock output to an external device. Before modify the counter value the Mclk divider must be disable. See Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228* to Section 5.5.14, *PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234*. $$Mclk = \frac{f_{system}}{MclkDiv[8:0] + 1}$$
IP bus Clock	Intellectual Property Clock for the internal IP bus system, 33, 66 or 132 MHz, see Section 5.5, *CDM Registers*

15.3.1 PSC in UART Mode

Select the UART mode by writing the corresponding value to the PSC Control (SICR) register. The PSC UART mode is the default mode after reset. The important registers to configure the PSC for UART mode are:

- SICR register - select the UART mode
- CSR register - enable the clock generation
- CTUR, CTLR register - select the Baud rate
- MR1 register - select the UART mode (parity mode, bits per character)
- MR2 register - select $\overline{RTS}$ and $\overline{CTS}$ control, Stop Bit Length
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- Port_config - select the right Pin-Muxing, see Chapter 2, *Signal Descriptions*

15.3.1.1 Block Diagram and Signal Definition for UART Mode

The Figure 1-1 shows the simplified Block Diagram of the PSC for UART mode.

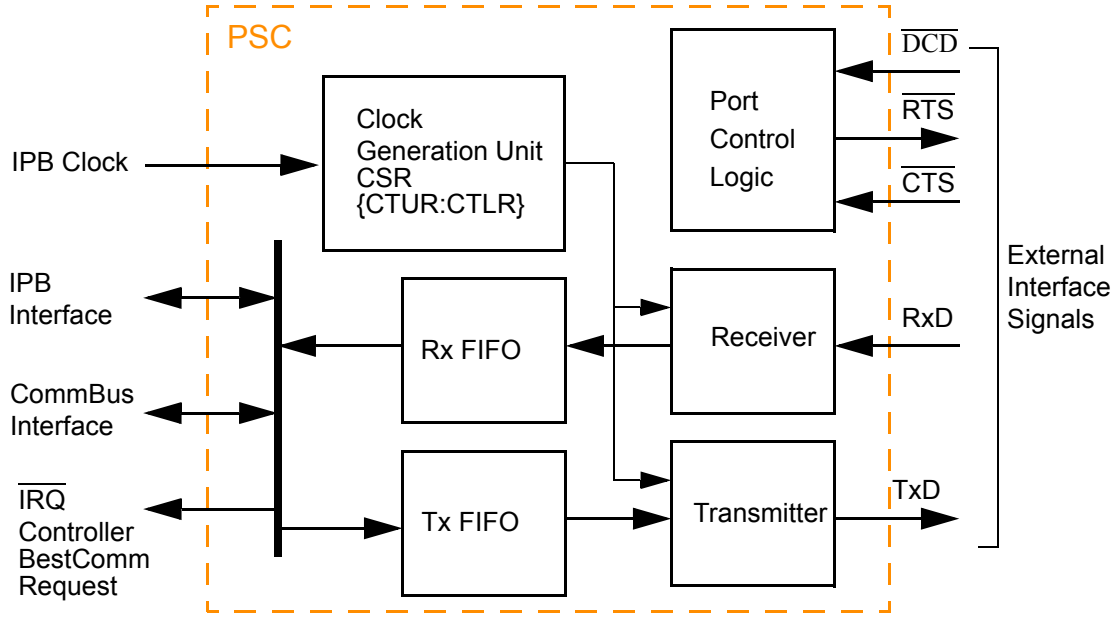


Figure 1-1PSC UART Block Diagram

An internal interrupt request signal ($\overline{\text{IRQ}}$) is provided to notify the Interrupt Controller of an interrupt condition. The output is the logical NOR of unmasked ISR bits. The interrupt level of a PSC module is programmed in the Interrupt Controller in the system integration module: Chapter 7, *System Integration Unit (SIU)*. The PSC uses the autovector for the programmed interrupt level.

The PSC can automatically transfer data using the BestComm, rather than interrupting the core. When IMR[FFULL] is 1 and Rx FIFO is full, it can send an interrupt to a BestComm channel so the FIFO data can be transferred to memory.

Table 15-71 briefly describes the PSC module signals.

NOTE

The terms "assertion" and "negation" are used to avoid confusion between active-low and active-high signals.

- *Asserted* indicates a signal is active, independent of the voltage level
- *Negated* indicates a signal is inactive.

Table 15-71. PSC Signal Description for UART Mode

Signal	Description
TxD	Transmitter Serial Data Output—TxD is held high (mark condition) when Tx is disabled, idle, or operating in the local loop-back mode. Data is shifted out on TxD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
RxD	Receiver Serial Data Input—Data received on RxD is sampled on the rising edge of the clock source, with the lsb received first.
$\overline{\text{CTS}}$	Clear-to-Send—This input can generate an interrupt on a change of state.
$\overline{\text{RTS}}$	Request-to-Send—This output can be programmed to be negated or asserted automatically by either Rx or Tx. When connected to a transmitter $\overline{\text{CTS}}$, $\overline{\text{RTS}}$ can control serial data flow.
$\overline{\text{DCD}}$	Data carrier detect Input — In the enhanced UART mode this signal must be assert during the data transmission.

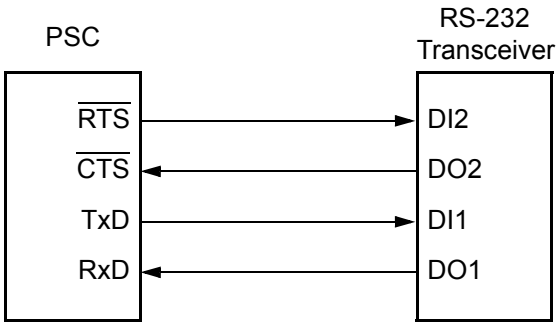


Figure 15-3. Signal configuration for a PSC/RS-232 interface

15.3.1.2 UART Clock Generation

IPB clock serves as the basic timing reference for the clock source generator logic, which consists of a Clock Generator and a programmable 16-bit divider dedicated to the PSC. The IPB clock passes through the 16-bit divider of the concatenated CTUR and CTLR registers. See also Figure 15-4. Using a 66 MHz IPB clock the Baud-rate calculation is as follows:

$$\text{Baud rate} = \frac{\text{IPB Clock}}{32 \times \text{divider } \{\text{CTUR:CTLR}\}}$$

Let Baud rate = 9600, the divider can be calculated as follows:

$$\text{Divider} = \frac{66\ \text{MHz}}{32 \times 9600} = 215(\text{decimal}) = 0\text{x}00\text{D}7$$

Therefore CTUR = 0x00 and CTLR = 0xD7.

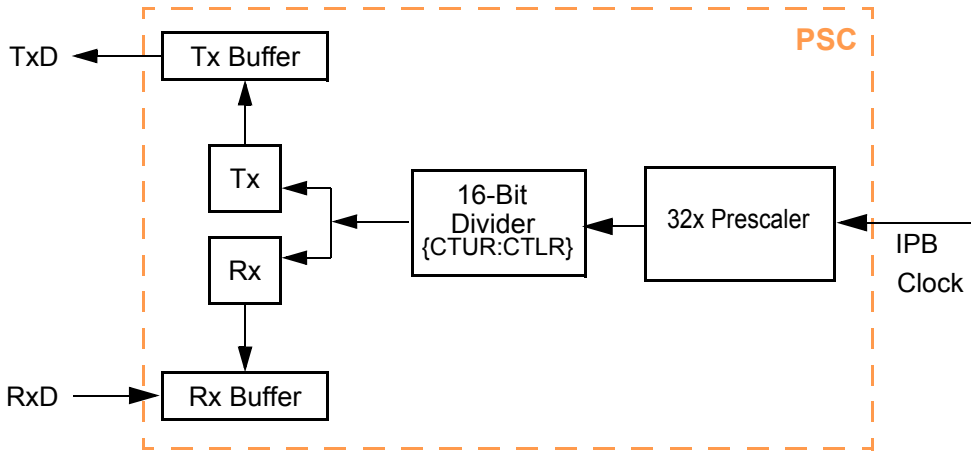


Figure 15-4. Clocking Source Diagram

15.3.1.3 Transmitting in UART Mode

After a hardware reset all PSC are in UART mode. The transmitter is enabled through the PSC command register (CR). When it is ready to accept a character, the PSC sets SR[TxRDY]. The transmitter converts parallel data from the CPU to a serial bit-stream on TxD. It automatically sends a start bit followed by:

- the programmed number of data bits
- an optional parity bit
- the programmed number of stop bits

The lsb is sent first. Data is shifted from the Tx output on the falling edge of the clock source.

After the stop bits are sent, if no new character is in the Tx holding register, the TxD output remains high (mark condition) and the Tx empty bit, SR[TxEMP], is set. Transmission resumes and TxEMP is cleared when the CPU loads a new character into the PSC Tx buffer (TB).

- If the transmitter receives a disable command, it continues until any character in the Tx shift register is completely sent.
- If the transmitter is reset through a software command, operation stops immediately.

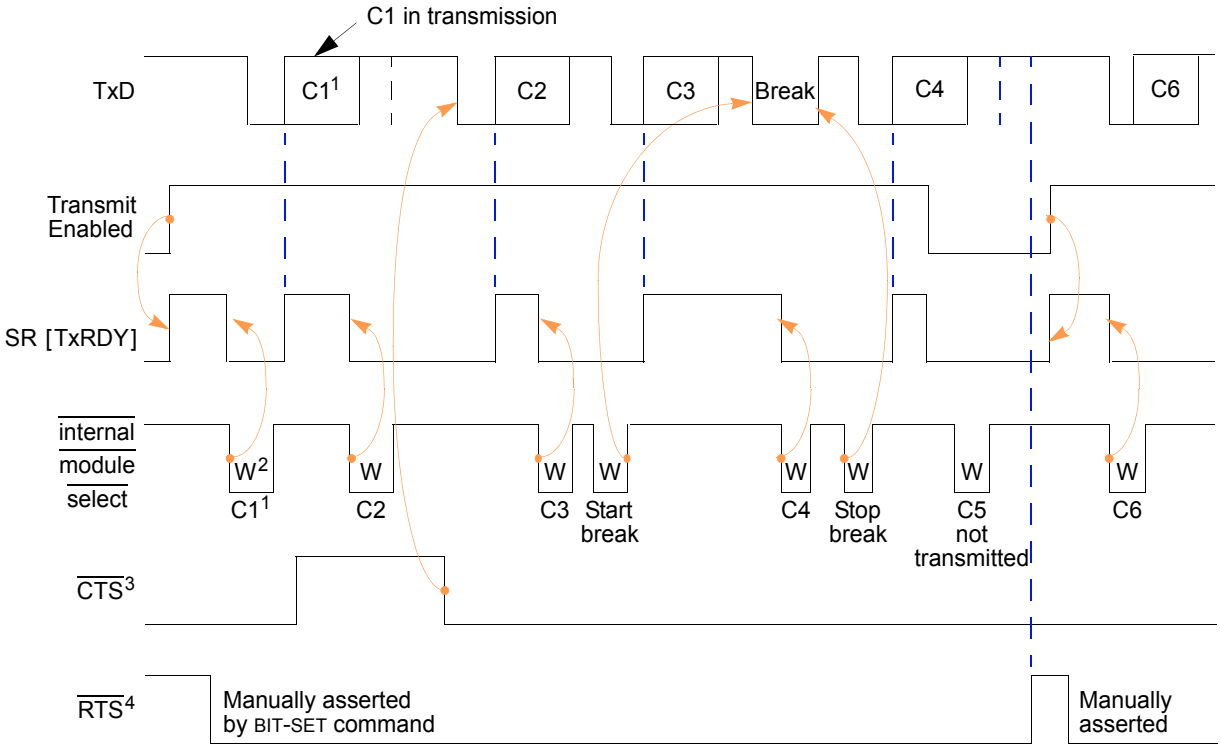
NOTE

The transmitter is re-enabled through the CR to resume operation after a disable or software reset.

- If the clear-to-send operation is enabled, $\overline{\text{CTS}}$ must be asserted for the character to be transmitted.
- If $\overline{\text{CTS}}$ is negated in the middle of a transmission, the character in the shift register is sent and TxD remains in mark state until $\overline{\text{CTS}}$ is reasserted.
- If the transmitter is forced to send a continuous low condition by issuing a send break command, the transmitter ignores the state of $\overline{\text{CTS}}$.
- If the transmitter is programmed to automatically negate $\overline{\text{RTS}}$ when a message transmission completes, $\overline{\text{RTS}}$ must be asserted manually before a message is sent.

In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{RTS}}$ is appropriately programmed, $\overline{\text{RTS}}$ is negated one bit-time after the character in the shift register is completely transmitted. The transmitter must be manually re-enabled by reasserting $\overline{\text{RTS}}$ before the next message is to be sent.

Figure 15-5 shows the transmitter functional timing information.



NOTE:
1. Cn = transmit characters
2. W = write
3. MR2[TxCTS] = 1
4. MR2[TxRTS] = 1

Figure 15-5. Timing Diagram—Transmitter

15.3.1.4 Receiving in UART Mode

After a hardware reset, all PSCs are in UART mode. The receiver is enabled through its CR, as described in Section 15.2.5, *Command Register (0x08)—CR*. Figure 15-6 shows the receiver functional timing.

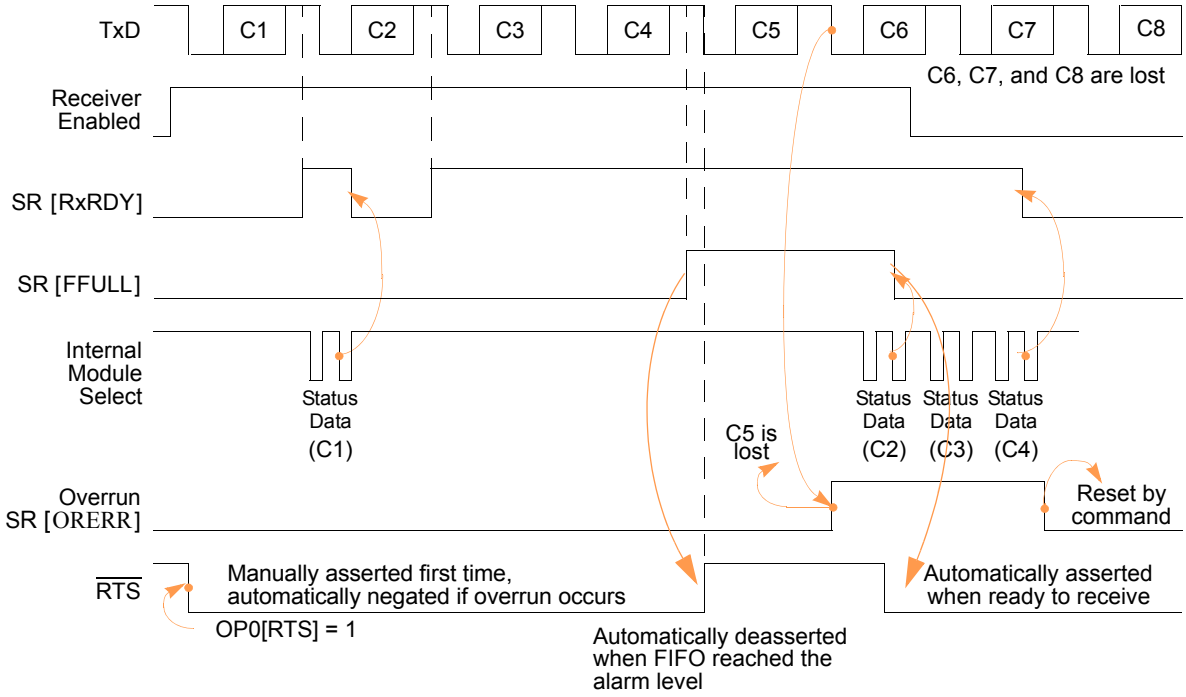


Figure 15-6. Timing Diagram—Receiver

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on RxD, the state of RxD is sampled. It samples each 16× clock for eight clocks, starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit-time clock (synchronous operation).

- If RxD is sampled high, start bit is invalid; a valid start bit search begins again.
- If RxD is still low, a valid start bit is assumed and receiver continues sampling input at 1-bit time intervals at the theoretical center of the bit. This continues until the proper number of data bits and parity, if any, is assembled and 1 stop bit is detected.

RxD input data is sampled on the rising edge of the programmed clock source. The lsb is received first. Data is then transferred to a receiver holding register and SR[RxRDY] is set. If the character is less than 8 bits, the most significant unused bits in the receiver holding register are cleared.

If the MR2[TxRTS] bit was set to one then the user must control the $\overline{\text{RTS}}$ line by writing to the output port register. For all user generated commands to the UART receiver, like enable RX, disable RX or set break, the user must set the associated $\overline{\text{RTS}}$ signal by writing the OP0 or OP1 register. But the UART receiver automatically deassert the $\overline{\text{RTS}}$ signal if the number of received data words reached the FIFO alarm level and deassert the $\overline{\text{RTS}}$ line if the number of words in the FIFO falls under the granularity level.

After the stop bit is detected, the receiver immediately looks for the next start bit.

- If a non-zero character is received without a stop bit (framing error) and RxD remains low for one-half of bit period after stop bit is sampled, the receiver operates as if a new start bit were detected. Parity error (PE), framing error (FE), overrun error (ORERR), and received break (RB) conditions set respective error and break flags in SR at the received character boundary and are valid only if SR[RxRDY] is set.
- If a break condition is detected (RxD is low for the entire character including the stop bit), a character of all 0s is loaded into the Receiver Shift Register and SR[RB,RxRDY] are set. RxD must return to a high condition for at least one-half bit-time before a search for the next start bit begins.

The receiver will detect the beginning of a break in the middle of a character, if the break persists through the next character time.

- If the break begins in the middle of a character, the receiver places the damaged character in the Rx FIFO stack and sets the corresponding SR error bits and SR[RxRDY].
- If the break lasts until the next character time, the receiver places an all-0 character into the Rx FIFO and sets SR[RB,RxRDY].

15.3.1.5 Configuration Sequence for UART Mode

Table 15-72 shows the configuration sequences. This list includes the UART mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations.

Table 15-72. General Configuration Sequence for UART mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
CSR	0xdd00	enable the clock generation
SICR	0x00000000 or 0x08000000	Select the UART mode
MR1	0xXX	Select Error Mode, Parity Mode and the Parity Type
MR2	0xXX	Select Channel Mode, Port Control and Stop-Bit Length
CTUR	0x00	set the Baud rate to 9600 with IPB clock frequency 66 MHz
CTLR	0xD7	
RFALARM	0x0XXX	Choose Rx FIFO “almost full” threshold level.
TFALARM	0x0XXX	Choose Tx FIFO “almost empty” threshold level.
IMR	0xXXXX	select the desired interrupt
Port_Config	0x00000005	Select the Pin-Muxing for UART mode for PSC1, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2 PSC in Codec Mode

After reset all PSCs are in UART mode. PSC1,2,3 or 6 can be put to one of the Codec modes by writing the appropriate value to the SICR register. The other values should be initialized at the same time. During Codec mode the PSC can connect to Codec interfaces with 8, 16, 24 or 32 bit data. For all these modes the PSC can be programmed to behave as a “normal soft modem” interface, SPI or I2S interface. The PSC Codec supports for all these modes the master mode (PSC drive the BitClk and Frame signal) and slave mode (PSC receive the BitClk and Frame signals) functionality. Independently from the mode (master or slave) the PSC can provide an Mclk (master clock) for an external Codec device. This behavior eliminates the need for an external crystal for the external Codec device. Figure 15-7 shows a simplified block Diagram for the PSC Codec mode. The Chapter 2, *Signal Descriptions* shows only the PSC signal names for the “normal” Codec mode. Table 15-73 shows the signal definition for all PSC Code modes.

Table 15-73. Signal Definition for all Codec Modes

Mode	Signal name			
“normal” Codec	TXD	RXD	CLK	Frame
I2S	SDATA_out	SDATA_in	SCK	LRCK
SPI Master	MOSI	MISO	SCK	SS
SPI Slave	MISO	MOSI	SCK	SS

The important register to configure the PSC for Codec mode are:

- SICR register - select the Codec mode
- for master mode:
  - cdm_pscX_bitclk_config - select Mclk frequency, see Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228*
  - cdm_clock_enable_register- enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
  - CCR- select BitClk and Frame Frequency
  - CTUR - select Frame width
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- Port_config - select the right Pin-Muxing, Chapter 2, *Signal Descriptions*

15.3.2.1 Block Diagram and Signal Definition for Codec Mode

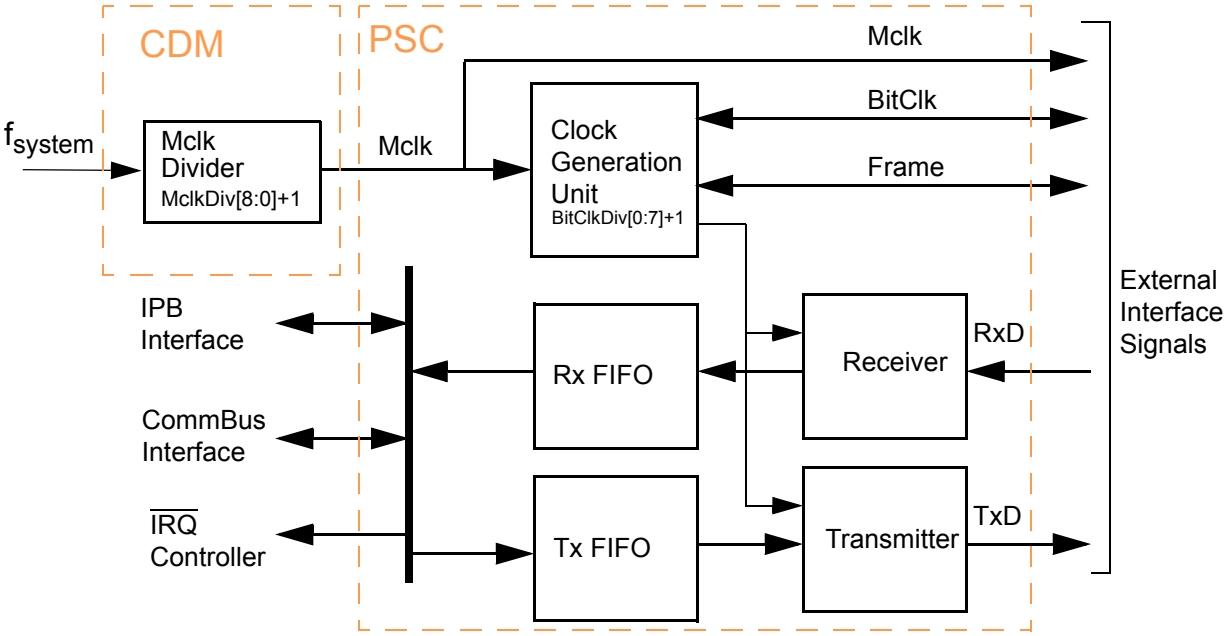


Figure 15-7. PSC Codec Block Diagram

NOTE

Here is important difference between PSC6 and the other PSCs. To work with PSC6 in slave mode (CODEC slave, SPI slave), the ext_48MHz_en bit in the cdm_48mhz_fractional_divider_configuration register must be set to one. If this bit was set to zero then the internal 48 Mhz clock generator drive the clock line. For more informations see Section 5.5.5, *CDM 48MHz Fractional Divider Configuration Register—MBAR + 0x0210*.

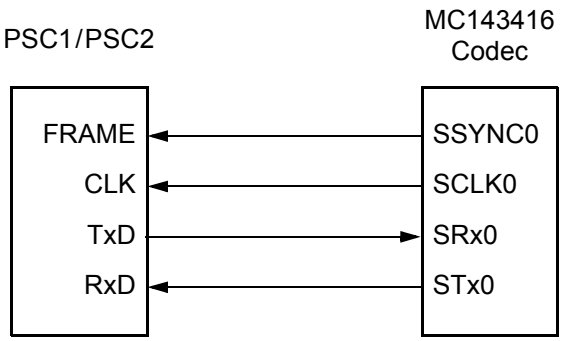


Figure 15-8. PSC Codec Interface in Slave Mode

Table 15-74. PSC Signal Description for Codec Mode

Signal	Description
TxD	Transmitter Serial Data Output—Data is shifted out on TxD on the falling or rising edge of the clock source. Transfers can be specified as either lsb or msb first. TxD is held low when Tx is disabled or idle. - data shifted out on the rising edge of CLK if SICR[ClkPol] = 0 - data shifted out on the falling edge of CLK if SICR[ClkPol] = 1 and - data send msb first if SICR[SHDIR] = 0 - data send lsb first if SICR[SHDIR] = 1
RxD	Receiver Serial Data Input—Data received on RxD is sampled on the falling or rising edge of the clock signal. Transfers can be specified as either lsb or msb first. - data sampled on the rising edge of CLK if SICR[ClkPol] = 1 - data sampled on the falling edge of CLK if SICR[ClkPol] = 0 and - data sampled msb first if SICR[SHDIR] = 0 - data sampled lsb first if SICR[SHDIR] = 1
Frame	Frame Sync—In Codec mode Frame can be driven from an external Codec or can be generate by the internal clock logic. Frame can be programmed as active High or active Low. - the frame sync input from the external Codec if SICR[GenClk] = 0 - the frame sync output to the external Codec if SICR[GenClk] = 1 and - frame sync is active low if SICR[SyncPol] = 0 - frame sync is active high if SICR[SyncPol] = 1
CLK	Bit Clock— In Codec mode CLK is: • - the clock input from the external Codec if SICR[GenClk] = 0 • - the clock output to the external Codec if SICR[GenClk] = 1
Mclk	Clock output for an external Codec

15.3.2.2 Codec Clock and Frame Generation

The serial BitClk and the Frame can either be inputs that come from an external Codec device, or they can be internally generated by the PSC and provided as outputs to the external device, under control of bit SICR[GenClk]. When the bit SICR[GenClk] is set to zero then the BitClk and the Frame are inputs. In this case the Frame width can be anything from one BitClk period up to the total frame length/period minus one BitClk. If the GenClk bit set to one, then the MPC5200 PSC generate the BitClk and the Frame signal. Figure 15-9 shows how the PSC generate the clocks.

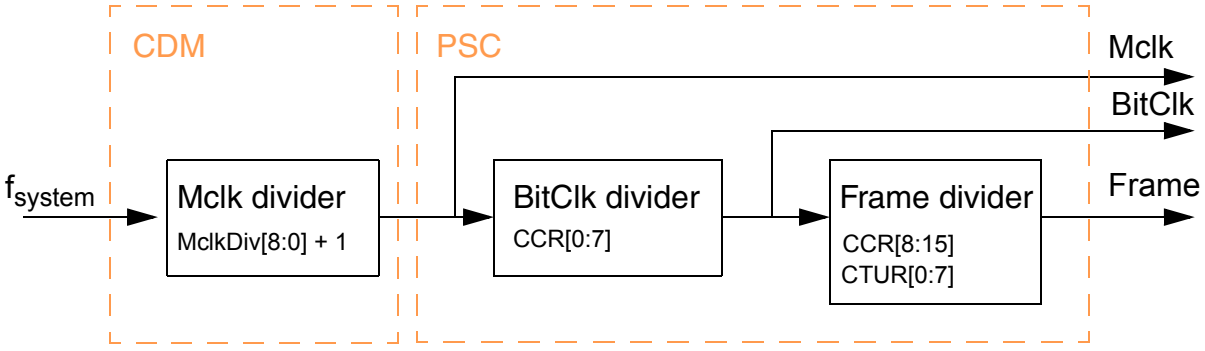


Figure 15-9. Clock Generation Diagram for Codec Mode

The source for the internal clock generation is the MclkDiv clock divider in CDM module. The CDM provides for each Codec PSC (1, 2, 3 and 6) a separate Mclk and MclkDiv clock divider. For more information about the f_{system} clock see also Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228*. The PSC provides the clock to the external Codec divided independently whether the PSC configured as a master (provide BitClk and Frame) or as a slave (receive the clock signals). These dividers generate the Mclks by dividing the f_{system} clock as follows:

$$Mclk = \frac{f_{system}}{MclkDiv\ [8:0] + 1}$$

15.3.2.2.1 BitClk and Frame in “normal” Codec and I2S Mode

Each PSC consists of an CCR register to generate a BitClk and a Frame signal. If the PSC is configured as a master and all necessary register (cdm_pscX_bitclk_config and CCR) are set to the right value the PSC generate both clock signals independent if the transmitter or receiver is enabled or not. As opposed to the SPI behavior. The equations below shows the calculation:

$$BitClk = \frac{Mclk}{CCR[8:15] + 1}$$

$$Frame = \frac{BitClk}{CCR[0:7] + 1}$$

When the frame sync is an output its pulse width can programmed by the register CTUR. This register defines the number of BitClk cycle during the Frame signal is active. The default reset value for this register is 0x00 therefore the default frame sync width is one BitClk. See the calculation below:

$$Frame\ sync\ width = CTUR[0:7] + 1$$

15.3.2.2.2 BitClk and Frame in “Cell Phone” Mode

A special Codec mode is the “Cell Phone” mode. The goal for this mode is that PSC2, 3 or 6 generate a BitClk which is synchronous to in the BitClk input on PSC1. The PSC use the configure and clock generation registers is the same as described in the section before. Only the internal clock distribution is different. The major deviation is, that the “cell phone” slave PSCs use the clock from PSC1 for BitClk and Frame generation instead of Mclk from the CDM module. This facts are described in Figure 15-10..

The SICR[CellSlave] and SICR[Cell2xClk] bits are used when the PSC is talking to a “slave” Codec that needs its frame rate to be synchronized with a cell phone “master” that is talking to PSC1. PSC2,3 or 6 are available for use as “slaves”, and PSC1 is the only PSC that can be used to master them. The SICR[CellSlave] and SICR[Cell2xClk] bits are only used in the slave PSCs, they must be set to’0’ in the master PSC. In this “cell phone” mode the master PSC1 must have its SICR[GenClk] bit cleared so that its bit clock and Frame will come from the cell phone, as opposed to being generated internally by PSC1’s CCR dividers. If for example PSC2 is to be used as a slave, then master PSC1’s bit clock will be used as slave PSC2’s Mclk in place of the Mclk that would normally be generated in the CDM for PSC2. This slave Mclk from PSC1 will typically be used to drive the slave Codec for frame synchronization between PSC1 and PSC2. If SICR[GenClk] = 1 in slave PSC2, then the slave Mclk will be divided down by PSC2’s BitClkDiv and CCR divider registers to generate the bit clock and frame sync that PSC2 will provide to the slave Codec device. If instead SICR[GenClk] = 0 in slave PSC2, then the slave Codec will provide the bit clock and frame sync signals to slave PSC2.

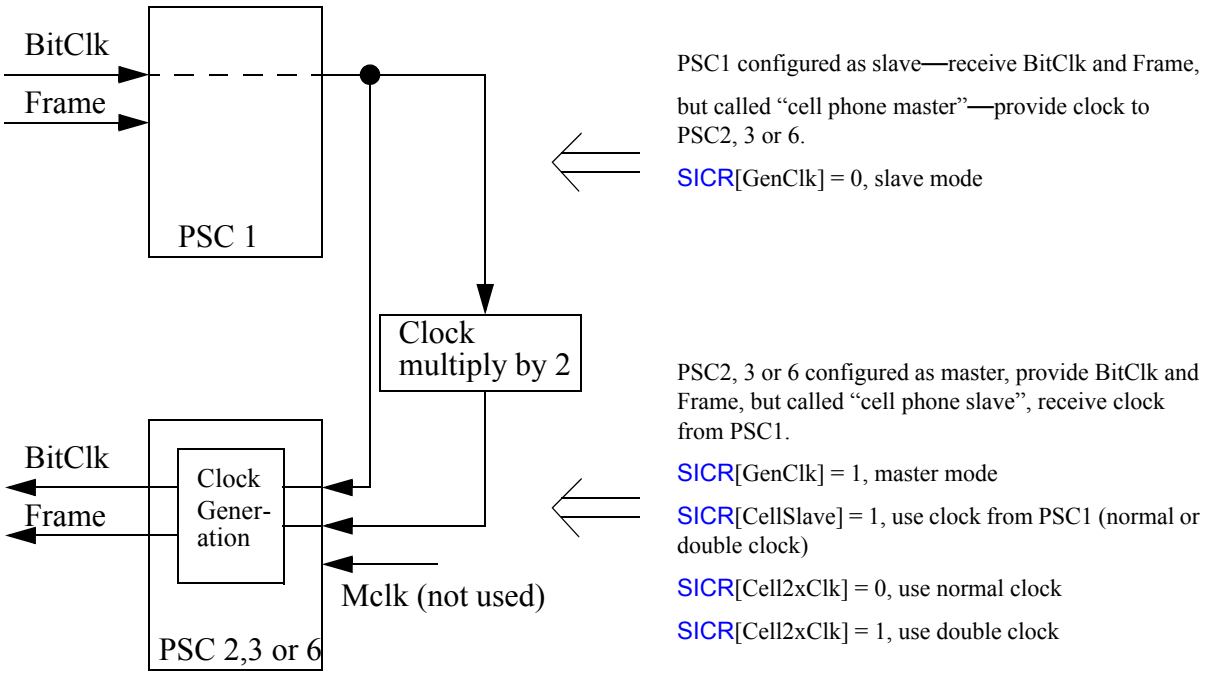


Figure 15-10. Clock distribution network in cell phone mode

15.3.2.2.3 BitClk and Frame in SPI Mode

An other Codec mode is the SPI mode. The PSC support a full duplex SPI interface. This mode is chosen by setting SICR[SPI] = 1, which must be true in order for the MSTR, CPOL, CPHA and UseEOF bits in the SICR register to take effect. In SPI Master mode the SICR[SIM] bits must also be set to select Codec8, Codec16, Codec24 or Codec32. In SPI Slave mode the SICR[SIM] bits must be set to select Codec8.

To configure the PSC to act like an SPI master set SICR[MSTR] = 1, or set SICR[MSTR] = 0 to configure the PSC as an SPI slave. When SICR[MSTR] = 1 (master mode) then SICR[GenClk] must also be set to’1’ since the PSC is driving the SPI clock line, “SCK.” When SICR[MSTR] = 0 (slave mode) then SICR[GenClk] must be set to “0” since the external SPI is driving the SCK clock line.

The CPOL and CPHA bits in the SICR register operate exactly the same way as they do in an SPI, and their values must be the same as the CPOL and CPHA bits in the SPI device that is communicating with the PSC.

The SICR[UseEOF] bit has an effect only when SICR[MSTR] = 1 for master mode. If the UseEOF bit is cleared then the number of bytes that will be transferred before Slave Select ($\overline{SS}$) goes high/inactive is either 1, 2 or 4, depending on whether Codec8, Codec16, Codec24 or Codec32 mode is selected by the SICR[SIM] field, respectively. When SICR[UseEOF] = 1, then SICR[SIM] = 001 (Codec8) should also be selected. In this mode the number of bytes transferred prior to $\overline{SS}$ going high is controlled by the BestComm task that fills the Tx FIFO. By using the "tfdOnExit" keyword in the for-loop that fills the Tx FIFO, the last byte written into the Tx FIFO by the for-loop is marked with an EOF flag. As the PSC reads bytes out of the Tx FIFO it will hold $\overline{SS}$ low/active until it transmits a byte whose EOF flag is set. In this mode there is virtually no limit on how many bytes can be sent in one SPI transfer.

To mark data word with the EOF flag during a IPB transfer set the Bit NXTEOF from the IRCR2 register before writing the last data word to the TX FIFO. This bit will be cleared after the TX FIFO access.

The SICR[SHDIR] bit controls the shift direction in SPI mode, just as it does in the non-SPI Codec modes. The DTS1, MultiWd, ClkPol, SyncPol, CellSlave and Cell2xClk bits in the SICR register have no effect in SPI mode. In SPI master mode the SCK frequency is generated by dividing down the Mclk frequency as follows

$$\text{Baud Rate} = \text{SCK} = \frac{\text{Mclk}}{\text{BitClkDiv[0:7]} + 1}$$

When in SPI master mode, the delay between $\overline{SS}$ going low/active and the first SCK transition of the serial transfer (DSCKL delay) is created by dividing down the Mclk frequency as follows:

$$\text{DSCKL delay} = \frac{\text{FrameSyncDiv[0:7]} + 1}{\text{Mclk}}$$

When in SPI master mode, the length of time that $\overline{SS}$ stays high/inactive between consecutive transfers (DTL delay) is created by dividing down the bus clock (IPB clock) frequency as follows:

$$\text{DTL delay} = \frac{\text{CT[15:0]}}{\text{IPB clock}}$$

where CT[15:0] is the concatenation of the CTUR and CTLR registers.

There is a separate cdm_pscX_bitclk_config register in the CDM for each of PSC1,2,3 and 6. This four PSCs the Codec mode. These cdm_pscX_bitclk_config registers are further described in the CDM chapter. For more details see also Figure 15-11. In SPI master mode the PSC controls the serial data transfers. In this mode if either the Tx FIFO becomes empty (underrun) or the Rx FIFO becomes full (overflow) in the middle of a multi-byte transfer, rather than set the Tx underrun or Rx overflow status bits the PSC will keep the Slave Select signal low/active and stop the SCK serial clock. When the Tx FIFO does not become empty and the Rx FIFO becomes not full the transfer proceeds.

In SPI slave mode the Mclk must be running/enabled even though it is not being used to generate the serial clock SCK, which is provided by the external master SPI device. The frequency of Mclk is not critical, as long as it is faster than the SCK frequency.

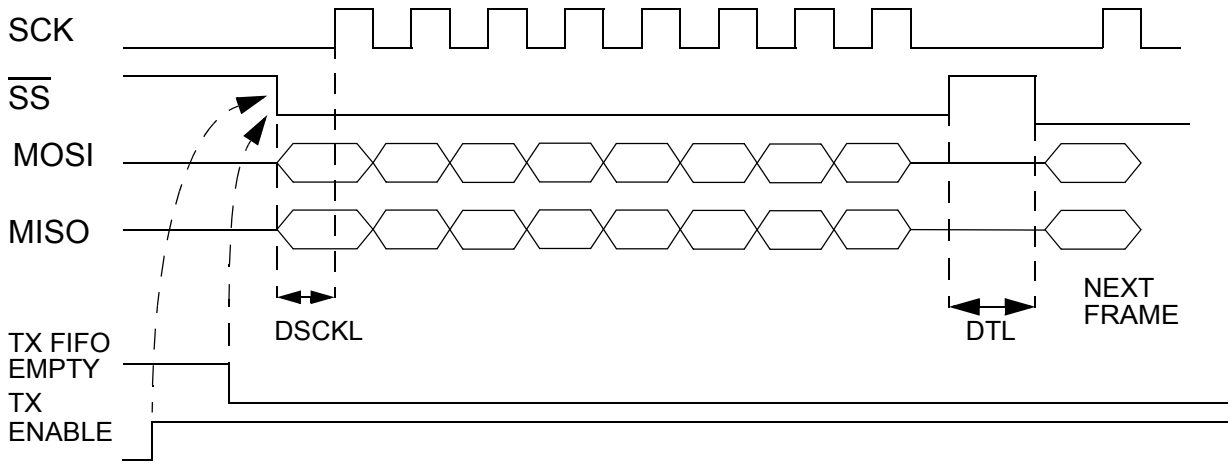


Figure 15-11. SPI Parameter

NOTE

The PSC starts to generate the SCK if the transmitter is enabled and the Tx FIFO is not empty!

15.3.2.3 Transmitting and Receiving in Codec Mode

To start the transmission the Tx and the Rx must be enable by writing the according value to the CR register.

NOTE

The transmitter works if the receiver is disabled, but to receive data the transmitter **and** receiver must be enabled.

If the receiver is enabled, the PSC samples data from the receive line after detecting the start of frame condition. If no data on the Rx line the receiver write zeros to the RxFIFO. If the TxFIFO is empty during the transmit state the Tx line will be zero.

When bit SICR[ClkPol] = 0 data is shifted out on the rising edge of bit clock; when SICR[ClkPol] = 1 data is shifted out on the falling edge of bit clock.When SICR[GenClk] = 1 the frequencies of the internally generated BitClk and Frame are divided down from Mclk.

The PSC starts to send a sample at either:

- the leading edge of frame sync, or
- 1 bit-clock cycle after the leading edge of frame sync, according to the SICR[DTS1] value.

The leading edge is defined as a rising edge if bit SICR[SyncPol] = 1, or a falling edge if SICR[SyncPol] = 0.

SICR[SHDIR] controls whether bits are shifted out msb or lsb first. After the 8-, 16-, 24- or 32-bit sample is sent, 0's are sent until the next frame sync.

In Codec 24 mode each 24-bit data sample uses an entire 32-bit longword in the Tx FIFO. The least significant (right-hand) byte is not used. Data should be written to the Tx FIFO four bytes at a time.

Figure 15-12 and Figure 15-13 shows a Codec interface timing diagram example.

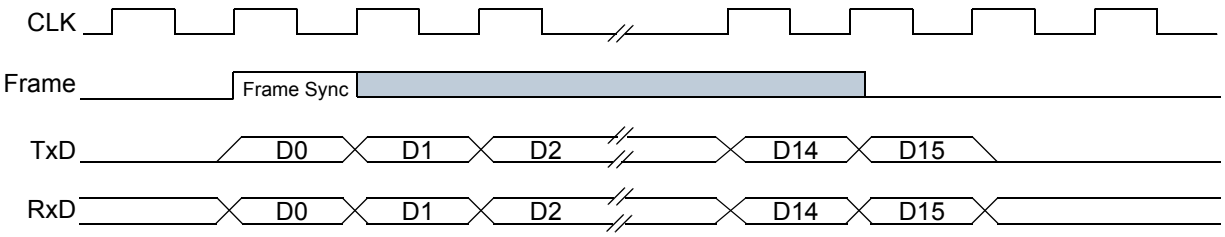


Figure 15-12. Timing Diagram—16-Bit Codec Interface (lsb First, DTS1 = 0)

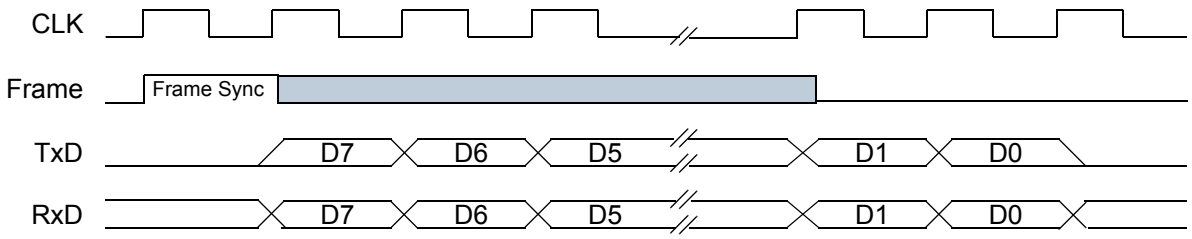


Figure 15-13. Timing Diagram—8-Bit Codec Interface (msb First)

The Frame pulse width makes no difference. SICR[SHDIR] controls whether the sample is shifted in msb or lsb first. After the complete sample is received, the receiver shift register shuts off until the next frame sync occurs.

15.3.2.4 Configuration Sequence Examples for Codec Modes

This section shows some configuration examples for different modes

15.3.2.4.1 PSC1 in 16-bit "soft Modem" Slave Mode

- use PSC1
- PSC in Slave mode
- 16 bit "soft Modem" mode
- Data are sampled on the falling edge of BitClk
- Frame is low true
- msb first, transfer starts with leading edge of Frame
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-75. 16-Bit "soft Modem" Slave Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x02100000	Select the 16 bit Codec mode, msb first, DTS1 = 0, slave mode
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt

Table 15-75. 16-Bit “soft Modem“ Slave Mode

Register	Value	Setting
Port_Config	0x00000006	Select the Pin-Muxing for PSC1 Codec mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2.4.2 PSC2 in 32-bit “soft Modem” Master Mode

- use PSC2
- PSC in Master mode
- 32bit “soft Modem” mode
- Data are sampled on the rising edge of BitClk
- Frame is low true
- lsb first, transfer starts one cycle after the leading edge of Frame
- set Mclk frequency to 33MHz
- set Bitclk frequency to 250 KHz
- Frame sync every 35 BitClk
- set Frame width to 3 BitClk
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-76. 32-Bit “soft Modem“ Master Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x3FA00000	Select the 32bit Codec mode, lsb first, DTS1 = 1, master mode
cdm_psc2_bitclk_config	0x800F	divide the f_{system} clock frequency from 528 to 33MHz Mclk, see Section 5.5.12, *PSC2 Mclock Config Register—MBAR + 0x022C*
cdm_clock_enable_register	0x00000040	enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
CCR	0x2283	select the BitClk and Frame frequency
CTUR	0x02	select the Frame width
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000070	Select the Pin-Muxing for PSC2 Codec mode, Mclk output enabled, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2.4.3 PSC 1 in Cell Phone Master Mode, PSC2 is Cell Phone Slave

- use PSC1 as cell phone master
- use PSC2 as cell phone slave
- both PSC work as 24bit data
- Data are sampled on the falling edge of BitClk
- Frame is high true
- msb first, transfer starts on the leading edge of Frame
- PSC2 divide the clock from PSC1 by 2

- Frame sync every 24BitClk, both PSCs
- set Frame width to 1BitClk
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-77. 24-Bit Cell Phone Master Mode for PSC1

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x07100000	Select the 24bit Codec mode, msb first, DTS1 = 0, slave mode
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000066	Select the Pin-Muxing for PSC1,PSC2 Codec mode, seeChapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

Table 15-78. 24-Bit Cell Phone Slave Mode for PSC2

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x07980000	Select the 24bit Codec mode, msb first, DTS1 = 0, master mode, cell phone master
CCR	0x0117	select the BitClk and Frame frequency
CTUR	0x00	select the Frame width, default value
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000066	Select the Pin-Muxing for PSC12, PSC2 Codec mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2.4.4 PSC2 in SPI Slave Mode

- use PSC2 as SPI slave
- 8bit data
- clock is active high, CPOL = 0;
- the first SCK edge is issued at the beginning of the data transfer; CPHA = 1
- msb first
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-79. 8-bit SPI Slave mode for PSC2

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x01009000	Select the 8bit Codec SPI slave mode, msb first, CPOL = 0; CPHA = 1
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000060	Select the Pin-Muxing for PSC2 Codec mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2.4.5 PSC3 in SPI Master Mode

- use PSC3 as SPI master
- 32bit data
- clock is active low, CPOL = 1;
- the first SCK edge is issued one half cycle into the data transfer; CPHA = 0
- msb first
- Baud Rate 1MBit
- DSCLK delay = 0.5 µs
- DTL delay = 2.0 µs
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-80. 32-bit SPI Master mode for PSC3

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x0F00E000	Select the 32bit Codec SPI master mode, msb first, CPOL = 1,CPHA = 0
cdm_psc345_bitclk_config	0x8020	divide the f_{system} clock frequency from 528 to 16 MHz Mclk, see Section 5.5.13, *PSC3 Mclock Config Register—MBAR + 0x0230*
cdm_clock_enable_register	0x00000080	enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
CCR	0x070F	set the SCK and DSCKL delay
CTUR	0x00	set the DTL delay 2us
CTLR	0x84	
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000600	Select the Pin-Muxing for PSC3 Codec mode, see: Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.2.4.6 PSC1 in I2S Master Mode

Figure 15-14 shows the select mode.

- use PSC1 as I2S master
- 32bit data
- SCLK frequency 1 MHz
- data shifted out on the falling edge of SCLK
- data transfer starts one CLK after the Frame is active
- Frame starts with LRCK low
- msb first
- set the TFALARM level to 0x010, alarm occurs if 16 byte are in the TxFIFO
- set the RFALARM level to 0x00C, alarm occurs if 12 byte space in the RxFIFO
- enable TxRDY interrupt

Table 15-81. 32-bit I2S Master Mode for PSC1

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x2FE00000	Select the 32bit Codec I2S master mode, msb first, DTS1 =1, send more than one data word per frame.
cdm_psc1_bitclk_config	0x8020	divide the f_{system} clock frequency from 528 to 16 MHz Mclk, see Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228*
cdm_clock_enable_register	0x00000020	enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
CCR	0x3F0F	set the SCK and SCKL frequency
CTUR	0x1F	set the frame width to 32 SCLKs
RFALARM	0x000C	set the RFALARM level to 0x00C
TFALARM	0x0010	set the TFALARM level to 0x010
IMR	0x0100	enable TxRDY interrupt
Port_Config	0x00000006	Select the Pin-Muxing for PSC1 Codec mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

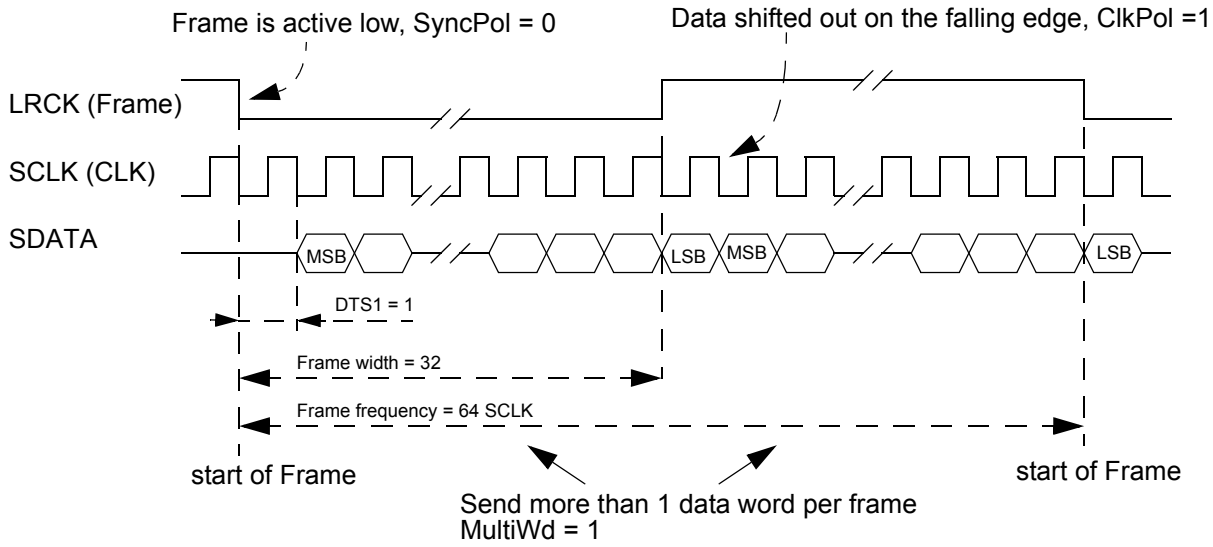


Figure 15-14. I2S Data Transmission

15.3.3 PSC in AC97 Mode

After reset all PSCs are in UART mode. AC97 mode is chosen by setting the SICR[SIM] =0x3. The other SICR field should be initialized at the same time. Only PSC1 and PSC2 support the AC97 mode. The important register to configure the PSC for AC97 mode are:

- SICR register - select the Codec mode
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- OP0, OP1 register - generate the reset pulse for the external device
- Port_config - select the right Pin-Muxing, see Chapter 2, *Signal Descriptions*

The follow sequence generate a reset pulse for the external AC97 device:

1. $\overline{\text{Res}}$ line is high, after power up
2. write 0x02 to the OP1 register - $\overline{\text{Res}}$ line goes low
3. write 0x02 to the OP0 register - $\overline{\text{Res}}$ line goes high

15.3.3.1 Block Diagram and Signal Definition for AC97 Mode

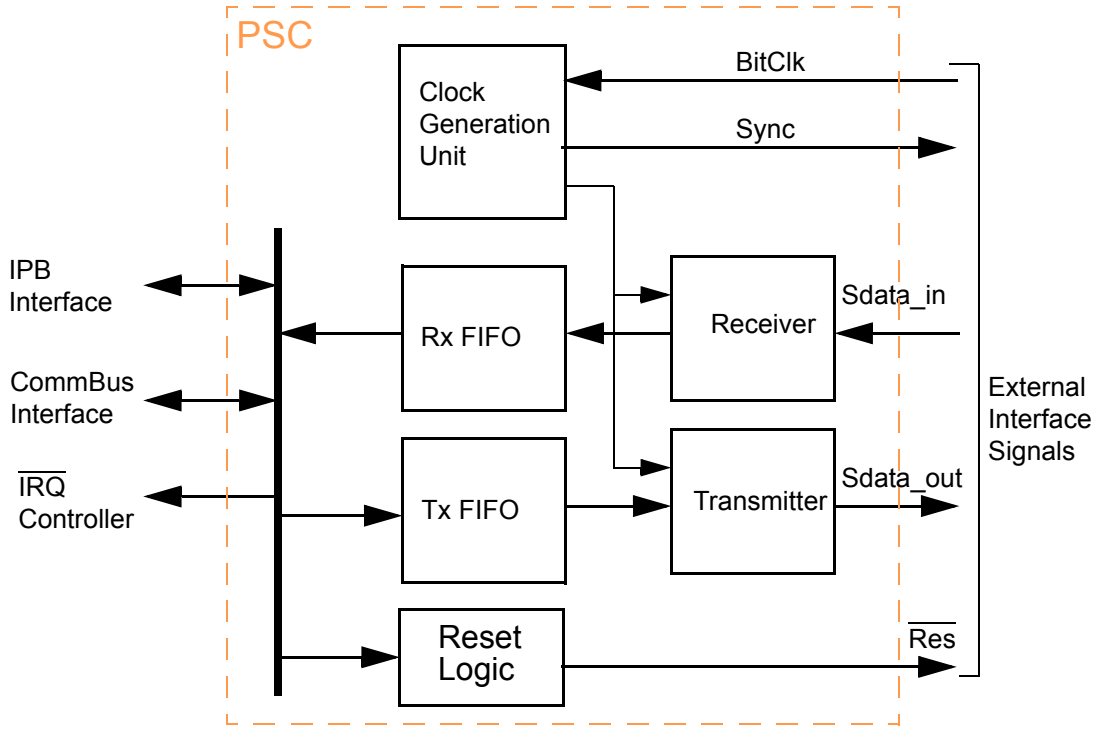


Figure 15-15. PSC AC97 Block Diagram

Figure 15-15 shows the simplified PSC Block Diagram for AC97 mode. The BitClk is an input from the external Codec. The PSC divide BitClk by 256 to generate a Frame pulse (Sync) that is high for 16 BitClk cycles. The table below shows the Pin definition for the AC97 mode and the Figure 15-16 shows an AC97 interface. An MPC5200 general-purpose I/O (GPIO) is used as a reset to the external AC97 device.

Table 15-82. PSC Signal Description for AC97 Mode

Signal	Description
Sdata_out	Transmitter Serial Data Output—Data is shifted out on TxD on the rising edge of the clock signal. Transfers must be specified as msb first.
Sdata_in	Receiver Serial Data Input—Data received on RxD is sampled on the falling edge of the clock signal. Transfers must be specified as msb first.
Sync	In AC97 mode Sync is the frame sync, or start-of-frame (SOF), output to the external AC97 Controller. In this mode the AC97 BitClk, which is input on CLK, is divided by 256 to generate the Sync.
BitClk	BitClk— In AC97 mode CLK must be driven by the serial bit-clock from the external AC97 Controller.
Res	Reset signal to the external AC97 device

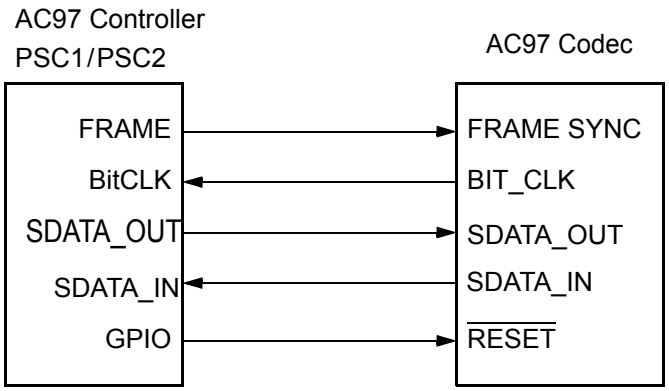


Figure 15-16. PSC - AC97 Interface

15.3.3.2 Transmitting and Receiving in AC97 Mode

When an AC97 Controller is specified (SICR[SIM]=0011), PSC1 (or PSC2) begins receiving time slot 1 data, 1 bit-clock cycle after the rising edge of frame sync, regardless of SICR[DTS1] value. However, SICR[SHDIR] must be 0, because the shift order must be msb first. The PSC divides the bit-clock by 256 to generate a frame sync pulse that is high for 16-bit clock cycles. The transmitter sends 0s until the receiver detects the Codec-ready condition (1 in the first bit of a new frame).

- Until the receiver detects the Codec ready condition (1 in the first bit of a new frame), no data is put into the Rx FIFO for that frame.
- When a Codec ready condition is detected, the receiver begins loading the Rx FIFO with the received time slot samples and continues to do so until a 0 is received in the first bit of a new frame.

Figure 15-17 shows a AC97 interface timing diagram example.

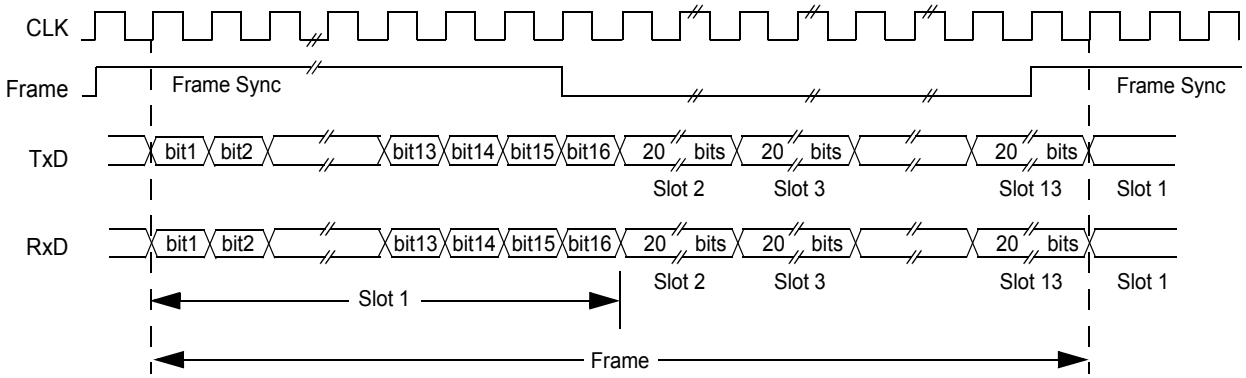


Figure 15-17. Timing Diagram—AC97 Interface

For more AC97 Controller interface information, refer to the *Audio Codec'97 Component Specification.*

Because Rx data is sampled on the falling edge of the BitClk, for transmit purposes, the frame has already started when the receiver detects a Codec-ready condition. For this reason, transmission starts at the next frame sync after the Codec-ready condition is detected. The PSC stops transmission at the end of the frame in which the first bit of the received frame is detected low (Codec not ready). During transmission, the PSC fills each of the 13 AC97 frame time slots with samples from the Tx FIFO.

15.3.3.3 AC97 Low-Power Mode

A General-Purpose I/O (GPIO) must be used as an AC97 reset output pin. PSC1 (or PSC2) monitors the first three time slots of each Tx frame to detect the power-down condition for the AC97 digital interface. The power-down condition is detected as follows:

1. The first 3 bits of slot 1 must be set, indicating Tx frame and slots 1 and 2 are valid.
2. Slot 2 holds the power-down register (0x26) address in the external AC97 device.
3. Slot 3 has "1" in the fourth bit (bit 12/PR4 in power-down register 1), as defined in the AC97 specification.

Low-power mode can be left through either a warm or cold reset. The CPU does a warm reset by setting SICR[AWR] for at least 1 µs. This asserts the FRAME frame sync output in AC97 mode. The CPU does a cold reset in two steps:

1. Writes 0 to whichever GPIO is being used as the active low AC97 reset pin for the minimum time specified in the AC97 specification.
2. Writes 0 to PSC1 or PSC2 SICR[ACRB]. CPU should set this bit after writing 1 to the GPIO used for the AC97 reset pin.

NOTE

Step 2 (above) is required so that the PSC knows when an AC97 cold reset is occurring.

15.3.3.4 Configuration Sequence for AC97 Mode

The Table 15-83 shows the configuration sequences. This list includes the AC97 mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations.

Table 15-83. General Configuration Sequence for AC97 Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x03000000	Select the AC97 mode
RFALARM	0x0XXX	Choose Rx FIFO “almost full” threshold level.
TFALARM	0x0XXX	Choose Tx FIFO “almost empty” threshold level.
IMR	0xXXXX	select the desired interrupt
Port_Config	0x00000020	Select the Pin-Muxing for AC97 mode PSC 2, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.4 PSC in SIR Mode

The SIR mode is one of the supported IrDA modes. This section will give some more informations about this mode. The imported register to configure the PSC6 (only this PSC support the IrDA modes) for SIR mode are:

- SICR register - select the SIR mode
- CTUR, CTLR register - select the Baud rate
- MR1 register - select the Receiver interrupt mode
- MR2 register - Channel Mode
- IRCR1 register - select the pulse width
- IRSDR register - select the counter for pulse width
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- Port_config - select the right Pin-Muxing, see Chapter 2, *Signal Descriptions*

15.3.4.1 Block Diagram and Signal Definition for SIR Mode

The Table 15-84 shows the interface signal definition. This definition is equal for all IrDA modes. The Figure 15-18. shows the PSC in SIR mode. The simplified block diagram describe the clock distribution.

Table 15-84. Signal Description for IrDa Mode

Signal	Description
IRDA_TX	Transmitter Serial Data Output—Transfers must be specified as msb first.
IRDA_RX	Receiver Serial Data Input—Transfers must be specified as msb first.

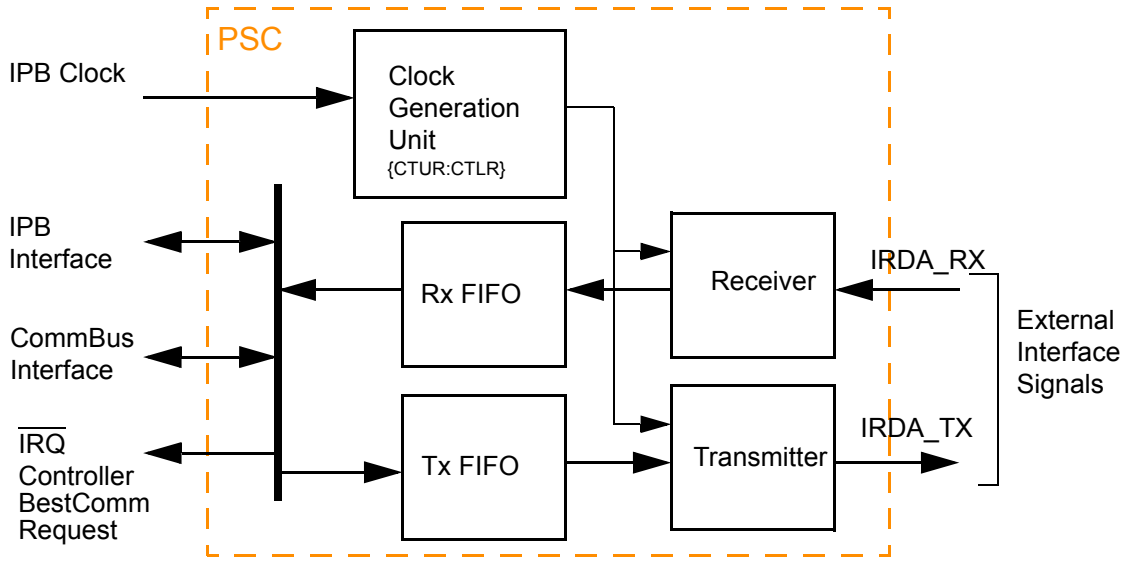


Figure 15-18. PSC SIR Block Diagram

15.3.4.2 Transmitting and Receiving in SIR Mode

This data format is similar to the UART. Each data consists of a start bit, 8 bit data and a stop bit. Each bit data is encoded so that a 0 is encoded as 3/16 of the bit time pulse (or 1.6 µs pulse) and a 1 is encoded as no pulse. Similarly, the received serial pulse is decoded as a 0 and an absence of a pulse is decoded as a 1. Like the UART mode, the SIR mode sends the lsb first. Here is an example of data stream of UART and SIR.

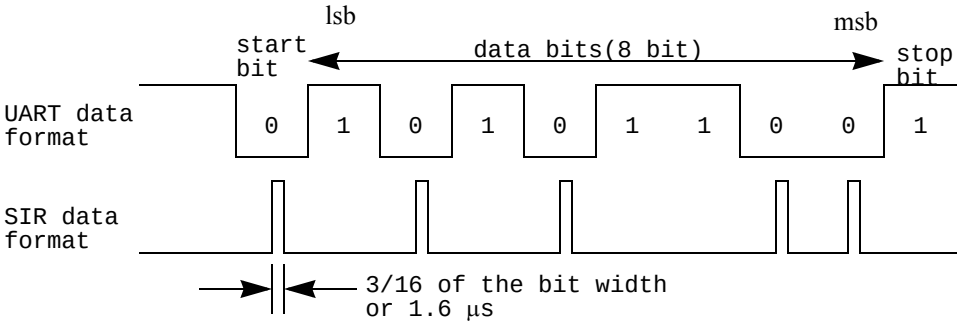


Figure 15-19. Data Format in SIR Mode

NOTE

Please choose first the desired mode (SIR mode) than configure the port (write to port_config register). This sequence will avoid pulses on the TX line during port configuration. This is very important for all IrDA (SIR, MIR, and FIR) modes.

For more informations regarding the pulse width and Baud rate calculations see Section 15.2.22, *Infrared SIR Divide Register (0x4C)—IRSDR* and Section 15.2.12, *Counter Timer Upper Register (0x18)—CTUR*.

15.3.4.3 Configuration Sequence Example for SIR Mode

The Table 15-85 shows the configuration sequences. This list includes the SIR mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations.

Table 15-85. Configuration Sequence Example for SIR Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x04000000	select the SIR mode
IRCR1	0x01	set SIR pulse width to 1.6 ms
IRSDR	0x6A	set counter for SIR pulse width for IPB clock 66 MHz
CTUR	0x00	set the Baud rate to 9600 with IPB clock frequency 66 MHz
CTLR	0xD7	
RFALARM	0x0XXX	Choose Rx FIFO “almost full” threshold level.
TFALARM	0x0XXX	Choose Tx FIFO “almost empty” threshold level.
IMR	0xXXXX	select the desired interrupt
Port_Config	0x00500000	Select the Pin-Muxing for IrDA mode, see Section 15.3.4, *PSC in SIR Mode*
CR	0x05	Enable Tx and Rx

15.3.5 PSC in MIR Mode

The MIR mode is the second IrDA mode, which the PSC supports. This section will give some more informations about this mode. The important register to configure the PSC6 (only this PSC supports the IrDA modes) for MIR mode are:

- SICR register - select the MIR mode
- MR2 register - Channel Mode
- If clock generate from the internal source:
  - cdm_irda_bitclk_config - select Mclk frequency, see Section 5.5.14, *PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234*
  - cdm_clock_enable_register - enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
  - CCR- select BitClk and Frame Frequency
- IRCR1 register - select full duplex and SIP mode
- IRMDR register - select the clock divider
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- Port_config - select the right Pin-Muxing, see Chapter 2, *Signal Descriptions*

15.3.5.1 Block Diagram and Signal Definition for MIR Mode

The signal definitions for MIR mode are the same as in SIR mode. Please see Table 15-84.

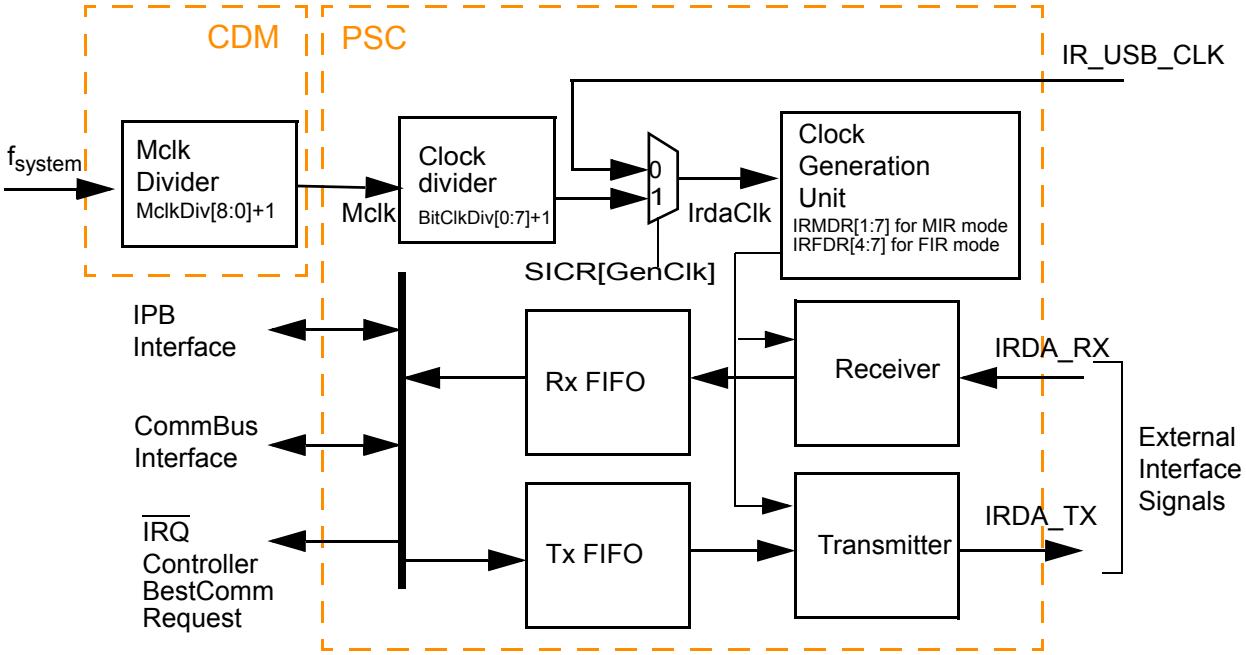


Figure 15-20. PSC MIR and FIR Block Diagram

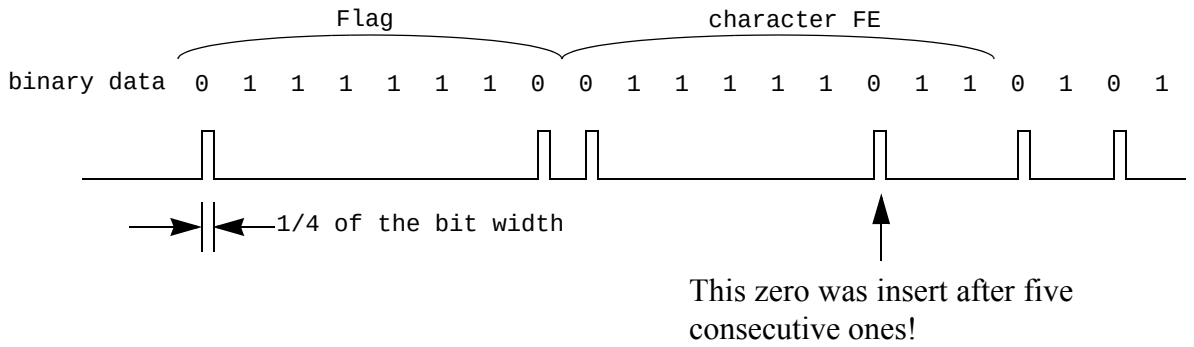
For MIR and FIR mode the clock for the transmitter and receiver is generated by dividing down from the internal Mclk or from an external clock. If the bit GenClk in the SICR was set to "1" then PSC generate the clock from the internal source. The clock from the Mclk generator goes through a predivider to the clock generation. See Section 15.2.23, *Infrared MIR Divide Register (0x50)—IRMDR* or Section 15.2.24, *Infrared FIR Divide Register (0x54)—IRFDR* for the possible frequencies for this mode. For more informations about the Mclk divider see Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228*. If the bit GenClk cleared then the PSC use the clock from an external source for the clock generation.

NOTE

If the CCR register was not changed (reset value 0x01) then the counter divide the clock (Mclk) by 2. This is the minimum value. 0x00 deactivate the clock generation.

15.3.5.2 Transmitting and Receiving in MIR Mode

Each bit data is encoded so that a 0 is encoded as 1/4 of the bit time pulse and a 1 is encoded as no pulse. Similarly, the received serial pulse is decoded as a 0 and an absence of a pulse is decoded as a 1. The PSC MIR mode use the HDLC bit stuffing after five consecutive ones to decode/encode the data, except the STA and STO flag. For example see the Figure below:



The packet format is:

STA	STA	DATA	FCS	STO
01111110	01111110	DATA	16 bit CRC	01111110

The STA represents the start of the frame and the STO represents the end of the frame. Both of STA and STO are defined as 01111110 in binary format. Like the UART mode, the MIR mode sends the lsb first.The FCS is a 16 bit CRC defined as

$$CRC(x) = x^{16} + x^{12} + x^{5} + 1$$

NOTE

The MIR module doesn't support the CRC generation. If the transfer require a CRC Field use the CRC generation from the BestComm module. See also Chapter 13, *BestComm*.

15.3.5.3 Serial Interaction Pulse (SIP)

The MIR and FIR system must emit SIP (Serial Interaction Pulse) at least once per 500ms while the connection lasts, in order to inform slower systems (SIR) not to interfere the link. If the SIPEN bit in IRCR1 is high, the transmitter automatically append one SIP after every frame. SIP can be also sent by writing 1 to SIPREQ bit in IRCR2. If SIPREQ is high and the transmitter is in idle state, one SIP is sent and SIPREQ bit is automatically cleared. The SIP is defined as:

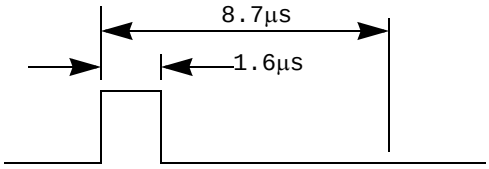


Figure 15-21. Serial Interaction Pulse (SIP)

15.3.5.4 Configuration Sequence Example for MIR Mode

This list includes the MIR mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations. The Table 15-86 shows the configuration sequences for follow example:

- PSC6 in IrDA MIR mode
- MIR mode: 1.152 Mbps, SIP pulse after every transfer
- Mclk frequency: 27.78 MHz
- IrdaClk: 9.26 MHz

NOTE

Please choose first the desired mode (MIR mode) than configure the port (write to port_config register). This sequence will avoid pulses on the TX line during port configuration. This is very important for all IrDA (SIR, MIR, and FIR) modes

Table 15-86. Configuration Sequence Example for MIR Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x05800000	Select the MIR mode, use internal clock
cdm_irda_bitclk_config	0x80012	set Mclk to 27.78 Mhz, see Section 5.5.11, *PSC1 Mclock Config Register—MBAR + 0x0228*
cdm_clock_enable_register	0x00000010	enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
CCR	0x0002	set IrdaClk to 9.26 MHz
IRCR1	0x02	enable SIP
IRMDR	0x07	set Baud rate to 1.152 Mbps
RFALARM	0x0XXX	Choose Rx FIFO "almost full" threshold level.
TFALARM	0x0XXX	Choose Tx FIFO "almost empty" threshold level.

Table 15-86. Configuration Sequence Example for MIR Mode

Register	Value	Setting
IMR	0xXXXX	select the desired interrupt
Port_Config	0x00F00000	Select the Pin-Muxing for IrDA mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.6 PSC in FIR Mode

The FIR mode is also a supported IrDA mode. This section will give some more informations about this mode. The important registers to configure the PSC6 (only this PSC support the IrDA modes) for FIR mode are:

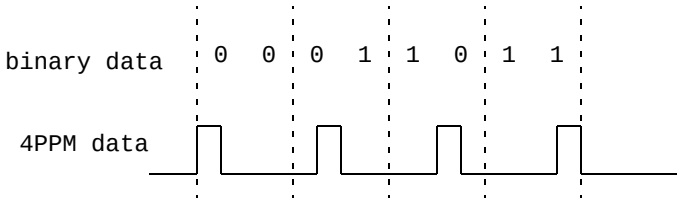
- SICR register - select the FIR mode
- MR2 register - Channel Mode
- If clock generate from the internal source:
  - cdm_irda_bitclk_config - select Mclk frequency, see Section 5.5.14, *PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234*
  - cdm_clock_enable_register - enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
  - CCR- select BitClk and Frame Frequency
- IRCR1 register - full duplex and SIP mode
- IRMDR register - select the clock divider
- RFALARM, TFALARM - select the FIFO “Alarm” level
- CR register - enable or disable receiver and transmitter
- Port_config - select the right Pin-Muxing, see Chapter 2, *Signal Descriptions*

15.3.6.1 Block Diagram and Signal Definition for FIR Mode

The signal definition for FIR mode is the same as in SIR mode. Please see Table 15-84. Figure 15-20. shows the Block diagram for FIR mode. The clock generation is the same as in MIR mode, see Section 15.3.5.1, *Block Diagram and Signal Definition for MIR Mode*.

15.3.6.2 Transmitting and Receiving in FIR Mode

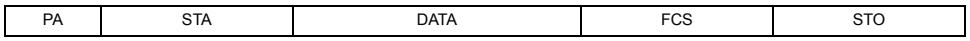
The data field is 4PPM encoded by the transmitter. Data encoding is done LSB first. Each chip duration is 125 ns.



bit pair	4PPM data
00	1000
01	0100
10	0010
11	0001

Figure 15-22. Data Format in FIR Mode

The packet format is defined as



The preamble (PA) field is used by a receiver to establish phase lock. After receiving the start flag (STA), the receiver begin to interpret the 4PPM encoded symbols. The receiver continues receiving until it receives the stop flag (STO). Like the UART mode, the FIR mode sends the lsb first. For more informations regarding the pulse width and Baud rate calculations see Section 15.2.24, *Infrared FIR Divide Register (0x54)—IRFDR*. The FCS is 32 bit CRC defined as:

$$CRC(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

NOTE

The FIR module doesn’t support the CRC generation. If the transfer require a CRC Field use the CRC generation from the BestComm module. See also Chapter 13, *BestComm*.

The chip patterns for PA, STA and STO are defined as:

PA	1000	0000	1010	1000	(16 times repeated)			
STA	0000	1100	0000	1100	0110	0000	0110	0000
STO	0000	1100	0000	1100	0000	0110	0000	0110
	first chip						last chip	

The FIR system must emit SIP (Serial Interaction Pulse), for more informations see Section 15.3.5.3, *Serial Interaction Pulse (SIP)*.

NOTE

Please choose first the desired mode (FIR mode) than configure the port (write to port_config register). This sequence will avoid pulses on the TX line during port configuration. This is very important for all IrDA (SIR, MIR, and FIR) modes.

15.3.6.3 Configuration Sequence Example for FIR Mode

The Table 15-87 shows the configuration sequences. This list includes the FIR mode related registers only, not the other configure values like interrupt and FIFO configurations. PSC module registers can be accessed by word or byte operations.

Table 15-87. Configuration Sequence Example for FIR Mode

Register	Value	Setting
CR	0x0A	Disable the Tx and Rx part for configuration if the PSC was enabled by the work before.
SICR	0x06800000	Select the FIR mode, use internal clock
cdm_irda_bitclk_config	0x8002	set Mclk to 176 Mhz, see Section 5.5.14, *PSC6 (IrDA) Mclock Config Register—MBAR + 0x0234*
cdm_clock_enable_register	0x00000010	enable Mclk, see Section 5.5.6, *CDM Clock Enable Register—MBAR + 0x0214*
CCR	0x0001	set IrdaClk to 88 MHz
IRCR1	0x02	enable SIP
IRFDR	0x0A	set bit clock frequency = 8 MHZ with Mclk frequency = 88 MHz
RFALARM	0x0XXX	Choose Rx FIFO “almost full” threshold level.
TFALARM	0x0XXX	Choose Tx FIFO “almost empty” threshold level.
IMR	0xXXXX	select the desired interrupt
Port_Config	0x00F00000	Select the Pin-Muxing for IrDA mode, see Chapter 2, *Signal Descriptions*
CR	0x05	Enable Tx and Rx

15.3.7 PSC FIFO System

The receive FIFO stack consists of the FIFO and a receiver shift register connected to the RxD. Data is assembled in the receiver shift register and loaded into the FIFO at the location pointed to by the FIFO Write Pointer.

Reading the Rx buffer produces an output of data from the location pointed to by the FIFO Read Pointer. After the read cycle data at the top of the FIFO stack is popped and the Rx shift register can add new data at the bottom of the FIFO. The standard FIFO Controller used in MPC5200 peripherals, such as the PSCs, was designed to control either:

- a transmit (Tx) FIFO
- a receive (Rx) FIFO

Depending on whether the FIFO is set for Tx or Rx, “Alarm” and “Granularity” are measured differently, either:

- valid data bytes (Tx FIFO)
- empty bytes (Rx FIFO)

For both Tx and Rx FIFOs:

- "Alarm" specifies a threshold at which the FIFO generates an interrupt to either:
  - BestComm
  - CPU (alternate)
- "Granularity" specifies a threshold at which the interrupt goes away.

Each PSC provide two control lines to the BestComm system, control the transfer from and to the PSC FIFO.

The FIFOs can be accessed as follows:

- 8-bit Codec mode or UART mode
  - Can access FIFOs either 1, 2, or 4 1-Byte samples at a time.
- 16-bit Codec mode:
  - Can access FIFOs 1 or 2 2-Byte samples at a time.
- 32-bit and 32-Bit Codec mode
  - Can access FIFOs 4-Byte samples at a time
- AC97 mode:
  - Must access FIFOs one sample at a time
  - In addition, when the Rx FIFO is being read, a "1" in bit 20 (21st bit of the sample) marks this sample as the first time slot of a new frame.

Block error mode is always selected because MR1[ERR] is hard-wired high. In block mode SR shows a logical OR of all characters received after the last RESET ERROR STATUS command. Block mode offers a data-reception speed advantage where the software overhead of error-checking each character cannot be tolerated. Errors are not detected until the check is done at the end of an entire message; the faulting character is not identified.

Reading SR does not affect the FIFO. FIFO is popped only when the Rx buffer is read. If the Rx FIFO is completely full a new character is held in the Rx shift register until space is available. However, if a second new character is received, contents of the character in the Rx shift register is lost. The FIFO's are unaffected, and SR[ORERR] sets when the receiver detects the start bit of the new overrunning character.

To support flow control, the receiver can be programmed to automatically negate and assert $\overline{\text{RTS}}$. In which case, the receiver automatically negates $\overline{\text{RTS}}$ when a valid start bit is detected and the FIFO stack is full. The receiver asserts $\overline{\text{RTS}}$ when a FIFO position becomes available. Overrun errors can be prevented by connecting $\overline{\text{RTS}}$ to the $\overline{\text{CTS}}$ input of the transmitting device.

NOTE

The receiver can still read characters in the FIFO stack if the receiver is disabled. If the receiver is reset, the FIFO stack, $\overline{\text{RTS}}$ control, all receiver status bits, and interrupt requests are reset. No more characters are received until the receiver is re-enabled.

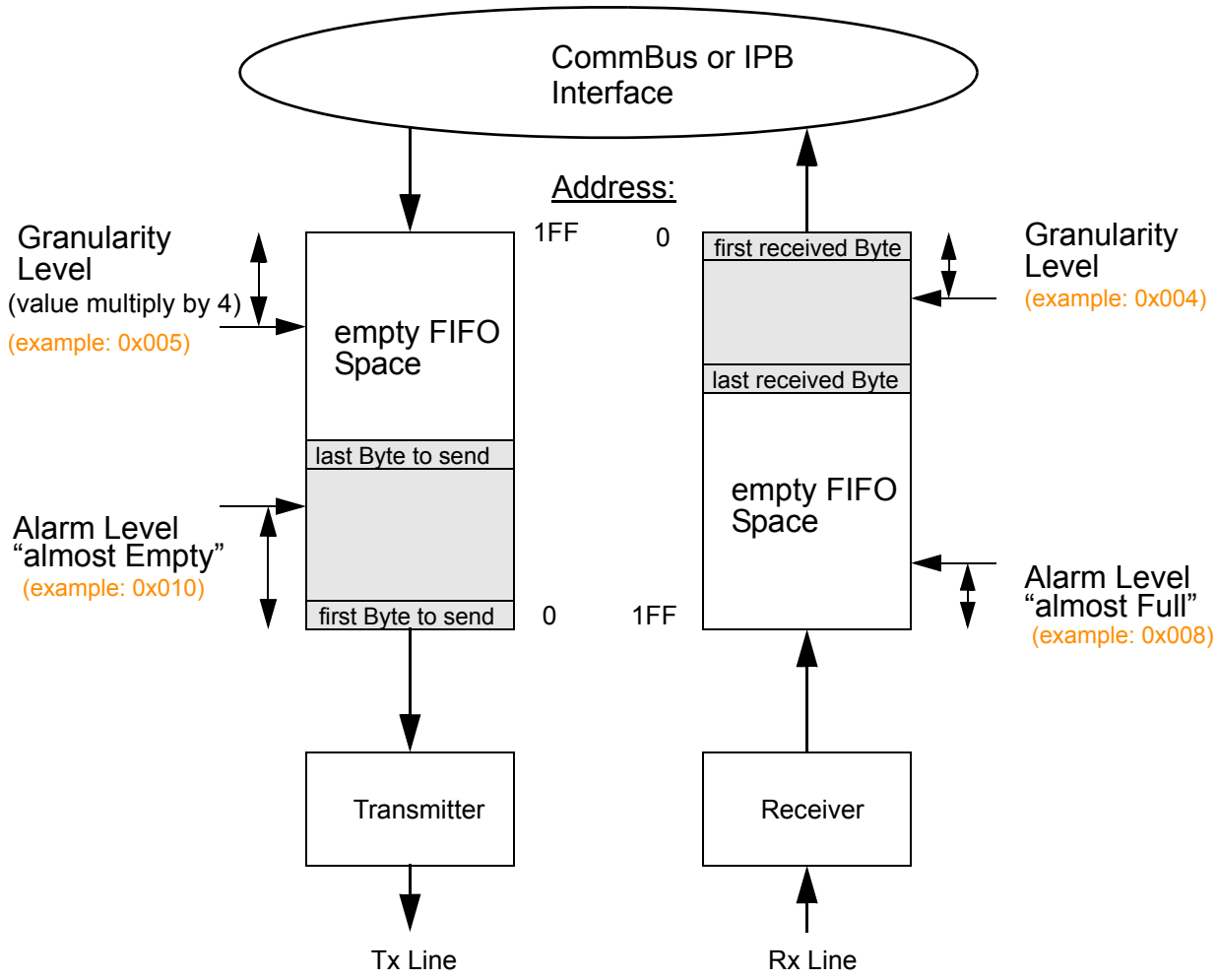


Figure 15-23. PSC FIFO System

15.3.7.1 RX FIFO

The RX FIFO space is 512 Byte. For an Rx FIFO, the "Alarm" value is **not** the amount of "data" in the Rx FIFO. Instead, an interrupt occurs as a result of the amount of empty space remaining in the Rx FIFO. These facts are described in Figure 15-23.

If it is known how much data is needed in the Rx FIFO to cause an interrupt, the value that must be written into the "Alarm" register is:

- the FIFO size, minus the number of data bytes in the FIFO

Unlike the "Alarm" value, "Granularity" value represents a number of data bytes, **not** empty space.

NOTE

In AC97, the number of data bytes are 4-times the number of timeslot samples in the FIFO. Because, each 20-bit sample uses an entire 32-bit longword in the FIFO.

For the Rx FIFO, the value can be between 0 and 7 bytes only. Therefore, the interrupt has hysteresis. For example, the interrupt goes active when the Rx FIFO is "almost full" (i.e., amount of empty space is less than the "Alarm" level). It stays active until enough data is read out of the Rx FIFO so that the amount of data left in the FIFO is less than the "Granularity" level.

For the example (see Figure 15-23) it means:

The requestor to the BestComm to emptying the RX FIFO becomes active if the empty space in the FIFO is less the 8 Bytes (504 date Bytes are in the FIFO).

The requester became inactive if 4Bytes are left in the FIFO. (508 Byte space now)

When BestComm is servicing the FIFO's, this process works well. However, if the CPU is servicing the FIFO's, the interrupt has no hysteresis. For Example, the "Alarm" level is used for both activating and deactivating the CPU interrupt.

When using BestComm you must specify a non-zero “Granularity” to get FIFO underrun errors. This is due to its internal pipelining. BestComm does not immediately stop accessing the FIFO when the FIFO interrupt goes away.

15.3.7.2 TX FIFO

The TX FIFO space is 512 Byte. For a Tx FIFO, the “Alarm” value specifies a threshold in terms of DATA bytes, **not** in terms of empty space as with the Rx FIFO. Once the amount of data in the Tx FIFO falls below the “Alarm” level, an interrupt activates. The interrupt indicates the Tx FIFO is “almost empty” and needs more data. Tx FIFO “Granularity” is specified in terms of empty bytes, **not** a number of data bytes as with the Rx FIFO. For more informations see also Figure 15-23. The “Granularity” value range is 0–7.

The Tx FIFO controller hardware multiplies this value by 4, to establish the actual level at which the FIFO alarm goes away.

For the Tx FIFO, the alarm goes away when the number of empty bytes left in the Tx FIFO is less than or equal to:

- 0 (Granularity value 0)
- 4 (Granularity value 1)
- 8 (Granularity value 2)
- 12 (Granularity value 3)
- 16 (Granularity value 4)
- 20 (Granularity value 5)
- 24 (Granularity value 6)
- 28 (Granularity value 7)

The FIFO interrupt stays active until BestComm writes enough data into the Tx FIFO to reach the Granularity level. Once the Granularity level is reached, the interrupt goes away.

For the example (see Figure 15-23) it means:

The requestor to the BestComm to filling the TX FIFO becomes active if the amount of data in the FIFO is less then 16 data.

The requester became inactive if less than 20 (5 * 4) bytes space in the FIFO.

15.3.8 Looping Modes

The PSC can be configured to operate in various loopback modes as shown in Figure 15-24. These modes are useful for local and remote system diagnostic functions and can be used by in Codec mode as well as UART mode. The modes are described below and in Section 15.2, *PSC Registers—MBAR + 0x2000, 0x2200, 0x2400, 0x2600, 0x2800, 0x2C00*.

The PSCs transmitter and receiver should be disabled when switching between modes. The selected mode is activated immediately on mode selection, regardless of whether a character is being received or transmitted.

15.3.8.1 Automatic Echo Mode

In automatic echo mode, shown in Figure 15-24, the PSC automatically resend received data bit-by-bit. The local CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled. In this mode, received data is clocked on the receiver clock and resent on TxD. The receiver must be enabled, but the transmitter need not be.

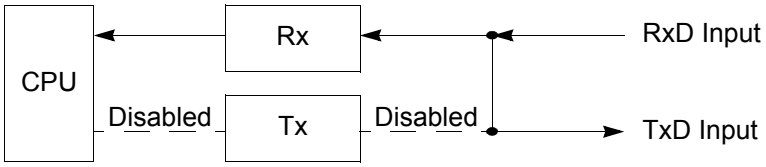


Figure 15-24. Automatic Echo

Because the transmitter is inactive, SR[TxEMP,TxRDY] is inactive and data is sent as it is received. Received parity is checked, but is not recalculated for transmission. Character framing is also checked, but stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

15.3.8.2 Local Loop-Back Mode

Figure 15-25 shows how TxD and RxD are internally connected in local loop-back mode. This mode is for testing the operation of a local PSC module channel by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.

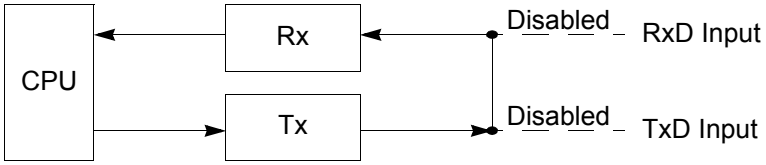


Figure 15-25. Local Loop-Back

Features of this local loop-back mode are:

- Transmitter and CPU-to-receiver communications continue normally.
- RxD input data is ignored.
- TxD data is held marking.
- The receiver is clocked by the transmitter clock.
- Transmitter must be enabled, but the receiver need not be enabled.

15.3.8.3 Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 15-26, the channel automatically transmits received data bit-by-bit on the TxD output. The local CPU-to-transmitter link is disabled. This mode is useful in testing receiver and transmitter operation of a remote channel. For this mode, the transmitter uses the receiver clock.

Because the receiver is not active, received data cannot be read by the CPU and error status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

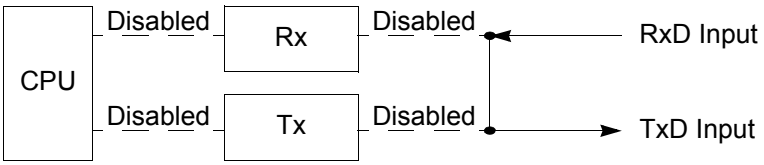


Figure 15-26. Remote Loop-Back

15.3.9 Multidrop Mode

Setting MR1[PM] programs the PSC to operate in a Walk-up mode for multidrop or multiprocessor applications. In this mode, a master can transmit an address character followed by a block of data characters targeted for one of up to 256 slave stations.

Although slave stations have their channel receivers disabled, they continuously monitor the masters data stream. When the master sends an address character, the slave receiver channel notifies its respective CPU by setting SR[RxRDY] and generating an interrupt (if programmed to do so). Each slave station CPU then compares the received address to its station address and enables its receiver if it wishes to receive the subsequent data characters or block of data from the master station. Slave stations not addressed continue monitoring the data stream. Data fields in the data stream are separated by an address character. After a slave receives a block of data, its CPU disables the receiver and repeats the process.

Figure 15-27 shows functional timing information for multidrop mode.

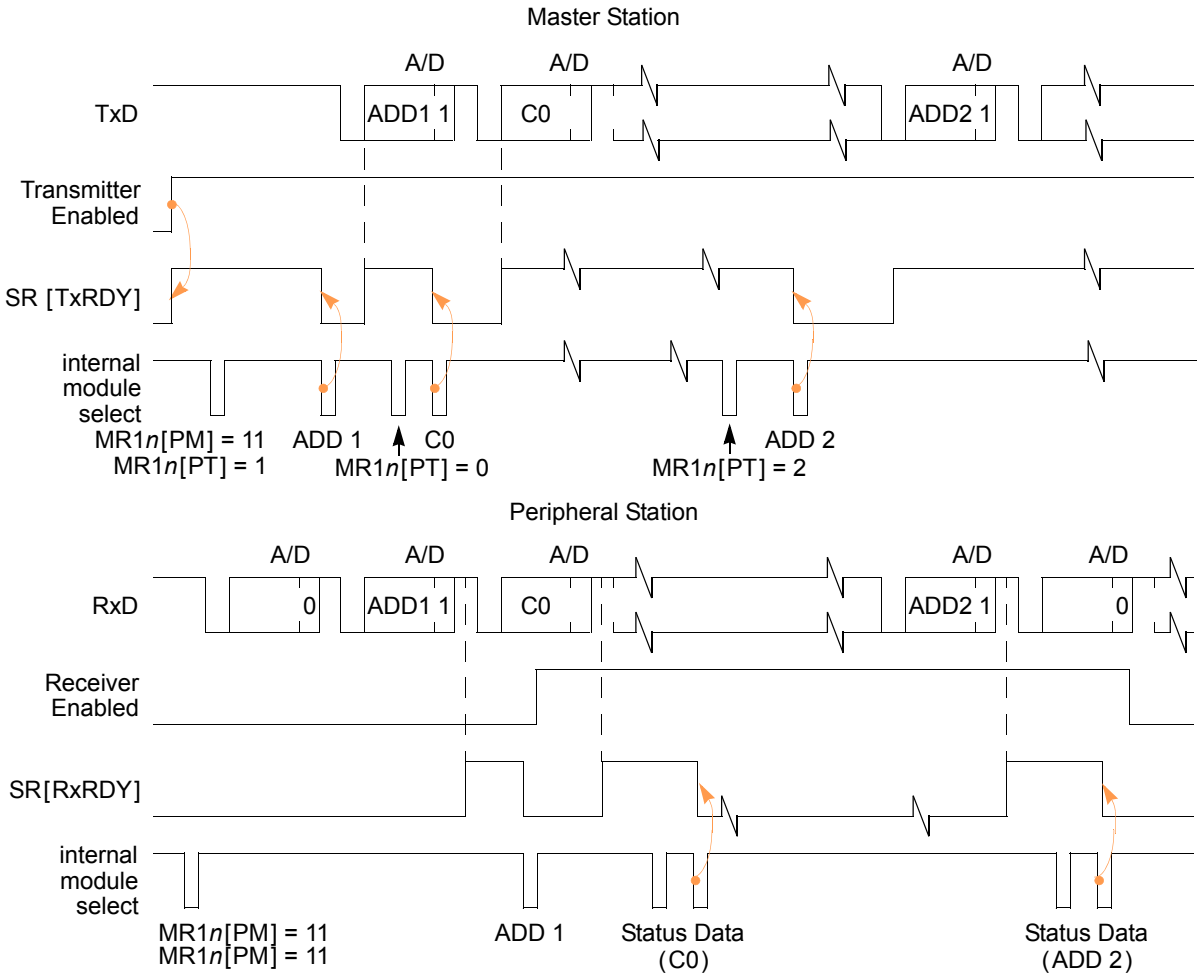


Figure 15-27. Timing Diagram—Multidrop Mode

A character sent from the master station consists of:

- a start bit
- a programmed number of data bits
- an address/data (A/D) bit flag
  - A/D=1 indicates an address character
  - A/D=0 indicates a data character
- a programmed number of stop bits

A/D polarity is selected through MR1[PT]. MR1 should be programmed before enabling the transmitter and loading the corresponding data bits into the Tx buffer.

In multidrop mode, the receiver continuously monitors the received data stream, regardless of whether it is enabled or disabled.

- If the receiver is disabled, it sets the RxRDY bit and loads the character into the receiver holding register FIFO stack, provided the received A/D bit is 1 (address tag). If the received A/D bit is 0 (data tag), the character is discarded.
- If the receiver is enabled, all received characters are transferred to the CPU through the receiver holding register stack during read operations.

In either case, data bits are loaded into the data portion of the stack while the A/D bit is loaded into the status portion of the stack normally used for a parity error (SR[PE]).

Framing error, overrun error, and break detection operate normally. The A/D bit takes the place of the parity bit. Parity is neither calculated nor checked. Messages in this mode may still contain error detection and correction information. One way to provide error detection if 8-bit characters are not required, is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

Notes

Chapter 16
XLB Arbiter

16.1 Overview

This document contains the following section:

- Section 16.1, *Overview*
- Section 16.2, *XLB Arbiter Registers—MBAR + 0x1F00*

16.1.1 Purpose

The purpose of the XLB Arbiter is to manage bus requests from the XLB masters (USB, PCI, BestComm, and G2_LE core), and determine which master should be granted the bus at any one time. The arbiter employs both master prioritization and a fair-share LRU (least-recently-used) algorithm to reduce access latency and starvation across all masters.

The XLB Arbiter consists of five functional blocks as shown below.

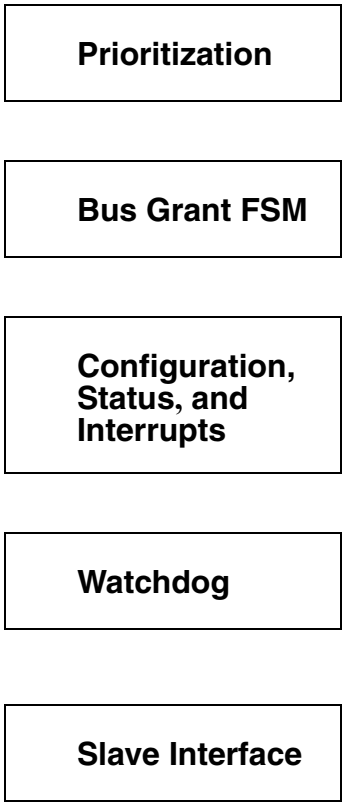


Figure 16-1. Block Diagram of XLB Arbiter

16.1.1.1 Prioritization

The prioritization block signals that a master is requesting the bus and which master has priority.

Priority is determined first by using the master priority level assigned by either the hardware-wired mNpr signals, or software-programmable Master N Priority bits in the Arbiter Master N Priority Register depending on the Master Priority Enable bit for each master. Masters at the same level of priority will be further sorted by a least recently used algorithm (LRU). Once a requesting master is identified as having priority and is granted the bus, that master will be continue to be granted the bus if:

1. It is requesting the bus. The request must occur immediately after the required one clock de-assertion after a qualified bus grant, and
2. It is the highest priority device, and
3. There is no address retry assertion.

Multiple masters at level 0 will only be able to perform one tenure before the bus is passed to the next master at level 0 using the LRU algorithm.

The priority level of each master may be changed while the arbiter is running. This allows dynamic changes in priority such as an aging scheme. It is possible for the G2_LE core to control priority by enabling the Master Priority Enable bits for a master. This causes the priority to be determined from the Master N Priority bits in the Arbiter Master N Priority Register. The G2_LE core then may write this register to set the master's priority.

16.1.1.2 Bus Grant Mechanism

16.1.1.2.1 Bus Grant

The Bus Grant mechanism will generate the address bus grant signals to the masters using the signals from the prioritization function as well as the XL bus signals (ts_b, aack_b, ta_b, tt[0:4], artry, tbst_b, tea_b, dbb_b). It will also generate required indicators of state to the prioritization and watchdog functions.

The Bus Grant mechanism will enforce a one level address pipeline, if pipelining is enabled (via the PLDIS bit in the Arbiter Configuration Register). The critical condition is that before a third address tenure is granted, the first tenure (address, and if needed, data) must be completed. The arbiter will assert bus grant to a master when the master is requesting, or if parking is enabled, and the one level pipeline condition is met.

16.1.1.2.2 Parking Modes

The Bus Grant mechanism will support the No Parking, Park on Programmed Master, and Park on Last Master bus parking modes. When in No Parking Mode, the arbiter will not assert bus grant when there are no masters asserting bus request. In Park on Programmed Master Mode, the arbiter will assert bus grant to the master indicated in the Select Parked Master bits (Arbiter Configuration Register, SP[2:0] bits) when no masters are asserting bus request, and the one level pipeline will not be violated. In Park on Last Master Mode, the arbiter will assert bus grant to the last master granted the bus when no masters are asserting bus request, and the one level pipeline will not be violated.

16.1.1.3 Configuration, Status, and Interrupt Generation

This block provides a set of status and configuration registers as well as interrupt generation for enabled interrupt conditions. These registers are detailed in the register section below. Some registers have reserved (unused, undefined) bits. These bits will always read as 0, and if written, should be written to 0 for future software compatibility.

16.1.1.4 Watchdog Functions

16.1.1.4.1 Timer Functions

There are three watchdog timers for address tenure, data tenure, and bus activity time out. Each has a programmable timer count and can be disabled. A timer time-out will set a status bit and trigger an interrupt if that interrupt is enabled.

The address tenure watchdog is a 32-bit timer. If an AACK is not detected by the programmed number of clocks after bus grant is accepted, the address watchdog timer will expire and the arbiter will issue AACK. The related data tenure will be terminated with TEA. The arbiter will set the Address Tenure Time-out Status bit in the Arbiter Status Register and issue an interrupt if that interrupt is enabled.

The upper 28-bits of address tenure time-out are programmed via the Address Tenure Time-out Register. The lower 4 bits are always 0xF.

The data tenure watchdog is a 32-bit timer. If a data tenure is not terminated, the data watchdog timer will expire and the arbiter will issue TEA. The arbiter will set the Data Tenure Time-out Status bit in the Arbiter Status Register and issue an interrupt if that interrupt is enabled.

Address Time-out (32 bits) = {Address Tenure Time-out Register (28-bits), 0xF}

Data Time-out (32 bits) = {Data Tenure Time-out Register (28-bits), 0xF}

NOTE

Enabling the data time-out will also enable the address time-out. This is required to prevent a data time-out before an AACK assertion.

The bus activity watchdog is a 32-bit timer. If no bus activity (no assertion of DBB or ABB) is detected by the programmed number of clocks, the bus activity watchdog timer will expire and the arbiter will set the Bus Activity Time-out Status bit in the Arbiter Status Register and issue an interrupt if that interrupt is enabled.

For any TEA assertion (from a watchdog time-out, or other source), a Machine Check exception will result in the G2_LE core. See the XLB Arbiter interrupt enablement recommendations below for the Arbiter Interrupt Enable Register. For more information on the Machine Check exception, see the 603e Users' Manual, Section 4.5.

16.1.1.4.2 Other Tenure Ending Conditions

In addition to the watchdog timers, this function will terminate tenures with or without TEA depending on the following conditions:

- AACK the address tenure for eciwx and ecowx transfer types, then TEA the ensuing data tenure. This sets the External Control Word Read/Write Status bit in the Arbiter Status Register.
- AACK the address tenure for address only and reserved transfer types. (no ensuing data tenure) This sets the TT Address Only Status bit or TT Reserved Status Bit, respectively, in the Arbiter Status Register.
- AACK the address tenure for invalid TBST/TSIZ combinations, then TEA the ensuing data tenure. This sets the TBST/TSIZ Mismatch Status bit in the Arbiter Status Register.

If enabled, an interrupt will be issued when a status bit is set.

When the arbiter ends a tenure or detects a TEA from a device, certain bus signals (Address, TT, TSIZ, TBST, GLB) are latched into the Arbiter Address Capture Register and Arbiter Bus Signal Capture Register. Additional errors will be handled but the original error related information is maintained until the error status is cleared.

16.2 XLB Arbiter Registers—MBAR + 0x1F00

The XLB Arbiter provides thirteen 32-bit active registers. All registers are located at an offset from the Module Base Address Register (MBAR). The XLB Arbiter offset from MBAR is 0x1F00. Therefore, the actual address for each XLB Arbiter register is: **MBAR + 0x1F00 + register address.**

The read/write nature of each register is shown in the descriptions that follow.

- Bit 0 in all registers is the most significant bit (MSB).
- Reserved bits cannot be written to, and will always read 0.
- Registers may be accessed on the following aligned boundaries:
  - 1 byte
  - 2 byte
  - word (32-bit)
  - double-word (64-bit)

Registers are organized on word boundaries to allow easy register mask operations.

When a bit enables or disables a function, the values are defined as:

- 0 = disabled
- 1 = enabled

The XLB Arbiter registers are listed below, followed by detailed descriptions of each register.

16.2.1 Arbiter Configuration Register (R/W)—MBAR + 0x1F40

The Arbiter Configuration Register is used to control the arbiter watchdog functionality and other XLB-related system configuration parameters.

Table 16-1. Arbiter Configuration Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PLDIS	Rsvd														
W																
RESET:	1	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 lsb
R	SE	USE_ WWF	TBEN	Rsvd	WS	SP			Rsvd	PM		Rsvd	BA	DT	AT	Rsvd
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Bit	Name	Description
0	PLDIS	Pipeline Disable. This bit is used to enable or disable transaction pipelining on the XLB. See note below. 0 = Enable pipelining 1 = Disable pipelining
1:15	—	Reserved
16	SE	Snoop Enable. This sets the address snooping enablement on the XLB. 0 = Disable address snooping. Internally on the XLB, the gbl_b signal is gated (always negated). This overrides any setting of the Arbiter Snoop Window Register (MBAR + 0x0070). 1 = Allow address snooping. Internally on the XLB, the gbl_b signal is not gated, and assertions of this signal during address tenures will be recognized.
17	USE_WWF	Force write-with-flush transfer type (TT) for PCI, BestComm, and USB interfaces to XLB. 0 = Write-with-kill operation is allowed on burst transactions. 1 = Always use write-with-flush on burst transactions.
18	TBEN	Timebase Enable. This bit is used as a “count enable” control input for the timebase counter in the 603e core. 0 = Timebase should stop clocking. 1 = Timebase should continue clocking.
19	—	Reserved
20	WS	Minimum Wait State. This bit sets the minimum number of wait states for slaves to respond with AACK assertion on the XLB. 0 = 0 minimum wait state. 1 = 1 minimum wait state.
21:23	SP[2:0]	Select Parked Master. These bits set the master that is used in Park on Programmed Master mode (000 = master 0, 001 = master 1, ..., 111 = master 7).
24	—	Reserved
25:26	PM[1:0]	Parking Mode. 00 = No parking (default). 01 = Reserved. 10 = Park on most recently used master. 11 = Park on programmed master as specifed by the Select Parked Master bits 21:23 above.
27	—	Reserved

Bit	Name	Description
28	BA	Bus Activity Time-out Enable. If enabled, the arbiter will set the Bus Activity Time-out Status bit (Arbiter Status Register, bit 29) when the Bus Activity Time-out is reached. Bus Activity Time-out is derived from the Arbiter Bus Activity Time Out Count register.
29	DT	Data Tenure Time-out Enable. If enabled, the arbiter will assert TEA when the Data Tenure Time-out is reached. Data Tenure Time-out is derived from the Arbiter Data Tenure Time Out Count Register. Also, the arbiter will set the Data Tenure Time-out Status bit (Arbiter Status Register, bit 30). Setting this bit will also enable the Address Tenure Time-out. This is required to ensure that a data time-out will not occur before an address acknowledge.
30	AT	Address Tenure Time-out Enable. If enabled, the arbiter will assert AACK and TEA (if required) when the Address Tenure Time-out is reached. Address Tenure Time-out is derived from the Arbiter Address Tenure Time Out Count register. Also, the arbiter will set the Address Tenure Time-out Status bit (Arbiter Status Register, bit 31). Address Tenure Time-out is also enabled by the DT bit above.
31	—	Reserved

NOTE

The PLDIS reset value is 1, which means the XLB Arbiter will prohibit transaction pipelining. In most applications, transaction pipelining will provide a significant performance increase, and therefore the customer should consider setting this bit to 0 to take advantage of this increase.

16.2.2 Arbiter Version Register (R)—MBAR + 0x1F44

The Arbiter Version Register contains the silicon version number for the arbiter hardware.

Table 16-2. Arbiter Version Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Version ID[0:15]															
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 lsb
R	Version ID[16:31]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	Name	Description
0:31	VER	Hardware version ID. The current version number is 0x0001.

16.2.3 Arbiter Status Register (R/W)—MBAR + 0x1F48

The Arbiter Status Register indicates the state of watchdog functions. When a monitored condition occurs, the respective bit is set to 1. The bit remains 1 until cleared by writing 1 into that bit position. Even if the causal condition is removed, the bit remains set until cleared.

Table 16-3. Arbiter Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd															
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 lsb
R	Rsvd							SEA	MM	TTA	TTR	ECW	TTM	BA	DT	AT
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:22	—	Reserved
23	SEA	Slave Error Acknowledge. This bit is set when an error is detected by any slave devices during the transfer.
24	MM	Multiple Masters at Priority 0. If more than one master is recognized at priority 0, this bit is set. Once this occurs, this bit will remain set until cleared. The arbiter recognizes priority by the hardware-wired mNpri signals or (if enabled) the Arbiter Master N Priority Register. This bit is intended to help in tuning dynamic priority algorithm development.
25	TTA	TT Address Only. The arbiter automatically AACKs for address only TT (transfer type) codes. This bit is set when this condition occurs. For a description of TT codes, see the MPC603e Users' Manual, Section 7.2.
26	TTR	TT Reserved. The arbiter automatically AACKs for reserved TT (transfer type) codes. This bit is set when this condition occurs. For a description of TT codes, see the MPC603e Users' Manual, Section 7.2.
27	ECW	External Control Word Read/Write. External Control Word Read/Write operations are not supported on the XLB. If either occur, the arbiter AACKs and TEAs the transaction, and sets this bit.
28	TTM	TBST/TSIZ mismatch. Set when an illegal/reserved TBST and TSIZ[0:2] combinations occur. These combinations are TBST asserted and TSIZ[0:2] = 000, 001, 011, or 1xx (where "x" is 0 or 1). For a description of TBST and TSIZ, see the MPC603e Users' Manual, Section 7.2.
29	BA	Bus Activity Tenure Time-out. Set when the bus activity time-out counter expires.
30	DT	Data Tenure Time-out. Set when the data tenure time-out counter expires.
31	AT	Address Tenure Time-out. Set when the address tenure time-out counter expires.

16.2.4 Arbiter Interrupt Enable Register (R/W)—MBAR + 0x1F4C

The Arbiter Interrupt Enable Register is used to enable a status bit to cause an interrupt. If the interrupt enable and corresponding status bits are set in the Arbiter Status Register and the Arbiter Interrupt Enable Register, the arbiter will assert the arb_int_b signal. Normally, an interrupt service routine would read the status register to determine the state of the arbiter. It is possible that multiple conditions exist that would cause an interrupt. Disabling an interrupt by writing a 0 to a bit in this register will not clear the status bit in the Arbiter Status Register.

NOTE

For SEAE, ECWE, TTME and ATE interrupt conditions, the arbiter also generates a TEA at a later time (dependent upon XL bus activity), which will cause a Machine Check exception. As a result, state information for the interrupted exception in the save/restore registers (SRR0 and SRR1) may be lost. Therefore, it is recommended that SEA, ECW, TTM, and ATE remain disabled at all times. It is possible to enable an arbiter interrupt for MME, TTAE, TTRE, as they do not result in a TEA; in case of DTE and BAE, arbiter interrupt can be enabled, as the TEA assertion always preceeds the interrupt.

Table 16-4. Arbiter Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd															
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 lsb
R	Rsvd							SEAE	MME	TTAE	TTRE	ECWE	TTME	BAE	DTE	ATE
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:22	—	Reserved
23	SEAE	Slave Error Acknowledge interrupt enable
24	MME	Multiple Masters at priority 0 interrupt enable
25	TTAE	TT Address Only interrupt enable
26	TTRE	TT Reserved interrupt enable
27	ECWE	External Control Word Read/Write interrupt enable
28	TTME	TBST/TSIZ mismatch interrupt enable
29	BAE	Bus Activity Tenure Time-out interrupt enable
30	DTE	Data Tenure Time-out interrupt enable
31	ATE	Address Tenure Time-out interrupt enable

16.2.5 Arbiter Address Capture Register (R)—MBAR + 0x1F50

The Arbiter Address Capture Register captures the address for a tenure that has either:

- an address time-out,
- a data time-out, or
- a TEA from another source

The captured value is held until unlocked by writing any value to the Arbiter Address Capture Register or Arbiter Bus Signal Capture Register. This value is also unlocked by writing a 1 to either the Arbiter Status Register, bit 30 (Data Tenure Time-out Status) or bit 31 (Address Tenure Time-Out Status). Unlocking the register does not clear its contents.

Table 16-5. Arbiter Address Capture Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Address[0:15]															
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 lsb
R	Address[16:31]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	ADRCAP	Address Capture Value. This is the address that is captured when a bus error occurs. This happens after an address time-out, data time-out, or any TEA assertion.

16.2.6 Arbiter Bus Signal Capture Register (R)—MBAR + 0x1F54

The Arbiter Bus Signal Capture Register captures TT, TBST, GBL, and TSIZ for an XLB address tenure that has either:

- an address time-out,
- a data time-out, or
- a TEA from another source

These values are held until unlocked by writing any value to the Arbiter Address Capture Register or Arbiter Bus Signal Capture Register. These values are also unlocked by writing 1 to either the Arbiter Status Register, bit 30 (Data Tenure Time-out Status) or bit 31 (Address Tenure Time-Out Status). Unlocking the register does not clear its contents.

Table 16-6. Arbiter Bus Signal Capture Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd															
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 lsb
R	Rsvd						TSIZ[0:2]			GBL	TBST	TT[0:4]				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:21	—	Reserved
22:24	TSIZ	Captured Value of TSIZ[0:2].
25	GBL	Captured Value of GBL.
26	TBST	Captured Value of TBST.
27:31	TT	Captured Value of TT[0:4].

16.2.7 Arbiter Address Tenure Time-Out Register (R/W)—MBAR + 0x1F58

The Arbiter Address Tenure Time-out Register provides an expiration value to the arbiter watchdog for address tenures. After an address tenure is initiated with a TS signal assertion by the master, the address tenure watchdog starts counting until either AACK is asserted, or the counter expires. If expiration occurs before the AACK is encountered, the arbiter issues an AACK assertion, followed by a TEA assertion for the corresponding data tenure. Subsequently, the Arbiter Status Register, bit 31 (AT) is set, and an interrupt is generated if the Arbiter Interrupt Enable Register, bit 31 (ATE) is set.

The Arbiter Address Tenure watchdog can be enabled/disabled via the Arbiter Configuration Register, bit 30 (AT).

Table 16-7. Arbiter Address Tenure Time-Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd				ADRTO[4:15]											
W																
RESET:	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	ADRTO[16:31]															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0:3	—	Reserved
4:31	ADRTO	Address Tenure Time-out. Contains the upper 28 bits of the Address Time-out Counter. Values represent increments of 16. Default value is 0xFFFFFFF.

16.2.8 Arbiter Data Tenure Time-Out Register (R/W)—MBAR + 0x1F5C

The Arbiter Data Tenure Time-out Register provides an expiration value to the arbiter watchdog for data tenures. After a data tenure is initiated with a DBB signal assertion by the master, the data tenure watchdog starts counting until either TA is asserted, or the counter expires. If expiration occurs before TA is encountered, the arbiter issues a TEA assertion for the data tenure. Subsequently, the Arbiter Status Register, bit 30 (DT) is set, and an interrupt is generated if the Arbiter Interrupt Enable Register, bit 30 (DTE) is set.

The Arbiter Data Tenure watchdog can be enabled/disabled via the Arbiter Configuration Register, bit 29 (AT).

Table 16-8. Arbiter Data Tenure Time-Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd				DATTO[4:15]											
W																
RESET:	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	DATTO[16:31]															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0:3	—	Reserved
4:31	DATTO	Data Tenure Time-out. Contains the upper 28 bits of the DataTime-out Counter. Values represent increments of 16. Default value is 0xFFFFFFF.

16.2.9 Arbiter Bus Activity Time-Out Register (R/W)—MBAR + 0x1F60

The Arbiter Bus Activity Time-out Register provides an expiration value to the arbiter watchdog for bus activity. The watchdog monitors bus activity, after the counter expires due to excessive bus idle time, the Arbiter Status Register, bit 29 (BA) is set, and an interrupt is generated if the Arbiter Interrupt Enable Register, bit 29 (BAE) is set.

The Arbiter Bus Activity watchdog can be enabled/disabled via the Arbiter Configuration Register, bit 28 (BA).

Table 16-9. Arbiter Bus Activity Time-Out Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUSTO[0:15]															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	BUSTO[16:31]															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0:31	BUSTO	Bus Activity Time-out. Contains the value of the Bus Activity Time-out Counter. Values represent increments of 1. Default value is 0xFFFFFFFF.

16.2.10 Arbiter Master Priority Enable Register (R/W)—MBAR + 0x1F64

The Arbiter Master Priority Enable Register determines whether the arbiter uses the hard-wired or software programmable priority for a master. The default is enabled for all masters. Both methods may be employed at the same time for different masters. This register may be written to at any time, and the change becomes effective one clock after the register is written.

When enabled, the software programmable value in the Arbiter Master N Priority Register is used as the priority for the master. When disabled, the priority assignment for each master is determined by the hardware-wired mNpri signals, as shown in Table 16-10.

Table 16-10. Arbiter Master Priority Enable Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd															
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 lsb
R	Rsvd								M7	M6	M5	M4	M3	M2	M1	M0
W																
RESET:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit	Name	Description
0:23	—	Reserved
24	M7	Master 7 Priority Register Enable
25	M6	Master 6 Priority Register Enable
26	M5	Master 5 Priority Register Enable
27	M4	Master 4 Priority Register Enable
28	M3	Master 3 Priority Register Enable
29	M2	Master 2 Priority Register Enable
30	M1	Master 1 Priority Register Enable
40	M0	Master 0 Priority Register Enable

Table 16-11. Hardware Assignments of Master Priority

Master	Priority	Description
M7–M4	—	Unused
M3	0	PCI Target Interface
M2	1	BestComm
M1	2	USB
M0	7	G2_LE Core

16.2.11 Arbiter Master Priority Register (R/W)—MBAR + 0x1F68

The Arbiter Master N Priority Register is used to set the software-programmable priority of each master. This register is used in conjunction with the Arbiter Master Priority Enable Register to enable software-programmable master priorites, consequently ignoring the hardware-wired mNpri signals. This register may be written at any time, and changes to this register become effective one clock after the register is written.

Valid priority values range from 0 to 7, with 0 being the highest priority. Each of the eight fields in the register has the upper (fourth) bit reserved. This allows for a possible future expansion to 16 priority levels. Currently, the reserved bits will always read as 0, and should be written as 0 for future software compatibility.

Table 16-12. Arbiter Master Priority Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Rsvd	M7 Priority			Rsvd	M6 Priority			Rsvd	M5 Priority			Rsvd	M4 Priority		
W																
RESET:	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Rsvd	M3 Priority			Rsvd	M2 Priority			Rsvd	M1 Priority			Rsvd	M0 Priority		
W																
RESET:	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0

Bit	Name	Description
0	—	Reserved
1:3	M7P	Master 7 Priority
4	—	Reserved
5:7	M6P	Master 6 Priority
8	—	Reserved
9:11	M5P	Master 5 Priority
12	—	Reserved
13:15	M4P	Master 4 Priority
16	—	Reserved
17:19	M3P	Master 3 Priority
20	—	Reserved
21:23	M2P	Master 2 Priority
24	—	Reserved
25:27	M1P	Master 1 Priority
28	—	Reserved
29:31	M0P	Master 0 Priority

16.2.12 Arbiter Snoop Window Register (RW)—MBAR + 0x1F70

The Arbiter Snoop Window Register is used by the PCI, BestComm, and USB Host interfaces to the XLB. This register dictates the size of an address range in memory that will allow or prohibit address snooping. Each master interface (MBI) monitors this register and determines if the master's address transferred to the XLB should be sent with gbl_b signal assertion.

The benefit of this implementation of a system-wide address snooping control mechanism is that address ranges that need not be cache-coherent will not be snooped by the core. Under certain conditions, the core can assert ARTRY when it is too busy to check the cache for a snooped address on the bus. Therefore, by specifying an address range that doesn't require snooping, the number of core-initiated "busy ARTRY" scenarios can be minimized, and the subsequent time penalty to eventually retry the transaction is avoided. The quantity of this incremental system performance increase is dependant upon the loading of core (especially as related to the caches), and the average number of accesses to the designated regions of the snoop window.

The MPC5200 implementation of this address snooping control is shown in the figure below. At the start of a master's address tenure, the master interface decodes the address and determines if it needs to be snooped, based on the configuration of the Arbiter Snoop Window Register. If the transaction requires snooping, the gbl_b signal is asserted; otherwise, gbl_b is negated. However, before the gbl_b signal reaches the XLB for the address tenure, it is gated by a mux, controlled by the Arbiter Configuration Register SE (snoop enable) bit. If SE is 0, gbl_b will always be negated, and no XLB transaction will be snooped. If SE is 1, the gbl_b signal generated by the master bus interface will be allowed to pass to the XLB.

For a more detailed description of address snooping and G2_LE cache-coherency, see the MCP603e Users' Manual, Section 3.6.

Table 16-13. Arbiter Snoop Window Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WINBASE[0:15]															
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 lsb
R	WINBASE[16:19]				Rsvd				DS	Rsvd		WINSIZE[0:4]				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:19	WINBASE	Window Base Address. Defines the base address of snoopable/non-snoopable addresses for all PCI, BestComm, and USB address transfers.
20:23	—	Reserved
24	DS	Default Snooping Policy: 0 = Addresses inside window are snooped. Default gbl_b = "negated" 1 = Addresses outside window are snooped. Default gbl_b = "asserted"

Bit	Name	Description
25:26	—	Reserved
27:31	WINSIZE	Window Size - Defines the size of window. The lower bits of WINBASE are effecttively ignored/masked from snooping address comparison depending on the value set in this field. WINSIZE = 00000: Window miss* WINSIZE = 00001 - 01011: 4KByte (non-mask) WINSIZE = 01100: 8KByte (1 bit mask) WINSIZE = 01101: 16KByte (2 bits mask) WINSIZE = 01110: 32KByte (3 bits mask) WINSIZE = 01111: 64KByte (4 bits mask) WINSIZE = 10000: 128KByte (5 bits mask) WINSIZE = 10001: 256KByte (6 bits mask) WINSIZE = 10010: 512KByte (7 bits mask) WINSIZE = 10011: 1MByte (8 bits mask) WINSIZE = 10100: 2MByte (9 bits mask) WINSIZE = 10101: 4MByte (10 bits mask) WINSIZE = 10110: 8MByte (11 bits mask) WINSIZE = 10111: 16MByte (12 bits mask) WINSIZE = 11000: 32MByte (13 bits mask) WINSIZE = 11001: 64MByte (14 bits mask) WINSIZE = 11010: 128MByte (15 bits mask) WINSIZE = 11011: 256MByte (16 bits mask) WINSIZE = 11100: 512MByte (17 bits mask) WINSIZE = 11101: 1GByte (18 bits mask) WINSIZE = : 2GByte (19 bits mask) WINSIZE = 11111: Reserved **Note:** *NOTE: Software should always write a non-zero value in this field. Otherwise, the address comparison does not take effect (treated as a “window miss”).

16.2.13 Arbiter Reserved Registers—MBAR + 0x1F00-1F3C, 0x1F74-1FFF

These are reserved registers and should not be accessed.

Table 16-14. Arbiter Reserved Registers

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	—	Reserved

Notes

Chapter 17
Serial Peripheral Interface (SPI)

17.1 Overview

The following sections are contained in this document:

- Section 17.2, *SPI Signal Description*
- Section 17.3, *SPI Registers—MBAR + 0x0F00*
- Section 17.4, *Functional Description*

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication between the MPC5200 and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

Figure 17-1 shows the SPI block diagram.

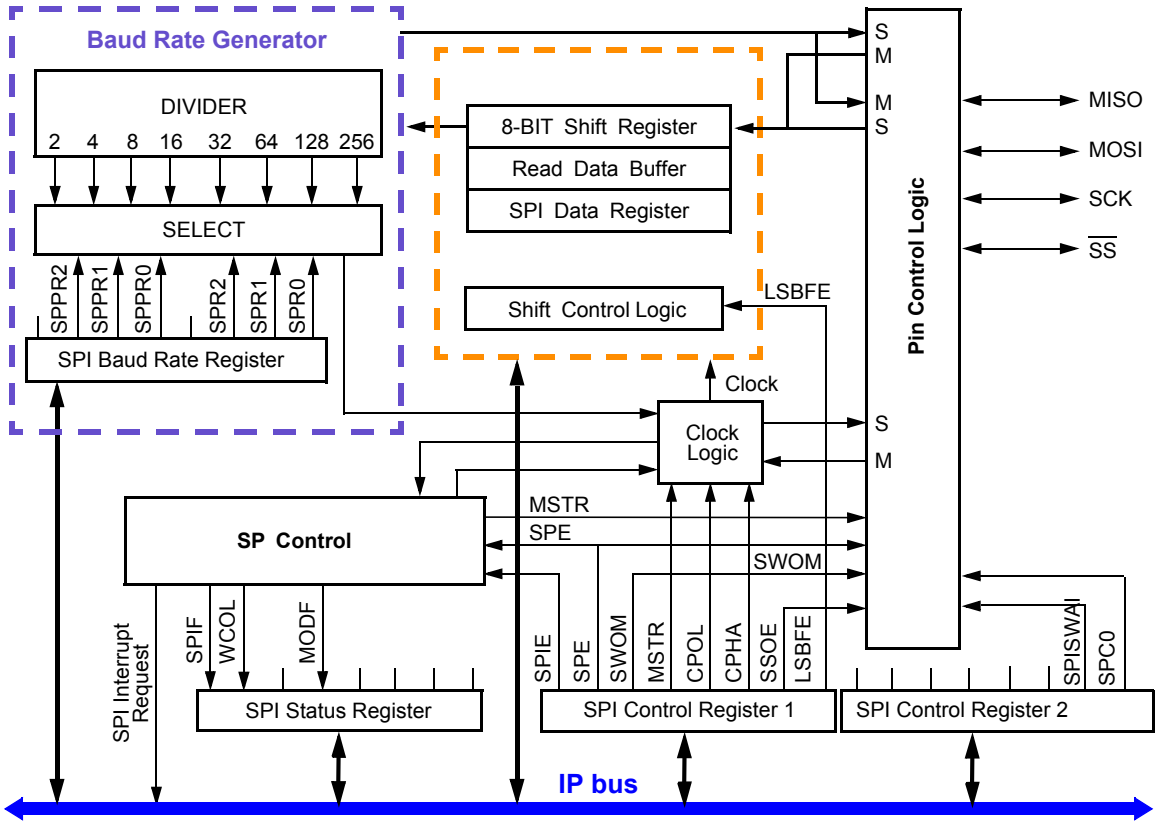


Figure 17-1. Block Diagram—SPI

17.1.1 Features

The SPI has the following features:

- Master mode and slave mode
- Bi-directional mode
- Slave-select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

17.1.2 Modes of Operation

The SPI functions in the following three modes:

- **Run Mode**—The normal mode of operation.
- **Wait Mode**—The SPI can be configured to operate in low-power mode. Based on the internal bit state, the SPI can operate normally when the CPU is in wait mode or the SPI clock generation can be turned off and the SPI module enters a power conservation state during wait mode. During wait mode, any master transmission in progress stops. Transmission and reception resumes when the SPI exits wait mode.
- **Stop Mode**—This mode is system dependent. The SPI enters the stop mode when the module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the processor enters stop mode, the transmission stops until the processor exits stop mode.

17.2 SPI Signal Description

Table 17-1 shows external SPI signals and their properties. These signals may connect off-chip. Detailed signal descriptions are given in the sections below.

Table 17-1. SPI External Signal Descriptions

Signal Name	Port	Function[1]	Reset State
MISO	SPIPORT[7]	Master Data In/Slave Data Out	0
MOSI	SPIPORT[6]	Master Data Out/Slave Data In	0
SCK	SPIPORT[5]	Serial Clock	0
SS	SPIPORT[4]	Slave Select	0

Note:
1. SPI ports MISO, MOSI, SCK, and $\overline{SS}$ are GPIO ports when SPI is disabled (SPE=0).

17.2.1 Master In/Slave Out (MISO)

MISO is one of two SPI module pins that transmit serial data. MISO is an input when the SPI is configured as a master and an output when the SPI is configured as a slave.

If the bidirectional serial pin mode is selected as a slave, MISO becomes a slave in/slave out (SISO) and the direction is controlled by the associated bit in the SPI port data direction register.

In a multiple-master system, all MISO pins are tied together.

17.2.2 Master Out/Slave In (MOSI)

MOSI is one of two SPI module pins that transmit serial data. MOSI is an output when the SPI is configured as a master and an input when the SPI is configured as a slave.

If the bidirectional serial pin mode is selected as a master, MOSI becomes master out/master in (MOMI) and the direction is controlled by the associated bit in the SPI port data direction register.

In a multiple-master system, all MOSI pins are tied together.

17.2.3 Serial Clock (SCK)

The serial clock synchronizes data transmissions between master and slave devices. SCK is an output if the SPI is configured as a master and SCK is an input if the SPI is configured as a slave.

In master mode the Serial Clock is derived from the IPB clock.

In a multiple-master system, all SCK pins are tied together.

17.2.4 Slave-Select ($\overline{SS}$)

The slave-select output or input provides a means of selectively enabling slaves so several may coexist in one system. $\overline{SS}$ is either a general-purpose output (SSOE = 0) or the slave select output (SSOE = 1) when the SPI is in master mode and the associated data direction bit is set.

The $\overline{SS}$ pin is the mode fault input when the SPI is in master mode and the associated data direction bit is clear. When the data direction bit is clear and SSOE = 1, the $\overline{SS}$ pin is a general-purpose input.

$\overline{SS}$ is always an input when the SPI is in slave mode, regardless of the state of the data direction bit for that pin. When the SPI is configured as a slave, the MISO (or SISO) output driver is three-stated until enabled by the slave select input (low true) so that many slaves may be wire-ORed to the same MISO (or SISO) line.

The directions of the MOSI and MISO pins are also determined by the serial pin control (SPC[0]) bit.

17.3 SPI Registers—MBAR + 0x0F00

This section gives a detailed description of memory and accessible registers.

SPI uses the first 16 bits of 4 32-bit registers. These registers are located at an offset from MBAR of 0x0F00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0F00 + register address**

Reads from a non-implemented address returns zeros, writes to a non-implemented address has no effect.

Hyperlinks to the SPI registers are provided below:

- SPI Control Register 1 (0x0F00)
- SPI Control Register 2 (0x0F01)
- SPI Baud Rate Register (0x0F04)
- SPI Status Register (0x0F05)
- SPI Data Register (0x0F09)
- SPI Port Data Register (0x0F0D)
- SPI Data Direction Register (0x0F10)

17.3.1 SPI Control Register 1—MBAR + 0x0F00

Table 17-2. SPI Control Register 1

	msb 0	1	2	3	4	5	6	7 lsb
R	SPIE	SPE	SWOM (unused)	MSTR	CPOL	CPHA	SSOE	LSBFE
W								
RESET:	0	0	0	0	0	1	0	0

Bit	Name	Description
0	SPIE	SPI Interrupt Enable—bit enables SPI interrupts each time the SPIF or MODF status flag is set. 0 = SPI interrupts disabled 1 = SPI interrupts enabled
1	SPE	SPI System Enable—bit enables the SPI system and dedicates SPI port pins 3–0 to SPI functions. When SPE is clear, the SPI system is initialized, but in a low-power disabled state. 0 = SPI system is in a low-power, disabled state 1 = SPI port pins 3–0 are dedicated to SPI functions
2	SWOM	Unused
3	MSTR	SPI Master/Slave Mode Select bit 0 = Slave mode 1 = Master mode
4	CPOL	SPI Clock Polarity—bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values 0 = Active-high clocks selected; SCK idles low 1 = Active-low clocks selected; SCK idles high
5	CPHA	SPI Clock Phase—bit is used to shift the SCK serial clock. 0 = The first SCK edge is issued one-half cycle into the 8-cycle transfer operation 1 = The first SCK edge is issued at the beginning of the 8-cycle transfer operation

Bit	Name	Description
6	SSOE	Slave Select ($\overline{SS}$) Output Enable—bit is enabled only in master mode by asserting SSOE and SPIDDR bit 3 as shown in Table 17-3.
7	LSBFE	SPI LSB-First Enable—bit does not affect the position of the msb and lsb in the data register. Reads and writes of the data register always have the msb in bit 7. 0 = Data is transferred most significant bit first. 1 = Data is transferred least significant bit first.

Table 17-3. $\overline{SS}$ Input/Output Selection

SPIDDR Bit 3	SSOE	Master Mode	Slave Mode
0	0	$\overline{SS}$ input with MODF feature	$\overline{SS}$ input
0	1	General-purpose input	$\overline{SS}$ input
1	0	General-purpose output	$\overline{SS}$ input
1	1	$\overline{SS}$ output	$\overline{SS}$ input

17.3.2 SPI Control Register 2—MBAR + 0x0F01

Table 17-4. SPI Control Register 2

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved						SPISWAI	SPC0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:5	—	Reserved
6	SPISWAI	SPI Stop in Wait Mode—bit is used for power conservation while in wait mode. 0 = SPI clock operates normally in wait mode 1 = Stop SPI clock generation when in wait mode
7	SPC0	Serial Pin Control Bit 0—working with the MSTR control bit, this bit enables bidirectional pin configurations as shown in Table 17-5.

Table 17-5. Bidirectional Pin Configurations

Pin Mode		SPC0	MSTR	MISO[1]	MOSI[2]	SCK[3]	$\overline{SS}$[4]
A	Normal	0	0	Slave Out	Slave In	SCK in	$\overline{SS}$ In
B			1	Master In	Master Out	SCK out	$\overline{SS}$ I/O
C	Bidirectional	1	0	Slave I/O	GP I/O[5]	SCK in	$\overline{SS}$ In
D			1	GP I/O	Master I/O	SCK out	$\overline{SS}$ I/O

Note:

1. Slave output is enabled if SPIDDR bit 0 = 1, $\overline{SS}$ = 0, and MSTR = 0 (A, C).
2. Master output is enabled if SPIDDR bit 1 = 1 and MSTR = 1 (B, D).
3. SCK output is enabled if SPIDDR bit 2 = 1 and MSTR = 1 (B, D).
4. $\overline{SS}$ output is enabled if SPIDDR bit 3 = 1, SSOE = 1, and MSTR = 1 (B, D).
5. GP I/O = General-Purpose Input/Output.

17.3.3 SPI Baud Rate Register—MBAR + 0x0F04

Table 17-6. SPI Baud Rate Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved	SPPR2	SPPR1	SPPR0	Reserved	SPR2	SPR1	SPR0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	—	Reserved
1:3	SPPR[0:2]	SPI Baud Rate Preselection bits
4	—	Reserved
5:7	SPR[0:2]	SPI Baud Rate Selection bits

The SPI baud rate is derived from the IPB clock. The SPI module clock divisor is calculated as following:

$$\text{SPI module clock divisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

The SPI Baud rate is calculated as following:

$$\text{SPI Baud Rate} = \frac{\text{IPB CLock}}{\text{SPI module clock divisor}}$$

Table 17-7 shows some Baud rates derived from the possible IPB clock values:

Table 17-7. SPI Baud Rate Selection

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	SPI Module Clock Divisor	Baud Rate IPB 33.0 MHz	Baud Rate IPB 66.0 MHz	Baud Rate IPB 132.0 MHz
0	0	0	0	0	0	2	16.50 MHz	33.00 MHz	66.00 MHz
0	0	0	0	0	1	4	8.250 MHz	16.50 MHz	33.00 MHz
0	0	0	0	1	0	8	4.125 MHz	8.250 MHz	16.50 MHz
0	0	0	0	1	1	16	2.063 MHz	4.125 MHz	8.250 MHz
....									
1	1	1	1	0	0	256	128.9 KHz	257.8 KHz	512.6 KHz
1	1	1	1	0	1	512	64.45 KHz	128.9 KHz	257.8 KHz
1	1	1	1	1	0	1024	32.23 KHz	64.45 KHz	128.9 KHz
1	1	1	1	1	1	2048	16.1 KHz	32.23 KHz	64.45 KHz

17.3.4 SPI Status Register —MBAR + 0x0F05

Table 17-8. SPI Status Register

	msb 0	1	2	3	4	5	6	7 lsb
RESET:	0	0	0	0	0	0	0	0

Table 17-8. SPI Status Register

R W	SPIF	WCOL	Reserved	MODF	Reserved			
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	SPIF	SPI Interrupt flag—bit sets after 8th SCK cycle in a data transfer. Bit is cleared by an SPISR register read (with SPIF set) followed by an SPI data register read or write access. 0 = Transfer not yet complete 1 = New data copied to SPIDR
1	WCOL	Write Collision flag—bit indicates a serial transfer was in progress when the MCU tried to write new data into the SPI data register. The flag is cleared automatically by an SPI status register read (with WCOL set) followed by a SPI data register read or write access. 0 = Write collision did not occur 1 = Write collision occurred
2	—	Reserved
3	MODF	Mode Fault flag—bit sets if SS input goes low while SPI is configured as a master. Flag is cleared automatically by an SPI status register read (with MODF set) followed by a SPI control register 1 write. 0 = Mode fault did not occur 1 = Mode fault occurred
4:7	—	Reserved

17.3.5 SPI Data Register—MBAR + 0x0F09

Table 17-9. SPI Data Register

	msb 0	1	2	3	4	5	6	7 lsb
R W	D7	D6	D5	D4	D3	D2	D1	D0
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	D[0:7]	The SPI Data register is both an input and output register for SPI data. Attempts to write to this register while data transfers are in progress sets the WCOL flag and disables the attempted write. Review the WCOL bit description in Table 17-8 for more information. Reading data can occur anytime, from after SPIF is set, to before the end of the next transfer. If SPIF is not serviced by the end of the successive transfers, those data bytes are lost and data within SPIDR retains the first byte until SPIF is serviced.

17.3.6 SPI Port Data Register—MBAR + 0x0F0D

Table 17-10. SPI Port Data Register

	msb 0	1	2	3	4	5	6	7 lsb
RESET:	0	0	0	0	0	0	0	0

Table 17-10. SPI Port Data Register

R	D7	D6	D5	D4	D3	D2	D1	D0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7 (Note 1)	D[0:7]	SPI Port Data bits—data written to SPIPORT drives pins only when they are configured as general-purpose outputs. Reading an input (data direction bit is clear) returns the pin level. Reading an output (data direction bit is set) returns the pin driver input level. Writes do not change the state of pins 0:3 when pin is configured for SPI output. SPIPORT I/O function depends upon the state of the SPE bit in SPI control register 1 and the state of each associated data direction bit in SPIDDR.

Note:
1. Bits 4:7 do not drive output pins. When programmed as inputs (data direction bit is set), they return "0".

17.3.7 SPI Data Direction Register—MBAR + 0x0F10

Table 17-11. SPI Data Direction Register

	msb 0	1	2	3	4	5	6	7 lsb
R	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
W								
RESET:	0	0	0	0	0	0	0	0
Pin Function					SS	SCK	MOSI	MISO

Bit	Name	Description
0:7	DDR[0:7]	In SPI slave mode, SPIDDR bit 3 has no meaning or effect. In SPI master mode, SPIDDR bit 3 determines if SPI port pin 3 is: • an error-detect input to SPI • a general-purpose output • a slave select output line **Note: NOTE:** When SPI is Enabled, MISO, MOSI, and SCK are: • inputs if expected to be inputs, regardless of associated data direction bit state. • outputs if expected to be outputs, only if associated data direction bit is set. SPIDDR bits 0:7—SPI Port Data Direction Control bits 0 = Associated pin is an input 1 = Associated pin is an output

17.4 Functional Description

17.4.1 General

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI control register 1. While SPE is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select (SS)
- Serial clock (SCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

While SPE is clear, SPI port pins 3, 2, 1, and 0 are general-purpose I/O (input/output) pins controlled by the SPI port data direction register.

The main element of the SPI system is the SPI data register. The 8-bit data register in the master and the 8-bit data register in the slave are linked by the MOSI and MISO pins to form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master; data is exchanged between the master and the slave. Data written to the master SPI data register becomes the output data for the slave, and data read from the master SPI data register after a transfer operation is the input data from the slave.

A write to the SPI data register puts data into a serial shifte. When a transfer is complete, received data is moved into a receive data register. Data may be read from this double-buffered system any time before the next transfer is complete. This 8-bit data register acts as the SPI receive data register for reads and as the SPI transmit data register for writes. A single SPI register address is used for reading data from the read data buffer and for writing data to the shifter.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI control register 1 select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by shifting the clock by a half cycle or by not shifting the clock **(17.4.4 Transmission Formats)**.

The SPI can be configured to operate as a master or as a slave. When MSTR in SPI control register 1 is set, the master mode is selected; when the MSTR bit is clear, the slave mode is selected.

17.4.2 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI data register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

The SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI baud rate register control the baud rate generator and determine the speed of the shift register. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and MSTR control bits.

The SS pin is normally an input which should remain in the inactive high state. However, in the master mode, if the associated data direction bit (SPIDDR bit 3) is set, then the SS pin is a general-purpose output or the slave select output depending on the state of the SSOE bit.

General-purpose output (SSOE = 0) or slave select output (SSOE = 1) is specified by the SSOE bit in SPI control register 1. When this pin is being used as the output pin with SPIDDR bit 3 set, the mode error function for the master is disabled (MODF in the SPI status register).

The SS output becomes low during each transmission and is high when the SPI is in the idling state. If the SS input becomes low while the SPI is configured as a master, it indicates a mode fault error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. In this case, the SPI immediately clears the data direction bits associated with the MISO, MOSI (or MOMI), and SCK pins so that these pins become inputs. This mode fault error also clears the SPE and MSTR control bits and sets the mode fault (MODF) flag in the SPI status register. If the SPI interrupt enable bit (SPIE) is set when the MODF bit gets set, then an SPI interrupt sequence is also requested

When a write to the SPI data register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI control register 1 **(17.4.4 Transmission Formats)**.

17.4.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI control register 1 is clear. In slave mode, SCK is the SPI clock input from the master, and SS is the slave select input. Before a data transmission occurs, the SS pin of the slave SPI must be at logic 0. SS must remain low until the transmission is complete.

In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit in SPI control register 2 and the MSTR control bit. While in slave mode, the SS input controls the serial data output pin; if SS is high (not selected), the serial data output pin is high impedance, and, if SS is low the msb (most significant bit) in the SPI data register is driven out of the serial data output pin. Also, if the slave is not selected (SS is high), then the SCK input is ignored and no internal shifting of the SPI shift register takes place.

Although the SPI is capable of full-duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin

NOTE

When peripherals with full-duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI control register 1 is clear, odd numbered edges on the SCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB of the SPI shifter.

If the CPHA bit is set, even numbered edges on the SCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB of the SPI shifter.

When CPHA is set, the first edge is used to get the most significant data bit onto the serial data output pin. When CPHA is clear and the SS input is low (slave selected), the msb of the SPI data is driven out of the serial data input pin. After the eighth shift, the transfer is considered complete and the received data is transferred into the SPI data register. To indicate transfer is complete, the SPIF flag in the SPI status register is set.

17.4.4 Transmission Formats

During an SPI transmission, data is transmitted (shifted out serially) and received (shifted in serially) simultaneously. The serial clock (SCK) synchronizes shifting and sampling of the information on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. Optionally, on a master SPI device, the slave select line can be used to indicate multiple-master bus contention.

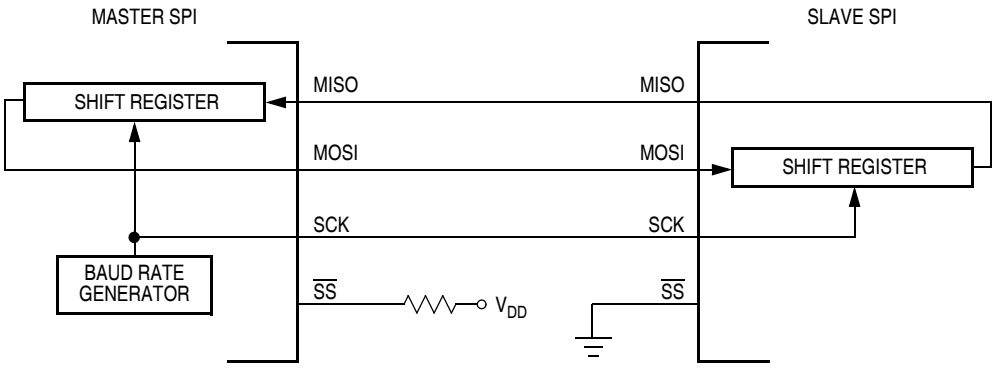


Figure 17-2. Master/Slave Transfer Block Diagram

17.4.4.1 Clock Phase and Polarity Controls

Using two bits in the SPI control register 1, software selects one of four combinations of serial clock phase and polarity.

The CPOL clock polarity control bit specifies an active high or low clock and has no significant effect on the transmission format.

The CPHA clock phase control bit selects one of two fundamentally different transmission formats.

Clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

17.4.4.2 CPHA = 0 Transfer Format

The first edge on the SCK line is used to clock the slave msb into the master and the master msb into the slave. In some peripherals, the msb of the slave's data is available at the slave data out pin as soon as the slave is selected. In this format, the first SCK edge is not issued until a half cycle into the 8-cycle transfer operation. The first edge of SCK is delayed a half cycle by clearing the CPHA bit.

The SCK output from the master remains in the inactive state for a half SCK period before the first edge appears. A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the LSB of the shifter.

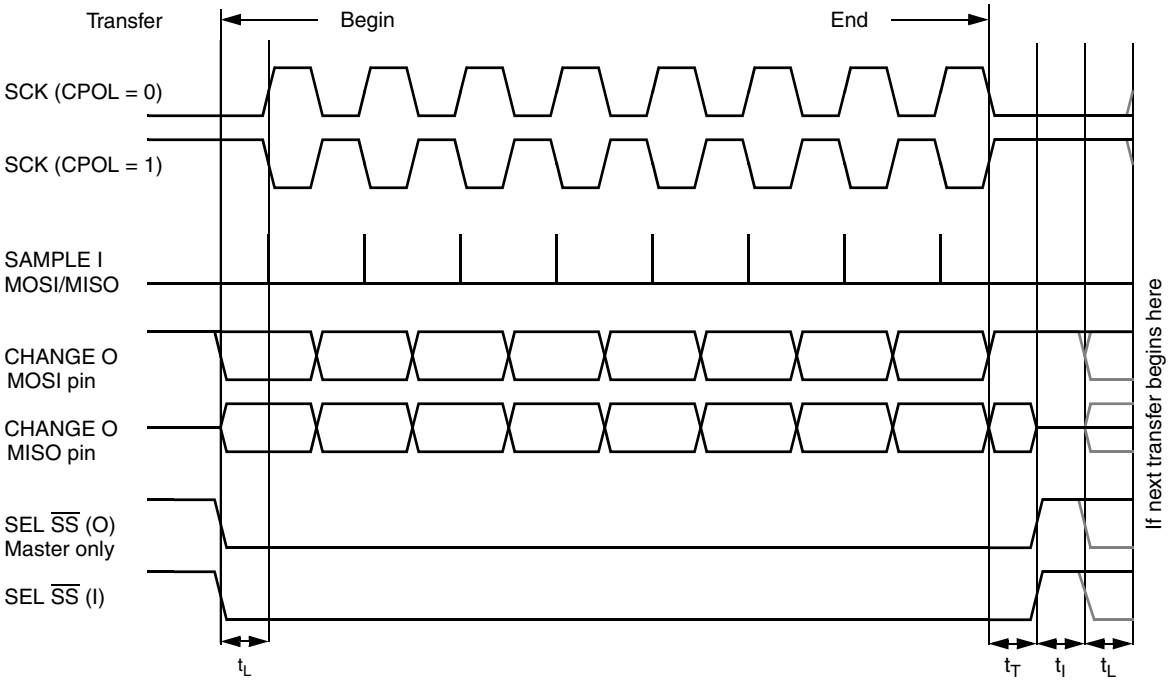
After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total of 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

Data reception is double buffered. Data is shifted serially into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After the 16th (last) SCK edge:

- Data that was previously in the master SPI data register should now be in the slave data register and the data that was in the slave data register should be in the master.
- The SPIF flag in the SPI status register is set and the clock is stopped, indicating that the transfer is complete.

Table 17-3 is a timing diagram of an SPI transfer where CPHA = 0. SCK waveforms are shown for CPOL = 0 and CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave and the MOSI signal is the output from the master. The SS pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.



t_L = Minimum leading time before the first SCK edge
t_T = Minimum trailing time after the last SCK edge
t_I = Minimum idling time between transfers (minimum $\overline{SS}$ high time)
t_L, t_T, and t_I are guaranteed for the master mode and required for the slave mode.

Figure 17-3. SPI Clock Format 0 (CPHA = 0)

In slave mode, if the SS line is not deasserted between the successive transmissions then the content of the SPI Data Register is not transmitted, instead the last received byte is transmitted. If the SS line is deasserted for at least minimum idle time (half SCK cycle) between successive transmissions content of the SPI Data Register is transmitted.

In master mode, with slave select output enabled the SS line is always deasserted and reasserted successive transfers for at least minimum idle time.

17.4.4.3 CPHA = 1 Transfer Format

Some peripherals require the first SCK edge before the msb becomes available at the data out pin; the second edge clocks data into the system. In this format, the first SCK edge is issued by setting the CPHA bit at the beginning of the 8-cycle transfer operation.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its most significant data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the LSB of the SPI shifter. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pins on the slave.

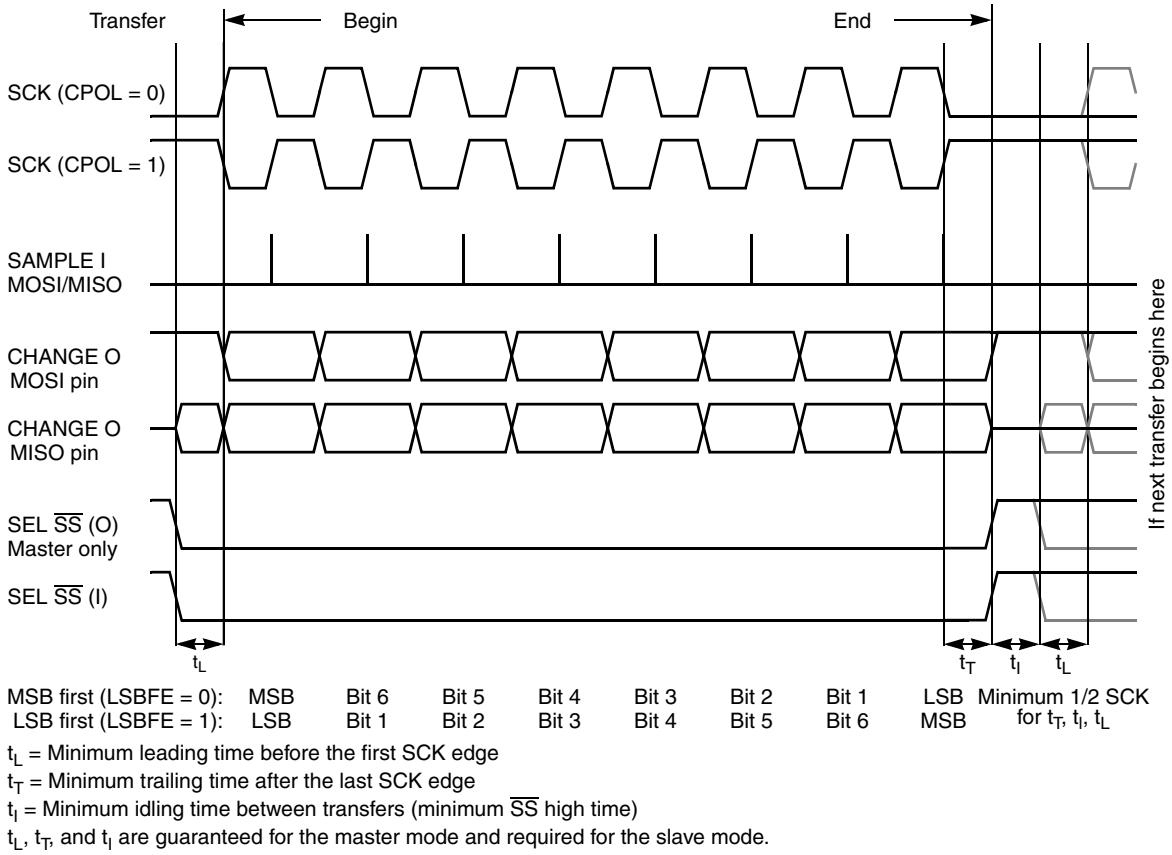
This process continues for a total of 16 edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

Data reception is double buffered; data is serially shifted into the SPI shift register during the transfer and is transferred to the parallel SPI data register after the last bit is shifted in.

After the 16th SCK edge:

- Data that was previously in the SPI data register of the master is now in the data register of the slave, and data that was in the data register of the slave is in the master.
- The SPIF flag bit in SPISR is set and the clock is stopped, indicating that the transfer is complete.

Table 17-4 shows two clocking variations for CPHA = 1. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO, and MOSI pins are connected directly between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The SS line is the slave select input to the slave. The SS pin of the master must be either high or reconfigured as a general-purpose output not affecting the SPI.



t_L = Minimum leading time before the first SCK edge
t_T = Minimum trailing time after the last SCK edge
t_I = Minimum idling time between transfers (minimum $\overline{SS}$ high time)
t_L, t_T, and t_I are guaranteed for the master mode and required for the slave mode.

Figure 17-4. SPI Clock Format 1 (CPHA = 1)

When CPHA = 1, the SS line can remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred in systems having a single fixed master and a single slave that drive the MISO data line.

The SPI interrupt request flag (SPIF) is common to both the master and slave modes. SPIF gets set after the last SCK cycle in a data transfer operation to indicate that the transfer is complete SPIF is cleared automatically when the SPI status register is read (with SPIF set) followed by a read or write to the SPI data register. If the SPIE bit is set when the SPIF flag is set, a hardware interrupt is requested.

A warning flag (WCOL) is set if a write to the SPI data register is attempted while a transfer is in progress. This is a conflict since the write would erroneously overwrite the current contents of the SPI serial shift register. If this situation arises, the write to the SPI data register is inhibited so as not to disturb the transfer in progress, and the WCOL flag is set to indicate the error. No interrupt is generated by WCOL because an interrupt comes at the end of the transfer that was in progress at the time of the error.

17.4.5 SPI Baud Rate Generation

Baud rate generation consists of a series of divider stages. Six bits in the SPI baud rate register (SPPR2, SPPR1, SPPR0, SPR2, SPR1, and SPR0) determine the divisor to the SPI module clock which results in the SPI baud rate.

The SPI clock rate is determined by the product of the value in the baud rate preselection bits (SPPR2–SPPR0) and the value in the baud rate selection bits (SPR2–SPR0). The module clock divisor equation is shown in **Table 17-5**.

When all bits are clear (the default condition), the SPI module clock is divided by 2. When the selection bits (SPR2–SPR0) are 001 and the preselection bits (SPPR2–SPPR0) are 000, the module clock divisor becomes 4. When the selection bits are 010, the module clock divisor becomes 8, etc.

When the preselection bits are 001, the divisor determined by the selection bits is multiplied by 2. When the preselection bits are 010, the divisor is multiplied by 3, etc. See Table 17-6 for baud rate calculations for all bit conditions, based on a 40 MHz SPI module clock. The two sets of selects allows the clock to be divided by a non-power of two to achieve other baud rates such as divide by 6, divide by 10, etc.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease IDD current

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)}$$

Figure 17-5. Baud Rate Divisor Equation

17.4.6 Special Features

17.4.6.1 SS Output

The SS output feature automatically drives the SS pin low during transmission to select external devices and drives it high during idle to deselect external devices. When SS output is selected, the SS output pin is connected to the SS input pin of the external device.

The SS output is available only in master mode during normal SPI operation by asserting SSOE and SPIDDR bit 3 as shown in Table 17-3.

The mode fault feature is disabled while SS output is enabled.

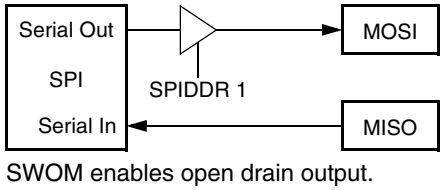
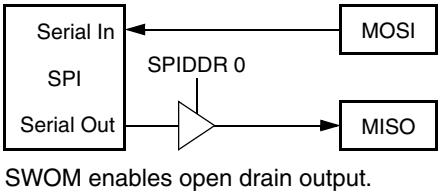
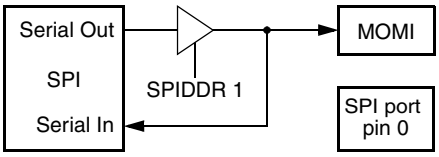
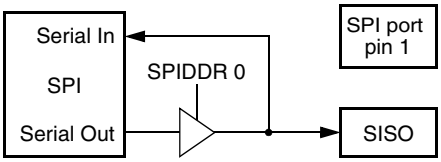
NOTE

Care must be taken when using the SS output feature in a multimaster system since the mode fault feature is not available for detecting system errors between masters.

17.4.6.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI control register 2 **(Table 17-12. Normal Mode and Bidirectional Mode)**. In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode The MISO pin in the master mode and MOSI pin in the slave mode become general-purpose I/O.

Table 17-12. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0	 SWOM enables open drain output.	 SWOM enables open drain output.
Bidirectional Mode SPC0 = 1	 SWOM enables open drain output. SPI port pin 0 becomes general-purpose I/O.	 SWOM enables open drain output. SPI port pin 1 becomes general-purpose I/O.

The direction of each serial I/O pin depends on the corresponding data direction register bit If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

If the pin is configured as an input, serial data from the shift register is discarded, but the external serial data through the pin is the serial input to the shift register.

The SCK is output for the master mode and input for the slave mode.

The SS is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SCK and SS functions; however, the SPIDDR bit 0 is not cleared by the mode fault error in the bidirectional mode.

17.4.7 Error Conditions

The SPI has two error conditions:

- Write collision error
- Mode fault error

17.4.7.1 Write Collision Error

The WCOL status flag in the SPI status register indicates that a serial transfer was in progress when the MCU tried to write new data into the SPI data register. The following list explains valid write times (reference **Table 17-3** and **Table 17-4** for definitions of tT and tI).

- In Master Mode, a valid write is defined as any write within tI (when SS is high).
- In Slave Phase 0, a valid write is defined as any write within tI (when SS is high).
- In Slave Phase 1, a valid write is defined as any write within tT or tI (after last SCK edge to when SS goes low) excluding the first two module clocks after the last SCK edge (beginning of tT is an illegal write).

A write during any other time will result in a WCOL error. The MCU write is disabled to avoid writing over the data being transmitted. No interrupt is generated because the error status flag can be read upon completion of the transfer that was in progress at the time of the error. This flag is cleared automatically by a read of the SPI status register (with WCOL set) followed by a read or write access to the SPI data register.

17.4.7.2 Mode Fault Error

If the SS input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. This condition is not permitted in normal operation; the MODF bit in the SPI status register is set automatically.

In the special case where SPIDDR bit 3 is set, the SS pin is either a general-purpose output pin or SS output pin rather than being dedicated as the SS input for the SPI system. In this special case, the mode error function is inhibited and MODF remains cleared.

When a mode fault error occurs, the SPE and MSTR bits are cleared and data direction bits controlling the output enable for the SCK, MISO, and MOSI (or MOMI) pins are cleared. This forces those pins to be high impedance inputs to avoid any possibility of conflict with another output driver.

If the mode fault error occurs in the bidirectional mode, the data direction bit associated with MISO (SISO) is not affected, since this bit is dedicated for general purpose.

This flag is cleared automatically by a read of the SPI status register (with MODF set) followed by a write to SPI control register 1.

17.4.8 Low Power Mode Options

17.4.8.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled. Since the SPI does not support other user modes (such as supervisory), any configurations to those modes will cause the SPI to act as it does in normal run mode. The supported modes are run, wait, and stop.

17.4.8.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI control register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SCK continues to be driven from the master. This keeps the slave synchronized to the master and the SCK.
  - If the master transmits several bytes while the slave is in wait mode, the slave will continue to send out bytes consistent with the its operation mode at the start of wait mode (i.e. If the slave is currently sending its SPIDR to the master, it will continue to send the same byte. Else if the slave is currently sending the last received byte from the master, it will continue to send each previous master byte).

NOTE

Care must be taken when expecting data from a master while the slave is in wait or stop mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e. a SPIF interrupt will not be generated until exiting stop or wait mode). Also, the byte from the shift register will not be copied into the SPIDR register until after the slave SPI has exited wait or stop mode. A SPIF flag and SPIDR copy is only generated if wait mode is entered or exited during a tranmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPIF nor a SPIDR copy will occur.

17.4.8.3 SPI in Stop Mode

Stop mode is dependent on the system. The SPI enters stop mode when the module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the MCU enters stop mode, the transmission is frozen until the MCU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is equivalent to the wait mode with the SPISWAI bit set except that the stop mode is not dependent on the SPISWAI bit.

17.4.9 SPI Interrupts

The Serial Peripheral Interface only originates interrupt requests. The following is a description of how the Serial Peripheral Interface makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

17.4.9.1 MODF Description

MODF occurs when the master detects an error on the SS pin. The master SPI must be configured for the MODF feature **(see Table 18-3. SS Input / Output Selection**). Once MODF is set, the current transfer is halted and the following bits are changed:

- SPE=0, The SPI automatically disables itself.
- MSTR=0, The master bit in SPICR1 resets.

The MODF interrupt is reflected in the status register MODF flag. Clearing the flag will also clear the interrupt. This interrupt will stay active while the MODF flag is set. MODF has an automatic clearing process which is described in **18.2.5.3.4 SPI Status Register**.

17.4.9.2 SPIF Description

SPIF occurs when the SPI receives/transmits the last SCK edge in a data transfer operation. Once SPIF is set, it does not clear until it is serviced. SPIF has an automatic clearing process which is described in **18.2.5.3.4 SPI Status Register** In the event that the SPIF is not serviced before the end of the next transfer (i.e. SPIF remains active throughout another transfer), the latter transfers will be ignored and no new data will be copied into the SPIDR

Chapter 18
Inter-Integrated Circuit (I^2C)

18.1 Overview

The following sections are contained in this document:

- Section 18.2, *I^2C Controller*
- Section 18.3, *I^2C Interface Registers*
- Section 18.4, *Initialization Sequence*
- Section 18.5, *Transfer Initiation and Interrupt*

The Inter-Integrated Circuit (I^2C) is a two-wire, bidirectional serial bus that provides a simple, efficient method for data exchange between devices. This two-wire bus minimizes the interconnection between devices.

The MPC5200 contains 2 identical and independent I^2C modules:

- I2C1 = MBAR + 0x3D00
- I2C2 = MBAR + 0x3D40

The I^2C module is connected to the IP bus, and the CommBus.

Each module operates up to 100 Kbps with a maximum bus load and timing. Both I^2C modules are capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading.

The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF. This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing more devices to be connected to the bus for further expansion and system development.

I^2C is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters attempt to control the bus simultaneously. This feature provides the capability for complex applications with multi-processor control. It may also be used for rapid testing and alignment of end products via external connections to an assembly-line computer.

18.1.1 Features

The I^2C module has the following key features:

- Compatible with I^2C bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven Byte-by-Byte data transfer
- Arbitration loss with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

Figure 18-1 shows a block diagram of the I^2C module.

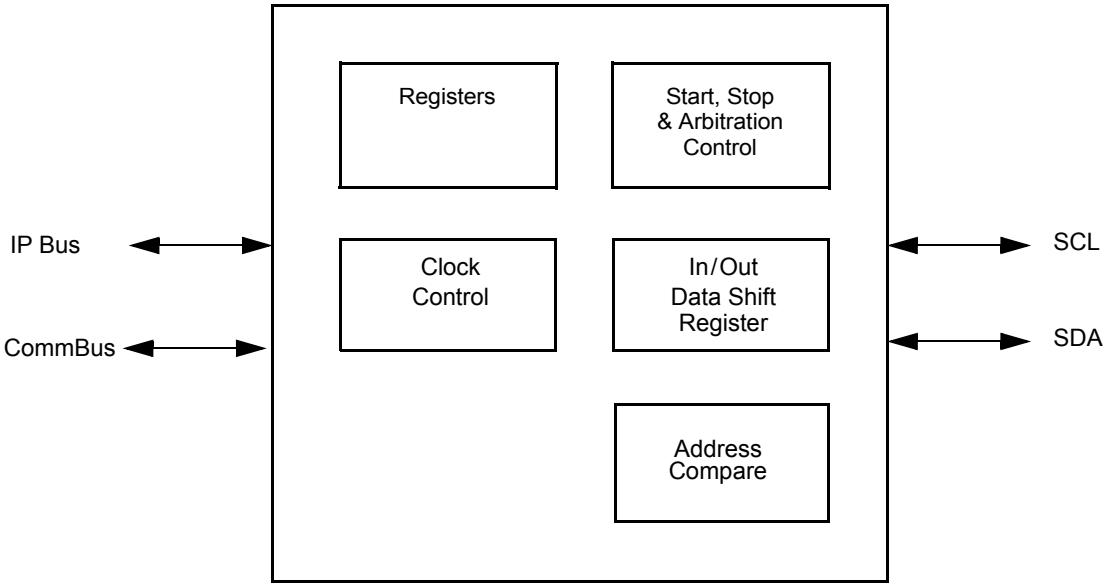


Figure 18-1. Block Diagram—I²C Module

18.2 I²C Controller

The I²C has simple bidirectional two-wire bus for efficient inter-IC control. The two wires, serial data line (SDA) and serial clock line (SCL), carry information between MPC5200 and other devices connected to the bus. Each device, including MPC5200, is recognized by a unique address, and can operate as either transmitter or receiver, depending on the function of the device. In addition to the transmitters and receivers, devices can be considered as masters or slaves. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave. See Table 18-1.

Table 18-1. I²C Terminology

Term	Description
Transmitter	Device that sends data to bus.
Receiver	Device that receives data from bus.
Master	Device that initiates transfer, generates SCL, and terminates transfer.
Slave	Device that is addressed by master.

Standard communication usually has 4 functional areas:

- START signal
- slave address transmission
- data transfer
- STOP signal

Activities listed above are briefly described in the sections below. Also see Figure 18-1.

18.2.1 START Signal

A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer and wakes up all slaves. Each data transfer may contain several data bytes.

When the bus is free, (i.e., no master device is engaging the bus) both SCL and SDA lines are at a logical high. A master initiates communication by sending a START signal.

18.2.2 STOP Signal

A STOP signal is defined as a low-to-high transition of SDA while SCL is high.

The master terminates communication by generating a STOP signal, which frees the bus. The master can generate a STOP even if the slave has generated an acknowledge, at which point the slave must release the bus.

The master can generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START.

18.2.2.1 Slave Address Transmission

The first byte of data transfer immediately after a START signal is the slave address transmitted by the master. This is a 7-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit tells the slave the desired direction of data transfer.

- 0 = Write transfer
- 1 = Read transfer

Only a slave with a calling address matching the address transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling SDA low at the 9th clock. See Figure 17 - 2.

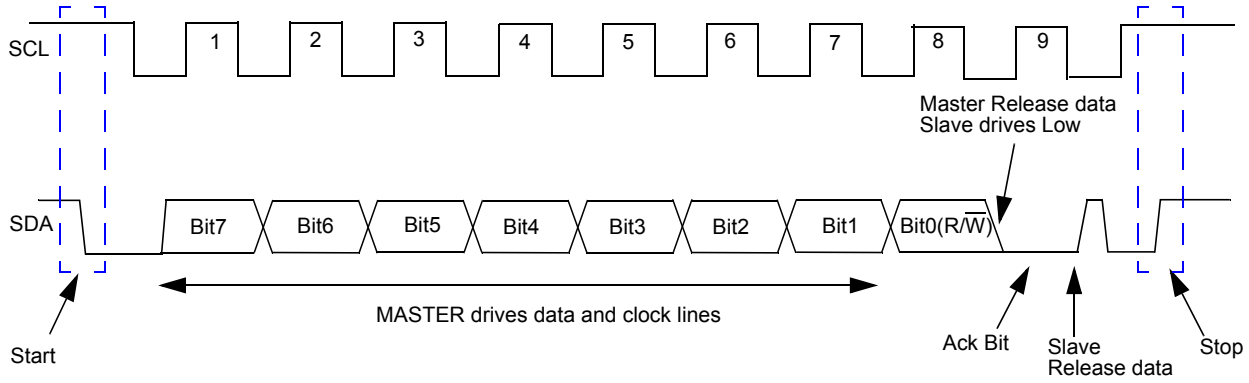


Figure 18-2. Timing Diagram—Start, Address Transfer and Stop Signal

18.2.2.2 Data Transfer

Data transfer proceeds Byte-by-Byte in a direction specified by the R/$\overline{W}$ bit sent by the calling master. Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high.

There is one clock pulse on SCL for each data bit. The MSB is transferred first. Each data byte must be followed by an acknowledge bit, which is signalled from the receiving device by pulling SDA low at the 9th clock. One complete data byte transfer needs nine clock pulses. See Figure 18-3.

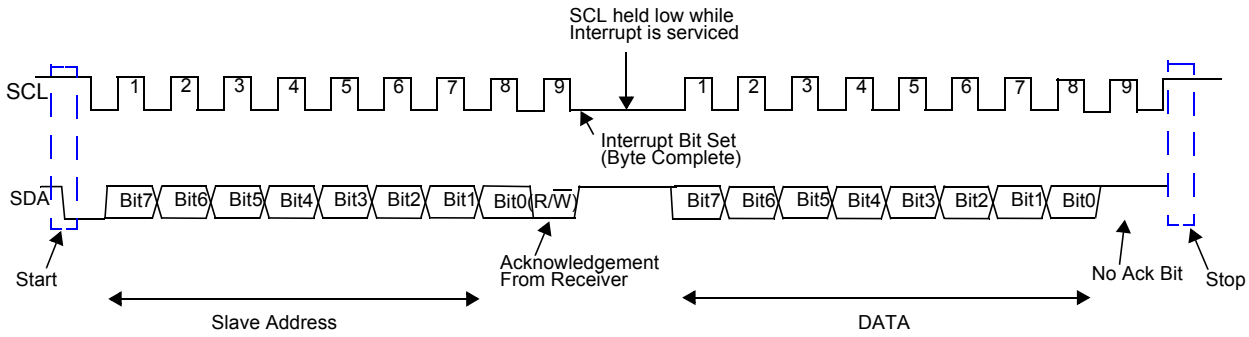


Figure 18-3. Timing Diagram—Data Transfer

18.2.2.3 Acknowledge

Figure 18-4 shows the transmitter releases the SDA line HIGH during the acknowledge clock pulse. The receiver pulls the SDA line low during the acknowledge clock pulse so that it remains stable LOW during the clock pulse high period.

If a slave-receiver does not acknowledge the byte transfer, SDA must be left HIGH by the slave. The master then generates a STOP condition to abort the transfer.

If a master-receiver does not acknowledge the slave transmitter after a byte transmission, it means End-Of-Data (EOD) to the slave. The slave then releases the SDA line for the master to generate a STOP or START signal.

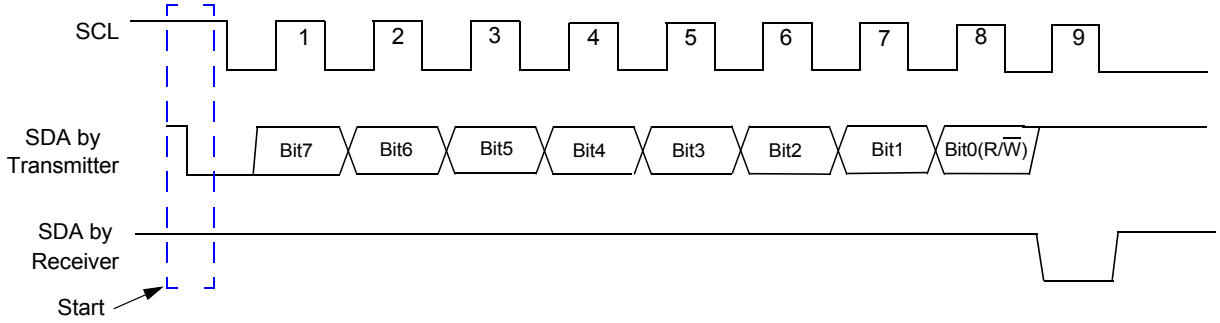


Figure 18-4. Timing Diagram—Receiver Acknowledgement

18.2.2.4 Repeated Start

A repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. The master uses this means to communicate with another slave or with the same slave in a different mode without releasing the bus.

Various combinations of read/write formats are possible. Figure 18-5 shows examples of:

- the master-transmitter transmitting to a slave-receiver. The transfer direction is not changed.
- the master reading a slave immediately after first byte. At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter.
- the START condition and slave address are both repeated using the repeated START signal. This communicates with same slave in a different mode without releasing the bus. The master transmits data to the slave first, then the master reads data from the slave by reversing the R/$\overline{W}$ bit.

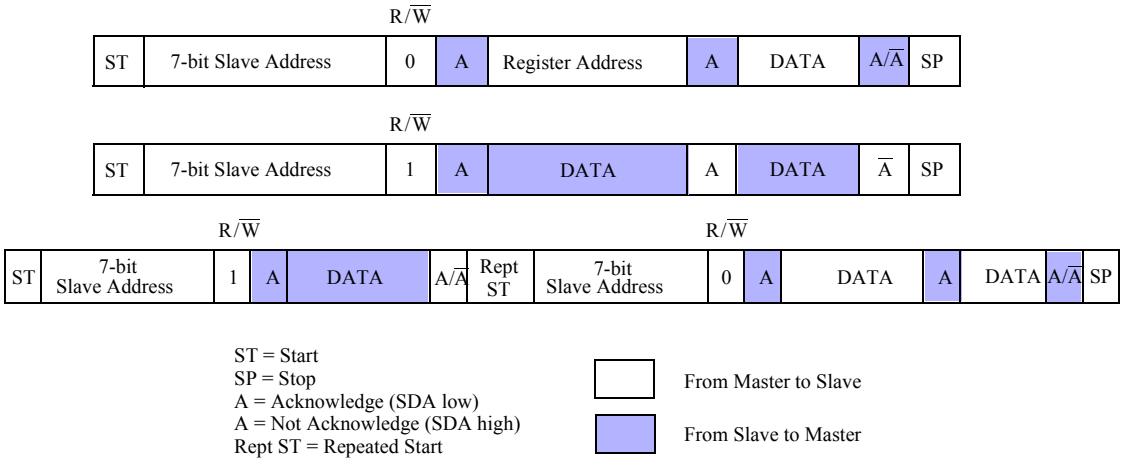


Figure 18-5. Data Transfer, Combined Format

18.2.2.5 Clock Synchronization and Arbitration

I^2C is a true multi-master bus; more than one master can be connected to the bus. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock.

Since wire-AND logic is used on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices start counting their low period. Once a device clock goes low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in this device clock may not change the SCL line state if another device clock is still within its low period. Therefore, the synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. See Figure 18-6.

When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. No difference exists between device clocks and the SCL line state. All devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

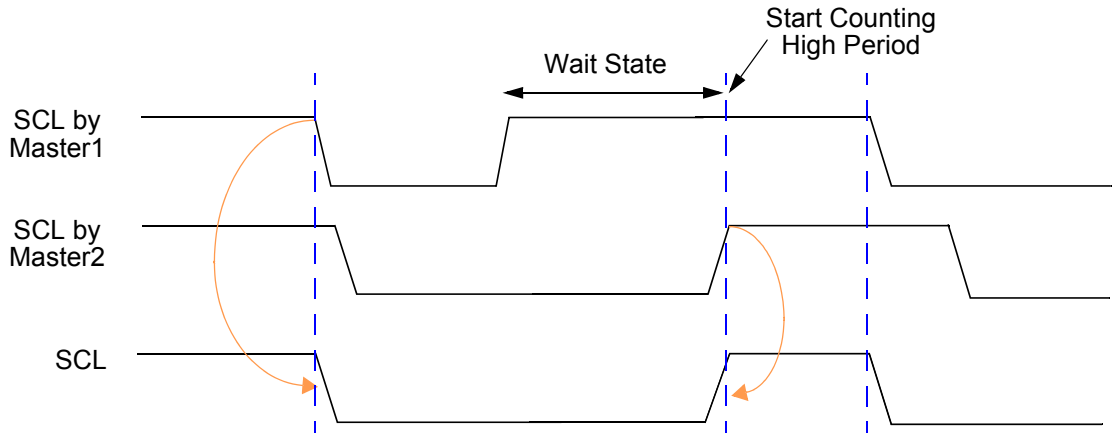


Figure 18-6. Timing Diagram—Clock Synchronization

A data arbitration procedure determines the relative priority of contending masters. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. Losing masters immediately switch to slave-receive mode and stop driving SDA output. In this case, transition from master to slave mode does not generate a STOP condition. A status bit is hardware set to indicate loss of arbitration. See Figure 18-7.

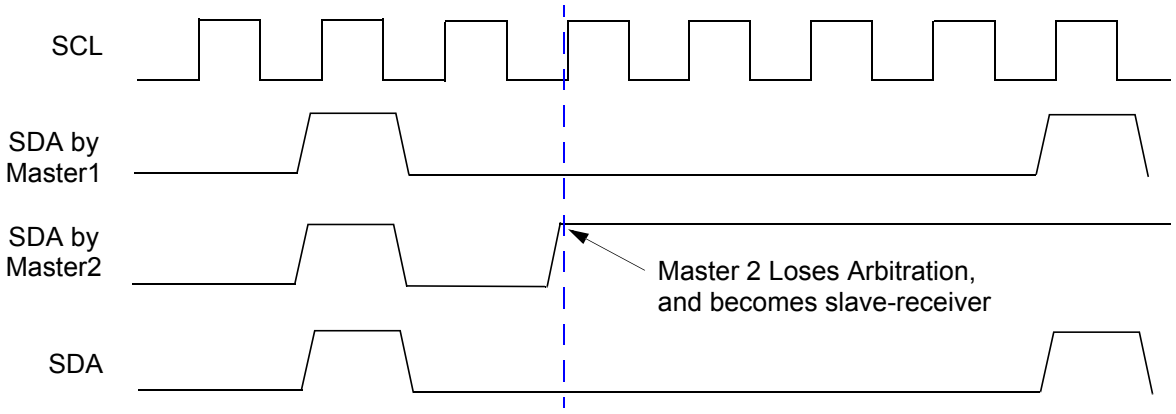


Figure 18-7. Timing Diagram—Arbitration Procedure

18.3 I²C Interface Registers

The I²C is controlled by 6 32-bit registers. The registers are located at an offset from MBAR of 0x3D00 (I2C1) and 0x3D40 (I2C2) . Register addresses are relative to this offset. There is one I²C Interrupt Control Register only for both I²C modules.

Hyperlinks to the I²C Interface registers are provided below:

- I²C Address Register (0x3D00)
- I²C Frequency Divider Register (0x3D04)
- I²C Control Register (0x3D08)
- I²C Status Register (0x3D0C)
- I²C Data I/O Register (0x3D10)
- I²C Interrupt Control Register (0x3D20)

18.3.1 I²C Address Register (MADR)—MBAR + 0x3D00

Table 18-2. I²C Address Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADR[7:1]							Reserved								
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	ADR[7:1]	Bits 0 to 6 contains the address I^2C responds to, when addressed as a slave. **Note:** This is not the address sent on the bus during address transfer.
7:31	—	Reserved

18.3.2 I^2C Frequency Divider Register (MFDR)—MBAR + 0x3D04

Table 18-3. I^2C Frequency Divider Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		FDR[5:0]						Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2:7	FDR[5:0]	This field is used to prescale the clock for bit-rate selection.
8:31	—	Reserved

The Frequency Divide register determines the SCL or serial bit-clock frequency. The bit-clock generator is implemented as a prescaled shift register. FDR bits are decoded to give the tap and prescale values as shown in Table 18-4.

- FDR[2:4] selects the prescaler divider.
- FDR[0:5] select the shift register tap point.

Table 18-4. I^2C Tap and Prescale Values

FDR 5,1,0 (bin)	SCL_Tap (clocks)	SDA_Tap (clocks)	FDR 4,3,2	scl2tap (clocks)	tap2tap (clocks)
000	9	3	0	4	1
001	10	3	1	4	2
010	12	4	10	6	4
011	15	4	11	6	8
100	5	1	100	14	16
101	6	1	101	30	32
110	7	2	110	62	64
111	8	2	111	126	128

Tap and prescale values are used to determine the SCL period and SDA hold time. Use the following equation to calculate the SCL period from FDR bits:

SCL Period = 2 x (scl2tap + [(SCL_Tap -1) x tap2tap] +2)

SDA hold time is defined as the delay from the SCL falling edge, to SDA changing. Use the following equation to generate the SDA hold value from FDR bits:

SDA Hold = scl2tap + [(SDA_Tap –1) x tap2tap] +3

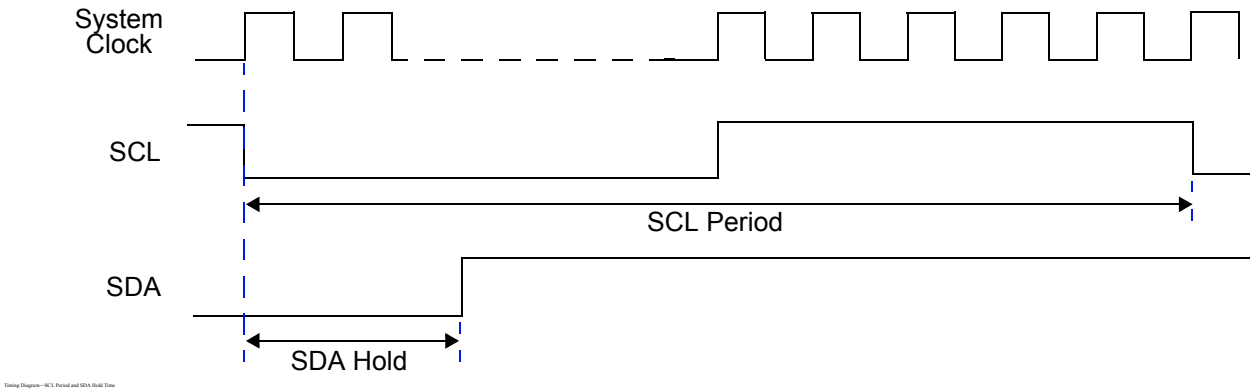
For example, if %000100 is selected for bits FDR5, FDR4, FDR3, FDR2, FDR1, FDR0] value, SCL period is:

SCL Period = 2 x (4 + [(9 –1) x 2] +2) = 44 clocks

The delay from the falling edge of SCL to SDA changing is:

SDA Hold = 4 + [(3 –1) x 2] +3 = 11 clocks wide

Serial bit clock frequency then equals system clock frequency divided by the SCL period.



Timing Diagram—SCL Period and SDA Hold Time

18.3.3 I²C Control Register (MCR)—MBAR + 0x3D08

Table 18-5. I²C Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	IEN	STA	TX	TXAK	RSTA	Reserved									
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	EN	I²C Enable—bit controls software reset of entire I²C module. If I²C module is enabled in the middle of a byte transfer, interface behaves as follows: • Slave mode ignores current bus transfer and starts operating when a subsequent start condition is detected. • Master mode is not aware if bus is busy. If a start cycle is initiated, current bus cycle may become corrupt. Ultimately this results in the current bus master or I²C module losing arbitration, after which bus operation returns to normal. 0 = module is reset and disabled. This is the Power-ON reset. When low the interface is held in reset, but registers can still be accessed. 1 = I²C module is enabled. Bit must be set before other CR bits have any effect.
1	IEN	I²C Interrupt Enable 0 = Interrupts from I²C module are disabled. This does not clear currently pending interrupt condition. 1 = Interrupts from I²C module are enabled. An I²C interrupt occurs, provided the status register IF bit is also set.
2	STA	Master/Slave mode select—bit clears on reset. • When bit changes from 0 to 1, a START signal is generated on the bus and master mode is selected. • When bit changes from 1 to 0, a STOP signal is generated and operation mode changes from master to slave. STA is cleared without generating a STOP signal when the master loses arbitration. 0 = Slave Mode 1 = Master Mode
3	TX	Transmit/Receive mode select—bit selects master/slave transfer direction. • When addressed as slave, software should set according to status register SRW bit. • When in master mode, bit should be set according to type of transfer required. For address cycles, bit is always high. 0 = Receive 1 = Transmit
4	TXAK	Transmit Acknowledge enable—bit specifies value driven to SDA during acknowledge cycles for both master and slave receivers. Values are used only when I²C is a receiver, not a transmitter. 0 = Acknowledge signal is sent to bus at 9th clock bit after receiving 1 Byte of data. 1 = No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5	RSTA	Repeat Start—writing 1 to this bit generates a repeated START condition on the bus, provided it is the current bus master. Bit is always read low. If the bus is owned by another master, attempting a repeated start at the wrong time results in loss of arbitration. 1 = Generate repeat start cycle
6:31	—	Reserved

18.3.4 I²C Status Register (MSR)—MBAR + 0x3D0C

Table 18-6. I²C Status Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CF	AAS	BB	AL	Rsvd	SRW	IF	RXAK	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	CF	Data transferring—bit clears while 1 Byte of data is being transferred. Bit is set by falling edge of 9th clock of a byte transfer. 0 = Transfer in progress 1 = Transfer complete
1	AAS	Addressed As Slave—bit sets when its own specific address (I²C Address Register) is matched with the calling address. The CPU is interrupted provided IEN is set. The CPU then needs to check the SRW bit and set its Tx/Rx mode accordingly. Writing to the I²C Control Register clears this bit. 0 = Not addressed 1 = Addressed as a slave
2	BB	Bus Busy—bit indicates bus status. When a START signal is detected, BB is set. If a STOP signal is detected, it is cleared. 0 = Bus is idle 1 = Bus is busy
3	AL	Arbitration Lost—bit is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: 1. SDA sampled low when master drives high during an address or data Tx cycle. 2. SDA sampled low when master drives high during a data Rx cycle acknowledge bit. 3. Start cycle is attempted when bus is busy. 4. A repeated start cycle is requested in slave mode. 5. Stop condition is detected when not requested by master. Software must clear bit by writing it low.
4	—	Reserved
5	SRW	Slave Read/Write—when set, bit indicates the R/W command bit value of the calling address sent from the master. **BE AWARE:** Bit is valid only when I²C is in slave mode, a complete address transfer occurred with an address match, and no other transfers were initiated. Checking this bit, the CPU can select slave Tx/Rx mode according to the master command. 0 = Slave receive, master writing to slave 1 = Slave transmit, master reading from slave
6	IF	I²C Interrupt—sets when an interrupt is pending. If IEN is set, a processor interrupt request is generated. IF sets when one of the following events occurs: 1. Complete 1 Byte transfer (set at falling edge of 9th clock). 2. A Rx calling address matches its own specific address in slave Rx mode. 3. Arbitration is lost. This bit must be cleared by software writing it low in the interrupt routine.
7	RXAK	Receive Acknowledge—SDA value during the bus cycle acknowledge bit. • If bit is low, it indicates an acknowledge signal was received after completion of 8 bits of data transmission on the bus. • If bit is high, it means no acknowledge signal is detected at the 9th clock. 0 = Acknowledge received 1 = No acknowledge received
8:31	—	Reserved
Note: This status register is read-only with the exception of bit6 (IF) and bit3 (AL), which are software clearable.		

18.3.5 I²C Data I/O Register (MDR)—MBAR+ x3D10

Table 18-7. I²C Data I/O Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	D7	D6	D5	D4	D3	D2	D1	D0	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	D[7:0]	In Master Transmit Mode—when data is written to this register, a data transfer is initiated. The most significant bit is sent first. **Note:** In this mode, the first data byte written to DR. Assertion of STA is used for the address transfer and should be comprise of the calling address (in position D[7]:D[1]) concatenated with the required R/$\overline{W}$ bit (in position D0). In Master Receive Mode—reading this register initiates next byte data receiving. In Slave Mode—the same functions are available after an address match occurs.
8:31	—	Reserved

18.3.6 I²C Interrupt Control Register—MBAR + 0x3D20

Table 18-8. I²C Interrupt Control Register

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BNBE2	TE2	RE2	IE2	BNBE1	TE1	RE1	IE1	Reserved							
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 lsb
R	Reserved															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	BNBE2	Bus Not Busy Enable 2—lets module 2 generate an interrupt when the bus is not busy. BNBE2 indicates an idle condition. To clear the interrupt, software must write 0 to the bit position. Reset condition disables BNBE2.
1	TE2	Transmit Enable 2—routes the interrupt for module 2 to the TX requestor at SDMA. Clear by writing 0 to this bit position. Reset condition disables TE2.

Bit	Name	Description
2	RE2	Receive Enable 2—routes the interrupt for module 2 to the RX requestor at SDMA. Clear by writing 0 to this bit position. Reset condition disables RE2.
3	IE2	Interrupt Enable 2—routes the interrupt for module 2 to the CPU. Clear by writing 0 to this bit position. Reset condition enables IE2.
4	BNBE1	Bus Not Busy Enable 1—lets module 1 generate an interrupt when the bus is not busy. BNBE1 indicates an idle condition. To clear the interrupt, software must write 0 to the bit position. Reset condition disables this bit.
5	TE1	Transmit Enable 1—routes the interrupt for module 1 to the TX requestor at SDMA. Clear by writing 0 to this bit position. Reset condition disables TE1.
6	RE1	Receive Enable 1—routes the interrupt for module 1 to the RX requestor at SDMA. Clear by writing a 0 to this bit position. Reset condition disables RE1.
7	IE1	Interrupt Enable 1—routes the interrupt for module 1 to the CPU. Clear by writing 0 to this bit position. Reset condition enables IE1.
8:31	—	Reserved

The Interrupt Control register is common to both MPC5200 I^2C modules. Each module generates an internal interrupt that can be routed as follows:

- To the CPU interrupt, if IE is set to 1.
- To the TX requestor at SDMA, if TE is set to 1.
- To the RX requestor at SDMA, if RE is set to 1.

Typically, only one (or none) of the above destinations would be specified. Although, it may be useful to send an interrupt to both the CPU and SDMA. Selecting between TX and RX is based on whether the module is:

- sending data (master or slave TX)
- receiving data (master or slave RX)

Individual requests trigger different SDMA tasks. Reset condition is, IE set and all other enable bits clear.

The BNBE bit lets the module generate an interrupt when the bus becomes not-busy. This implies receipt of a STOP condition, for which the module normally does not generate an interrupt. Because bus-not-busy is an idle condition, it is necessary for software responding to this interrupt to clear the BNBE bit to clear the interrupt condition. Otherwise, the interrupt condition persists until another I^2C transaction is initiated.

18.4 Initialization Sequence

Reset puts the I^2C Control register to its default status. Before the interface can be used to transfer serial data, the following initialization procedure must be done:

Step 1. Update the Frequency Divider register and select the required division ratio to obtain the SCL frequency from the system clock.
Step 2. Update the I^2C Address register to define a slave address.
Step 3. Set the Control register EN bit to enable the I^2C interface system.
Step 4. Modify the Control register bits to select master/slave mode, transmit/receive mode and interrupt enable or not.

18.5 Transfer Initiation and Interrupt

In master transmit mode, a data transfer is initiated when data is written to the DATA register. The most significant bit is sent first.

In master receive mode, reading this register initiates next byte data receiving.

In slave mode, the same functions as are available after an address match occurs. Data transfer is initiated by:

- writing to the DATA register for slave transmits, or
- a dummy reading from the DATA register in slave receive mode occurs.

The I^2C interrupt STATUS register bit is set when an interrupt is pending. If the CONTROL register interrupt enable bit is set, setting the I^2C interrupt STATUS register bit causes a processor interrupt request. The interrupt bit sets when one of the following events occurs:

- A complete 1 Byte transfer (set at falling edge of 9th clock) occurs.
- A receive calling address matches its own specific address in slave receive mode.
- Arbitration is lost.

18.5.1 Post-Transfer Software Response

In the interrupt service routine, software must clear the IF status bit first. The CF status bit will be cleared automatically by reading from the Data I/O Register (MBDR) in receive mode or writing to MBDR in transmit mode.

Software may service the bus I/O in the main program by monitoring the IF status bit if the interrupt function is disabled. Polling should monitor the IF status bit rather than the CF bit since their operation is different when arbitration is lost.

When an interrupt occurs at the end of the address cycle the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in the DATA register, then the TX control bit should be toggled at this stage.

During slave mode address cycles (AAS = 1) the SRW bit in the STATUS register is read to determine the direction of the subsequent transfer and the TX control bit is programmed accordingly. For slave mode, data cycles (AAS = 0) the SRW bit is not valid, therefore the TX bit in the control register should be read to determine the direction of the current transfer.

18.5.2 Slave Mode

In the slave interrupt service routine, the AAS bit should be tested to determine if a calling of its own address was received. If AAS is set, software should set the Tx/Rx mode select bit (Control register Tx bit) according to the R/$\overline{W}$ command bit (SRW). Writing to the CONTROL register automatically clears AAS. A data transfer can then be initiated by writing information to the DATA register for slave transmits, or dummy reading from the DATA register, in slave receive mode. The slave drives SCL low between byte transfers. SCL is released when the DATA register is accessed in the required mode.

In slave transmitter routine, RXAK must be tested before transmitting the next data byte. Setting RXAK means an end of data signal from the master receiver. After which, software causes a switch from transmitter mode to receiver mode. A dummy read then releases the SCL line letting the master generate a STOP signal.

Chapter 19
Controller Area Network (MSCAN)

19.1 Overview

The following sections are contained in this document:

- Section 19.1, *Overview*
- Section 19.2, *Features*
- Section 19.3, *External Signals*
- Section 19.4, *CAN System*
- Section 19.5, *Memory Map / Register Definition*
- Section 19.6, *Programmer's Model of Message Storage*
- Section 19.7, *Functional Description*

The MPC5200 contains 2 identical and independent MSCAN Controller :

- MSCAN1 = MBAR + 0x0900
- MSCAN2 = MBAR + 0x0980

The Motorola Scalable Controller Area Network (MSCAN) definition is based on the MSCAN12 definition which is the specific implementation of the Motorola Scalable CAN concept targeted for the Freescale Semiconductor, Inc. (formerly Motorola) MC68HC12 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991. For users to fully understand the MSCAN specification, it is recommended that the Bosch specification be read first to familiarize the reader with the terms and concepts contained within this document.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN utilizes an advanced buffer arrangement resulting in a predictable real-time behavior and simplifies the application software.

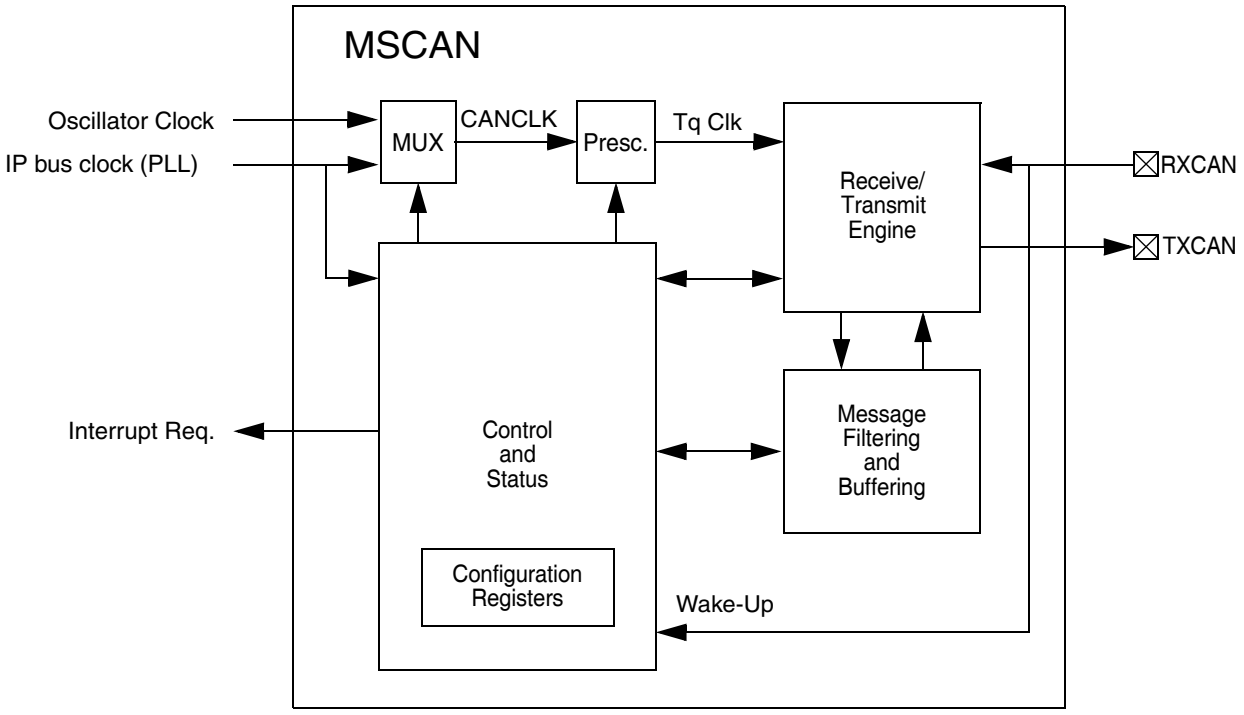


Figure 19-1. MSCAN Block Diagram

19.2 Features

The basic features of the MSCAN are as follows:

- Implementation of the CAN protocol - Version 2.0A/B
  - — Standard and extended data frames
  - — 0 - 8 bytes data length
  - — Programmable bit rate up to 1 Mbps (Depending on the actual bit timing and the clock jitter of the PLL)
  - — Support for remote frames
  - — 4 receive buffers with FIFO storage scheme
- 3 transmit buffers with internal prioritization using a “local priority” concept
- Flexible maskable identifier filter supports two full size extended identifier filters (two 32-bit) or four 16-bit filters or eight 8-bit filters
- Programmable wake-up functionality
- Programmable loop back mode supports self-test operation
- Programmable listen-only mode for monitoring of CAN bus
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (Warning, Error Passive, Bus-Off)
- Programmable MSCAN clock source either IP bus clock or Oscillator clock
- Internal timer for time-stamping of received and transmitted messages
- Three low power modes: Sleep, Power Down and MSCAN Enable
- Global initialization of configuration registers

19.3 External Signals

The MSCAN uses two external pins. In the MPC5200 the MSCAN pins are shared with other funtionality and can be available at two different groups of pins. The configuration of the pin-muxing is controlled by the Port Configuration Register, see Section 7.

19.3.1 RXCAN — CAN Receiver Input Pin

RXCAN is the MSCAN receiver input pin.

19.3.2 TXCAN — CAN Transmitter Output Pin

TXCAN is the MSCAN transmitter output pin. The TXCAN output pin represents the logic level on the CAN bus:

0 = Dominant state

1 = Recessive state

19.4 CAN System

A typical CAN system with MSCAN is shown in Figure 19-2. Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN bus and has current protection against defected CAN or defected stations.

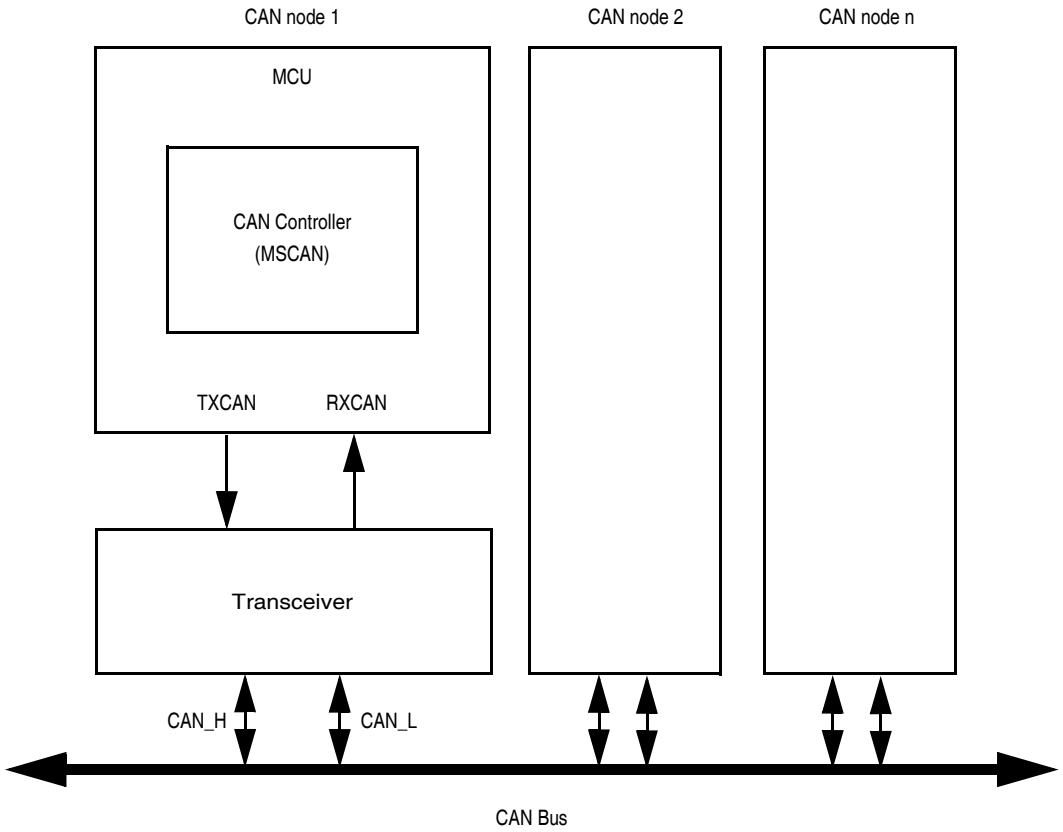


Figure 19-2. The CAN System

19.5 Memory Map / Register Definition

The MPC5200 contains 2 independent MSCAN Controller :

- MSCAN1 = MBAR + 0x0900
- MSCAN2 = MBAR + 0x0980

19.5.1 Module Memory Map

Table 19-1 and Table 19-2 give an overview on all registers and their individual bits in the MSCAN memory map. The *register address* results from the addition of *base address* and *address offset*. The *base address* is determined at the MPC5200 MCU level. The *address offset* is defined at the module level.

The MSCAN occupies 64 bytes in the memory space. The base address of the MSCAN module is determined at the MPC5200 MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset.

Table 19-1. MSCAN Register Organization

Address Offset	
$__00	CONTROL REGISTERS 12 BYTES
$__15	
$__18	RESERVED 2 BYTES
$__19	
$__1C	ERROR COUNTERS 2 BYTES
$__1D	

Table 19-1. MSCAN Register Organization (continued)

$__20 $__3F	IDENTIFIER FILTER 16 BYTES
$__40 $__5F	RECEIVE BUFFER 16 BYTES (Window)
$__60 $__7F	TRANSMIT BUFFER 16 BYTES (Window)

Table 19-1 shows the individual registers associated with the MSCAN and their relative offset from the base address. The detailed register descriptions follow in the order they appear in the register map (see Table 19-2).

Table 19-2. Module Memory Map

Address	Use	Access
$__00	MSCAN Control Register 0 (CANCTL0)	R/W[a]
$__01	MSCAN Control Register 1 (CANCTL1)	R/W[1]
$__04	MSCAN Bus Timing Register 0 (CANBTR0)	R/W
$__05	MSCAN Bus Timing Register 1 (CANBTR1)	R/W
$__08	MSCAN Receiver Flag Register (CANRFLG)	R/W[1]
$__09	MSCAN Receiver Interrupt Enable Register (CANRIER)	R/W
$__0C	MSCAN Transmitter Flag Register (CANTFLG)	R/W[1]
$__0D	MSCAN Transmitter Interrupt Enable Register (CANTIER)	R/W[1]
$__10	MSCAN Transmitter Message Abort Control (CANTARQ)	R/W[1]
$__11	MSCAN Transmitter Message Abort Control (CANTAAK)	R
$__14	MSCAN Transmit Buffer Selection (CANTBSEL)	R/W[1]
$__15	MSCAN Identifier Acceptance Control Register (CANIDAC)	R/W[1]
$__18 -$__19	RESERVED	
$__1C	MSCAN Receive Error Counter Register (CANRXERR)	R
$__1D	MSCAN Transmit Error Counter Register (CANTXERR)	R
$__20	MSCAN Identifier Acceptance Register 0 (CANIDAR0)	R/W
$__21	MSCAN Identifier Acceptance Register 1 (CANIDAR1)	R/W
$__24	MSCAN Identifier Acceptance Register 2 (CANIDAR2)	R/W
$__25	MSCAN Identifier Acceptance Register 3 (CANIDAR3)	R/W
$__28	MSCAN Identifier Mask Register 0 (CANIDMR0)	R/W
$__29	MSCAN Identifier Mask Register 1 (CANIDMR1)	R/W
$__2C	MSCAN Identifier Mask Register 2 (CANIDMR2)	R/W
$__2D	MSCAN Identifier Mask Register 3 (CANIDMR3)	R/W
$__30	MSCAN Identifier Acceptance Register 4 (CANIDAR4)	R/W
$__31	MSCAN Identifier Acceptance Register 5 (CANIDAR5)	R/W
$__34	MSCAN Identifier Acceptance Register 6 (CANIDAR6)	R/W

Table 19-2. Module Memory Map (continued)

$__35	MSCAN Identifier Acceptance Register 7 (CANIDAR7)	R/W
$__38	MSCAN Identifier Mask Register 4 (CANIDMR4)	R/W
$__39	MSCAN Identifier Mask Register 5 (CANIDMR5)	R/W
$__3C	MSCAN Identifier 6 Mask Register 6 (CANIDMR6)	R/W
$__3D	MSCAN Identifier Mask Register 7 (CANIDMR7)	R/W
$__40 -$__5F	Foreground Receive Buffer (CANRXFG)	R[b]
$__60 -$__7F	Foreground Transmit Buffer (CANTXFG)	R[2]/W

a Refer to detailed register description for write access restrictions on per bit basis.

b Reserved bits and unused bits within the TX- & RX-Buffers (CANTXFG, CANRXFG) will be read as “X”, because of RAM based implementation.

19.5.2 Register Descriptions

This section describes in detail all the registers and register bits in the MSCAN module. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. All bits of all registers in this module are completely synchronous to internal clocks during a register read.

The registers are located at an offset from MBAR of 0x0900 (MSCAN1) and 0x0980 (MSCAN2) . Register addresses are relative to this offset.

19.5.3 MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900

Table 19-3. MSCAN Control Register 0

	msb 0	1	2	3	4	5	6	7 lsb
R	RXFRM	RXACT	CSWAI	SYNCH	TIME	WUPE	SLPRQ	INITRQ
W								
RESET:	0	0	0	0	0	0	0	1

The MSCAN Control Register 0, CANCTL0, provides for various control of the MSCAN Module.

NOTE: The MSCAN Control Register 0, except the WUPE, INITRQ and SLPRQ bits, is held in the reset state when the Initialization Mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the Initialization Mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when out of Initialization; exceptions are bits RXACT and SYNCH which are read-only and bit RXFRM which is set by the module. A write of ‘1’ to the RXFRM register clears the flag and a write of ‘0’ is ignored.

Bit	Name	Description
0	RXFRM	Received Frame—flag bit is read and clear only. It is set when a receiver has received a valid message correctly, independently of the filter configuration. Once set, it remains set until cleared by software or reset. Clear by writing 1 to the bit. This bit is not valid in loop-back mode. 0 = No valid message was received since last clearing this flag 1 = A valid message was received since last clearing of this flag
1	RXACT	Receiver Active Status—flag bit indicates MSCAN is receiving a message. The flag is controlled by the receiver front end. This bit is not valid in loop-back mode. 0 = MSCAN is transmitting or idle1 1 = MSCAN is receiving a message (including when arbitration is lost)
2	CSWAI	CAN Stops in Wait Mode—enabling this bit allows lower power consumption in wait mode by disabling all clocks at the bus interface to the MSCAN module. 0 = Module is not affected during WAIT mode 1 = Module ceases to be clocked during WAIT mode
3	SYNCH	Synchronized Status—flag bit indicates whether MSCAN is synchronized to the CAN bus and, as such, can participate in the communication process. It is set and cleared by MSCAN. 0 = MSCAN is not synchronized to the CAN bus 1 = MSCAN is synchronized to the CAN bus
4	TIME	Timer Enable—bit activates an internal 16-bit wide free running timer, clocked by the bit-clock. If timer is enabled, a 16-bit time stamp is assigned to each transmitted/received message within the active Tx/Rx buffer. As soon as a message is acknowledged on CAN, the time stamp is written to the highest bytes ($_E, $_F) in the appropriate buffer. The internal timer is reset (all bits set to “0”) when Initialization Mode is active. 0 = Disable internal MSCAN timer 1 = Enable internal MSCAN timer
5	WUPE	WakeUp Enable—bit lets MSCAN restart when being locked in idle state during sleep mode and traffic on CAN is detected. 0 = WakeUp disabled—MSCAN ignores traffic on CAN 1 = WakeUp enabled—MSCAN is able to restart
6	SLPRQ	Sleep Mode Request—bit requests MSCAN enter sleep mode, an internal power saving mode. If a CAN message transfer is occurring when receiving this request, MSCAN waits until end of current message before entering sleep mode. The module indicates entry to Sleep Mode by setting SLPAK=1. MSCAN Control 1 Register (CANCTL1). Sleep mode is active until the CPU clears SLPRQ or, depending on the WUPE bit setting, MSCAN detects CAN bus activity and clears SLPRQ. 0 = Running—MSCAN functions normally 1 = Sleep Mode Request—MSCAN locks in idle state
7	INITRQ	Initialization Mode Request—When the CPU sets this bit, MSCAN skips to initialization mode. Any ongoing transmission or reception is aborted and bus synchronization lost. The module indicates entry to initialization mode by setting INITAK=1

19.5.4 MSCAN Control Register 1 (CANCTL1)—MBAR + 0x0901

Table 19-4. MSCAN Control Register 1

	msb 0	1	2	3	4	5	6	7 lsb
R	CANE	CLKSRC	LOOPB	LISTEN	Rsvd	WUPM	SLPAK	INITAK
W								
RESET:	0	0	0	1	0	0	0	1

The MSCAN Control Register 1 provides for various control and handshake status information of the MSCAN module.

READ: Anytime

WRITE: Anytime when INITRQ = 1 and INITAK = 1, except CANE which is write once in normal modes and anytime in special modes when the MSCAN is in Initialization Mode (INITRQ = 1 and INITAK = 1).

Bit	Name	Description
0	CANE	MSCAN Enable 0 = MSCAN module is disabled 1 = MSCAN module is enabled
1	CLKSRC	MSCAN Clock Source—bit defines MSCAN module clock source (only for systems with a system clock generation module. 0 = MSCAN clock source is the IP bus clock (IP CLK) 1 = MSCAN clock source is the oscillator clock (SYS_XTAL_IN) NOTE: Both MSCAN modules can have only the same selected clock source.To select the oscillator clock the CLKSRC bit in the CANCTL1 register must be set in MSCAN1 OR/AND in MSCAN2.
2	LOOPB	Loop-Back Self-Test Mode—when bit is set, MSCAN does an internal loop-back that can be used for self test operation. Tx bit-stream output feeds back to receiver internally. RxCAN input pin is ignored and TxCAN output goes to recessive state (logic ‘1’). MSCAN behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state, MSCAN ignores bit sent during ACK slot in CAN frame acknowledge field to ensure proper reception of its own message. Both Tx and Rx interrupts are generated.
3	LISTEN	Listen-Only Mode—bit configures MSCAN as bus monitor1. When bit is set, all valid CAN messages with matching ID are received, but no acknowledgement or error frames are sent out. In addition, error counters are frozen. Listen-only mode supports applications that require “hot plugging” or throughput analysis. MSCAN is unable to transmit any messages, when listen-only mode is active. 0 = normal operation 1 = Listen Only Mode activated
4	—	Reserved
5	WUPM	WakeUp Mode—bit defines whether the integrated low-pass filter is applied to protect the MSCAN from spurious WakeUp. 0 = MSCAN wakes-up the CPU after any recessive to dominant edge on the CAN bus and WUPE=1 in CANCTL0 1 = MSCAN wakes-up the CPU only in case of a dominant pulse on the bus which has a length of Twup and WUPE=1 in CANCTL0
6	SLPAK	Sleep Mode Acknowledge—flag indicates whether MSCAN module has entered sleep mode. It is used as a handshake flag for SLPRQ sleep mode request. Sleep mode is active when INITRQ=1 and INITAK=1. Depending on the WUPE bit setting, MSCAN clears the flag if it detects bus activity on CAN while in Sleep Mode. 0 = Running—MSCAN operates normally 1 = Sleep Mode Active—MSCAN has entered Sleep Mode
7	INITAK	Initialization Mode Acknowledge—flag indicates whether MSCAN module is in initialization mode. It is used as a handshake flag for the INITRQ initialization mode request. Initialization mode is active when INITRQ=1 and INITAK=1. The registers CANCTL1, CANBTR0, CANBTR1, CANIDAC, CANIDAR0-7, CANIDMR0-7 can only be written by the CPU when the MSCAN is in initialization mode. 0 = Running – The MSCAN operates normally 1 = Initialization Mode Active – The MSCAN has entered initialization mode

19.5.5 MSCAN Bus Timing Register 0 (CANBTR0)—MBAR + 0x0904

Table 19-5. MSCAN Bus Timing Register 0

	msb 0	1	2	3	4	5	6	7 lsb
R	SJW[1:0]		BRP[5:0]					
W								
RESET:	0	0	0	0	0	0	0	0

The MSCAN Bus Timing Register 0 provides for various bus timing control of the MSCAN module.

Read: Anytime

Write: Anytime in Initialization Mode (INITRQ = 1 and INITAK = 1)

Bit	Name	Description
0:1	SJW[1:0]	Synchronization Jump Width—defines the maximum number of time quanta (Tq) clock cycles a bit can be shortened or lengthened to achieve re-synchronization to data transitions on the bus. 00 = 1 Tq clock cycle 10 = 2 Tq clock cycles 01 = 3 Tq clock cycles 11 = 4 Tq clock cycles
2:7	BRP[5:0]	Baud Rate Prescaler—bits determine time quanta (Tq) clock used to build up individual bit timing, see Table 19-6.

Table 19-6. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler Value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
.....						
1	1	1	1	1	0	63
1	1	1	1	1	1	64

19.5.6 MSCAN Bus Timing Register 1 (CANBTR1)—MBAR + 0x0905

Table 19-7. MSCAN Bus Timing Register 1

	msb 0	1	2	3	4	5	6	7 lsb
R	SAMP	TSEG[22:20]			TSEG[13:10]			
W								
RESET:	0	0	0	0	0	0	0	0

The MSCAN Bus Timing Register 1 provides for various bus timing control of the MSCAN module.

Read: Anytime

Write: Anytime in Initialization Mode (INITRQ = 1 and INITAK = 1)

Bit	Name	Description
0	SAMP	Sampling—bit determines number of serial bus samples taken per bit-time. If set, three samples per bit are taken; the regular one (sample point) and two preceding samples using a majority rule. For higher bit-rates, it is recommended that SAMP be cleared, which means only one sample is taken per bit. 0 = One sample per bit 1 = Three samples per bit
1:3	TSEG[22:20]	Time Segment 2—time segments within the bit-time, fix the number of clock cycles per bit-time and the location of the sample point. Time segment 2 (TSEG2) values are programmable as shown in Table 19-9.
4:7	TSEG[13:10]	Time Segment 1—time segments within the bit-time, fix the number of clock cycles per bit-time and the location of the sample point. Time segment 1 (TSEG1) values are programmable as shown in Table 19-8.

Read: Anytime

Write: Anytime in Initialization Mode (INITRQ = 1 and INITAK = 1)

Bit-time, as shown below, is determined by:

- oscillator frequency
- baud rate prescaler
- number of time quanta (Tq) clock cycles per bit

Table 19-8 and Table 19-9 give time segment values.

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{CANCLK}} \bullet (\text{Number of Time Quanta})$$

Table 19-8. Time Segment 1 Values

TSEG13	TSEG12	TSEG11	TSEG10	Time segment 1
0	0	0	0	1 Tq clock cycle (a)
0	0	0	1	2 Tq clock cycles (1)
0	0	1	0	3 Tq clock cycles (1)
0	0	1	1	4 Tq clock cycles
.....				
1	1	1	0	15 Tq clock cycles
1	1	1	1	16 Tq clock cycles

Table 19-9. Time Segment 2 Values

TSEG22	TSEG21	TSEG20	Time segment 2
0	0	0	1 Tq clock cycle (a)
0	0	1	2 Tq clock cycles
.....			
1	1	0	7 Tq clock cycles
1	1	1	8 Tq clock cycles

19.5.7 MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908

Table 19-10. MSCAN Receiver Flag Register

	msb 0	1	2	3	4	5	6	7 lsb
R	WUPIF	CSCIF	RSTAT[1:0]		TSTAT[1:0]		OVRIF	RXF
W								
RESET:	0	0	0	0	0	0	0	0

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

A flag can only be cleared when the condition which caused the setting is no longer valid and can only be cleared by software (writing a “1” to the corresponding bit position). Every flag has an associated interrupt enable bit in the MSCAN Receive Interrupt Enable Register.

Read: Anytime

Write: Anytime when out of initialization mode, except RSTAT(1:0) & TSTAT(1:0) flags which are read only; write of “1” clears flag; write of “0” ignored.

Bit	Name	Description
0	WUPIF	WakeUp Interrupt Flag—If MSCAN detects bus activity while in sleep mode and WUPE=1 in CANTCTL0, it sets the WUPIF flag. If not masked, a WakeUp interrupt is pending while this flag is set. 0 = No WakeUp activity observed while in Sleep Mode. 1 = MSCAN detected bus activity and requested WakeUp.
1	CSCIF	CAN Status Change Interrupt Flag—flag is set when MSCAN changes its current bus status due to actual value of Tx error counter (TEC) and Rx error counter (REC). An additional 4-bit (RSTAT[1:0], TSTAT[1:0]) status register, split into separate sections for TEC/REC, notifies system of actual bus status. If not masked, an error interrupt is pending while this flag is set. CSCIF provides a blocking interrupt. That guarantees the Rx/Tx status bits (RSTAT/TSTAT) are updated only when no CAN Status Change interrupt is pending. If TECs/RECs change their current value after CSCIF is asserted and therefore cause an additional state change in RSTAT/TSTAT bits, these bits keep their old state bits until the current CSCIF interrupt is again cleared. 0 = No change in bus status occurred since last interrupt 1 = MSCAN changed current bus status.
2:3	RSTAT[1:0]	Receiver Status bits—values of the error counters control the actual bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set these bits indicate the appropriate receiver related bus status of the MSCAN. The coding for the bits RSTAT1, RSTAT0 is: 00 = RxOK: 0 ≤ Receive Error Counter ≤ 96 01 = RxWRN: 96 < Receive Error Counter ≤ 127 10 = RxERR: 127 < Receive Error Counter 11 = BusOff: 255 > Transmit Error Counter
4:5	TSTAT[1:0]	Transmitter Status bits—values of the error counters control the actual bus status of the MSCAN. As soon as the status change interrupt flag (CSCIF) is set these bits indicate the appropriate transmitter related bus status of the MSCAN. The coding for the bits TSTAT1, TSTAT0 is: 00 = TxOK: 0 ≤ Transmit Error Counter ≤ 96 01 = TxWRN: 96 < Transmit Error Counter ≤ 127 10 = TxERR: 127 < Transmit Error Counter ≤ 255 11 = BusOff: 255 > Transmit Error Counter

Bit	Name	Description
6	OVRIF	Overrun Interrupt Flag—flag is set when a data overrun condition occurs. If not masked, an Error interrupt is pending while this flag is set. 0 = No data overrun condition. 1 = data overrun detected.
7	RXF	Receive Buffer Full—flag is set by MSCAN when a new message is shifted into RX FIFO. Flag indicates whether the shifted buffer is loaded with a correctly received message (matching identifier, matching cyclic redundancy code (CRC) and no other errors detected). After CPU reads message from RxFG buffer in Rx FIFO, RxF flag must be cleared to release the buffer. A set RxF flag prohibits shifting of next FIFO entry into foreground buffer (RxFG). If not masked, RX interrupt is pending while this flag is set. To ensure data integrity, do not read the Rx buffer registers while RxF flag is cleared. For MCUs with dual CPUs, reading Rx buffer registers while RxF flag is cleared may result in a CPU fault condition. 0 = No new message available within RxFG. 1 = Rx FIFO not empty. New message is available in RxFG.

Note:

1. Every flag has an associated interrupt enable bit in the CANRIER register. A flag can only be cleared:
   - when the condition that caused the setting is no longer valid.
   - by software writing 1 to the corresponding bit position.

WARNING: To ensure data integrity, do not read the receive buffer registers while the RX Flag is cleared.

19.5.8 MSCAN Receiver Interrupt Enable Register (CANRIER)—MBAR + 0x0909

Table 19-11. MSCAN Receiver Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7 lsb
R	WUPIE	CSCIE	RSTATE[1:0]		TSTATE[1:0]		OVRIE	RXFIE
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	WUPIE	WakeUp Interrupt Enable 0 = No interrupt request is generated from this event. 1 = A WakeUP event causes a WakeUp interrupt request.
1	CSCIE	CAN Status Change Interrupt Enable 0 = No interrupt request is generated from this event 1 = A CAN Status Change event causes an error interrupt request
2:3	RSTATE[1:0]	Receiver Status Change Enable—bits control sensitivity level in which Rx state changes cause CSCIF interrupts. Independent of the chosen sensitivity level, RSTATE flags still indicate the actual Rx state and are only updated if no CSCIF interrupt is pending. 00 = Do not generate CSCIF interrupt caused by Rx state changes. 01 = Generate CSCIF interrupt only if receiver enters or leaves “BusOff” state. Discard other Rx state changes for generating CSCIF interrupt. 10 = Generate CSCIF interrupt only if receiver enters or leaves “RxErr” or “BusOff” state. Discard other Rx state changes for generating CSCIF interrupt. 11 = Generate CSCIF interrupt on all state changes.

Bit	Name	Description
4:5	TSTATE[1:0]	Transmitter Status Change Enable—bits control sensitivity level in which Tx state changes cause CSCIF interrupts. Independent of the chosen sensitivity level, TSTATE flags still indicate the actual Tx state and are only updated if no CSCIF interrupt is pending. 00 = Do not generate CSCIF interrupt caused by Tx state changes. 01 = Generate CSCIF interrupt only if transmitter enters or leaves “BusOff” state. Discard other Tx state changes for generating CSCIF interrupt. 10 = Generate CSCIF interrupt only if transmitter enters or leaves “TxErr” or “BusOff” state. Discard other Tx state changes for generating CSCIF interrupt. 11 = Generate CSCIF interrupt on all state changes.
6	OVRIE	Overrun Interrupt Enable 0 = No interrupt request is generated from this event. 1 = An overrun event causes an error interrupt request
7	RXFIE	Receiver Full Interrupt Enable 0 = No interrupt request is generated from this event 1 = Rx buffer full (successful message reception) event causes Rx interrupt request

Note: The MSCAN Receive Interrupt Enable Register is held in reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

19.5.9 MSCAN Transmitter Flag Register (CANTFLG)—MBAR + 0x090C

Table 19-12. MSCAN Transmitter Flag Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					TXE[2:0]		
W								
RESET:	0	0	0	0	0	1	1	1

Bit	Name	Description
0:4	—	Reserved
5:7	TXE[2:0]	Transmitter Buffer Empty—flag indicates the associated Tx message buffer is empty, and thus not scheduled for transmission. CPU must clear the flag after a message is set up in the Tx buffer and is due for transmission. MSCAN sets flag after message is successfully sent. Flag is also set by MSCAN when Tx request is successfully aborted due to a pending abort request. If not masked, a Tx interrupt is pending while this flag is set. Clearing a TxEx flag also clears the corresponding ABTAKx. When a TxEx flag is set, the corresponding ABTRQx bit is cleared. When listen-mode is active TxEx flags cannot be cleared and no transmission is started. 0 = associated message buffer full (loaded with message due for Tx) 1 = associated message buffer empty (not scheduled)

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime for TXEx flags when not in Initialization Mode; write of “1” clears flag, write of ‘0’ is ignored.

19.5.10 MSCAN Transmitter Interrupt Enable Register (CANTIER)—MBAR+0x090D

Table 19-13. MSCAN Transmitter Interrupt Enable Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					TXEIE[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5:7	TXEIE[2:0]	Transmitter Empty Interrupt Enable 0 = No interrupt request generated from this event. 1 = Transmitter empty (Tx buffer available) event causes Tx empty interrupt request.

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when not in Initialization Mode.

19.5.11 MSCAN Transmitter Message Abort Request (CANTARQ)—MBAR + 0x0910

Table 19-14. MSCAN Transmitter Message Abort Request Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					ABTRQ[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5:7	ABTRQ[2:0]	Abort Request—CPU sets bit to request a scheduled message buffer (TxEx=0) be aborted. MSCAN grants request if message has not already started transmission, or if transmission is not successful (lost arbitration or error). When message is aborted, the associated TxE and abort acknowledge flags (ABTAK) are set and a Tx interrupt occurs if enabled. The CPU cannot reset ABTRQx. ABTRQx is reset whenever the associated TxE flag is set. 0 = No abort request 1 = Abort request pending

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

Read: Anytime

Write: Anytime when not in Initialization Mode; write of “1” clears flag, write of ‘0’ is ignored.

19.5.12 MSCAN Transmitter Message Abort Ack (CANTAAK)—MBAR +0x0911

Table 19-15. MSCAN Transmitter Message Abort Acknowledgement Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					ABTAK[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1).

READ: Anytime

WRITE: Not writable at any time

Bit	Name	Description
0:4	—	Reserved
5:7	ABTAK[2:0]	Abort Acknowledge—flag acknowledges message was aborted due to pending CPU abort request. After a specific message buffer is flagged empty, application software can use this flag to identify whether message was successfully aborted or was sent. Flag is cleared whenever the corresponding TxE flag is cleared. 0 = message not aborted 1 = message aborted

19.5.13 MSCAN Transmit Buffer Selection (CANTBSEL)—MBAR + 0x0914

Table 19-16. MSCAN Transmit Buffer Selection Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					TX[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5:7	TX[2:0]	Transmit Buffer Select—lowest numbered bit places respective Tx buffer in CANTxFG register space (e.g., Tx1=1 and Tx0=1 selects Tx buffer Tx0, Tx1=1 and Tx0=0 selects Tx buffer Tx1) 0 = associated message buffer deselected 1 = associated message buffer selected, if lowest numbered bit

Note: This register is held in the reset state when the initialization mode is active (INITRQ = 1 and INITAK = 1). This register is writable again as soon as the initialization mode is exited (INITRQ = 0 and INITAK = 0).

READ: Find the lowest ordered bit set to ‘1’, all other bits will be read as ‘0’

WRITE: Anytime when not in Initialization Mode

19.5.14 MSCAN ID Acceptance Control Register (CANIDAC)—MBAR + 0x0915

Table 19-17. MSCAN ID Acceptance Control Register

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved		IDAM[1:0]		Rsvd	IDHIT[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

READ: Anytime

WRITE: Anytime in Initialization Mode (INITRQ = 1 and INITAK =1)

Bit	Name	Description
0:1	—	Reserved
2:3	IDAM[1:0]	Identifier Acceptance Mode—CPU sets these flags to define the identifier acceptance filter organization. In filter closed mode, no message is accepted such that the foreground buffer is never reloaded. See Table 19-19.
4	—	Reserved
5:7	IDHIT[2:0]	Identifier Acceptance Hit Indicator—MSCAN sets these flags to indicate an identifier acceptance hit. See Table 19-18.

Table 19-18. Identifier Acceptance Hit Indication

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 Hit
0	0	1	Filter 1 Hit
0	1	0	Filter 2 Hit
0	1	1	Filter 3 Hit
1	0	0	Filter 4 Hit
1	0	1	Filter 5 Hit
1	1	0	Filter 6 Hit
1	1	1	Filter 7 Hit

Table 19-19. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	2 32-bit Acceptance Filters
0	1	4 16-bit Acceptance Filters
1	0	8 8-bit Acceptance Filters
1	1	Filter Closed

19.5.15 MSCAN Receive Error Counter Register (CANRXERR)—MBAR + 0x091C

Table 19-20. MSCAN Receive Error Counter Register

	msb 0	1	2	3	4	5	6	7 lsb
R	RxERR[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

Note: This register reflects the status of the MSCAN receive error counter.

READ: Only when in Sleep Mode (SLPRQ = 1 and SLPAK = 1) or Initialization Mode (INITRQ = 1 and INITAK =1).

WRITE: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or Initialization may return an incorrect value.

NOTE

Writing to these registers when in special modes can alter the MSCAN functionality.

Bit	Name	Description
0:7	RxERR[7:0]	This register reflects the status of the MSCAN receive error counter.

19.5.16 MSCAN Transmit Error Counter Register (CANTXERR)—MBAR + 0x091D

Table 19-21. MSCAN Transmit Error Counter Register

	msb 0	1	2	3	4	5	6	7 lsb
R	TxERR[7:0]							
W								
RESET:	0	0	0	0	0	0	0	0

Note: This register reflects the status of the MSCAN transmit error counter.

READ: Only when in Sleep Mode (SLPRQ = 1 and SLPAK = 1) or Initialization Mode (INITRQ = 1 and INITAK =1).

WRITE: Unimplemented

NOTE

Reading this register when in any other mode other than sleep or Initialization may return an incorrect value.

NOTE

Writing to these registers when in special modes can alter the MSCAN functionality.

Bit	Name	Description
0:7	TxERR[7:0]	This register reflects the status of the MSCAN receive error counter.

19.5.17 MSCAN ID Acceptance Registers (CANIDAR0-7)—MBAR + 0x0915

Table 19-22. MSCAN ID Acceptance Registers (1st Bank)

	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x20						CANIDR0
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x21						CANIDR1
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x24						CANIDR2
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x25						CANIDR3
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0

Table 19-23. MSCAN ID Acceptance Registers (2nd Bank)

	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x30						CANIDR4	
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x31						CANIDR5	
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x34						CANIDR6	
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x35						CANIDR7	
R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	AC[7:0]	Acceptance Code—bits comprise a user defined sequence with which corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. Result of this comparison is then masked with the corresponding identifier mask register.

READ: Anytime

WRITE: Anytime in initialization mode (INITRq + 1 and INITAK = 1).

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message is overwritten by the next message (dropped).

The acceptance registers of the MSCAN are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers, only the first two (CANIDAR0/1, CANIDMR0/1) are applied.

19.5.18 MSCAN ID Mask Register (CANIDMR0-7)—MBAR + 0x0928

Table 19-24. MSCAN ID MaskRegisters (1st Bank)

	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x28						CANIDMR0
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x29						CANIDMR1
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x2C						CANIDMR2
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset		0x2D						CANIDMR3
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0

Table 19-25. MSCAN ID MaskRegisters (2nd Bank)

	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x38						CANIDMR4	
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x39						CANIDMR5	
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x3C						CANIDMR6	
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0
	msb 0	1	2	3	4	5	6	7 lsb
Adress Offset	0x3D						CANIDMR7	
R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
W								
RESET:	0	0	0	0	0	0	0	0

READ: Anytime

WRITE: Anytime in initialization mode (INITRq + 1 and INITAK = 1).

Bit	Name	Description
0:7	AM[7:0]	Acceptance Mask bits—If a particular bit in this register is cleared, this indicates the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates the state of the corresponding bit in the identifier acceptance register does not affect whether or not message is accepted. 0 = Match corresponding acceptance code register and identifier bits 1 = Ignore corresponding acceptance code register bit

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

- To receive standard identifiers in 32-bit filter mode, the last three bits (AM[0:2]) in the following mask registers must be programmed as "don't care":
  - CANIDMR1
  - CANIDMR5
- To receive standard identifiers in 16-bit filter mode, the last three bits (AM[0:2]) in the following mask registers must be programmed as "don't care":
  - CANIDMR1
  - CANIDMR3
  - CANIDMR5
  - CANIDMR7

19.6 Programmer's Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure. An additional Transmit Buffer Priority Register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900*). The Time Stamp register is written by the MSCAN. The CPU can only read these registers.

Table 19-26. Message Buffer Organization

Addr	Register Name
$__00	Identifier Register 0
$__01	Identifier Register 1
$__04	Identifier Register 2
$__05	Identifier Register 3
$__08	Data Segment Register 0
$__09	Data Segment Register 1
$__0C	Data Segment Register 2
$__0D	Data Segment Register 3
$__10	Data Segment Register 4
$__11	Data Segment Register 5
$__14	Data Segment Register 6
$__15	Data Segment Register 7
$__18	Data Length Register
$__19	Transmit Buffer Priority Register[a]
$__1C	Time Stamp Register (High Byte)[b]
$__1D	Time Stamp Register (High Byte)[c]

[a] Not Applicable for Receive Buffers
[b] Read-Only for CPU
[c] Read-Only for CPU

Figure 19-27 shows the common 13 byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 19-28. All bits of the receive and transmit buffers are 'x' out of reset because of RAM based implementation[1].All reserved or unused bits of the receive and transmit buffers are always read 'x'.

Table 19-27. Receive / Transmit Message Buffer Extended Identifier

Register		Bit 7	6	5	4	3	2	1	Bit 0	ADDR
IDR0	Read:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	$__00
	Write:									
IDR1	Read:	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15	$__01
	Write:									

= Unused[a]

1. Exception: The Transmit Priority Registers are "0" out of reset

Table 19-27. Receive / Transmit Message Buffer Extended Identifier (continued)

Register		Bit 7	6	5	4	3	2	1	Bit 0	ADDR
IDR2	Read:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	$__04
	Write:									
IDR3	Read:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR	$__05
	Write:									
DSR0	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__08
	Write:									
DSR1	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__09
	Write:									
DSR2	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__0C
	Write:									
DSR3	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__0D
	Write:									
DSR4	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__10
	Write:									
DSR5	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__11
	Write:									
DSR6	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__14
	Write:									
DSR7	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	$__15
	Write:									
DLR	Read:					DLC3	DLC2	DLC1	DLC0	$__18
	Write:									

= Unused[a]

[a] Unused bits are always read 'x'

Read: anytime for transmit buffers; only when RXF flag is set for receive buffers (see Section 19.5.7, *MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908*).

Write: anytime for transmit buffers when TXEx flag is set (see Section 19.5.9, *MSCAN Transmitter Flag Register (CANTFLG)—MBAR + 0x090C*) and the corresponding transmit buffer is selected in CANTBSEL (see Section 19.5.13, *MSCAN Transmit Buffer Selection (CANTBSEL)—MBAR + 0x0914*); unimplemented for receive buffers

Reset: $xx because of RAM based implementation

Table 19-28. Standard Identifier Mapping

Register		Bit 7	6	5	4	3	2	1	Bit 0	ADDR
IDR0	Read:	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	$__x0
	Write:									

= Unused[a]

Table 19-28. Standard Identifier Mapping (continued)

Register		Bit 7	6	5	4	3	2	1	Bit 0	ADDR
IDR1	Read:	ID2	ID1	ID0	RTR	IDE (=0)				$__x1
	Write:									
IDR2	Read:									$__x4
	Write:									
IDR3	Read:									$__x5
	Write:									

= Unused[a]

[a]Unused bits are always read 'x'

19.6.1 Identifier Registers (IDR0-3)

The identifier registers for an extended format identifier consist of a total of 32 bits; ID28 - ID0, SRR, IDE, and RTR bits. The identifier registers for a standard format identifier consist of a total of 13 bits; ID10 - ID0, RTR, and IDE bits.

ID28 - ID0 — Extended format identifier

The identifiers consist of 29 bits (ID28 - ID0) for the extended format. ID28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

ID10 - ID0 — Standard format identifier

The identifiers consist of 11 bits (ID10 – ID0) for the standard format. ID10 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR — Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and is stored as received on the CAN bus for receive buffers.

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send.

1=Extended format (29 bit)

0=Standard format (11 bit)

RTR — Remote Transmission Request

This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1=Remote frame

0=Data frame

19.6.2 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB7-DB0, contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

DB7 - DB0 — Data Bits 7-0

19.6.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

DLC3 - DLC0 — Data Length Code bits

The data length code contains the number of bytes (data byte count) of the respective message. During the transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted data bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 19-29 shows the effect of setting the DLC bits.

Table 19-29. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

19.6.4 MSCAN Transmit Buffer Priority Register (TBPR)—MBAR + 0x0979

Table 19-30. MSCAN Transmit Buffer Priority Register

	msb 0	1	2	3	4	5	6	7 lsb
R	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	PRIO[7:0]	Register defines local priority of associated message buffer. Local priority is used for MSCAN internal prioritization process and is defined to be highest for the smallest binary number. MSCAN implements the following internal prioritization mechanisms: • All transmission buffers with a cleared TXEx flag participate in prioritization immediately before start of frame (SOF) is sent. • Transmission buffer with lowest local priority field wins prioritization. • If more than one buffer has the same lowest priority, message buffer with lower index number wins.

19.6.5 MSCAN Time Stamp Register High (TSRH)—MBAR + 0x097C

Table 19-31. MSCAN Time Stamp Register (High Byte)

	msb 0	1	2	3	4	5	6	7 lsb
R	TSR15	TSR14	TSR13	TSR12	TSR11	TSR10	TSR9	TSR8
W								
RESET:	0	0	0	0	0	0	0	0

READ: Anytime

WRITE: Unimplemented

Bit	Name	Description
0:7	TSR[15:8]	If TIME bit is enabled, MSCAN writes a special time stamp to respective registers in active Tx or Rx buffer as soon as a message is acknowledged on the CAN bus. Time stamp is written on bit sample point for recessive bit of ACK delimiter in CAN frame. If Tx, CPU can only read time stamp after respective Tx buffer is flagged empty. Timer value, used for stamping, is taken from a free running internal CAN bit-clock. Timer overrun is not indicated by MSCAN. Timer is reset (all bits set to 0) during initialization mode. CPU can only read time stamp registers.

19.6.6 MSCAN Time Stamp Register Low (TSRL)—MBAR + 0x097D

Table 19-32. MSCAN Time Stamp Register (Low Byte)

	msb 0	1	2	3	4	5	6	7 lsb
R	TSR7	TSR6	TSR5	TSR4	TSR3	TSR2	TSR1	TSR0
W								
RESET:	0	0	0	0	0	0	0	0

READ: Anytime

WRITE: Unimplemented

Bit	Name	Description
0:7	TSR[7:0]	If TIME bit is enabled, MSCAN writes a special time stamp to respective registers in active Tx or Rx buffer as soon as message is acknowledged on CAN bus. Time stamp is written on bit sample point for recessive bit of ACK delimiter in CAN frame. If Tx, CPU can only read time stamp after respective Tx buffer is flagged empty. Timer value, used for stamping, is taken from a free running internal CAN bit-clock. Timer overrun is not indicated by MSCAN. Timer is reset (all bits set to 0) during initialization mode. CPU can only read time stamp registers.

19.7 Functional Description

19.7.1 General

This section provides a complete functional description of the MSCAN. It describes each of the features and modes listed in the introduction.

19.7.2 Message Storage

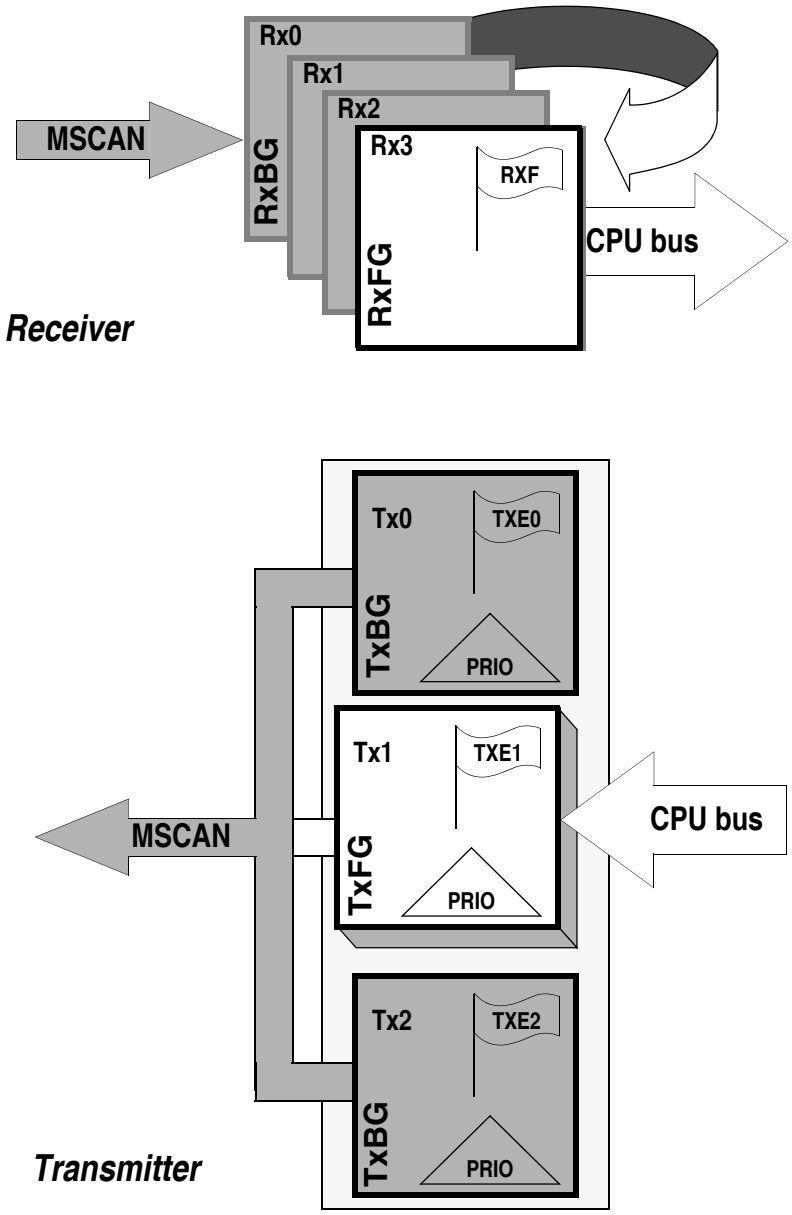


Figure 19-3. User Model for Message Buffer Organization

MSCAN facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

19.7.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the bus between the two messages. Such nodes arbitrate for the bus immediately after sending the previous message and only release the bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message is sent. This loading process lasts a finite amount of time and has to be completed within the Inter-Frame Sequence (IFS)[1] to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU react with short latencies to the transmit interrupt.

1. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, as such, reduces the reactiveness requirements on the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the “local priority” concept described in Section 19.7.2.2, *Transmit Structures*.

19.7.2.2 Transmit Structures

The MSCAN has a triple transmit buffer scheme which allows multiple messages to be set up in advance and achieve an optimized real-time performance. The three buffers are arranged as shown in Section Figure 19-3., *User Model for Message Buffer Organization*.

All three buffers have a 13 byte data structure similar to the outline of the receive buffers Section 19.6, *Programmer’s Model of Message Storage*. An additional MSCAN Transmit Buffer Priority Register (TBPR)—MBAR + 0x0979 contains an 8-bit “Local Priority” field (PRIO). The remaining two bytes are used for time stamping of a message, if required (see Section 19.6.5, *MSCAN Time Stamp Register High (TSRH)—MBAR + 0x097C*, Section 19.6.6, *MSCAN Time Stamp Register Low (TSRL)—MBAR + 0x097D*).

To transmit a message, the CPU has to identify an available transmit buffer which is indicated by a set Transmitter Buffer Empty (TXEx) flag Section 19.5.9, *MSCAN Transmitter Flag Register (CANTFLG)—MBAR + 0x090C*. If a transmit buffer is available, the CPU has to set a pointer to this buffer by writing to the CANTBSEL register (Section 19.5.13, *MSCAN Transmit Buffer Selection (CANTBSEL)—MBAR + 0x0914*). This makes the respective buffer accessible within the CANTXFG address space Section 19.6, *Programmer’s Model of Message Storage*. The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition this scheme makes the handler software simpler as only one address area is applicable for the transmit process. In addition the required address space is minimized.

The CPU then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt Section 19.7.9.1, *Transmit Interrupt* is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the “local priority” setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. As messages that are already in transmission cannot be aborted, the user has to request the abort by setting the corresponding Abort Request bit (ABTRQ) Section 19.5.12, *MSCAN Transmitter Message Abort Ack (CANTAAK)—MBAR +0x0911*. The MSCAN then grants the request, if possible, by: 1) setting the corresponding Abort Acknowledge flag (ABTAK) in the CANTAAK register, 2) setting the associated TXE flag to release the buffer, and 3) generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was aborted (ABTAK=1) or sent (ABTAK=0).

19.7.2.3 Receive Structures

The received messages are stored in a four stage input FIFO. The four message buffers are alternately mapped into a single memory area Section Figure 19-3., *User Model for Message Buffer Organization*. While the background receive buffer (RxBG) is exclusively associated with the MSCAN, the foreground receive buffer (RxFG) is addressable by the CPU Section Figure 19-3., *User Model for Message Buffer Organization*. This scheme simplifies the handler software as only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents and a time stamp, if enabled (for details Section 19.6, *Programmer’s Model of Message Storage*)[2].

The Receiver Full flag (RXF) Section 19.5.7, *MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908* signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see if it passes the filter (Section 19.7.3, *Identifier Acceptance Filter*) and in parallel, is written into the active RxBG. After successful reception of a valid message the MSCAN shifts the content of RxBG into the receiver FIFO[3], sets the RXF

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.
2. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for details.
3. Only if the RXF flag is not set.

flag, and generates a receive interrupt Section 19.7.9.2, *Receive Interrupt* to the CPU[1]. The user's receive handler has to read the received message from the RxFG and then reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loop back mode Section 19.5.4, *MSCAN Control Register 1 (CANCTL1)—MBAR + 0x0901* where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration[2]. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled Section 19.7.9.4, *Error Interrupt*. The MSCAN is still able to transmit messages while the receiver FIFO being filled, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

19.7.3 Identifier Acceptance Filter

The MSCAN Identifier Acceptance Registers (Section 19.5.14, *MSCAN ID Acceptance Control Register (CANIDAC)—MBAR + 0x0915*) define the acceptable patterns of the standard or extended identifier (ID10 - ID0 or ID28 - ID0). Any of these bits can be marked 'don't care' in the MSCAN Identifier Mask Registers Section 19.5.18, *MSCAN ID Mask Register (CANIDMR0-7)—MBAR + 0x0928*.

A filter hit is indicated to the application software by a set Receive Buffer Full flag (RXF=1) and three bits in the CANIDAC register Section 19.5.17, *MSCAN ID Acceptance Registers (CANIDAR0-7)—MBAR + 0x0915*. These Identifier Hit flags (IDHIT2-0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case more than one hit occurs (two or more filters match), the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes[3]:

- Two identifier acceptance filters, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame: Remote Transmission Request (RTR), Identifier Extension (IDE), and Substitute Remote Request (SRR) or b)[4] the 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure 19-4 shows how the first 32-bit filter bank (CANIDAR0-3, CANIDMR0-3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4-7, CANIDMR4-7) produces a filter 1 hit.
- Four identifier acceptance filters, each to be applied to a) the 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages or b) the 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 19-5 shows how the first 32-bit filter bank (CANIDAR0-3, CANIDMR0-3) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4-7, CANIDMR4-7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier. Figure 19-6 shows how the first 32-bit filter bank (CANIDAR0-3, CANIDMR0-3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4-7, CANIDMR4-7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

1. The receive interrupt occurs only if not masked. A polling scheme can be applied on RXF also.
2. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for details.
3. For a better understanding of references made within the filter mode description, reference the Bosch specification dated September 1991 which details the CAN 2.0A/B protocol.
4. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

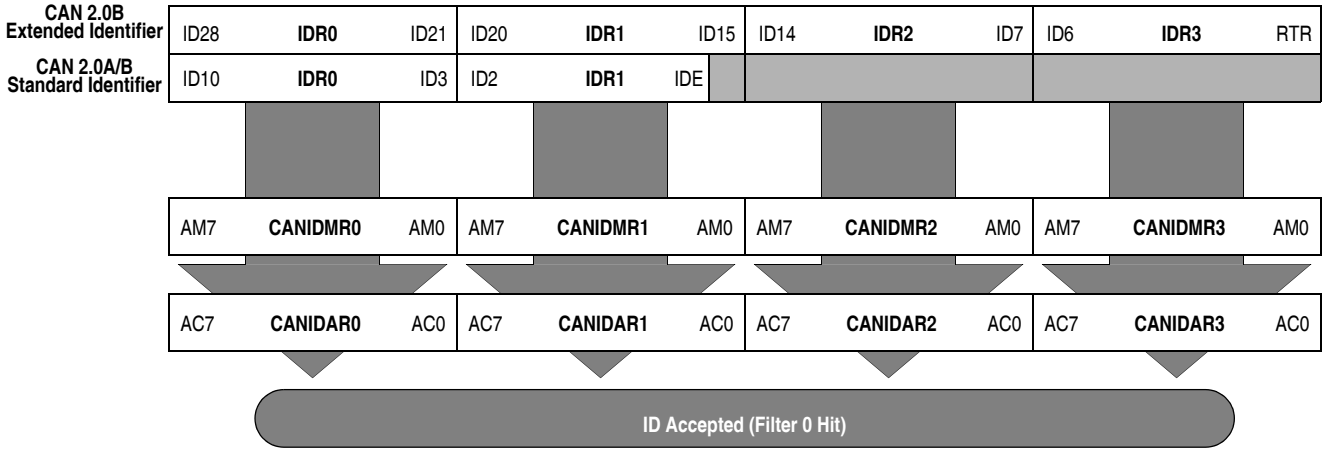


Figure 19-4. 32-bit Maskable Identifier Acceptance Filter

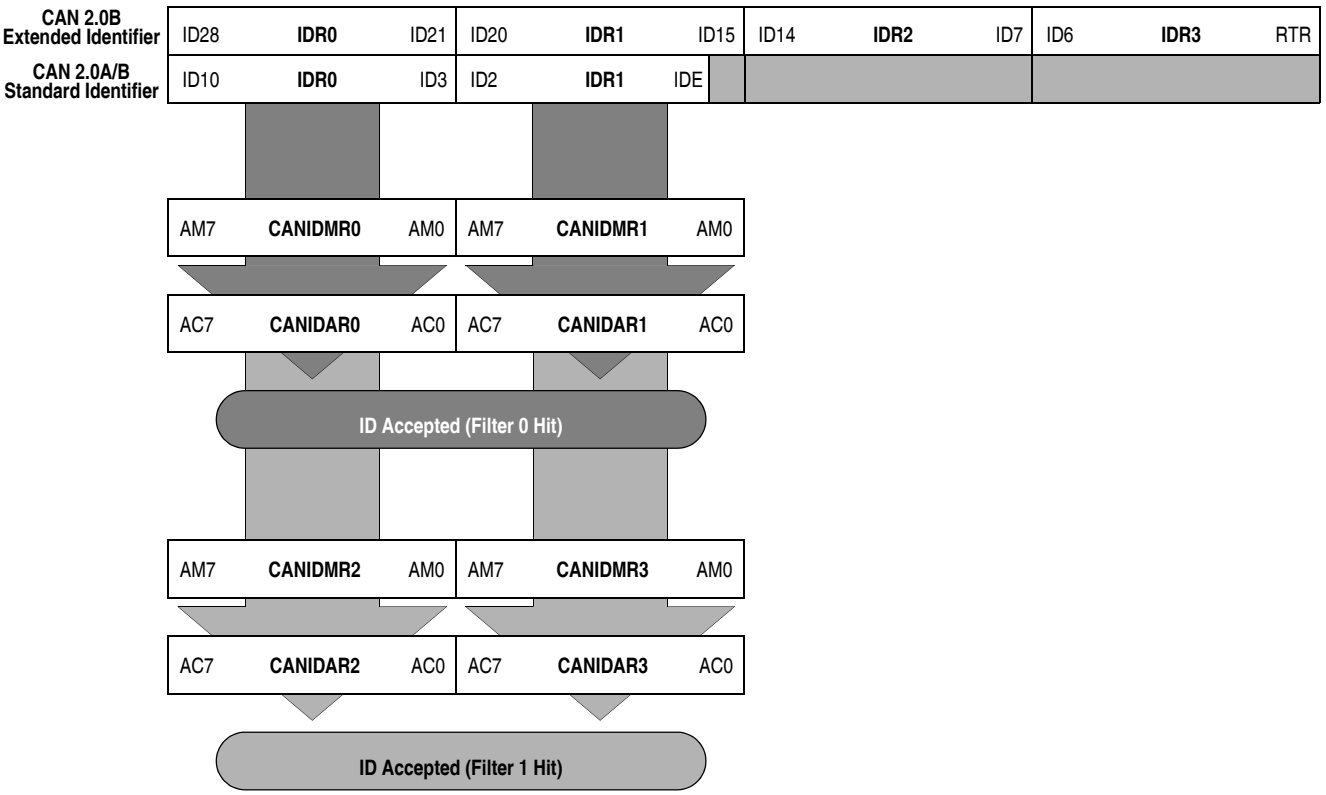


Figure 19-5. 16-bit Maskable Identifier Acceptance Filters

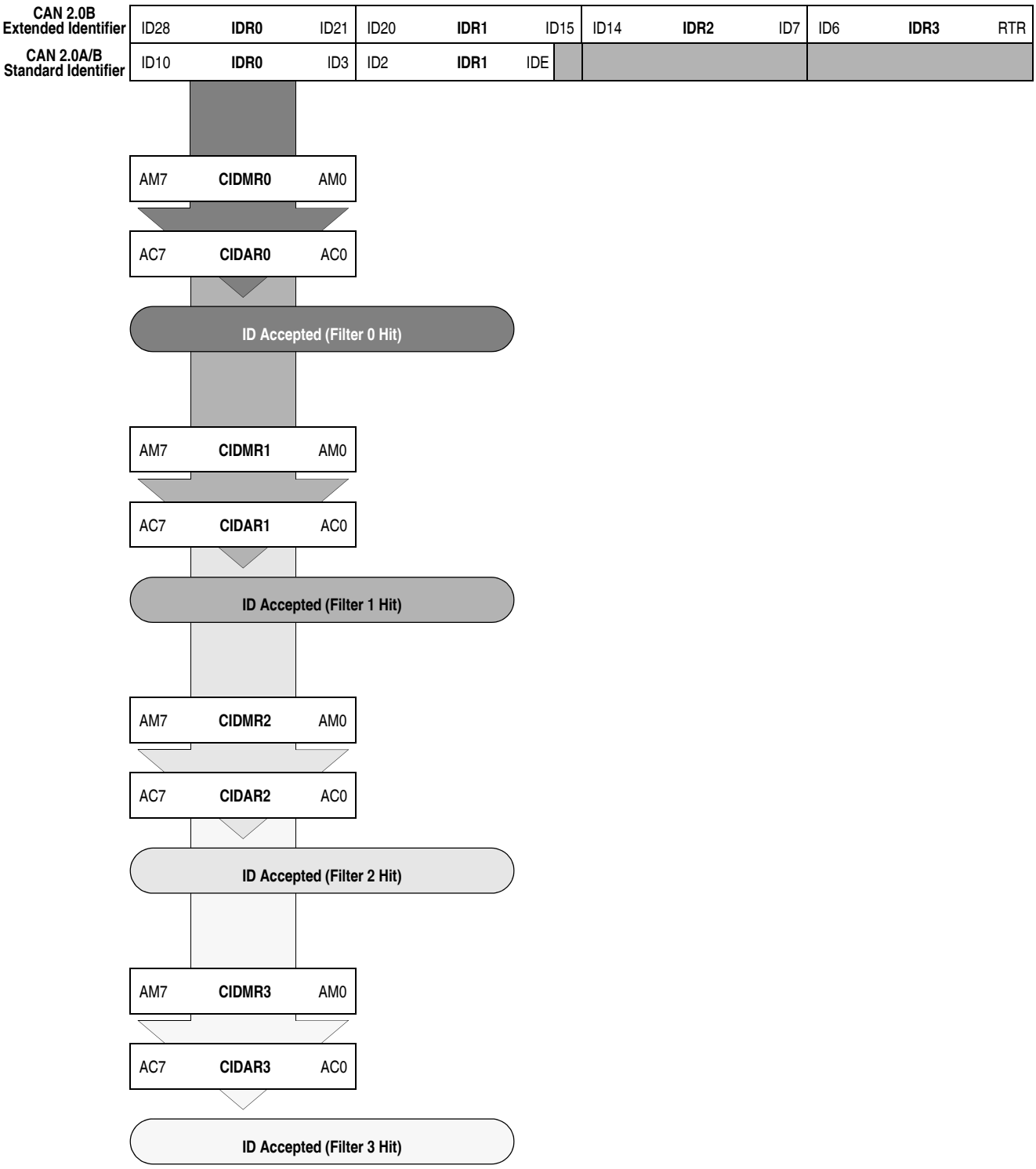


Figure 19-6. 8-bit Maskable Identifier Acceptance Filters

19.7.4 Protocol Violation Protection

The MSCAN protects the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.

- All registers which control the configuration of the MSCAN cannot be modified while the MSCAN is on-line. The MSCAN has to be in Initialization Mode. The corresponding INITRQ/INITAK handshake bits in the CANCTL0/CANCTL1 registers Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900* serve as a lock to protect the following registers:
  - MSCAN Control 1 Register (CANCTL1)
  - MSCAN Bus Timing Registers 0 and 1 (CANBTR0, CANBTR1)
  - MSCAN Identifier Acceptance Control Register (CANIDAC)
  - MSCAN Identifier Acceptance Registers (CANIDAR0-7)
  - MSCAN Identifier Mask Registers (CANIDMR0-7)
- The TXCAN pin is immediately forced to a recessive state when the MSCAN goes into the Power Down Mode or Initialization Mode (see Section 19.7.8.6, *MSCAN Power Down Mode* and Section 19.7.8.5, *MSCAN Initialization Mode*).
- The MSCAN enable bit (CANE) is only writable once in normal modes as further protection against inadvertently disabling the MSCAN.

19.7.5 Clock System

Figure 19-7 shows the structure of the MSCAN clock generation circuitry. With this flexible clocking scheme, the MSCAN is able to handle CAN bus rates ranging from 10 Kbps up to 1 Mbps.

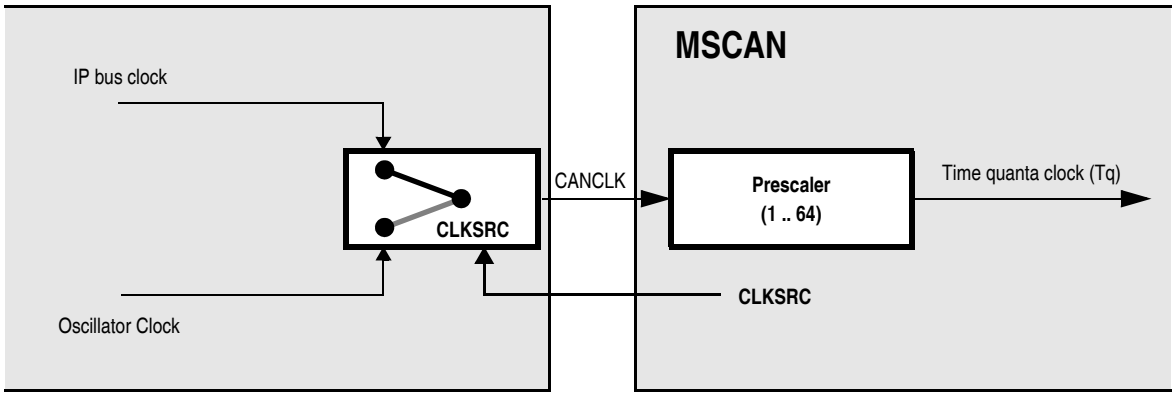


Figure 19-7. MSCAN Clocking Scheme

The clock source bit (CLKSRC) in the CANCTL1 register Section 19.5.4, *MSCAN Control Register 1 (CANCTL1)—MBAR + 0x0901* defines whether the internal CANCLK is connected to the output of the system oscillator clock (SYS_XTAL_IN) or to the IP bus clock.

NOTE

Both MSCAN modules can have only the same selected clock source. To select the oscillator clock the CLKSRC bit in the CANCTL1 register must be set in MSCAN1 OR/AND in MSCAN2.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 45%-55% duty cycle of the clock is required.

Because the Bus Clock is generated from a PLL, it is recommended to select the Oscillator Clock rather than the Bus Clock due to jitter considerations, especially at the faster CAN bus rates.

A programmable prescaler generates the time quanta (Tq) clock from CANCLK. A time quantum is the atomic unit of time handled by the MSCAN.

$$f_{Tq} = \frac{f_{CANCLK}}{(Prescaler\ \ value)}$$

A bit time is subdivided into three segments[1] [2] (reference Figure 19-8):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.

1. For further explanation of the under-lying concepts please refer to ISO/DIS 11519-1, Section 10.3.
2. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

- Time Segment 2: This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

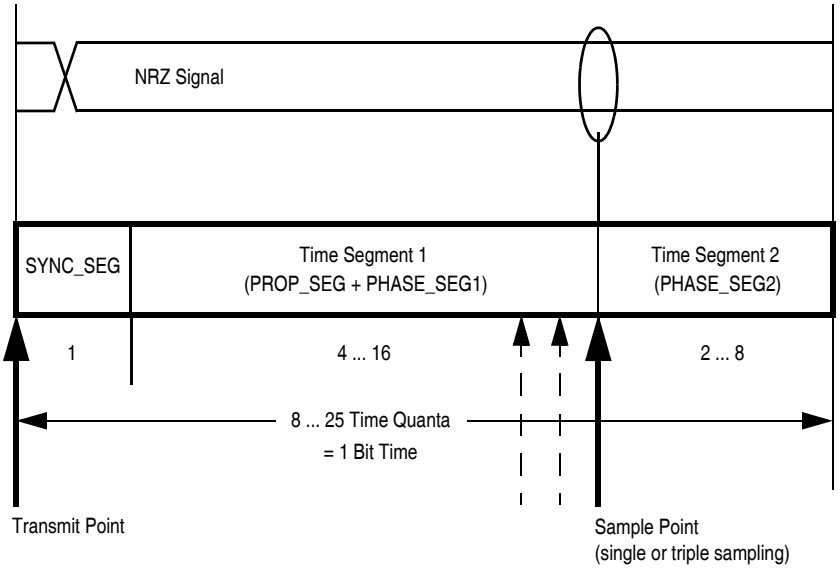


Figure 19-8. Segments within the Bit Time

Table 19-33. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node in receive mode samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The Synchronization Jump Width[1] can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The above parameters are set by programming the MSCAN Bus Timing Registers (CANBTR0, CANBTR1) (see Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900* and Section 19.5.6, *MSCAN Bus Timing Register 1 (CANBTR1)—MBAR + 0x0905*).

Table 19-34 gives an overview of the CAN compliant segment settings and the related parameter values.

NOTE

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard.

Table 19-34. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3

1. Reference the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

Table 19-34. CAN Standard Compliant Bit Time Segment Settings (continued)

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronization Jump Width	SJW
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

19.7.6 Timer Link

The MSCAN generates an internal time stamp whenever a valid frame is received or transmitted and the TIME bit is enabled. Because the CAN specification defines a frame to be valid if no errors occur before the End of Frame (EOF) field is transmitted successfully, the actual value of an internal timer is written at EOF to the appropriate time stamp position within the transmit buffer. For receive frames the time stamp is written to the receive buffer.

19.7.7 Modes of Operation

19.7.7.1 Normal Modes

The MSCAN module behaves as described within this specification in all normal modes.

19.7.7.2 Listen-Only Mode

In an optional bus monitoring mode (Listen-Only), the CAN node is able to receive valid data frames and valid remote frames, but it sends only “recessive” bits on the CAN bus. In addition it cannot start a transmission. If the MAC sub-layer is required to send a “dominant” bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MAC sub-layer monitors this “dominant bit, although the CAN bus may remain in recessive state externally.

19.7.8 Low Power Options

If the MSCAN is disabled (CANE=0), the MSCAN clocks are stopped for power savings.

If the MSCAN is enabled (CANE=1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: Sleep and Power Down Mode. In Sleep Mode power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In Power Down Mode, all clocks are stopped and no power is consumed.

Table 19-35 summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

For all modes, an MSCAN Wake-Up interrupt can only occur if the MSCAN is in Sleep Mode (SLPRQ=1 and SLPAK=1), wake-up functionality is enabled (WUPE=1) and the Wake-Up interrupt is enabled (WUPIE=1).

Table 19-35. CPU vs. MSCAN Operating Modes

Power Mode	MSCAN Mode			
	Normal	Power Down	Sleep	(CANE=0)
Full Power	CSWAI = X[a] SLPRQ = 0 SLPAK = 0		CSWAI = X SLPRQ = 1 SLPAK = 1	CSWAI = X SLPRQ = X SLPAK = X
Sleep	CSWAI = 0 SLPRQ = 0 SLPAK = 0	CSWAI = 1 SLPRQ = X SLPAK = X	CSWAI = 0 SLPRQ = 1 SLPAK = 1	CSWAI = X SLPRQ = X SLPAK = X
Deep Sleep		CSWAI = X SLPRQ = X SLPAK = X	CSWAI = X SLPRQ = 1 SLPAK = 1	CSWAI = X SLPRQ = X SLPAK = X

[a] ‘X’ means don’t care.

19.7.8.1 CPU Run Mode

As can be seen in Table 19-35, only MSCAN Sleep Mode is available as low power option, when CPU is in run mode.

19.7.8.2 CPU Sleep Mode

While the CPU is in Sleep Mode, the MSCAN can be operated in Normal Mode and generate interrupts (registers can be accessed via background debug mode). The MSCAN can also operate in any of the low power modes depending on the values of the SLPRQ/SLPAK and CSWAI bits as seen in Table 19-35.

19.7.8.3 CPU Deep Sleep Mode

In Deep Sleep Mode, the MSCAN operates in Power Down mode regardless of the value of the SLPRQ/SLPAK and CSWAI bits Table 19-35.

19.7.8.4 MSCAN Sleep Mode

The CPU can request the MSCAN to enter this low power mode by asserting the SLPRQ bit in the CANCTL0 register. The time when the MSCAN enters Sleep Mode depends on a fixed synchronization delay and its current activity:

- If it is transmitting, it continues to transmit until the entire message is transmitted and then goes into Sleep Mode.
- If it is receiving, it waits for the end of this message and then goes into Sleep Mode.
- If it is neither transmitting nor receiving, it immediately goes into Sleep Mode.

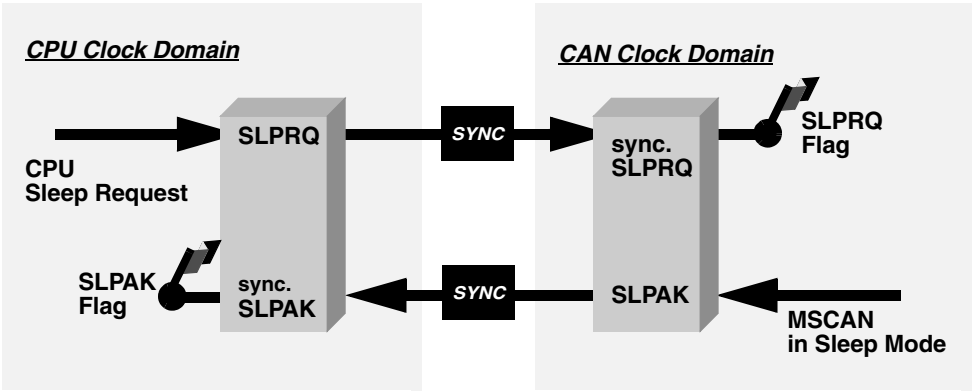


Figure 19-9. Sleep Request / Acknowledge Cycle

NOTE

The application software must avoid setting up a transmission (by clearing one or more TXEx flag(s)) and immediately request Sleep Mode (by setting SLPRQ). It depends on the exact sequence of operations whether the MSCAN starts transmitting or goes into Sleep Mode directly.

If Sleep Mode is active, the SLPRQ and SLPAK bits are set (Figure 19-9). The application software must use SLPAK as a handshake indication for the request (SLPRQ) to go into Sleep Mode.

When in Sleep Mode (SLPRQ=1 and SLPAK=1), the MSCAN stops its internal clocks. However, clocks to allow register accesses from the CPU side still run. If the MSCAN is in Bus-Off state, it stops counting the 128*11 consecutive recessive bits due to the stopped clocks. The TXCAN pin remains in a recessive state. If RXF=1, the message can be read and RXF can be cleared. Shifting a new message into the foreground buffer of the receiver FIFO (RxFG) does not take place while in Sleep Mode. It is possible to access the transmit buffers and to clear the associated TXE flags. No message abort takes place while in Sleep Mode. If the WUPE bit in CANCLT0 is not asserted, the MSCAN will mask any activity it detects on CAN. The RXCAN pin is therefore held internally in a recessive state. This locks the MSCAN in Sleep Mode (Section Figure 19-10., *Simplified State Transitions for Entering/Leaving Sleep Mode*).

The MSCAN is only able to leave Sleep Mode (wake-up) when

- bus activity occurs and WUPE=1 or
- the MCU clears the SLPRQ bit

NOTE

The MCU cannot clear the SLPRQ bit before Sleep Mode (SLPRQ=1 and SLPAK=1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before Sleep Mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN is still in Bus-Off state after Sleep Mode was left, it continues counting the 128*11 consecutive recessive bits.

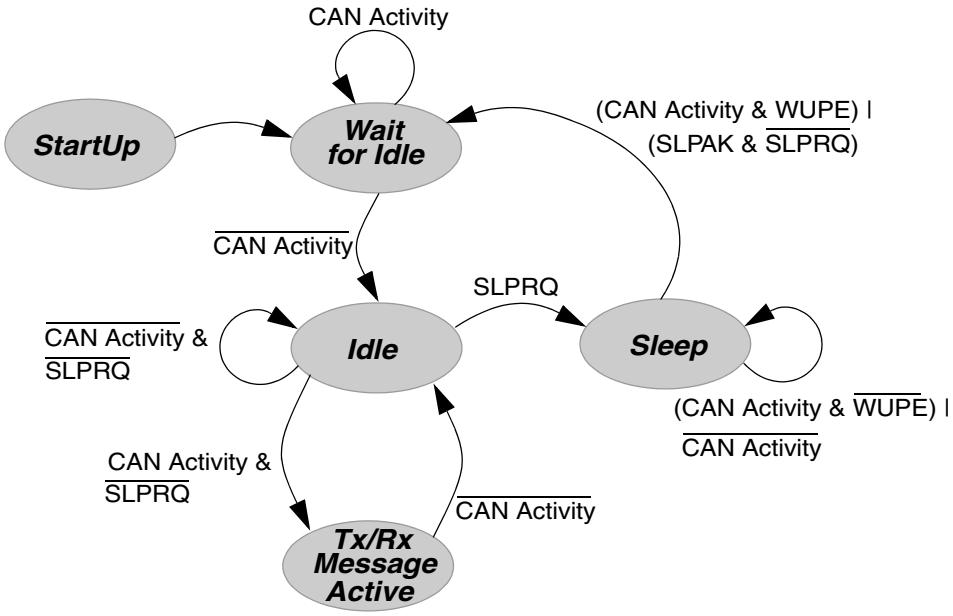


Figure 19-10. Simplified State Transitions for Entering/Leaving Sleep Mode

19.7.8.5 MSCAN Initialization Mode

In Initialization Mode, any ongoing transmission or reception is immediately aborted and synchronization to the bus is lost potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations, the MSCAN immediately drives the TXCAN pin into a recessive state.

NOTE

The user is responsible for ensuring that the MSCAN is not active when Initialization Mode is entered. The recommended procedure is to bring the MSCAN into Sleep Mode (SLPRQ=1 and SLPAK=1) before setting the INITRQ bit in the CANCTL0 register. Otherwise the abort of an ongoing message can cause an error condition and can have an impact on the other bus devices.

In Initialization Mode, the MSCAN is stopped. However, interface registers can still be accessed. This mode is used to reset the CANTCTL0, CANRFLG, CANRIER, CANTFLG, CANTIER, CANTARQ, CANTAAK, CANTBSEL registers to their default values. In addition it enables the configuration of the CANBTR0, CANBTR1 bit timing registers, CANIDAC and the CANIDAR, CANIDMR message filters. Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900* for a detailed description of the Initialization Mode.

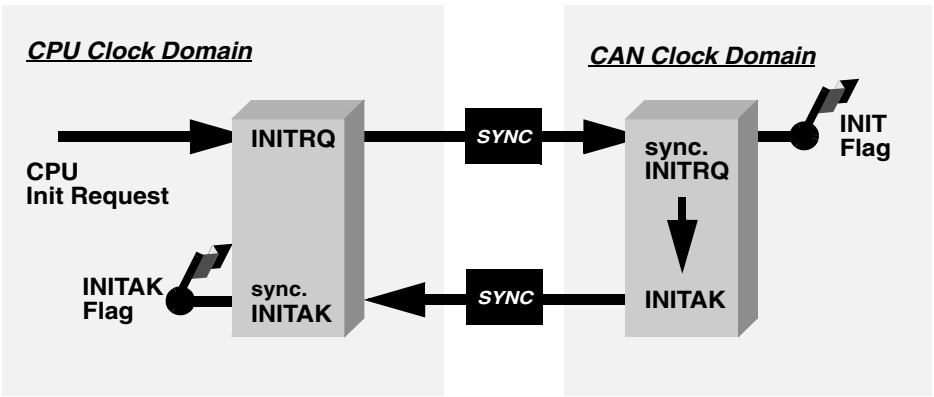


Figure 19-11. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN the INITRQ has to be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (). If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in Initialization Mode the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into Initialization Mode.

NOTE

The MCU cannot clear the INITRQ bit before Initialization Mode (INITRQ=1 and INITAK=1) is active.

19.7.8.6 MSCAN Power Down Mode

The MSCAN is in Power Down Mode when Table 19-35

- the CPU is in Deep Sleep Mode or
- the CPU is in Wait Mode and the CSWAI bit is set.

When entering the Power Down Mode, the MSCAN immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations. To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN immediately drives the TXCAN pin into a recessive state.

NOTE

The user is responsible for ensuring that the MSCAN is not active when Power Down Mode is entered. The recommended procedure is to bring the MSCAN into Sleep Mode before the STOP or WAI instruction (if CSWAI is set) is executed. Otherwise the abort of an ongoing message can cause an error condition and can have an impact on the other bus devices.

In Power Down Mode, all clocks are stopped and no registers can be accessed. If the MSCAN was not in Sleep Mode before Power Down Mode became active, the module would perform an internal recovery cycle after powering up. This causes some fixed delay before the module enters Run Mode again.

19.7.8.7 Programmable Wake-Up Function

The MSCAN can be programmed to wake-up the MSCAN as soon as bus activity is detected (see control bit WUPE in Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900*). The sensitivity to existing bus action can be modified by applying a low-pass filter function to the RXCAN input line while in Sleep Mode (see control bit WUPM in Section 19.5.4, *MSCAN Control Register 1 (CANCTL1)—MBAR + 0x0901*). This feature can be used to protect the MSCAN from wake-up due to short glitches on the CAN bus lines. Such glitches can result e.g. from electromagnetic interference within noisy environments.

19.7.9 Description of Interrupt Operation

The MSCAN supports one interrupt vector mapped onto eight different interrupt sources, any of which can be individually masked (for details see sections Section 19.5.8, *MSCAN Receiver Interrupt Enable Register (CANRIER)—MBAR + 0x0909* to Section 19.5.10, *MSCAN Transmitter Interrupt Enable Register (CANTIER)—MBAR+0x090D*):

19.7.9.1 Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXEx flag of the empty message buffer is set.

19.7.9.2 Receive Interrupt

A message is successfully received and shifted into the foreground buffer (RxFG) of the receiver FIFO. This interrupt is generated immediately after receiving the EOF symbol. The RXF flag is set. If there are multiple messages in the receiver FIFO, the RXF flag is set as soon as the next message is shifted to the foreground buffer.

19.7.9.3 Wake-Up Interrupt

Activity on the CAN bus occurred during MSCAN internal Sleep Mode and WUPE Section 19.5.3, *MSCAN Control Register 0 (CANCTL0)—MBAR + 0x0900* enabled.

19.7.9.4 Error Interrupt

An overrun of the receiver FIFO, error, warning or Bus-Off condition occurred. The Section 19.5.7, *MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908* indicates one of the following conditions:

- Overrun

  An overrun condition of the receiver FIFO as described in Section 19.7.2.3, *Receive Structures* occurred.

- CAN Status Change

  The actual value of the Transmit and Receive Error Counters control the bus state of the MSCAN.
  As soon as the error counters skip into a critical range (Tx/Rx-Warning, Tx/Rx-Error, Bus-Off) the MSCAN flags an error condition. The status change, which caused the error condition, is indicated by the TSTAT and RSTAT flags (see section Section 19.5.7, *MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908* and Section 19.5.8, *MSCAN Receiver Interrupt Enable Register (CANRIER)—MBAR + 0x0909*).

19.7.10 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the Section 19.5.7, *MSCAN Receiver Flag Register (CANRFLG)—MBAR+0x0908* or the Section 19.5.9, *MSCAN Transmitter Flag Register (CANTFLG)—MBAR + 0x090C*. Interrupts are pending as long as one of the corresponding flags is set. The flags in the above registers must be reset within the interrupt handler to handshake the interrupt. The flags are reset by writing a “1” to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

NOTE

It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (*BSET*) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

19.7.11 Recovery from STOP or WAIT

The MSCAN can recover from Sleep Mode via the Wake-Up interrupt. This interrupt can only occur if the MSCAN is in Sleep Mode (SLPRQ=1 and SLPAK=1), the wake-up option is enabled (WUPE=1) and the Wake-Up interrupt is enabled (WUPIE=1).

Notes

Chapter 20
Byte Data Link Controller (BDLC)

20.1 Overview

The BDLC module is a serial communication module which allows the user to send and receive messages across a Society of Automotive Engineers (SAE) J1850 serial communication network. The user's software handles each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

It is recommended that the reader be familiar with the operation and requirements of the SAE J1850 protocol as described in the document "SAE Standard J1850 Class B Data Communications Network Interface" prior to proceeding with this specification.

The BDLC module is designed in a modular structure for use as an IP block. A general working knowledge of the IP bus signals and bus control is assumed in the writing of this document.

20.2 Features

Features of the BDLC module include the following:

- SAE J1850 Class B Data Communications Network Interface Compatible and ISO Compatible for Low-Speed ($\leq$ 125 Kbps) Serial Data Communications in Automotive Applications
- 10.4 Kbps Variable Pulse Width (VPW) Bit Format
- Digital Noise Filter
- Digital Loopback Mode
- 4X Receive and Transmit Mode, 41.6 Kbps, Supported
- BREAK symbol generation Supported
- Block Mode Receive and Transmit Supported
- Collision Detection
- Hardware Cyclical Redundancy Check (CRC) Generation and Checking
- Dedicated Register for Symbol Timing Adjustments
- IP bus Interface
- In-Frame Response (IFR) Types 0, 1, 2, and 3 Supported
- Polling and CPU Interrupt Generation with Vector Lookup Available

20.3 Modes of Operation

The BDLC module has 6 main modes of operation which interact with the power supplies, pins, and the rest of the MCU as shown below.

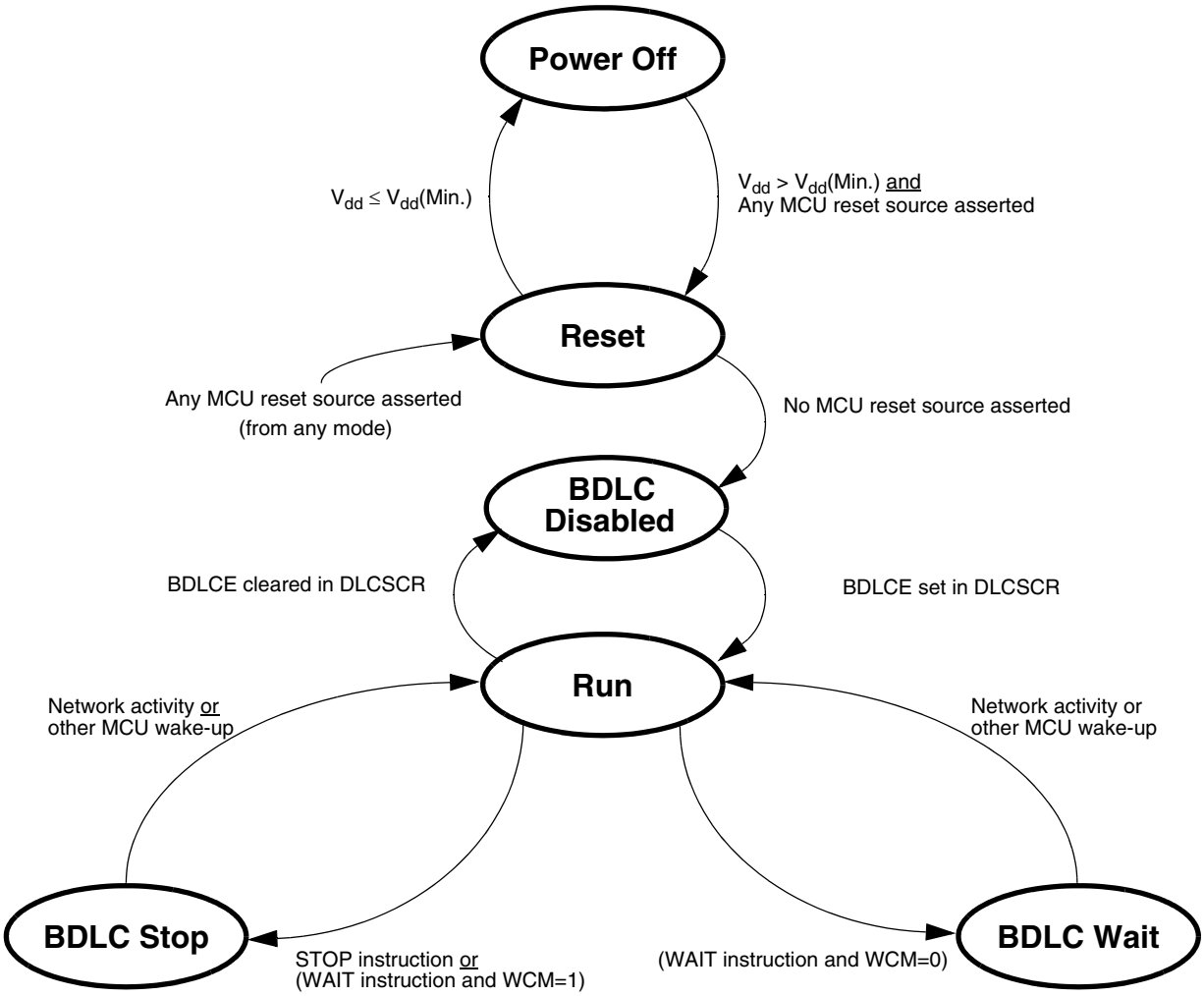


Figure 20-1. BDLC Operating Modes State Diagram

- Power Off

  This mode is entered from the Reset mode whenever the BDLC module supply voltage V_{dd} drops below its minimum specified value for the BDLC module to guarantee operation. The BDLC module will be placed in the Reset mode by a system Low Voltage Reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

- Reset

  This mode is entered from the Power Off mode whenever the BDLC module supply voltage V_{dd} rises above its minimum specified value ($V_{dd(MIN)}$) and some MCU reset source is asserted. To prevent the BDLC from entering an unknown state, the internal MCU reset is asserted while powering up the BDLC module. BDLC Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (e.g. LVR, POR, COP watchdog, Reset pin etc.) is asserted.

  In this mode, the internal BDLC module voltage references are operative, V_{dd} is supplied to the internal circuits, which are held in their reset state and the internal BDLC module system clock is running. Registers will assume their reset condition. Outputs are held in their programmed Reset state, inputs and network activity are ignored.

- BDLC Disabled

  This mode is entered from the Reset mode after all MCU reset sources are no longer asserted. It is entered from the Run mode whenever the BDLCE bit in the BDLC Control Register is cleared.

  In this mode the mux interface clock (f_{bdlc}) is stopped to conserve power and allow the BDLC module to be configured for proper operation on the J1850 bus. The IP bus interface clocks are left running in this mode to allow access to all BDLC module registers for initialization.

- Run

  This mode is entered from the BDLC Disabled mode when the BDLCE bit in the BDLC Control Register is set. It is entered from the BDLC Wait mode whenever activity is sensed on the J1850 bus or some other MCU source wakes the CPU out of Wait mode.

  It is entered from the BDLC Stop mode whenever network activity is sensed or some other MCU source wakes the CPU out of Stop mode. Messages will not be received properly until the clocks have stabilized and the CPU is also in the Run mode.

- BDLC Wait

  This power conserving mode is automatically entered from the Run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the BDLC Control Register 1 register is previously cleared. In this mode, the BDLC module internal clocks continue to run. Any activity on the J1850 network will cause the BDLC module to exit BDLC Wait mode and generate an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the BDLC State Vector Register, encoded as the highest priority interrupt. This interrupt can be cleared by the CPU with a read of the BDLC State Vector Register.

  – Wakeup from BDLC Wait with CPU in WAIT

  If the CPU executes the WAIT instruction and the BDLC module enters the WAIT mode (WCM = 0), the clocks to the BDLC module as well as the clocks in the MCU continue to run. Therefore, the message which wakes up the BDLC module from WAIT and the CPU from WAIT mode will also be received correctly by the BDLC module. This is because all of the required clocks continue to run in the BDLC module in WAIT mode.The wakeup behavior of the BDLC module applies regardless of whether the BDLC module is in normal or 4X mode when the WAIT instruction is executed.

- BDLC Stop

  This power conserving mode is automatically entered from the Run mode whenever the CPU executes a STOP instruction, or if the CPU executes a WAIT instruction and the WCM bit in the BDLC Control Register 1 register is previously set. In this mode, the BDLC internal clocks are stopped. Any activity on the network will cause the BDLC module to exit BDLC Stop mode and generate an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the BDLC State Vector Register, encoded as the highest priority interrupt. This interrupt can be cleared by the CPU with a read of the BDLC State Vector Register. Depending upon which low-power mode instruction the CPU executes to cause the BDLC module to enter BDLC Stop, the message which wakes up the BDLC module (and the CPU) may or may not be received. There are two different possibilities, both of which is described below. These descriptions apply regardless of whether the BDLC module is in normal or 4X mode when the STOP or WAIT instruction is executed.

  – Wakeup from BDLC Stop with CPU in STOP

  When the CPU executes the STOP instruction, all clocks in the MCU, including clocks to the BDLC module, are turned off. Therefore, the message which wakes up the BDLC module and the CPU from STOP mode will not be received. This is due primarily to the amount of time required for the MCU's oscillator to stabilize before the clocks can be applied internally to the other MCU modules, including the BDLC module.

  – Wakeup from BDLC Stop with CPU in WAIT

  If the CPU executes the WAIT instruction and the BDLC module enters the Stop mode (WCM = 1), the clocks to the BDLC module are turned off, but the clocks in the MCU continue to run. Therefore, the message which wakes up the BDLC module from Stop and the CPU from WAIT mode will be received correctly by the BDLC module. This is because very little time is required for the CPU to turn the clocks to the BDLC module back on once the wakeup interrupt occurs.

NOTE

While the BDLC module will correctly receive a message which arrives when the BDLC module is in Stop mode or Wait mode and the MCU is in WAIT mode, if the user enters this mode while a message is being received, the data in the message will become corrupted. This is due to the steps required for the BDLC module to resume operation upon exiting Stop mode or Wait mode, and its subsequent resynchronization with the SAE J1850 bus.

- Digital Loopback

  When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node's internal circuits or elsewhere in the network, including the node's analog physical interface. In this mode, the input to the digital filter is disconnected from the receive pin input (RXB). The input to the digital filter is then connected to the transmitter output to form the loopback connection. The transmit pin (TXB) is negated and will always drive a passive state onto the bus. Digital loopback mode is entered by setting the DLOOP bit in Section 20.7.3.3, *BDLC Control Register 2 (DLCBCR2) - MBAR + 0x1304*.

- Normal and Emulation Mode Operation

  The BDLC module operates in the same manner in all Normal and Emulation Modes. All BDLC module registers can be read and written except those that are reserved, unimplemented, or write once. The user must be careful not to unintentionally write a register when using 16-bit writes in order to avoid unexpected BDLC module behavior.

- Special Mode Operation

  Some aspects of BDLC module operation can be modified in special test mode. This mode is reserved for internal use only.

- Low Power Options

  The BDLC module can save power in Disabled, Wait, and Stop modes. A complete description of what the BDLC module does while in a low power mode can be found in Section 20.3, *Modes of Operation*.

20.4 Block Diagram

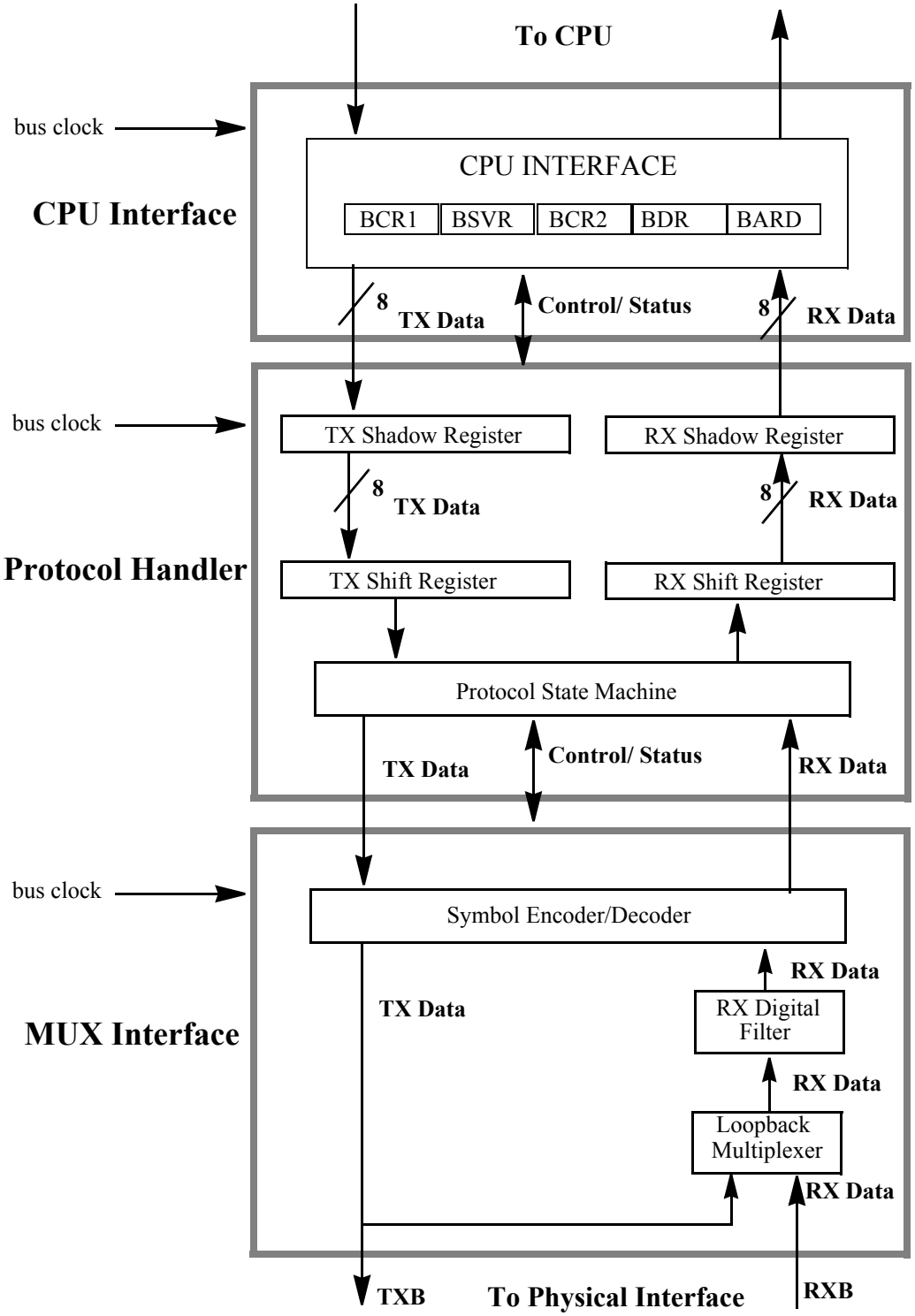


Figure 20-2. BDLC Block Diagram

Figure 20-2 shows the organization of the BDLC module. The Buffers provide storage for data received and data to be transmitted onto the J1850 bus. The Protocol Handler is responsible for the encoding and decoding of data bits and special message symbols during transmission

and reception. The MUX Interface provides the link between the BDLC digital section and the analog Physical Interface. The wave shaping, driving and digitizing of data is performed by the Physical Interface.

NOTE

The Physical Interface is not implemented in the BDLC module and must be provided externally.

The main functional blocks of the BDLC module are explained in greater detail in the following sections.

Use of the BDLC module in message networking fully implements the “SAE Standard J1850 Class B Data Communication Network Interface” specification.

20.5 Signal Description

20.6 Overview

The BDLC module has a total of 2 external pins.

20.6.1 Detailed Signal Descriptions

20.6.1.1 TXB - BDLC Transmit Pin

The TXB pin serves as the transmit output channel for the BDLC module.

20.6.1.2 RXB - BDLC Receive Pin

The RXB pin serves as the receive input channel for the BDLC module.

20.7 Memory Map and Registers

20.7.1 Overview

This section provides a detailed description of all memory and registers accessible to the end user.

20.7.2 Module Memory Map

Table 20-1. Module Memory Map

Address	Use	Access
MBAR + 0x1300	BDLC Control Register 1 (DLCBCR1)	R/W
MBAR + 0x1301	BDLC State Vector Register (DLCBSVR)	R
MBAR + 0x1304	BDLC Control Register 2 (DLCBCR2)	R/W
MBAR + 0x1305	BDLC Data Register (DLCBDR)	R/W
MBAR + 0x1308	BDLC Analog Round Trip Delay Register (DLCBARD)	R/W
MBAR + 0x1309	BDLC Rate Select Register (DLCBRSR)	R/W
MBAR + 0x130C	BDLC Control Register (DLCSCR)	R/W
MBAR + 0x130D	BDLC Status Register (DLCBSTAT)	R

20.7.3 Register Descriptions

20.7.3.1 BDLC Control Register 1 (DLCBCR1)—MBAR + 0x1300

This register is used to configure and control the BDLC module.

Table 20-2. BDLC Control Register 1

	msb 0	1	2	3	4	5	6	7 lsb
R	IMSG	CLKS	0	0	0	0	IE	WCM
W								
RESET:	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

READ: any time

WRITE: IMSG, IE, and WCM any time.

CLKS write once in normal and emulation modes.

CLKS bit has modified functionality in special test mode.

Writes to unimplemented bits 5-2 are ignored.

IMSG — Ignore Message (Bit 7)

This bit allows the CPU to ignore messages by disabling updates of the BDLC State Vector Register register until a new Start of Frame (SOF) or a BREAK symbol is detected. BDLC module transmitter and receiver operation are unaffected by the state of the IMSG bit.

1 = Disable BDLC State Vector Register Updates. When set, all BDLC interrupt sources (exceptions are described below) will be prevented from updating BDLC State Vector Register status bits. Setting IMSG does not clear pending interrupt flags, the behavior of which will still be as described in Section , *BDLC State Vector Register (DLCBSVR)*. If this bit is set while the BDLC is receiving or transmitting a message, state vector register updates will be inhibited for the rest of the message.

0 = Enable BDLC State Vector Register Updates. This bit is automatically cleared by the reception of a SOF symbol or a BREAK symbol. It will then allow updates of the state vector register to occur.

There are two situations in which interrupts will not be masked by the IMSG bit: when a wakeup interrupt occurs; and when a receiver error occurs which causes a byte pending transmission to be flushed from the transmit shadow register. See Section 20.7.3.4, *BDLC Data Register (DLCBDR) - MBAR + 0x1305* for a description of the conditions which cause a pending transmission to be flushed.

CLKS — Clock Select (Bit 6)

The nominal BDLC operating frequency (mux interface clock frequency - f_{bdlc}) **must** always be 1.048576 MHz or 1 MHz in order for J1850 bus communications to take place properly. The CLKS register bit is provided to allow the user to indicate to the BDLC module which frequency (1.048576 MHz or 1 MHz) is used so that each symbol time can be automatically adjusted.

The CLKS bit is a write once bit. All writes to this bit will be ignored after the first one.

Binary frequency (1.048576 MHz) is used for f_{bdlc}.

Integer frequency (1 MHz) is used. for f_{bdlc}

Section 20.8.1.3, *J1850 VPW Valid/Invalid Bits & Symbols* describes the transmitter and receiver VPW symbol timing for integer and binary frequencies.

IE — Interrupt Enable (Bit 1)

This bit determines whether the BDLC module will generate CPU interrupt requests. It does not affect CPU interrupt requests when exiting the BDLC module Stop or Wait modes. Interrupt requests will be maintained until all of the interrupt request sources are cleared, by performing the specified actions upon the BDLC module's registers. Interrupts that were pending at the time that this bit is cleared may be lost.

1 = Enable interrupt requests from BDLC module

0 = Disable interrupt requests from BDLC module

If the programmer does not wish to use the interrupt capability of the BDLC module, the BDLC State Vector Register (BDLC State Vector Register) can be polled periodically by the programmer to determine BDLC module states. Refer to Section 20.7.3.2, *BDLC State Vector Register (DLCBSVR) - MBAR + 0x1300* for a description of BDLC State Vector Register register and how to clear interrupt requests.

WCM — Wait Clock Mode (Bit 0) *(Provided CPU has Low Power Mode Options)*

This bit determines how the BDLC module responds when the CPU enters WAIT mode. As described in Section 20.3, *Modes of Operation*, the BDLC module can respond by either entering BDLC_STOP mode, where all internal clocks are stopped, or entering BDLC_WAIT mode where internal clocks are allowed to run.

1 = Stop BDLC internal clocks during CPU wait mode (BDLC_STOP)

0 = Run BDLC internal clocks during CPU wait mode (BDLC_WAIT)

20.7.3.2 BDLC State Vector Register (DLCBSVR) - MBAR + 0x1300

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a MUX protocol. It provides a index offset that is directly related to the BDLC module's current state, which can be used with a user supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Table 20-3. BDLC State Vector Register

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	I3	I2	I1	I0	0	0
W								
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

READ: any time

WRITE: ignored

I[3:0] — Interrupt State Vector (Bits 5- 2)

These bits indicate the source of the interrupt request that is currently pending.

Table 1-1. Interrupt Summary

BSVR	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No Interrupts Pending	0 (Lowest)
$04	0	0	0	1	Received EOF	1
$08	0	0	1	0	Received IFR byte	2
$0C	0	0	1	1	Rx data register full	3
$10	0	1	0	0	Tx data register empty	4
$14	0	1	0	1	Loss of arbitration	5
$18	0	1	1	0	CRC error	6
$1C	0	1	1	1	Symbol invalid or out of range	7
$20	1	0	0	0	Wakeup	8 (Highest)

The state encoding of the interrupt sources mean that only one interrupt source is dealt with at a time. Once the highest priority interrupt source is dealt with, if another interrupt event of a lower priority has also occurred, the value corresponding to that interrupt source appears in the BDLC State Vector Register. This continues until all BDLC interrupt sources have been dealt with and all bits in the BDLC State Vector Register are cleared.

- Wakeup

  The BDLC has two different power-conserving modes, stop and wait. Wakeup from these modes is described below.
- Wakeup from BDLC Wait with CPU in Wait

  If the CPU executes a WAIT instruction and the BDLC enters the BDLC wait mode, the clocks to the BDLC as well as the clocks in the MCU continue to run. The message which generates a Wake-up interrupt of the BDLC and the CPU will be received correctly.
- Wakeup from BDLC Stop with CPU in Wait

  If the CPU executes a WAIT instruction and the BDLC enters the BDLC stop mode, the clocks to the BDLC are turned off, but the clocks in the MCU continue to run. The message which generates a Wake-up interrupt of the BDLC and the CPU will be received correctly. To ensure this, the EOF following the last message appearing on the bus must be received; otherwise, the message will not be received correctly.
- Wakeup from BDLC Stop with CPU in Stop

If the CPU executes a STOP all clocks to the BDLC as well as the clocks in the MCU are turned off including clocks to the BDLC. The message which generates a Wake-up interrupt of the BDLC and the CPU will not be received correctly.

- Symbol Invalid or Out of Range
- CRC Error

The Cyclical Redundancy Check Byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. If the message is not error free, the CRC error status is shown in the BDLC State Vector Register.

- Loss of Arbitration

The Loss of Arbitration status is entered when a loss of arbitration occurs while the BDLC is transmitting onto the bus.

- Tx Data Register Empty

The Tx Data Register Empty (TDRE) Byte is used to tell when data has been unloaded from the BDLC Data Register.

- Rx Data Register Full

The Rx Data Register Full (RDRF) Byte is used to tell when data has been loaded in the BDLC Data Register.

- Received IFR Byte

The BDLC can transmit and receive all four types of in-frame responses. As each byte of an IFR is received, the BDLC State Vector Register indicates this by setting this state.

- Received EOF

When a 280us passive period on the bus is received, it signifies an EOF. Whenever this occurs, the EOF flag is set.

- No Interrupts Pending

This interrupt cannot generate an interrupt of the CPU.

20.7.3.3 BDLC Control Register 2 (DLCBCR2) - MBAR + 0x1304

This register controls transmitter operations of the BDLC module.

Table 20-4. BDLC Control Register 2

	msb 0	1	2	3	4	5	6	7 lsb
R	SMRST	DLOOP	4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
W								
RESET:	0	1	0	0	0	0	0	0

READ: any time

WRITE: any time

SMRST — State Machine Reset (Bit 7)

The programmer can use this bit to reset the BDLC state machines to an initial state after the user put the off-chip analog transceiver in loop back mode.

1 = Setting SMRST arms the state machine reset generation logic. Setting SMRST does not affect BDLC module behavior in any way.

0 = Clearing SMRST after it has been set will cause the generation of a state machine reset. After SMRST is cleared, the BDLC requires the bus to be idle for a minimum of an End of Frame symbol (EOF) time before allowing the reception of a message. The BDLC requires the bus to be idle for a minimum of an Inter-Frame Separator symbol (IFS) time before allowing any message to be transmitted.

DLOOP — Digital Loopback Mode (Bit 6)

This bit determines the source to which the input of the digital filter is connected and can be used to isolate bus fault conditions. If a fault condition has been detected on the bus, this control bit allows the programmer to disconnect the digital filter from input from the receive pin (RXB) and connect it to the transmit output to the pin (TXB). In this configuration, data sent from the transmit buffer should be reflected back into the receive buffer. If no faults exist in the digital block, the fault is in the physical interface block or elsewhere on the J1850 bus.

1 = When set, digital filter input is connected to the transmitter output. The BDLC module is now in Digital Loopback Mode of operation. The transmit pin (TXB) is driven low and not driven by the transmitter output.

0 = When cleared, digital filter input is connected to receive pin (RXB) and the transmitter output is connected to the transmit pin (TXB). The BDLC module is taken out of Digital Loopback Mode and can now drive and receive from the J1850 bus normally. After writing DLOOP to zero, the BDLC module requires the bus to be idle for a minimum of an End of Frame symbol time before allowing a reception of a message. The BDLC module requires the bus to be idle for a minimum of an Inter-Frame Separator symbol time before allowing any message to be transmitted.

NOTE

The DLOOP bit is a fault condition aid and should never be altered after the BDLC Data Register is loaded for transmission. Changing DLOOP during a transmission may cause corrupted data to be transmitted onto the J1850 network.

4XE — 4X Mode Enable (Bit 5)

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbps) or in 4X Mode at 41.6 kbps. This feature is useful for fast download of data into a J1850 node for diagnostic or factory programming of the node.

1 = When set, the BDLC module is put in 4X (41.6 kbps) operation.

0 = When cleared, the BDLC module transmits and receives at 10.4 kbps. Reception of a BREAK symbol automatically clears this bit and sets the symbol invalid or out of range flag BDLC State Vector Register = $1C).

The effect of 4X receive operation on receive symbol timing boundaries is described in Section 20.8.1.3, *J1850 VPW Valid/Invalid Bits & Symbols*.

NBFS — Normalization Bit Format Select (Bit 4)

This bit controls the format of the Normalization Bit (NB). SAE J1850 strongly encourages the use of an active long: ‘0’ for In-Frame Responses containing CRC and active short, ‘1’ for In-Frame Responses without CRC.

1 = NB that is received or transmitted is a ‘0’ when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘1’ when the response part of an In-Frame Response (IFR) does not end with a CRC byte.

0 = NB that is received or transmitted is a ‘1’ when the response part of an In-Frame Response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘0’ when the response part of an In-Frame Response (IFR) does not end with a CRC byte.

TEOD — Transmit End of Data (Bit 3)

This bit is set by the programmer to indicate the end of a message being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte in the Tx Shift Register followed by the EOD symbol. If the transmit shadow register (refer to Section 20.8.3.1, *Protocol Architecture* for a description of the transmit shadow register) is full when TEOD is set, the CRC byte and EOD will be transmitted after the current byte in the Tx Shift Register and the byte in the Tx Shadow Register have been transmitted. Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC State Vector Register (BDLC State Vector Register) is cleared to allow lower priority interrupts to occur. This bit is also used to end an IFR. Bits TSIFR, TMIFR1, and TMIFR0 determine whether a CRC byte is appended before EOD transmission for IFRs.

1 = Transmit EOD symbol.

0 = The TEOD bit will be automatically cleared after the first CRC bit is sent, or if an error or loss of arbitration is detected on the bus. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol, or an error condition or loss of arbitration occurs.

TSIFR, TMIFR1, TMIFR0 — Transmit In-Frame Response Control (Bits 2-0)

These three bits control the type of In-Frame Response being sent. The programmer should not set more than one of these control bits to a one at any given time. However, if more than one of these three control bits are set to one, the priority encoding logic will force the internal register bits to a known value as shown in the following table. But, when these bits are read, they will be the same as written earlier. For instance, if “011” is written to TSIFR, TMIFR1, TMIFR0, then internally, they’ll be encoded as “010”. However, when these bits are later read back, it’ll still be “011”.

Table 1-2. Transmit In-Frame Response Control Bit Priority Encoding

WRITE			READ			ACTUAL (internal register)		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0	0	0	0
1	X	X	1	X	X	1	0	0
0	1	X	0	1	X	0	1	0
0	0	1	0	0	1	0	0	1

The BDLC supports the In-frame Response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown in Figure 20-3. The purpose of the in-frame response modes is to allow single or multiple nodes to acknowledge receipt of the data by responding to a received message after they have seen the EOD symbol. For VPW modulation, the first bit of the IFR is always passive; therefore, an active normalization bit must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync their responses.

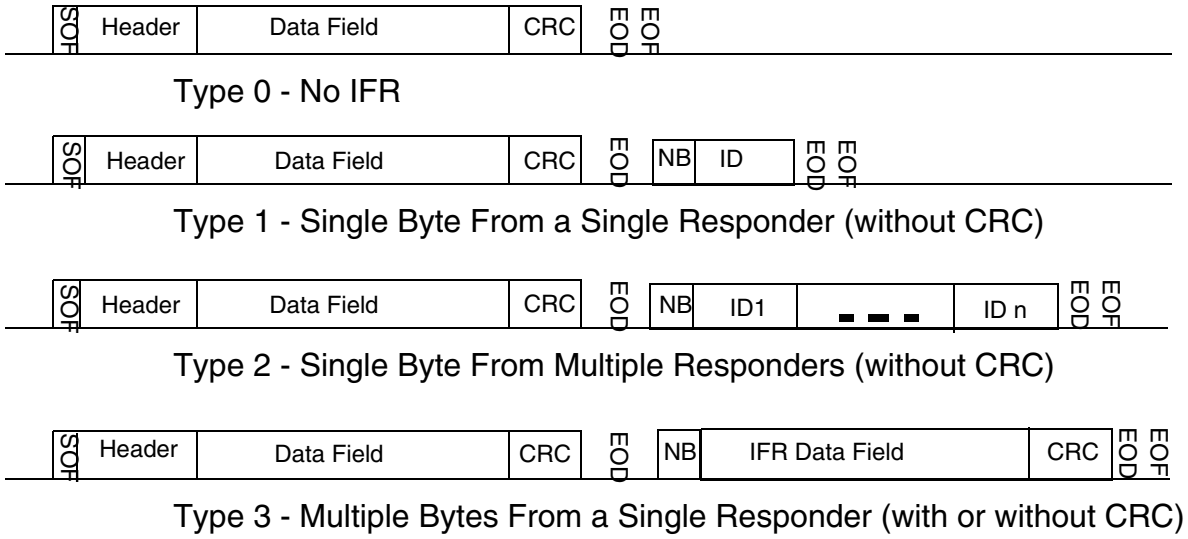


Figure 20-3. Types of In-Frame Response

TSIFR — Transmit Single Byte IFR with no CRC (Type 1 or 2)

This bit is used to request the BDLC to transmit the byte in the BDLC Data Register as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC module will attempt to transmit the appropriate normalization bit followed by the byte in the BDLC Data Register.

0 = The TSIFR bit will be automatically cleared once the EOD following one or more IFR bytes has been received or an error is detected on the bus.

The user must set the TSIFR bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TSIFR bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TSIFR bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a loss of arbitration occurs when the BDLC module attempts transmission, after the IFR byte winning arbitration completes transmission, the BDLC module will again attempt to transmit the byte in the BDLC Data Register (with no normalization bit). The BDLC module will continue transmission attempts until an error is detected on the bus, or TEOD is set by the CPU, or the BDLC transmission is successful.

NOTE

Setting the TEOD bit before transmission of the IFR byte will direct the BDLC to make only one attempt at transmitting the byte.

If loss of arbitration occurs in the last bit of the IFR byte, two additional ‘1’ bits **will not** be sent out because the BDLC will attempt to retransmit the byte in the transmit shift register after the IFR byte winning arbitration completes transmission.

TMIFR1 — Transmit Multiple Byte IFR with CRC (Type 3)

This bit requests the BDLC module to transmit the byte in the BDLC Data Register (BDLC Data Register) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC.Response IFR bytes are still subject to J1850 message length maximums.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC module will attempt to transmit the appropriate normalization bit followed by IFR bytes.The programmer should set TEOD after the last IFR byte has been written into BDLC Data Register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

0 = The TMIFR1 bit will be automatically cleared once the BDLC module has successfully transmitted the CRC byte and EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration.

After the byte in the BDLC Data Register has been loaded into the transmit shift register, the TDRE flag will be set in the BDLC State Vector Register register, similar to the main message transmit sequence. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated.The programmer should then load the next byte of the IFR into the BDLC Data Register for transmission. When the last byte of the IFR has been loaded into the BDLC Data Register, the programmer should set the TEOD bit in the BDLC control register 2. This will instruct the BDLC module to transmit a CRC byte once the byte in the BDLC Data Register is transmitted, and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if the programmer wishes to transmit a single byte followed by a CRC byte, the programmer should load the byte into the BDLC Data Register and then set the TMIFR1 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in BDLC Control Register 2. This will result in the byte in the BDLC Data Register being the only byte transmitted before the IFR CRC byte.

The user must set the TMIFR1 bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TMIFR1 bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR1 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by the programmer not writing another byte to the BDLC Data Register following the TDRE flag being set) the BDLC module will automatically disable the transmitter after the byte currently in the shifter plus two extra 1-bits have been transmitted. The receiver will pick this up as an framing error and relay it in the State Vector Register as an invalid symbol error. The TMIFR1 bit will also be cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR with CRC, the BDLC module will go to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR1 bit will be cleared and no attempt will be made to retransmit the byte in the BDLC Data Register. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) **will** be sent out.

NOTE

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.

TMIFR0 — Transmit Multiple Byte IFR with no CRC (Type 3)

This bit is used to request the BDLC module to transmit the byte in the BDLC Data Register as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC module will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into BDLC Data Register. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the BDLC Data Register.

0 = The TMIFR0 bit will be automatically cleared once the BDLC module has successfully transmitted the EOD symbol, by the detection of an error on the multiplex bus, a transmitter underrun, or loss of arbitration.

After the byte in the BDLC Data Register has been loaded into the transmit shift register, the TDRE flag will be set in the BDLC State Vector Register register, similar to the main message transmit sequence. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. The programmer should then load the next byte of the IFR into the BDLC Data Register for transmission. When the last byte of the IFR has been loaded into the BDLC Data Register, the programmer should set the TEOD bit in the BDLC Control Register 2. This will instruct the BDLC to transmit an EOD symbol, indicating the end of the IFR portion of the message frame. The BDLC module will not append a CRC.

However, if the programmer wishes to transmit a single byte, the programmer should load the byte into the BDLC Data Register and then set the TMIFR0 bit before the EOD symbol has been received. Once the TDRE flag is set and interrupt occurs (if enabled), the programmer should then set the TEOD bit in BDLC Control Register 2. This will result in the byte in the BDLC Data Register being the only byte transmitted.

The user must set the TMIFR0 bit before the EOF following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node transmits an IFR to this message, the user must set the TMIFR0 bit before the normalization bit is received or no IFR transmit attempts will be made for the message. If another node does transmit a successful IFR or a reception error occurs, the TMIFR0 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

If a transmitter underrun error occurs during transmission (caused by the programmer not writing another byte to the BDLC Data Register following the TDRE flag being set) the BDLC module will automatically disable the transmitter after the byte currently in the shifter plus two extra 1-bits have been transmitted. The receiver will pick this up as an framing error and relay it in the State Vector Register as an invalid symbol error. The TMIFR0 bit will also be cleared.

If a loss of arbitration occurs when the BDLC module is transmitting a multiple byte IFR without CRC, the BDLC module will go to the loss of arbitration state, set the appropriate flag and cease transmission. The TMIFR0 bit will be cleared and no attempt will be made to retransmit the byte in the BDLC Data Register. If loss of arbitration occurs in the last bit of the IFR byte, two additional one bits (a passive long followed by an active short) **will** be sent out.

NOTE

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.

20.7.3.4 BDLC Data Register (DLCBDR) - MBAR + 0x1305

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC module. It is also used to pass data received from the J1850 bus to the CPU.

Table 20-5. BDLC Data Register

	msb 0	1	2	3	4	5	6	7 lsb
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
RESET:	0	0	0	0	0	0	0	0

READ: any time

WRITE: any time

D7:D0 — Receive/Transmit Data (Bits 7 - 0)

While transmitting, each data byte (after the first one) should be written only after a “Tx Data Register Empty” (TDRE) interrupt has occurred, or the BDLC State Vector Register register has been polled indicating this condition.

Data read from this register will be the last data byte received from the J1850 bus. This received data should only be read after a “Rx Data Register Full” (RDRF) or “Received IFR byte” (RXIFR) interrupt has occurred or the BDLC State Vector Register register has been polled indicating either of these two conditions.

The BDLC Data Register is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next byte of data.

The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag (or RXIFR flag if the received byte is part of an IFR) is set to indicate that a new byte of data has been received. The programmer has one BDLC module byte reception time to read the shadow register and clear the RDRF or RXIFR flag before the shadow register is overwritten by the newly received byte.

If the user writes the first byte of a message to be transmitted to the BDLC Data Register and then determines that a different message should be transmitted, the user can write a new byte to the BDLC Data Register up until the transmission begins.This new byte will replace the original byte in the BDLC Data Register.

From the time a byte is written to the BDLC Data Register until it is transferred to the transmit shift register, the transmit shadow register is considered full and the byte pending transmission. If one of the IFR transmission control bits (TSIFR, TMIFR1, or TMIFR0 in BDLC Control Register 2) is also set, the byte is pending transmission as an IFR. A byte pending transmission will be flushed from the transmit shadow register and the transmission canceled if one of the following occurs: a loss of arbitration or transmitter error on the byte currently being transmitted; a symbol error, framing error, bus fault, or BREAK symbol is received. If the byte pending transmission is an IFR byte, the reception of a message with a CRC error will also cause the byte in the transmit shadow register to be flushed.

To abort an in-progress transmission, the programmer should simply stop loading more data into the BDLC Data Register. This will cause a transmitter underrun error and the BDLC module will automatically disable the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte (plus two extra 1-bits) has been transmitted. The receiver will pick this up as an error and relay it in the State Vector Register as an invalid symbol error.

20.7.3.5 BDLC Analog Round Trip Delay Register (DLCBARD) - MBAR + 0x1308

This register is used to program the BDLC module so that it compensates for the round trip delays of different external transceivers. Also the polarity of the receive pin (RXB) is set in this register.

Table 20-6. BDLC Analog Round Trip Delay Register

	msb 0	1	2	3	4	5	6	7 lsb
R	0	RXPOL	0	BO4	BO3	BO2	BO1	BO0
W								
RESET:	0	1	0	1	0	0	0	0

= Unimplemented

READ: any time

WRITE: write once in normal and emulation modes.

Register functionality modified in special test mode.

Writes to unimplemented bits 7, 5 are ignored.

RXPOL — Receive Pin Polarity (Bit 6)

The Receive pin Polarity bit is used to select the polarity of incoming signal on the receive pin. Some external analog transceiver inverts the receive signal from the J1850 bus before feeding back to the digital receive pin.

1 = Select normal/true polarity; true non-inverted signal from J1850 bus, i.e., the external transceiver does not invert the receive signal.

0 = Select inverted polarity, where external transceiver inverts the receive signal.

BO4-BO0 — BDLC Analog Roundtrip Delay Offset Field (Bits 4-0)

BO[4:0] adjust the transmitted symbol timings to account for the differing roundtrip delays found in different SAE J1850 analog transceivers. The allowable delay range is from 0 µs to 31 µs, with a nominal target of 16 µs (reset value). Refer to Table 20-7 for the BO[4:0] values corresponding to the expected transceiver delays and the resultant transmitter timing adjustment (in mux interface clock periods (t_{bdlc})). Refer to the analog transceiver device specification for the expected roundtrip delay through both the transmitter and the receiver. The sum of these two delays makes up the total roundtrip delay value.

NOTE

For Digital Loopback test, the Analog Roundtrip Delay Offset Field should be set to 0µs.

Table 20-7. BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment

BARD Offset Bits BO[4:0]	Corresponding Expected Transceiver's delays (µs)	Transmitter Symbol Timing Adjustment (t_{bdlc})[1]
00000	0	0
00001	1	1
00010	2	2
00011	3	3
00100	4	4
00101	5	5
00110	6	6
00111	7	7
01000	8	8
01001	9	9
01010	10	10
01011	11	11
01100	12	12
01101	13	13
01110	14	14

Table 20-7. BARD Values vs. Transceiver Delay and Transmitter Timing Adjustment (continued)

BARD Offset Bits BO[4:0]	Corresponding Expected Transceiver's delays (µs)	Transmitter Symbol Timing Adjustment (t_{bdlc})[1]
01111	15	15
10000	16	16
10001	17	17
10010	18	18
10011	19	19
10100	20	20
10101	21	21
10110	22	22
10111	23	23
11000	24	24
11001	25	25
11010	26	26
11011	27	27
11100	28	28
11101	29	29
11110	30	30
11111	31	31

Note:
1. The transmitter symbol timing adjustment is the same for binary and integer bus frequencies.

20.7.3.6 BDLC Rate Select Register (DLCBRSR) - MBAR + 0x1309

This register determines the divider prescaler value for the mux interface clock (f_{bdlc}). Only integer multiple of the 1 MHz or 1.048576 MHz f_{bdlc} are supported as input clock.

Table 20-8. BDLC Rate Select Register

	msb 0	1	2	3	4	5	6	7 lsb
R	R7	R6	R5	R4	R3	R2	R1	R0
W								
RESET:	0	0	0	0	0	0	0	0

READ: any time

WRITE: write once in normal and emulation modes.

R7-R0 — Rate Select (Bits 7-0)

These bits determine the amount by which the frequency of the system clock signal is divided to generate the MUX Interface clock (f_{bdlc}) which defines the basic timing resolution of the MUX Interface. The value programmed into these bits is dependent on the chosen system clock frequency. See Table 20-9 and Table 20-10 for example rate selects for different bus frequencies. All divisor values from divide by 1 to divide by 256 are possible, but are not shown in the tables.

NOTE

Although the maximum divider is 256, a divider which will generate a 1 MHz or 1.048576 MHz f_{bdlc} must be selected in order for J1850 communications to occur.

Table 20-9. BDLC Rate Selection for Binary Frequencies [CLKS = 1]

IP bus clock frequency	R[7:0]	division	f_{bdlc}
f_{CLOCK}=1.048576 MHz	$00	1	1.048576 MHz

Table 20-10. BDLC Rate Selection for Integer Frequencies [CLKS = 0]

IP bus clock frequency	R[7:0]	division	f_{bdlc}
f_{CLOCK}=132.00000 MHz	$83	132	1.000000 MHz
f_{CLOCK}=66.00000 MHz	$41	66	1.000000 MHz
f_{CLOCK}=54.00000 MHz	$35	54	1.000000 MHz
f_{CLOCK}=33.00000 MHz	$20	33	1.000000 MHz
f_{CLOCK}=27.00000 MHz	$1A	27	1.000000 MHz

20.7.3.7 BDLC Control Register (DLCSCR) - MBAR + 0x130C

The following register enables the BLDC module.

Table 20-11. BDLC Control Register

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	BDLCE	0	0	0	BREAK
W								
RESET:	0	0	0	0	0	0	0	0

= Unimplemented

READ: any time

WRITE: any time

BDLCE — BDLC Enable (Bit 4)

This bit serves as a mux interface clock (f_{bdlc}) enable/disable for power savings.

1 = The mux interface clock (f_{bdlc}) and BDLC module are enabled to allow J1850 communications to take place.

0 = The mux interface clock (f_{bdlc}) is disabled, shutting down the BDLC module for power saving. Bus clocks are still running allowing registers to be accessed.

BREAK — send BREAK signal (Bit 0)

This bit determines whether the BDLC module will generate a BREAK symbol.

1 = The BDLC module will immediately send a Break signal on the bus, regardless of its current transmit or receive status.

0 = The BDLC module does not generate a BREAK symbol.

After setting the BREAK bit it will automatically be cleared after two IPB clock cycles.

The active Break signal causes any other transmitting module to stop transmitting immediately because it loses arbitration. It is at least 240 µs long.

When the BDLC is operating at the high bus speed all 4X symbol times are one fourth that shown, except for Break, which will be transmitted the same length in 1X or 4X mode.

20.7.3.8 BDLC Status Register (DLCBSTAT) - MBAR + 0x130D

This register Indicates the status of the BLDC module.

Table 20-12. BDLC Status Register

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	0	0	IDLE
W	Unimplemented					Reserved	Unimplemented	
RESET:	0	0	0	0	0	0	0	0

= Unimplemented

READ: any time

WRITE: ignored in normal and emulation modes

Register functionality is modified in special test mode.

IDLE Idle (Bit 0)

This bit indicates when the BDLC module is idle.

1 = BDLC module has received IFS and no data is being transmitted or received.

0 = BDLC module is either transmitting or receiving data.

NOTE

BDLC module is only idle after receiving IFS. The IDLE bit is 0 during reset since the BDLC module needs to wait for an IFS before becoming idle. Noise on the bus will be filtered and the IDLE bit will remain unchanged.

20.8 Functional Description

20.8.1 General

The BDLC module is a serial communication module which allows the user to send and receive messages across a Society of Automotive Engineers (SAE) J1850 serial communication network. The user's software handles each transmitted or received message on a byte-by-byte basis, while the BDLC performs all of the network access, arbitration, message framing and error detection duties.

20.8.1.1 J1850 Frame Format

As noted above and in Section 20.2, *Features*, the BDLC module communicates across an SAE J1850 network. As such, all messages transmitted on the J1850 bus are structured using the format below. The following sections describe this format and it's meanings.

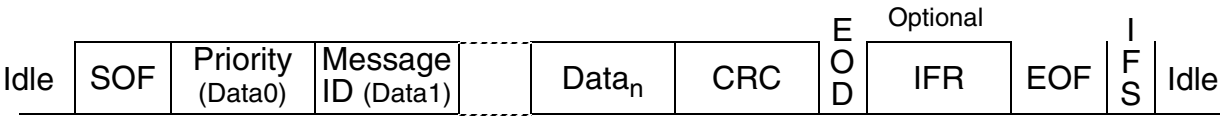


Figure 20-4. J1850 Bus Message Format (VPW)

SAE J1850 states that each message has a maximum length of 101 bit times or 12 bytes (excluding SOF, EOD, NB and EOF).

- SOF - Start of Frame Symbol

  All messages transmitted onto the J1850 bus must begin with an long active SOF symbol. This indicates to any listeners on the J1850 bus the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

- Data - In Message Data Bytes

  The data bytes contained in the message include the message priority/type, message I.D. byte, and any actual data being transmitted to the receiving node. See SAE J1850 - Class B Data Communications Network Interface, for more information about 1 and 3 Byte Headers.

Messages transmitted by the BDLC module onto the J1850 bus must contain at least one data byte, and therefore can be as short as one data byte and one CRC byte. Each data byte in the message is 8 bits in length, transmitted MSB to LSB.

- CRC - Cyclical Redundancy Check Byte

  This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus, and also performs CRC detection on any messages it receives from the J1850 bus.

  CRC generation uses the divisor polynomial $X^8+X^4+X^3+X^2+1$. The remainder polynomial is initially set to all ones, and then each byte in the message after the SOF symbol is serially processed through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes, in MSB to LSB order.

  When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and EOD symbols, but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial will equal $X^7+X^6+X^2$ ($C4), regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC will recognize this as a CRC error and set the CRC error flag in the BDLC State Vector Register.
- EOD - End of Data Symbol

  The EOD symbol is a long passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.
- IFR - In Frame Response Bytes

  The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the "SAE J1850 Class B Data Communications Network Interface" specification.
- EOF - End of Frame Symbol

  This symbol is a passive period on the J1850 bus, longer than an EOD symbol, which signifies the end of a message. Since an EOF symbol is longer than an EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.
- IFS - Inter-Frame Separation Symbol

  The IFS symbol is a passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node following the completion of the EOF period.

  When the last byte of a message has been transmitted onto the J1850 bus, and the EOF symbol time has expired, all nodes must then wait for the IFS symbol time to expire before transmitting an SOF, marking the beginning of another message.

  However, if the BDLC module is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it will internally synchronize to that edge.

  A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. Receivers must synchronize to any SOF occurring during an IFS period to allow for individual clock tolerances.
- Break

  If the BDLC module is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred, and halts transmission.The BDLC module can transmit a BREAK symbol. If while receiving a message the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error and sets the invalid symbol flag. If while receiving a message in 4X mode, the BDLC module detects a BREAK symbol, it treats the BREAK as a reception error, sets BDLC State Vector Register register to $1C, and exits 4X mode.The 4XE bit in BDLC Control Register 2 is automatically cleared upon reception of the BREAK symbol.
- Idle Bus

  An idle condition exists on the bus during any passive period after expiration of the IFS period. Any node sensing an idle bus condition can begin transmission immediately.

20.8.1.2 J1850 VPW Symbols

Variable Pulse Width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions, and by the level of the bus between transitions, active or passive. Active and passive bits are used alternately. This encoding technique is used to reduced the number of bus transitions for a given bit rate. See Section 20.2, *Features*.

The symbol values shown below are nominal values. Refer to the electrical specification for a more complete description of symbol values. Each logic one or logic zero contains a single transition, and can be at either the active or passive level and one of two lengths, either 64µs or 128µs (T_{NOM} at 10.4kbps baud rate), depending upon the encoding of the previous bit. The SOF, EOD, EOF and IFS symbols will always be encoded at an assigned level and length. See Figure 20-5.

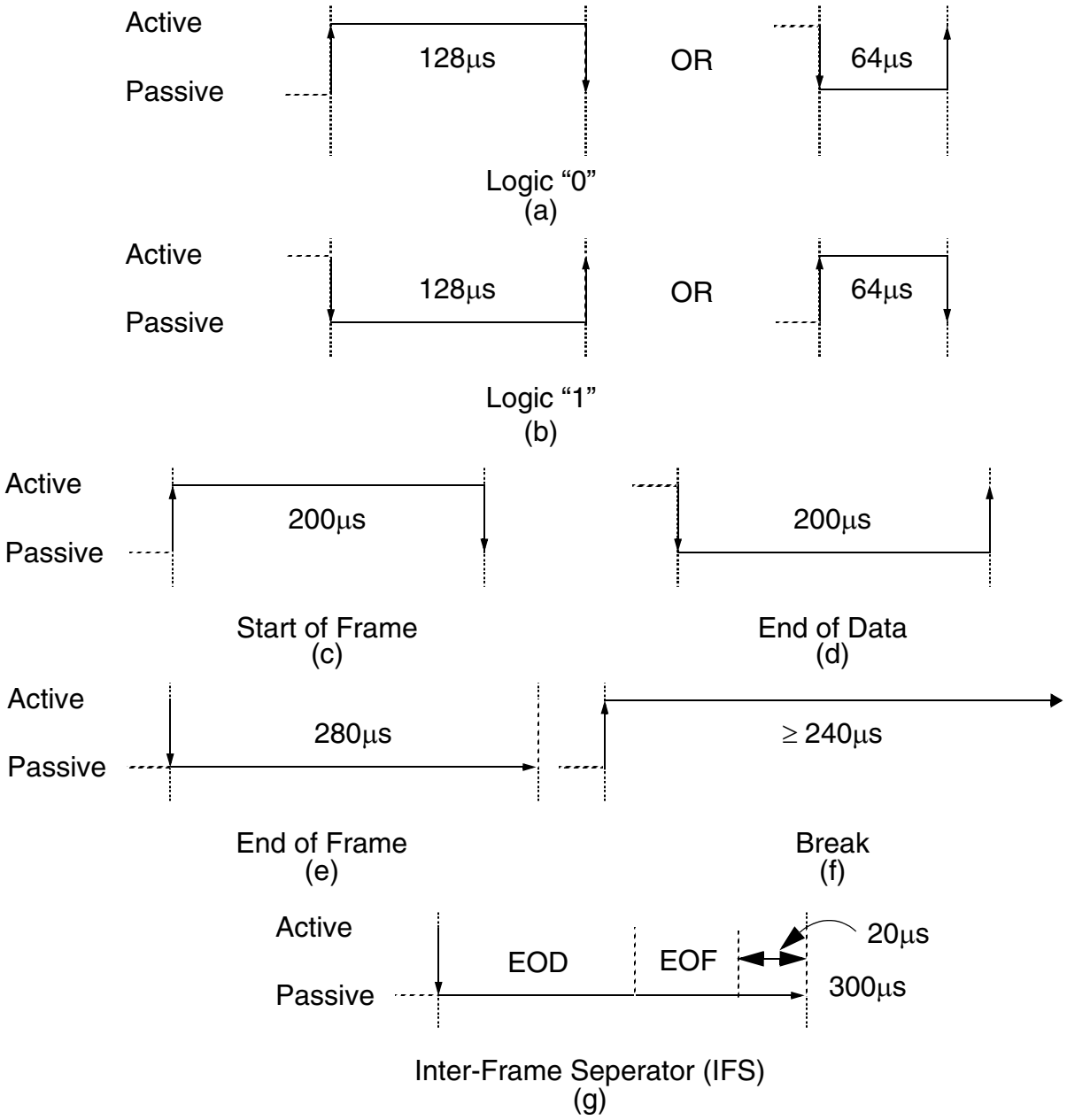


Figure 20-5. J1850 VPW Symbols

Each message will begin with an SOF symbol, an active symbol, and therefore each data byte (including the CRC byte) will begin with a passive bit, regardless of whether it is a logic one or a logic zero. All VPW bit lengths stated in the following descriptions are typical values at a 10.4kbps bit rate.

- Logic “0”

  A logic zero is defined as either an active to passive transition followed by a passive period 64µs in length, or a passive to active transition followed by an active period 128µs in length (Figure 20-5(a)).
- Logic “1”

  A logic one is defined as either an active to passive transition followed by a passive period 128µs in length, or a passive to active transition followed by an active period 64µs in length (Figure 20-5(b)).
- NB - Normalization Bit

  The NB symbol has the same property as a logic “1” or a logic “0”.It is only used in IFR message responses. This bit is defined as an active bit.
- SOF - Start of Frame Symbol

The SOF symbol is defined as passive to active transition followed by an active period 200µs in length (Figure 20-5(c)). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic one or a logic zero.

- EOD - End of Data Symbol

  The EOD symbol is defined as an active to passive transition followed by a passive period 200µs in length (Figure 20-5(d)).
- EOF - End of Frame Symbol

  The EOF symbol is defined as an active to passive transition followed by a passive period 280µs in length (Figure 20-5(e)). If there is no IFR byte transmitted after an EOD symbol is transmitted, after another 80µs the EOD becomes an EOF, indicating the completion of the message.
- IFS - Inter-Frame Separation Symbol

  The IFS symbol is defined as a passive period 300µs in length. The IFS symbol contains no transition, since when used it always follows an EOF symbol.(Figure 20-5(g))
- BREAK - Break Signal

  The BREAK signal is defined as a passive to active transition followed by an active period of at least 240µs (Figure 20-5(f)).
- IDLE

  An IDLE is defined as a passive period greater than 300µs in length.

20.8.1.3 J1850 VPW Valid/Invalid Bits & Symbols

The timing tolerances for receiving data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC module for determining what symbol is being received is equal to a single period of the MUX Interface clock, (t_{bdlc}). i.e. the receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

This clock resolution of 1 t_{bdlc} allows the BDLC module to properly differentiate between the different bits and symbols, without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus having varying oscillator frequencies.

- Transmit and Receive Symbol Timing Specifications

  Table 20-13 throughTable 20-18 contain the SAE J1850 transmit and receive symbol timing specifications for the BDLC module. The units used in these tables are mux interface clock periods (t_{bdlc}). The mux interface clock is a divided down version of the bus clock input to the module (see Section 20.7.3.6, *BDLC Rate Select Register (DLCBRSR) - MBAR + 0x1309*). The mux interface clock drives the transmit and receive counters which control symbol generation and identification. The symbol timing in effect during J1850 operations is dependent the state of two control bits: the CLKS bit BDLC Control Register 1, which indicates whether the bus clock is an integer frequency or a binary frequency; the 4XE bit in BDLC Control Register 2, which is used to select 4X operation.

Table 20-13 andTable 20-15 indicate the transmit and receive timing for integer bus frequencies (CLKS = 0) and 4X operation disabled (4XE = 0). It is assumed that for integer bus frequencies the divided down mux interface clock frequency will be 1MHz (t_{bdlc} = 1 µs).

Table 20-14 and Table 20-16 indicated the transmit and receive timing for binary bus frequencies (CLKS = 1) and 4X operation disabled (4XE = 0). It is assumed that for binary bus frequencies the divided down mux interface clock frequency will be 1.048576 MHz (t_{bdlc} = 0.953674 µs). The symbol timing values are adjusted to compensate for the shortening of the mux interface clock period.

Table 20-17 and Table 20-18 show how the receive symbol timing values are adjusted when 4X operation is enabled (4XE = 1) for both integer bus frequencies (CLKS = 0) and binary bus frequencies (CLKS = 1), respectively.

The values specified in the tables are for the symbols appearing on the SAE J1850 bus. These values assume the BDLC module is communicating on the SAE J1850 bus using an external analog transceiver, and that the BDLC module analog roundtrip delay value programed into the BDLC Analog Round Trip Delay Register register is the appropriate value for the transceiver being used. If these conditions are not met, the symbol timings being measured on the SAE J1850 bus will be significantly affected. For a detailed description of how symbol timings are measured on the SAE J1850 bus, refer to the appropriate SAE documents.

Table 20-13. BDLC Transmitter VPW Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{tvp1}	62	64	66	t_{bdlc}
2	Passive Logic 1	T_{tvp2}	126	128	130	t_{bdlc}
3	Active Logic 0	T_{tva1}	126	128	130	t_{bdlc}
4	Active Logic 1	T_{tva2}	62	64	66	t_{bdlc}

Table 20-13. BDLC Transmitter VPW Symbol Timing for Integer Frequencies (continued)

Number	Characteristic	Symbol	Min	Typ	Max	Unit
5	Start of Frame (SOF)	T_{tva3}	198	200	202	t_{bdlc}
6	End of Data (EOD)[1]	T_{tvp3}	162	164	166	t_{bdlc}
7	End of Frame (EOF)[1]	T_{tv4}	238	240	242	t_{bdlc}
8	Inter-Frame Separator (IFS)[1]	T_{tv5}	298	300	302	t_{bdlc}

Note:
1. The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.

Table 20-14. BDLC Transmitter VPW Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{tvp1}	65	67	69	t_{bdlc}
2	Passive Logic 1	T_{tvp2}	132	134	136	t_{bdlc}
3	Active Logic 0	T_{tva1}	132	134	136	t_{bdlc}
4	Active Logic 1	T_{tva2}	65	67	69	t_{bdlc}
5	Start of Frame (SOF)	T_{tva3}	208	210	212	t_{bdlc}
6	End of Data (EOD)[1]	T_{tvp3}	170	172	174	t_{bdlc}
7	End of Frame (EOF)[1]	T_{tv4}	250	252	254	t_{bdlc}
8	Inter-Frame Separator (IFS)[1]	T_{tv5}	313	315	317	t_{bdlc}

Note:
1. The transmitter timing for this symbol depends upon the minimum detection time of the symbol by the receiver.

Table 20-15. BDLC Receiver VPW Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	32	64	95	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	96	128	163	t_{bdlc}
3	Active Logic 0	T_{rva1}	96	128	163	t_{bdlc}
4	Active Logic 1	T_{rva2}	32	64	95	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	164	200	239	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	164	200	239	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	240	280	299	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	281	---	---	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	240	---	---	t_{bdlc}

Note:
1. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 20-16. BDLC Receiver VPW Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	34	67	100	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	101	134	171	t_{bdlc}
3	Active Logic 0	T_{rva1}	101	134	171	t_{bdlc}
4	Active Logic 1	T_{rva2}	34	67	100	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	172	210	251	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	172	210	251	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	252	293	314	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	315	---	---	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	252	---	---	t_{bdlc}

Note:

1. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 20-17. BDLC Receiver VPW 4X Symbol Timing for Integer Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	8	16	23	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	24	32	40	t_{bdlc}
3	Active Logic 0	T_{rva1}	24	32	40	t_{bdlc}
4	Active Logic 1	T_{rva2}	8	16	23	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	41	50	59	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	41	50	59	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	60	70	74	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	75	---	---	t_{bdlc}
9	Break Signal (BREAK)	T_{rv6}	60	---	---	t_{bdlc}

Note:

1. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

Table 20-18. BDLC Receiver VPW 4X Symbol Timing for Binary Frequencies

Number	Characteristic	Symbol	Min	Typ	Max	Unit
1	Passive Logic 0	T_{rvp1}	9	17	25	t_{bdlc}
2	Passive Logic 1	T_{rvp2}	26	34	42	t_{bdlc}
3	Active Logic 0	T_{rva1}	26	34	42	t_{bdlc}
4	Active Logic 1	T_{rva2}	9	17	25	t_{bdlc}
5	Start of Frame (SOF)	T_{rva3}	43	53	62	t_{bdlc}
6	End of Data (EOD)	T_{rvp3}	43	53	62	t_{bdlc}
7	End of Frame (EOF)	T_{rv4}	63	74	78	t_{bdlc}
8	Inter-Frame Separator (IFS)	T_{rv5}	79	---	---	t_{bdlc}
[9]	Break Signal (BREAK)	T_{rv6}	63	---	---	t_{bdlc}

Note:

1. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{bdlc} due to sampling considerations.

The min and max symbol limits shown in the following sections (Invalid Passive Bit - Valid BREAK Symbol) and figures (Figure 20-6 - Figure 20-9) refer to the values listed in Table 20-13 throughTable 20-18.

- Invalid Passive Bit

  If the passive to active transition beginning the next data bit or symbol occurs between the active to passive transition beginning the current data bit or symbol and $T_{rvp1(Min)}$, the current bit would be invalid. See Figure 20-6(1).

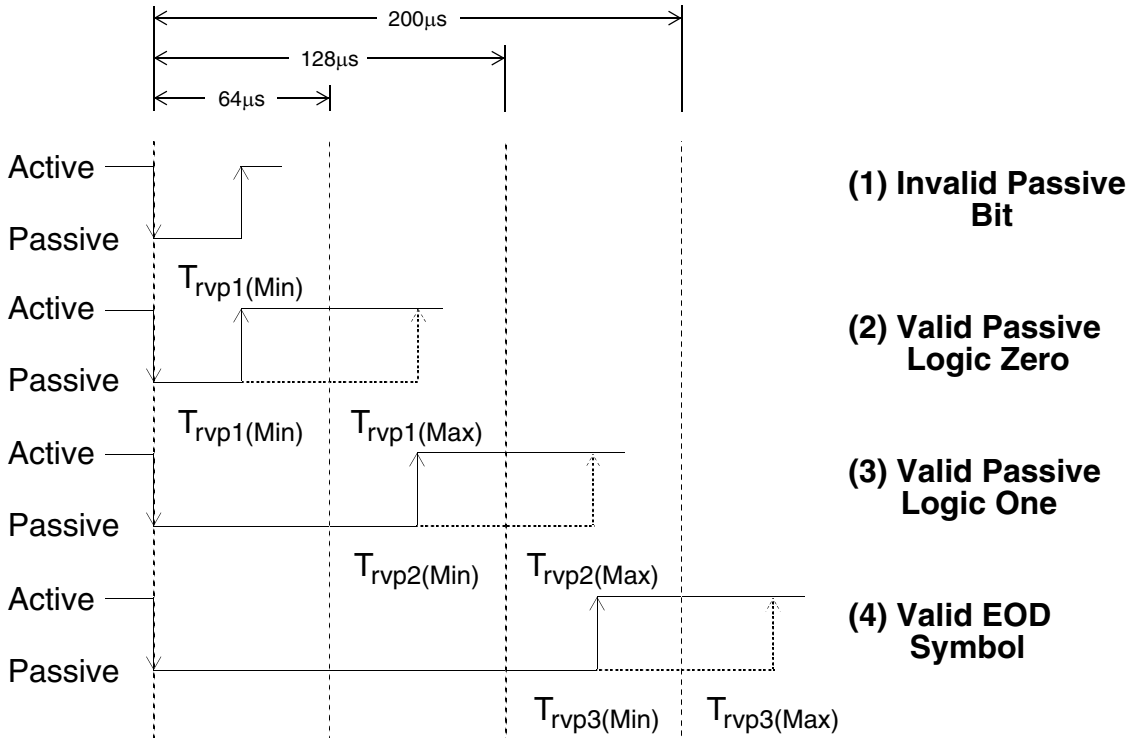


Figure 20-6. J1850 VPW Passive Symbols

- Valid Passive Logic Zero

  If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp1(Min)}$ and $T_{rvp1(Max)}$, the current bit would be considered a logic zero. See Figure 20-6(2).

- Valid Passive Logic One

  If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp2(Min)}$ and $T_{rvp2(Max)}$, the current bit would be considered a logic one. See Figure 20-6(3).

- Valid EOD Symbol

  If the passive to active transition beginning the next data bit or symbol occurs between $T_{rvp3(Min)}$ and $T_{rvp3(Max)}$, the current symbol would be considered a valid EOD symbol. See Figure 20-6(4).

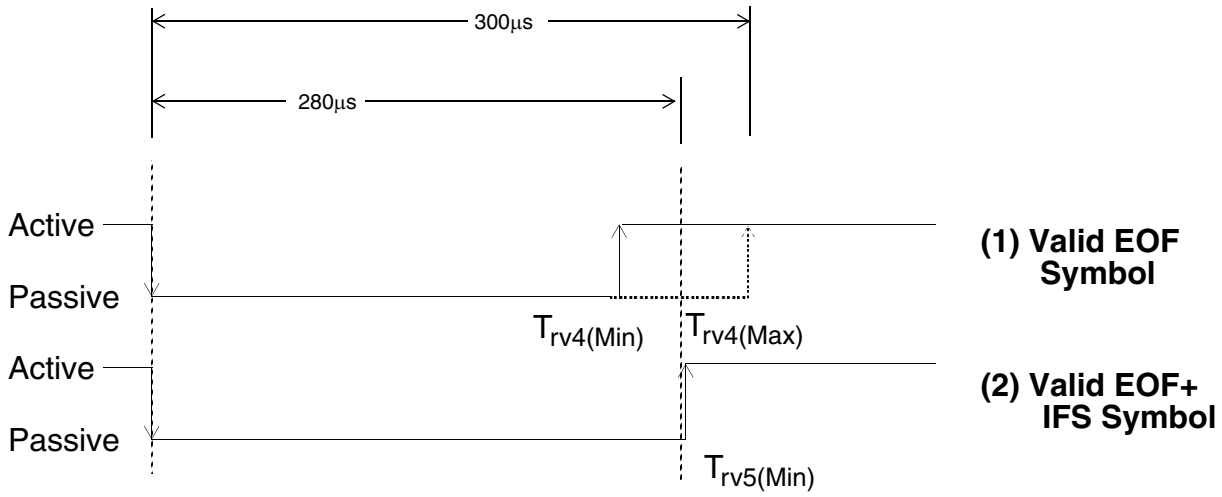


Figure 20-7. J1850 VPW EOF and IFS Symbols

- Valid EOF & IFS Symbol

  In Figure 20-7(1), if the passive to active transition beginning the SOF symbol of the next message occurs between $T_{rv4(Min)}$ and $T_{rv4(Max)}$, the current symbol will be considered a valid EOF symbol.

  If the passive to active transition beginning the SOF symbol of the next message occurs after $T_{rv5(Min)}$, the current symbol will be considered a valid EOF symbol followed by a valid IFS symbol. See Figure 20-7(2). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others, and immediately begin transmitting. Therefore, anytime a node waiting to transmit detects a passive to active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.
- Idle Bus

  If the passive to active transition beginning the SOF symbol of the next message does not occur before $T_{tv5(Min)}$, the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

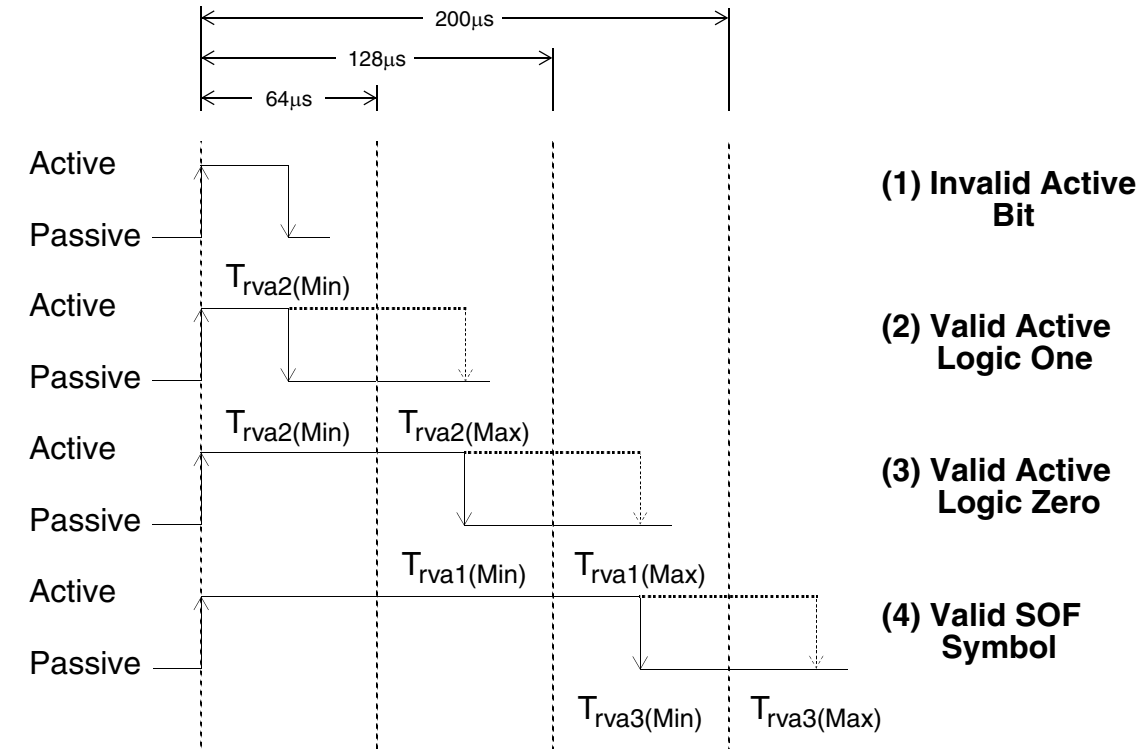


Figure 20-8. J1850 VPW Active Symbols

- Invalid Active Bit

  If the active to passive transition beginning the next data bit or symbol occurs between the passive to active transition beginning the current data bit or symbol and $T_{rva2(Min)}$, the current bit would be invalid. See Figure 20-8(1).
- Valid Active Logic One

  If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva2(Min)}$ and $T_{rva2(Max)}$, the current bit would be considered a logic one. See Figure 20-8(2).
- Valid Active Logic Zero

  If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva1(Min)}$ and $T_{rva1(Max)}$, the current bit would be considered a logic zero. See Figure 20-8(3).
- Valid SOF Symbol

  If the active to passive transition beginning the next data bit or symbol occurs between $T_{rva3(Min)}$ and $T_{rva3(Max)}$, the current symbol would be considered a valid SOF symbol. See Figure 20-8(4).

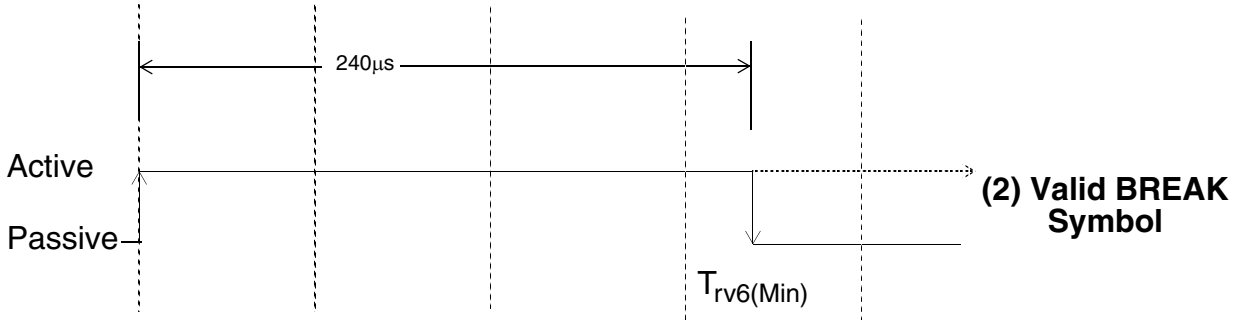


Figure 20-9. J1850 VPW BREAK Symbol

- Valid BREAK Symbol

  If the next active to passive transition does not occur until after $T_{rv6(Min)}$, the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by a SOF symbol beginning the next message to be transmitted onto the J1850 bus. See Figure 20-9.

- Message Arbitration

  Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

  If the BDLC module wishes to transmit onto the J1850 bus, but detects that another message is in progress, it automatically waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration will occur beginning with the first bit after the SOF symbol and continue with each bit thereafter.

  The VPW symbols and J1850 bus electrical characteristics are carefully chosen so that a logic zero (active or passive type) will always dominate over a logic one (active or passive type) simultaneously transmitted. Hence logic zeroes are said to be ‘dominant’ and logic ones are said to be ‘recessive’.

  Whenever a node transmits a recessive bit and detects a dominant bit, it loses arbitration, and immediately stops transmitting. This is known as ‘bitwise arbitration’.The loss of arbitration flag (in BDLC State Vector Register) is set when arbitration is lost. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.

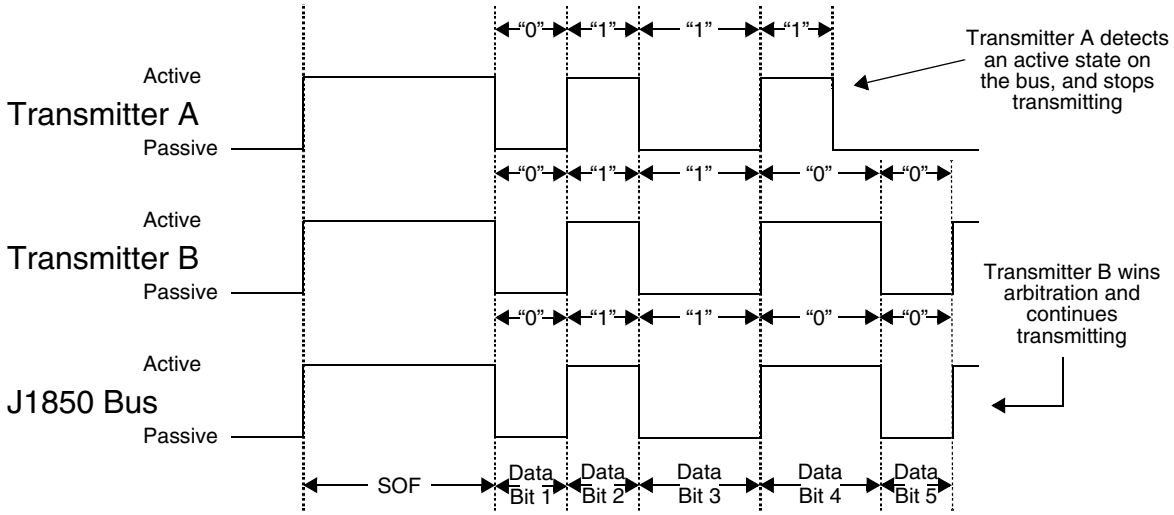


Figure 20-10. J1850 VPW Bitwise Arbitrations

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is immediately stopped unless it occurs on the 8th bit of a byte. In this case the BDLC module will automatically append up to two extra 1 bits and then stop transmitting. These two extra bits will be arbitrated normally and thus will not interfere with another message. The second 1 bit will not be sent if the first loses arbitration. If the BDLC module has lost arbitration to another valid message then the two extra ones will not corrupt the current message. However, if the BDLC module has lost arbitration due to noise on the bus, then the two extra ones will ensure that the current message will be detected and ignored as a noise-corrupted message.

Since a “0” dominates a “1”, the message with the lowest value will have the highest priority, and will always win arbitration, i.e. a message with priority 000 will win arbitration over a message with priority 011. This method of arbitration will work no matter how many bits of priority encoding are contained in the message.

20.8.1.4 J1850 Bus Errors

The BDLC module detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

- Transmission Error

  If the BDLC module is transmitting a message and the message received contains a symbol error, a framing error, a bus fault, a BREAK symbol, or a logic ‘1’ symbol when a logic “0” is being transmitted, this constitutes a transmission error. Receiving a logic ‘0’ symbol when transmitting a logic ‘1’ is considered a loss of arbitration condition (See Message Arbitration) and not a transmission error. When a transmission error is detected, the BDLC module will immediately cease transmitting. Further transmission or reception will be disabled until a valid EOF symbol is detected on the J1850 bus. The error condition is reflected by setting the symbol invalid or out of range flag in the BDLC State Vector Register register. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.
- CRC Error

  A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed, and the CRC calculation result is not equal to $C4.The CRC code should detect any single and 2 bit errors, as well as all 8 bit burst errors, and almost all other types of errors. The CRC error flag (in BDLC State Vector Register) is set when a CRC error is detected. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.
- Symbol Error

  A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. See sections Invalid Passive Bit and Invalid Active Bit which define invalid symbols.The symbol invalid or out of range flag (in BDLC State Vector Register) is set when a symbol error is detected. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.
- Framing Error

  A framing error is detected when a received symbol occurs in an inappropriate location in the message frame. The following situations result in framing errors:
  - An active logic “0” or logic “1” received as the first symbol of the frame.
  - An SOF symbol received in any location other than the first symbol of a frame. Erroneous locations include: Within the data portion of a message or IFR; Immediately following the EOD in a message or IFR.
  - An EOD symbol received on a non-byte boundary in a message or IFR.
  - An active logic “0” or logic “1” received immediately following the EOD at the end of an IFR.

  The symbol invalid or out of range flag (in BDLC State Vector Register) is set when a framing error is detected. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.
- Bus Fault

  If a bus fault occurs, the response of the BDLC module will depend upon the type of bus fault.

  If the bus is shorted to V_{DD}, the BDLC module will wait for the bus to fall to a passive state before it will attempt to transmit a message. As long as the short remains, the BDLC will never attempt to transmit a message onto the J1850 bus.

  If the bus is shorted to ground, the BDLC module will see an idle bus, begin to transmit the message, and then detect a transmission error, since the short to ground would not allow the bus to be driven to the active (dominant) state. The BDLC module will wait for assertion of the receive pin for (64 - analog round trip delay) t_{bdlc} cycles, after assertion of the transmit pin, before detecting the error. If the transmission is an IFR, the BDLC module will wait for (280 - analog round trip delay) t_{bdlc} cycles before detecting an error. The “analog round trip delay” is determined by the value stored in the BDLC Analog Round Trip Delay Register register. The BDLC module will set the symbol invalid or out of range flag (in BDLC State Vector Register), abort that transmission and wait for the next CPU command to transmit. In this case, the transmitter does not have to wait for an EOF symbol to be received to be enabled. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.

  In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC module will resume normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus.
- BREAK - Break

  Any BDLC transmitting at the time a BREAK is detected will treat the BREAK as if a transmission error had occurred, and halt transmission.

  If while receiving a message the BDLC module detects a BREAK symbol, it will treat the BREAK as a reception error.

If a BREAK symbol is received while the BDLC module is transmitting or receiving, the symbol invalid or out of range flag (in BDLC State Vector Register) is set. Further transmission/reception will be disabled until the J1850 bus returns to the passive state and a valid EOF symbol is detected on the J1850 bus. If the interrupt enable bit (IE in BDLC Control Register 1) is set, an interrupt request from the BDLC module is generated. Reading the BDLC State Vector Register register will clear this flag.

The BDLC module can transmit a BREAK symbol. And it can receive a BREAK symbol from the J1850 bus.

- Bus Error Summary

The possible J1850 bus errors and the actions taken by the BDLC module are summarized in Table 20-19.

Table 20-19. BDLC module J1850 Error Summary

Error Condition	BDLC Module Function
Transmission Error	BDLC module will immediately cease transmitting. Further transmission and reception will be disabled until a valid EOF symbol is detected. The symbol invalid or out of range flag will be set and interrupt generated if enabled.
Cyclical Redundancy Check (CRC) Error	CRC error flag set and interrupt generated if enabled.
Symbol Error	The symbol invalid or out of range flag will be set and interrupt generated if enabled. Transmission and reception will be disabled until a valid EOF symbol is detected.
Framing Error	The symbol invalid or out of range flag will be set and interrupt generated if enabled. Transmission and reception will be disabled until a valid EOF symbol is detected.
Bus short to V_{DD}.	The BDLC module will not transmit until short is corrected and a valid EOF is detected. Depending upon when short occurs and is corrected, this error condition may set the symbol invalid or out of range, crc error, or loss of arbitration flags.
Bus short to GND.	Short will be seen as an idle bus by BDLC module. If a transmission attempt is made before short is corrected, the symbol invalid or out of range flag will be set and interrupt generated if enabled. Another transmission can be initiated as soon as short is corrected.
BREAK symbol reception	If doing so, the BDLC module will immediately cease transmitting. Symbol invalid or out of range flag set and interrupt generated if enabled.Transmission and reception will be disabled until a valid EOF symbol is detected.

20.8.2 Mux Interface

The MUX Interface is responsible for bit encoding/decoding and digital noise filtering between the Protocol Handler and the Physical Interface. Refer to Figure 20-2.

20.8.2.1 Mux Interface - Rx Digital Filter

The Receiver section of the BDLC module includes a digital low pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 20-11.

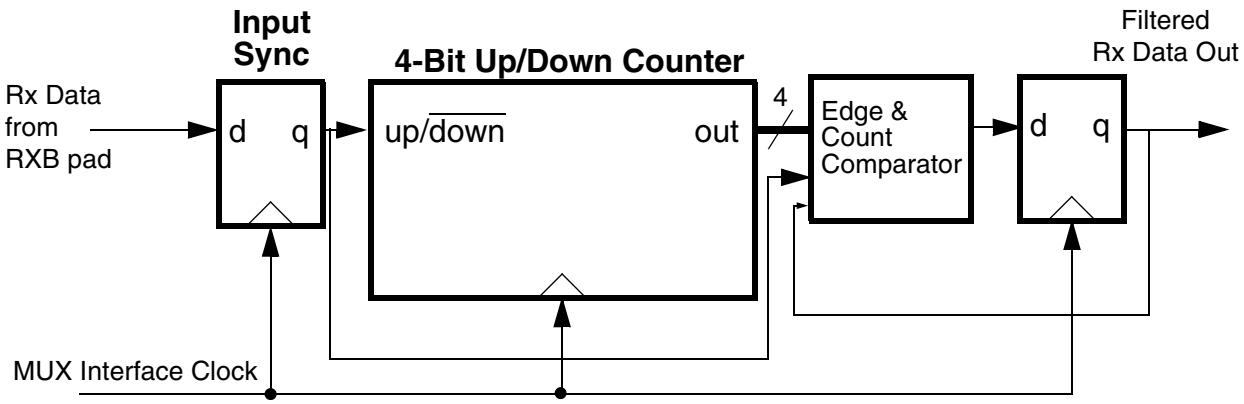


Figure 20-11. BDLC Module Rx Digital Filter Block Diagram

- Operation

  The clock for the digital filter is provided by the MUX Interface clock.At each positive edge of the clock signal, the current state of the Receiver input signal from the RXB pad is sampled.The RXB signal state is used to determine whether the counter should increment or decrement at the next positive edge of the clock signal.

  The counter will increment if the input data sample is high but decrement if the input sample is low.The counter will thus progress up towards ‘15’ if, on average, the RXB signal remains high or progress down towards ‘0’ if, on average, the RXB signal remains low.

  When the counter eventually reaches the value ‘15’, the digital filter decides that the condition of the RXB signal is at a stable logic level one and the Data Latch is set, causing the Filtered Rx Data signal to become a logic level one. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

  Alternatively, should the counter eventually reach the value ‘0’, the digital filter decides that the condition of the RXB signal is at a stable logic level zero and the Data Latch is reset, causing the Filtered Rx Data signal to become a logic level zero. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

  The Data Latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the RXB signal.

- Performance

  The performance of the digital filter is best described in the time domain rather than the frequency domain.

  If the signal on the RXB signal transitions, then there will be a delay before that transition appears at the Filtered Rx Data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This ‘filter delay’ must be taken into account when performing message arbitration.

  For example, if the frequency of the MUX Interface clock (f_{bdlc}) is 1.0486MHz, then the period (t_{bdlc}) is 954ns and the maximum filter delay in the absence of noise will be 15.259us.

  The effect of random noise on the RXB signal depends on the characteristics of the noise itself. Narrow noise pulses on the RXB signal will be completely ignored if they are shorter than the filter delay. This provides a degree of low pass filtering.

  If noise occurs during a symbol transition, the detection of that transition may be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

  Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length will be detected by the next stage of the BDLC module’s receiver as an invalid symbol.

  Noise pulses that are longer than the shortest allowable symbol length will normally be detected as an invalid symbol or as invalid data when the frame’s CRC is checked.

20.8.3 Protocol Handler

The Protocol Handler is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The Protocol Handler conforms to SAE J1850 - Class B Data Communications Network Interface. Refer to Figure 20-2

20.8.3.1 Protocol Architecture

The Protocol Handler contains the State Machine, Rx Shadow Register, Tx Shadow Register, Rx Shift Register, Tx Shift Register, and Loopback Multiplexer as shown in Figure 20-12. Each block will now be described in more detail.

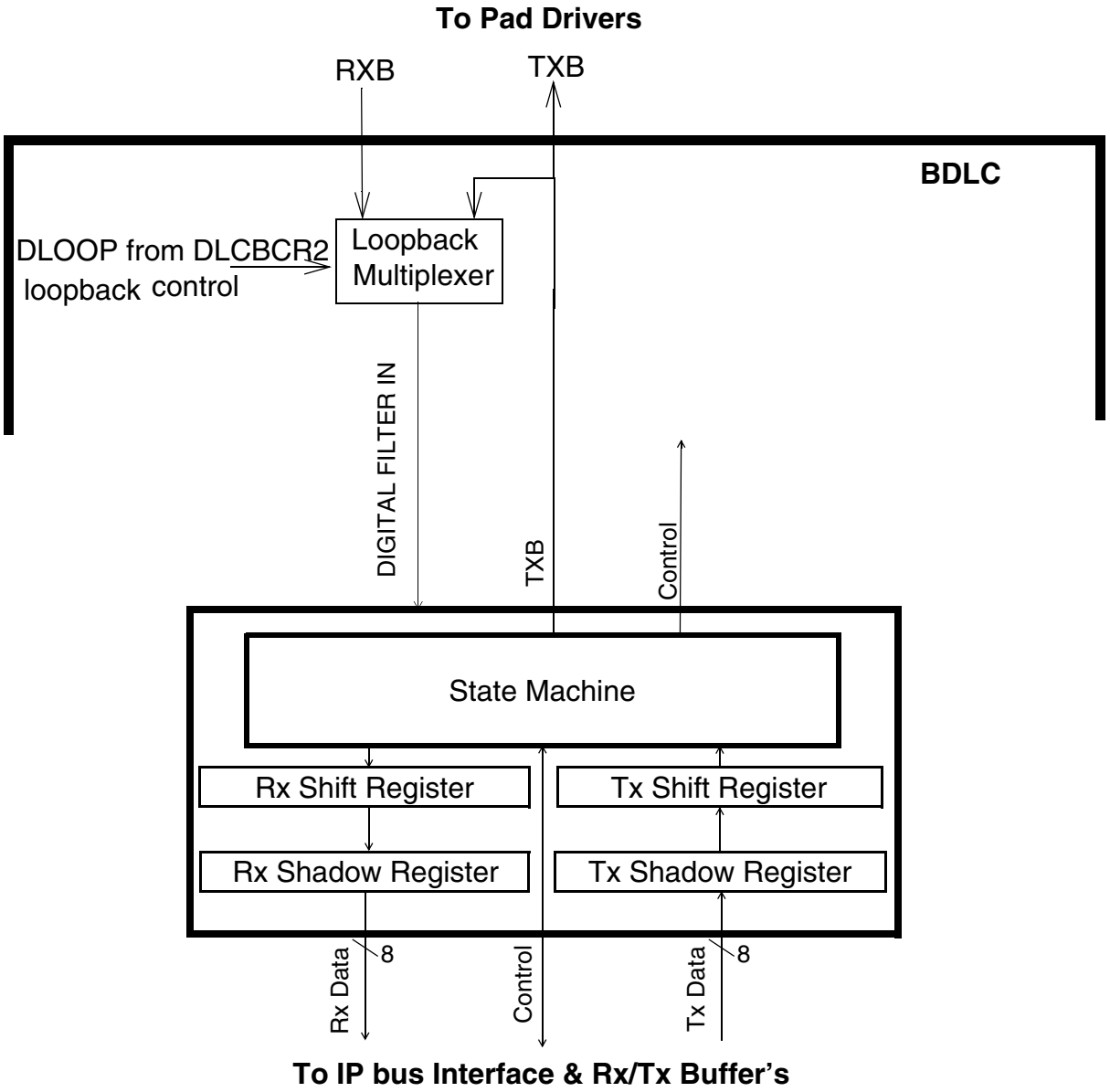


Figure 20-12. BDLC Protocol Handler Outline

- Rx & Tx Shift Registers

  The Rx Shift Register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx Shadow Register. The Tx Shift Register takes data, in parallel form, from the Tx Shadow Register and presents it serially to the State Machine so that it can be transmitted onto the J1850 bus.

- Rx & Tx Shadow Registers

  Immediately after the Rx Shift Register has completed shifting in a byte of data, this data is transferred to the Rx Shadow Register and RDRF or RXIFR is set and interrupt is generated if the interrupt enable bit (IE) in BDLC Control Register 1 is set. After the transfer takes place, this new data byte in the Rx Shadow Register is available to the CPU, and the Rx Shift Register is ready to shift in the next byte of data. Data in Rx Shadow Register must be retrieved by the CPU before it is overwritten by new data from the Rx Shift Register.

  Once the Tx Shift Register has completed its shifting operation for the current byte, the data byte in the Tx Shadow Register is loaded into the Tx Shift Register. After this transfer takes place, the Tx Shadow Register is ready to accept new data from the CPU.

- Digital Loopback Multiplexer

  The Digital Loopback Multiplexer connects the input of the receive digital filter (See Figure 20-12) to either the transmit signal out to the pad (TXB) or the receive signal from the pad (RXB), depending on the state of the DLOOP bit in BDLC Control Register 2 register.
- State Machine

  All of the functions associated with performing the protocol are executed or controlled by the State Machine. The State Machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following sections describe the BDLC module's actions in a variety of situations.
- 4X Mode

  The BDLC module can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 VPW operation. The BDLC module can transmit and receive messages in 4X mode, if the 4XE bit is set in BDLC Control Register 2. If the 4XE bit is not set in the BDLC Control Register 2, any 4X message on the J1850 bus is treated as noise by the BDLC module and is ignored. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode will be interpreted as noise on the network by the BDLC module.
- Receiving a Message in Block Mode

  Although not a part of the SAE J1850 protocol, the BDLC module does allow for a special "Block Mode" of operation of the receiver. As far as the BDLC module is concerned, a Block Mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

  Another node wishing to send a Block Mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.
- Transmitting a Message in Block Mode

  A Block mode message is transmitted inherently by simply loading the bytes one by one into the BDLC Data Register register until the message is complete. The programmer should wait until the TDRE flag is set prior to writing a new byte of data into the BDLC Data Register register. The BDLC module does not contain any predefined maximum J1850 message length requirement.

20.8.4 Transmitting A Message

The design of the BDLC module enables the user to easily handle message reception and message transmission separately. This can greatly simplify the communication software, as all received messages can be handled virtually the same, regardless of their origin.

This chapter will therefore describe only the steps necessary for transmitting a message, and will not address the resulting reception of that message by the BDLC module. Message reception is described in Section 20.8.5, *Receiving A Message*. Later sections will deal with transmitting and receiving In-Frame Responses on the SAE J1850 bus.

20.8.4.1 BDLC Transmission Control Bits

There is only one BDLC module control bit which is used when transmitting a message onto the SAE J1850 bus. This bit, the Transmit End of Data (TEOD) bit, is set by the user to indicate to the BDLC module that the last byte of that part of the message frame has been loaded into the BDLC Data Register. The TEOD bit, located in BDLC Control Register 2, is also used when transmitting an In-Frame Response (IFR), but that usage is described in Section 20.8.6, *Transmitting An In-Frame Response (IFR)*. Setting the TEOD bit indicates to the BDLC module that the last byte written to the BDLC Data Register is the final byte to be transmitted, and that following this byte a CRC byte and EOD symbol should be transmitted automatically. Setting the TEOD bit will also inhibit any further TDRE interrupts until TEOD is cleared. The TEOD bit will be cleared on the rising edge of the first bit of the transmitted CRC byte, or if an error or loss of arbitration is detected on the bus.

- BDLC Data Register

  The BDLC Data Register is a double-buffered register which is used for handling the transmitted and received message bytes.Bytes to be transmitted onto the SAE J1850 bus are written to the BDLC Data Register, and bytes received from the bus by the BDLC module are read from the BDLC Data Register. Since this register is double buffered, bytes written into it cannot be read by the CPU. If this is attempted, the byte which is read will be the last byte placed in the BDLC Data Register by the BDLC module, not the last byte written to the BDLC Data Register by the CPU. For an illustration of the BDLC Data Register, refer to Section 20.7.3.4, *BDLC Data Register (DLCBDR) - MBAR + 0x1305*.
- Transmitting a Message with the BDLC

  To transmit a message using the BDLC module, the user just writes the first byte of the message to be transmitted into the BDLC Data Register, initiating the transmission process. When the TDRE status appears in the BDLC State Vector Register, the user writes the next byte into the BDLC Data Register. Once all of the bytes have been loaded into the BDLC Data Register, the user sets the TEOD bit, and the BDLC module completes the message transmission. What follows is an overview of the basic steps required to transmit a message onto an SAE J1850 network using the BDLC module. For an illustration of this sequence, refer to Figure 20-13.

NOTE

Due to the byte-level architecture of the BDLC module, the 12-byte limit on message length as defined in SAE J1850 must be enforced by the user's software. The number of bytes in a message (transmitted or received) has no meaning to the BDLC module.

— Step 1: Write the First Byte into the BDLC Data Register

To initiate a message transmission, the CPU simply loads the first byte of the message to be transmitted into the BDLC Data Register. The BDLC module will then perform the necessary bus acquisition duties to determine when the message transmission can begin.

Once the BDLC module determines that the SAE J1850 bus is free, a Start of Frame (SOF) symbol will be transmitted, followed by the byte written to the BDLC Data Register. Once the BDLC module readies this byte for transmission, the BDLC State Vector Register will reflect that the next byte can be written to the BDLC Data Register (TDRE interrupt).

NOTE

If the user writes the first byte of a message to be transmitted to the BDLC Data Register and then determines that a different message should be transmitted, the user can write a new byte to the BDLC Data Register up until the transmission begins. This new byte will replace the original byte in the BDLC Data Register.

— Step 2: When TDRE is Indicated, Write the Next Byte into the BDLC Data Register

When a TDRE state is reflected in the BDLC State Vector Register, the CPU writes the next byte to be transmitted into the BDLC Data Register. This step is repeated until the last byte to be transmitted is written to the BDLC Data Register.

NOTE

Due to the design and operation of the BDLC module, when transmitting a message the user may write two, or possibly even three of the bytes to be transmitted into the BDLC Data Register before the first RDRF interrupt occurs. For this reason, the user should never use receive interrupts to control the sequencing of bytes to be transmitted.

— Step 3: Write the Last Byte to the BDLC Data Register and Set TEOD

Once the user has written the last byte to be transmitted into the BDLC Data Register, the user then sets the TEOD bit in BDLC Control Register 2. When the TEOD bit is set, once the byte written to the BDLC Data Register is transmitted onto the bus, the BDLC module will begin transmitting the 8-bit CRC byte, as specified in SAE J1850. Following the CRC byte, the BDLC module will transmit an EOD symbol onto the SAE J1850 bus, indicating that this part of the message has been completed. If no IFR bytes are transmitted following the EOD, an EOF will be recognized and the message will be complete.

Setting the TEOD bit is the last step the CPU needs to take to complete the message transmission, and no further transmission-related interrupts will occur. Once the message has been completely received by the BDLC module, an EOF interrupt will be generated. However, this is technically a receive function which can be handled by the message reception routine.

NOTE

While the TEOD bit is typically set immediately following the write of the last byte to the BDLC Data Register, it is also acceptable to wait until a TDRE interrupt is generated before setting the TEOD bit. While the example flowchart in Figure 20-13 shows the TEOD bit being set after the write to the BDLC Data Register, either method is correct. If a TDRE interrupt is pending, it will be cleared when the TEOD bit is set.

20.8.4.2 Transmitting Exceptions

While this is the basic transmit flow, at times the message transmit process will be interrupted. This can be due to a loss of arbitration to a higher priority message or due to an error being detected on the network. For the transmit routine, either of these events can be dealt with in a similar manner.

- Loss of Arbitration

  If a loss of arbitration (LOA) occurs while the BDLC module is transmitting onto the SAE J1850 bus, the BDLC module will immediately stop transmitting, and a LOA status will be reflected in the BDLC State Vector Register. If the loss of arbitration has occurred on a byte boundary, an RDRF interrupt may also be pending once the LOA interrupt is cleared.

  When a loss of arbitration occurs, the J1850 message handling software should immediately switch into the receive mode. If the TEOD bit was set, it will be cleared automatically. **If another attempt is to be made to transmit the same message, the user must start the transmit sequence over from the beginning of the message.**

- Error Detection

Similar to a loss of arbitration, if any error (except a CRC error) is detected on the SAE J1850 bus during a transmission, the BDLC module will stop transmitting immediately. The byte which was being transmitted will be discarded, and the “Symbol Invalid or Out of Range” status will be reflected in the BDLC State Vector Register. As with the loss of arbitration, if the TEOD bit was set, it will be cleared automatically, and any attempt to transmit the same message will have to start from the beginning.

If a CRC error occurs following a transmission, this will also be reflected in the BDLC State Vector Register. However, since the CRC error is really a receive error based on the received CRC byte, at this point all bytes of the message will have been transmitted. It is therefore up to the user’s software to determine if another attempt should be made to transmit the message in which the error occurred.

- Transmitter Underrun

  A transmitter underrun can occur when a TDRE interrupt is not serviced in a timely fashion. If the last byte loaded into the BDLC Data Register is completely transmitted onto the network before the next byte is loaded into the BDLC Data Register, a transmitter underrun will occur. If this does happen, the BDLC module will transmit two additional logic ones to ensure that the partial message which was transmitted onto the bus does not end on a byte boundary. This will be followed by an EOD and EOF symbol. The only indication to the CPU that an underrun occurred is the Symbol Invalid or Out of Range error which will be indicated in the BDLC State Vector Register. As with the other errors, it is up to the user’s software to determine if another transmission attempt should be made.

- In-Frame Response to a Transmitted Message

  If an In-Frame Response (IFR) is received following the transmission of a message, the status indicating that an IFR byte has been received will be indicated in the BDLC State Vector Register before an EOF is indicated. Refer to Section 20.8.7, *Receiving An In-Frame Response (IFR)* for a description of how to handle the reception of IFR bytes.

20.8.4.3 Aborting a Transmission

The BDLC module does not have a mechanism designed specifically for aborting a transmission. Since the module transmits each message on a byte-by-byte basis, there is little need to implement an abort mechanism. If the user has loaded a byte into the BDLC Data Register to initiate a message transmission and decides to send a different message, the byte in the BDLC Data Register can be replaced, right up to the point that the message transmission begins.

If the user has loaded a byte into the BDLC Data Register and then decides not to send any message at all, the user can let the byte transmit, and when the TDRE interrupt occurs let the transmitter underrun. This will cause two extra logic ones followed by an EOF to be transmitted. While this method may require a small amount of bus bandwidth, the need to do this should be very rare. Replacing the byte originally written to the BDLC Data Register with $FF will also increase the probability of the transmitter losing arbitration if another node begins transmitting at the same time, also reducing the bus bandwidth needed.

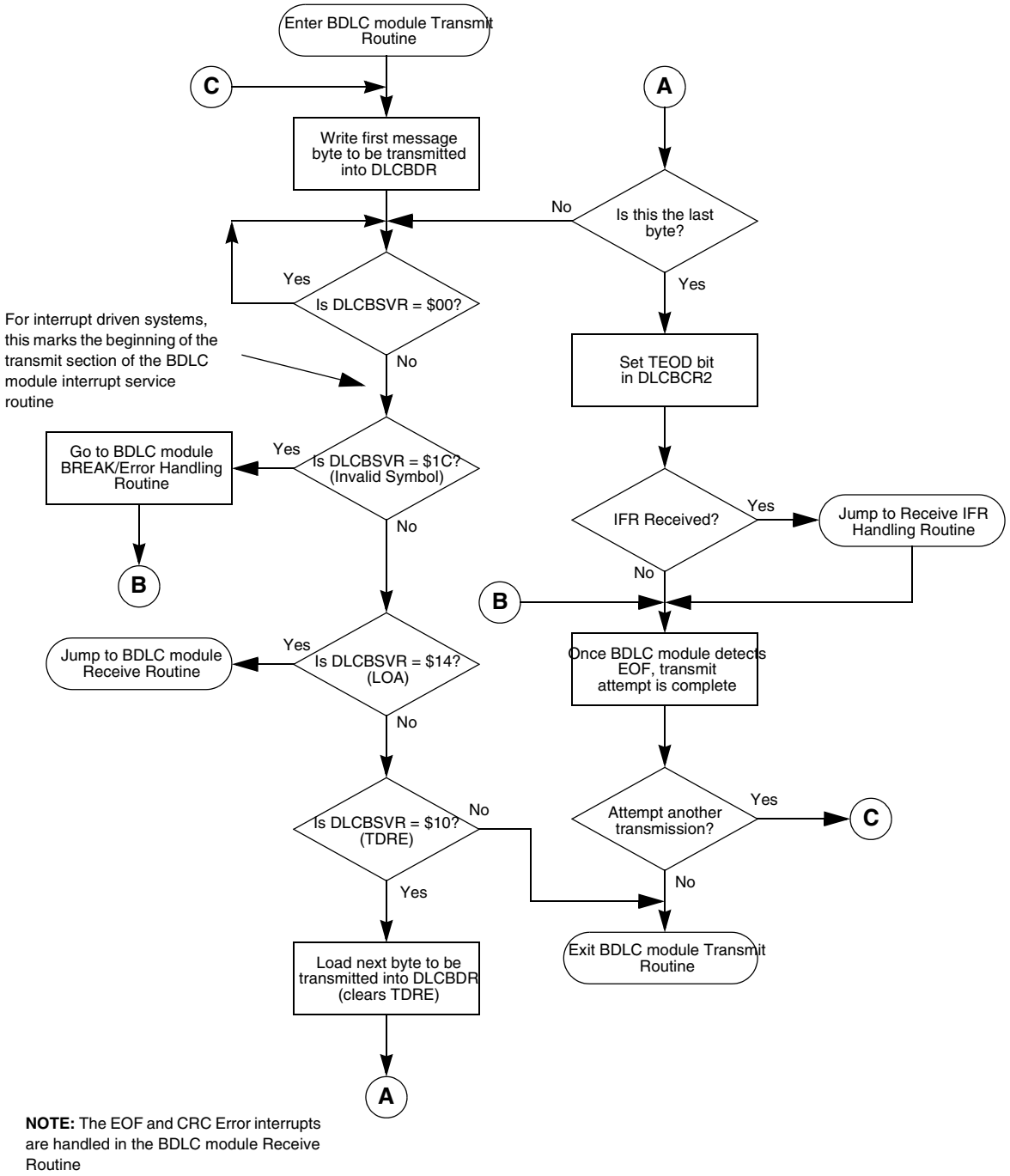


Figure 20-13. Basic BDLC Transmit Flowchart

20.8.5 Receiving A Message

The design of the BDLC module makes it especially easy to use for receiving messages off of the SAE J1850 bus. When the first byte of a message comes in, the BDLC State Vector Register will indicate to the CPU that a byte has been received. As each successive byte is received, that will in turn be reflected in the BDLC State Vector Register. When the message is complete and the EOF has been detected on the bus, the BDLC State Vector Register will reflect this, indicating that the message is complete.

The basic steps required for receiving a message from the SAE J1850 bus are outlined below. For more information on receiving IFR bytes, refer to Section 20.8.7, *Receiving An In-Frame Response (IFR)*.

20.8.5.1 BDLC Reception Control Bits

The only control bit which is used for message reception, the IMSG bit, is actually used to prevent message reception. When the IMSG bit is set BDLC module interrupts of the CPU are inhibited until the next SOF symbol is received. This allows the BDLC module to ignore the remainder of a message once the CPU has determined that it is of no interest. This helps reduce the amount of CPU overhead used to service messages received from the SAE J1850 network, since otherwise the BDLC module would require attention from the CPU for each byte broadcast on the network. The IMSG bit is cleared when the BDLC module receives an SOF symbol, or it can also be cleared by the CPU.

NOTE

While the IMSG bit can be used to prevent the CPU from having to service the BDLC module for every byte transmitted on the SAE J1850 bus, **the IMSG bit should never be used to ignore the BDLC module's own transmission**. Because setting the IMSG bit prevents all BDLC State Vector Register bits from being updated until an SOF is received, the user would not receive any further transmit-related interrupts until another SOF was received, making it very difficult for the CPU to complete the transmission correctly.

20.8.5.2 Receiving a Message with the BDLC module

Receiving a message using the BDLC module is extremely straight-forward. As each byte of a message is received and placed into the BDLC Data Register, the BDLC module will indicate this to the CPU with an Rx Data Register Full (RDRF) status in the BDLC State Vector Register. When an EOF symbol is received, indicating to the CPU that the message is complete, this too will be reflected in the BDLC State Vector Register.

Outlined below are the basic steps to be followed for receiving a message from the SAE J1850 bus with the BDLC module. For an illustration of this sequence, refer to Section Figure 20-14., *Basic BDLC Receive Flowchart*.

- Step 1: When RDRF Interrupt Occurs, Retrieve Data Byte

  When the first byte of a message following a valid SOF symbol is received that byte is placed in the BDLC Data Register, and an RDRF state is reflected in the BDLC State Vector Register. No indication of the SOF reception is made, since the end of the previous message is marked by an EOF indication. The first RDRF state following this EOF indication should allow the user to determine when a new message begins.

  The RDRF interrupt is cleared when the received byte is read from the BDLC Data Register. Once this is done, no further CPU intervention is necessary until the next byte is received, and this step is repeated.

  All bytes of the message, including the CRC byte, will be placed into the BDLC Data Register as they are received for the CPU to retrieve.
- Step 2: When an EOF is Received, the Message is Complete

  Once all bytes (including the CRC byte) have been received from the bus, the bus will be idle for a time period equal to an EOD symbol. Once the EOD symbol is received, the BDLC module will verify that the CRC byte is correct. If the CRC byte is not correct, this will be reflected in the BDLC State Vector Register.

  If no In-Frame Response bytes are transmitted following the EOD symbol, the EOD will transition into an EOF symbol. When the EOF is received it will be reflected in the BDLC State Vector Register, indicating to the user that the message is complete. If IFR bytes do follow the first EOD symbol, once they are complete another EOD will be transmitted, followed by an EOF.

  Once the EOF state is reflected in the BDLC State Vector Register, this indicates to the user that the message is complete, and that when another byte is received it is the first byte of a new message.

20.8.5.3 Filtering Received Messages

No message filtering hardware is included on the BDLC module, so all message filtering functions must be performed in software. Because the BDLC module handles each message on a byte-by-byte basis, message filtering can be done as each byte is received, rather than after the entire message is complete. This enables the CPU to decide while a message is still in progress whether or not that message is of any interest.

At any point during a message, if the CPU determines that the message is of no interest the IMSG bit can be set. Setting the IMSG bit commands the BDLC module not to update the BDLC State Vector Register until the next valid SOF is received. This prevents the CPU from having to service the BDLC module for each byte of every message sent over the network.

20.8.5.4 Receiving Exceptions

As with a message transmission, this basic message reception flow can be interrupted if errors are detected by the BDLC module. This can occur if an incorrect CRC is detected or if an invalid or out of range symbol appears on the SAE J1850 bus. A problem can also arise if the CPU fails to service the BDLC Data Register in a timely manner during a message reception.

- Receiver Overrun

Once a message byte has been received, the CPU must service the BDLC Data Register before the next byte is received, or the first byte will be lost. If the BDLC Data Register is not serviced quickly enough, the next byte received will be written over the previous byte in the BDLC Data Register. No receiver overrun indication is made to the CPU. If the CPU fails to service the BDLC module during the reception of an entire message, the byte remaining in the BDLC Data Register will be last byte received (usually a CRC byte).

Once a receiver overrun occurs, there is no way for the CPU to recover the lost byte(s), so the entire message should be discarded. To prevent receiver overrun, the user should ensure that a BDLC RDRF interrupt will be serviced before the next byte can be received. When polling the BDLC State Vector Register, the user should select a polling interval which will provide timely monitoring of the BDLC module.

- CRC Error

  If a CRC error is detected during a message reception, this will be reflected in the BDLC State Vector Register once an EOD time is recognized by the BDLC module. Since all bytes of the message will have been received when this error is detected, it is up to the user to ensure that all the received message bytes are discarded.

- Invalid or Out of Range Symbol

  If an invalid or out of range symbol, a framing error or a BREAK symbol is detected on the SAE J1850 bus during the reception of a message, the BDLC module will immediately stop receiving the message and discard any partially received byte. The “Symbol Invalid or Out of Range” status will immediately be reflected in the BDLC State Vector Register. Following this the BDLC module will wait until the bus has been idle for a time period equal to an EOF symbol before receiving another message. As with the CRC error, the user should discard any partially received message if this occurs.

- In-Frame Response to a Received Message

  As mentioned above, if one or more IFR bytes are received following the reception of a message, the status indicating the reception of the IFR byte(s) will be indicated in the BDLC State Vector Register before the EOF is indicated. Refer to Section 20.8.7, *Receiving An In-Frame Response (IFR)* for a description of how to deal with the reception of IFR bytes.

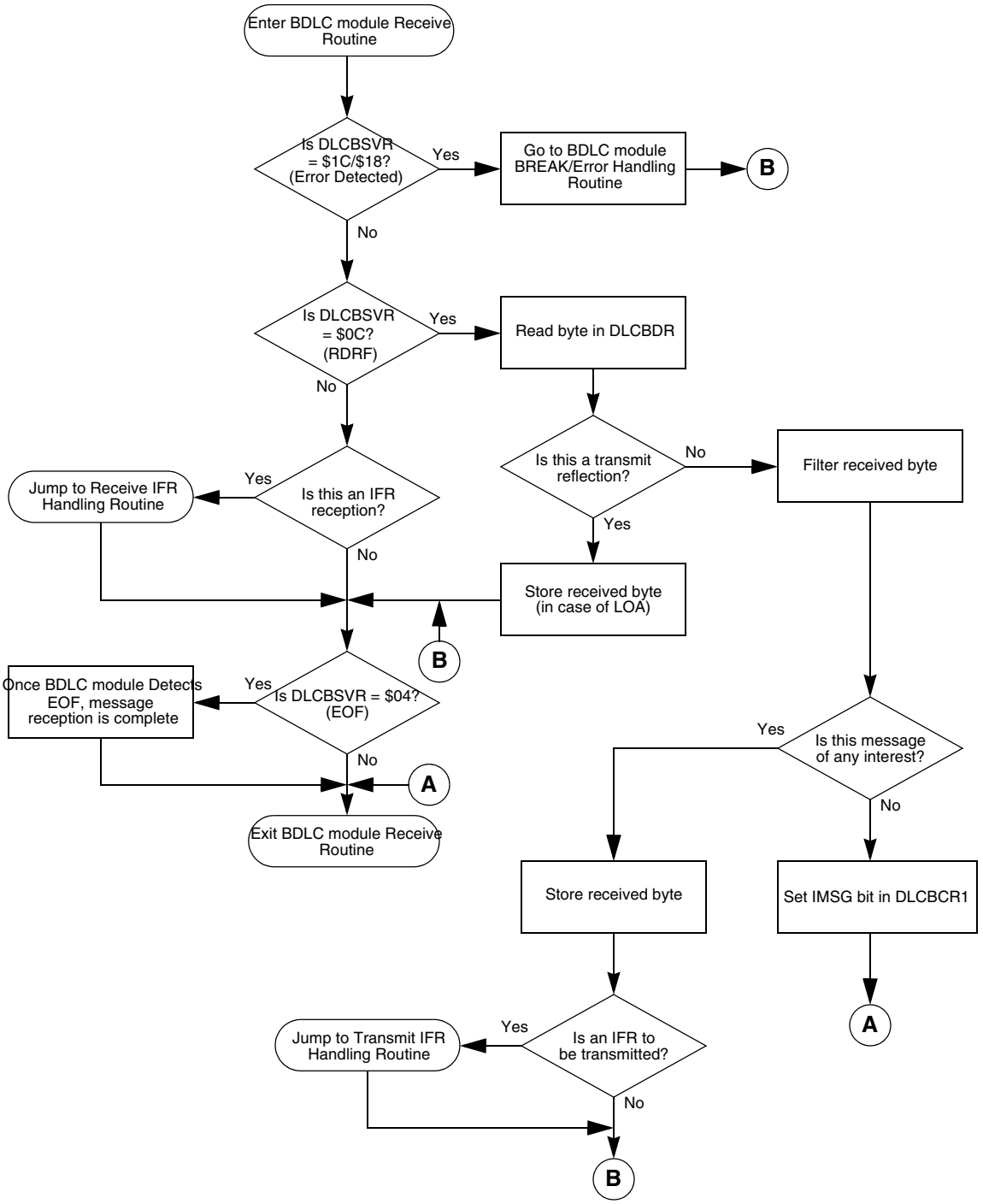


Figure 20-14. Basic BDLC Receive Flowchart

20.8.6 Transmitting An In-Frame Response (IFR)

The BDLC module can be used to transmit all four types of In-Frame Response (IFR) which are defined in SAE J1850. A very brief definition of each IFR type is given below. For a more detailed description of each, refer the SAE J1850 document.

The explanation regarding IFR support by the BDLC module which assumes the user is familiar with the use of IFRs as defined in SAE J1850, and understands the message header bit encoding and normalization bit formats which are used with the different types of IFRs. For more information on this, refer to the SAE J1850 document.

20.8.6.1 IFR Types Supported by the BDLC module

SAE J1850 defines four distinct types of IFR. The first (and most basic) IFR is Type 0, or no IFR. IFR types 1, 2 & 3 are each made up of one or more bytes and, depending upon the type used, may be followed by a CRC byte. The BDLC module is designed to allow the user to transmit and receive all types of SAE J1850 IFRs, but only the network framing/error checking/bus acquisition duties are performed by the BDLC module. The user is responsible for determining the type of IFR to be transmitted, the number of retries to be made (if allowed), and the allowable number of bytes to be transmitted.

- IFR Type 0: No Response

  The most basic type of IFR is no IFR. The Type 0 IFR, as defined in SAE J1850, is no response. The EOD and EOF symbols follow directly after the CRC byte at the end of the message frame being transmitted. This type of IFR is, of course, inherently supported by the BDLC module with no additional user intervention required.

- IFR Type 1: Single Byte from a Single Responder

  SAE J1850 defines the Type 1 IFR as a single byte from a single receiver. This type of IFR is used to acknowledge to the transmitter that the message frame was transmitted successfully on the network, and that at least one receiver received it correctly. A Type 1 IFR generally consists of the physical node ID of the receiver responding to the message, with no CRC byte appended. This type of response is used for Broadcast-type messages, where there may be several intended receivers for a message but the transmitter only wants to know that at least one node received it. In this case, all receivers will begin transmitting their node ID following the EOD. Since all nodes on an SAE J1850 network have a unique node ID, if multiple nodes begin transmitting their node ID simultaneously, arbitration takes place. The node with the highest priority (lowest value) ID wins this arbitration process, and that node's ID makes up the IFR. No retries are attempted by the nodes which lose arbitration during a Type 1 IFR transmission.

  A Type 1 IFR can also be used as a response to a physically addressed message, where the only intended receiver is the one which responds. In this case, no arbitration would take place during the IFR transmission, but the resulting IFR would still consist of a single byte.

- IFR Type 2: Single Byte from Multiple Responders

  The Type 2 IFR, as defined in SAE J1850, is a series of single bytes, each transmitted by a different responder. This IFR type not only acknowledges to the transmitter that the message was transmitted successfully, but also reveals which receivers actually received the message. As with the Type 1 IFR, no CRC byte is appended to the end of a Type 2 IFR.

  This IFR type is typically used with Function-type messages, where the original transmitter may need to know which nodes actually received the message. The basic difference between this type of IFR and the Type 1 IFR is that the nodes which lose arbitration while attempting to transmit their node ID during a Type 2 IFR wait until the byte which wins arbitration is transmitted and then again attempt to transmit their node ID onto the bus. The result is a series of node IDs, one from each receiver of the original message.

- IFR Type 3: Multiple Bytes from a Single Responder

  The last type of IFR defined by SAE J1850 is the Type 3 IFR. This IFR type consists of one or more bytes from a single responder. This type of IFR is used to return data to the original transmitter within the original message frame. This type of IFR may or may not have a CRC byte appended to it.

  The Type 3 IFR is typically used with Function Read-type or Function Query-type messages, where the original transmitter is requesting data from the intended receiver. The node requesting the data transmits the initial portion of the message, and the intended receiver responds by transmitting the desired data in an IFR. In most cases, the original message requiring a Type 3 IFR is addressed to one particular node, so no arbitration should take place during the IFR portion of the message.

20.8.6.2 BDLC IFR Transmit Control Bits

The BDLC module has three bits which are used to control the transmission of an In-Frame Response. These bits, all located in BDLC Control Register 2, are TSIFR, TMIFR1 and TMIFR0. Each is used in conjunction with the TEOD bit to transmit one of three IFR types defined in SAE J1850. What follows is a brief description of each bit.

Because each of the bits used for transmitting an IFR with the BDLC module is used to transmit a particular type of IFR, only one bit should be set by the CPU at a time. However, should more than one of these bits get set at one time, a priority encoding scheme is used to determine which type of IFR is sent. This scheme prevents unpredictable operation caused by conflicting signals to the BDLC module. Table 20-20 illustrates which IFR bit will actually be acted upon by the BDLC module should multiple IFR bits get set at the same time.

NOTE

As with transmitted messages, IFRs transmitted by the BDLC module will also be received by the BDLC module. For a description of how IFR bytes received by the BDLC module should be handled, refer to Section 20.8.7, *Receiving An In-Frame Response (IFR)*.

Table 20-20. IFR Control Bit Priority Encoding

READ/WRITE			ACTUAL		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0
1	X	X	1	0	0
0	1	X	0	1	0
0	0	1	0	0	1

20.8.6.3 Transmit Single Byte IFR

The Transmit Single Byte IFR (TSIFR) bit in BDLC Control Register 2 is used to transmit Type 1 and Type 2 IFRs onto the SAE J1850 bus. If this bit is set after a byte is loaded into the BDLC Data Register, the BDLC module will attempt to send that byte, preceded by the appropriate Normalization Bit, as a single byte IFR without a CRC. If arbitration is lost, the BDLC module will automatically attempt to transmit the byte again (without a Normalization Bit) as soon as the byte winning arbitration completes transmission. Attempts to transmit the byte will continue until either the byte is successfully transmitted, the TEOD bit is set by the user or an error is detected on the bus.

The user must set the TSIFR bit before the EOD following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TSIFR bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

The TSIFR bit will be automatically cleared once the EOD following one or more IFR bytes has been received or an error is detected on the bus.

20.8.6.4 Transmit Multi-Byte IFR 1

The Transmit Multi-Byte IFR 1 (TMIFR1) bit is used to transmit an SAE J1850 Type 3 IFR with a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the BDLC Data Register, the BDLC module will begin transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. Once this happens a TDRE interrupt will occur, indicating to the user that the next IFR byte should be loaded into the BDLC Data Register. When the last byte to be transmitted is written to the BDLC Data Register, the user sets the TEOD bit. This will cause a CRC byte and an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR bit, the TMIFR1 bit must be set before the EOD symbol is received, or it will remain cleared and no IFR transmit attempt will be made. The TMIFR1 bit will be cleared once the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission will halt immediately and the loss of arbitration will be indicated in the BDLC State Vector Register.

20.8.6.5 Transmit Multi-Byte IFR 0

The Transmit Multi-Byte IFR 0 (TMIFR0) bit is used to transmit an SAE J1850 Type 3 IFR without a CRC byte appended. If this bit is set after the user has loaded the first byte of a multi-byte IFR into the BDLC Data Register, the BDLC module will begin transmitting that byte, preceded by the appropriate Normalization Bit, onto the SAE J1850 bus. Once this happens a TDRE interrupt will occur, indicating to the user that the next IFR byte should be loaded into the BDLC Data Register. When the last byte to be transmitted is written to the BDLC Data Register, the user sets the TEOD bit. This will cause an EOD symbol to be transmitted following the last IFR byte.

As with the TSIFR and TMIFR1 bits, the TMIFR0 bit must be set before the EOD symbol is received, or it will remain cleared and no IFR transmit attempt will be made. The TMIFR0 bit will be cleared once the CRC byte and EOD are transmitted, if an error is detected on the bus, if a loss of arbitration occurs during the IFR transmission or if a transmitter underrun occurs when the user fails to service the TDRE interrupt in a timely manner. If a loss of arbitration occurs while the Type 3 IFR is being transmitted, transmission will halt immediately and the loss of arbitration will be indicated in the BDLC State Vector Register.

NOTE

The TMIFR0 bit should not be used to transmit a Type 1 IFR. If a loss of arbitration occurs on the last bit of a byte being transmitted using the TMIFR0 bit, two extra logic ones will be transmitted to ensure that the IFR will not end on a byte boundary. This can cause an error in a Type 1 IFR.

20.8.6.6 Transmitting An IFR with the BDLC module

While the design of the BDLC module makes the transmission of each type of IFR similar, the steps necessary for sending each will be discussed. Again, a discussion of the bytes making up any particular IFR is not within the scope of this document. For a more detailed description of the use of IFRs on an SAE J1850 network, refer to the SAE J1850 document.

- Transmitting a Type 1 IFR

  To transmit a Type 1 IFR, the user loads the byte to be transmitted into the BDLC Data Register and sets both the TSIFR bit and the TEOD bit. This will direct the BDLC module to attempt transmitting the byte written to the BDLC Data Register one time, preceded by the appropriate Normalization Bit. If the transmission is not successful, the byte will be discarded and no further transmission attempts will be made. For an illustration of the steps described below, refer to Section Figure 20-15., *Transmitting A Type 1 IFR*.

- Step 1: Load the IFR Byte into the BDLC Data Register

  The user begins initiation of a Type 1 IFR by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

- Step 2: Set the TSIFR and TEOD Bits

  The final step in transmitting a Type 1 IFR with the BDLC module is to set the TSIFR and TEOD bits in BDLC Control Register 2. Setting both bits will direct the BDLC module to make one attempt at transmitting the byte in the BDLC Data Register as an IFR. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TEOD and TSIFR will be cleared and no further transmit attempts will be made.

- Transmitting a Type 2 IFR

  To transmit a Type 2 IFR, the user loads the byte to be transmitted into the BDLC Data Register and sets the TSIFR bit. Once this is done, the BDLC module will attempt to transmit the byte in the BDLC Data Register as a single byte IFR, preceded by the appropriate Normalization Bit. If the first BDLC module loses arbitration on the first attempt, it will make repeated attempts to transmit this byte until it is successful, an error occurs or the user sets the TEOD bit.

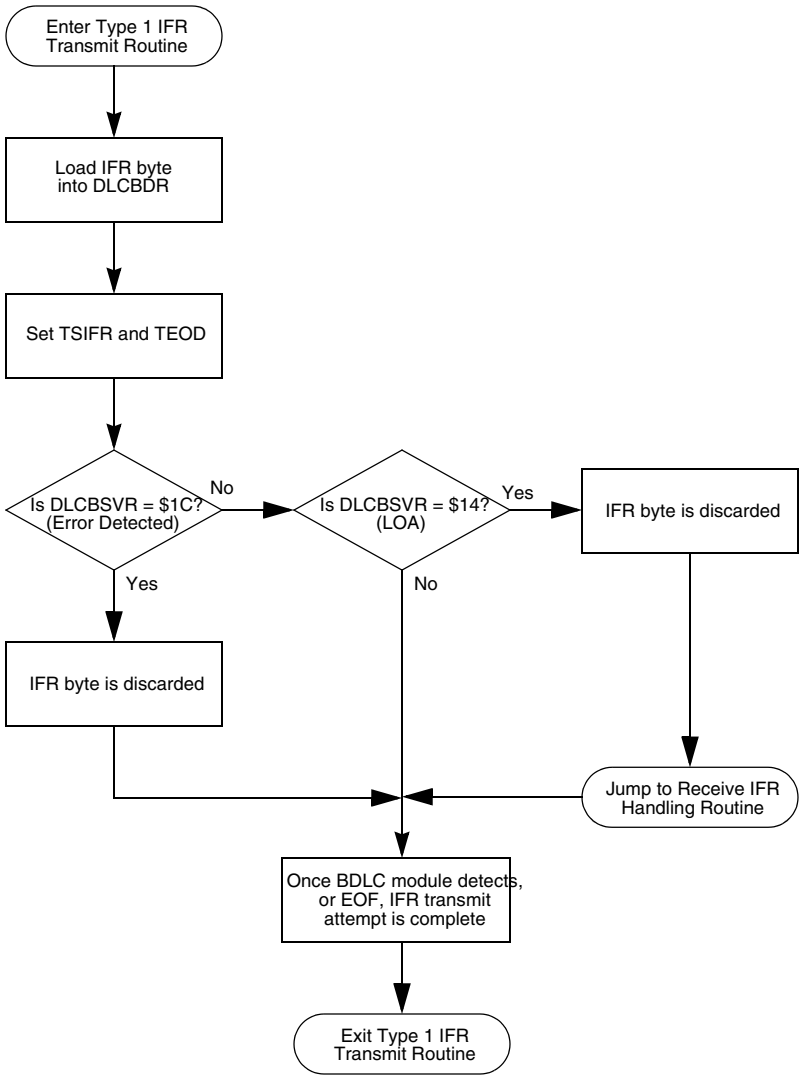


Figure 20-15. Transmitting A Type 1 IFR

— Step 1: Load the IFR Byte into the BDLC Data Register

As with the Type 1 IFR, the user begins initiation of a Type 2 IFR by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

— Step 2: Set the TSIFR Bit

The second step necessary for transmitting a Type 2 IFR is to set the TSIFR bit in BDLC Control Register 2. Setting this bit will direct the BDLC module to attempt to transmit the byte in the BDLC Data Register as an IFR until it is successful. If the byte is transmitted successfully, or if an error or loss of arbitration occurs, TSIFR will be cleared and no further transmit attempts will be made.

— Step 3: If Necessary, Set the TEOD Bit

The third step in transmitting a Type 2 IFR is only necessary if the user wishes to halt the transmission attempts. This may be necessary if the BDLC module's attempt to transmit the byte loaded into the BDLC Data Register continually loses arbitration, and the overall message length approaches the 12-byte limit as defined in SAE J1850.

If it becomes necessary to halt the IFR transmission attempts, the user simply sets the TEOD bit in BDLC Control Register 2. If the BDLC module is between transmission attempts, it will make one more attempt to transmit the IFR byte. If it is transmitting the

byte when TEOD is set, the BDLC module will continue the transmission until it is successful or it loses arbitration to another transmitter. At this point it will then discard the byte and make no more transmit attempts.

NOTE

When transmitting a Type 2 IFR, the user should monitor the number of IFR bytes received to ensure that the overall message length does not exceed the 12-byte limit for the length of SAE J1850 messages. The user should set the TEOD bit when the 11th byte is received, which will prevent the 12-byte limit from being exceeded.

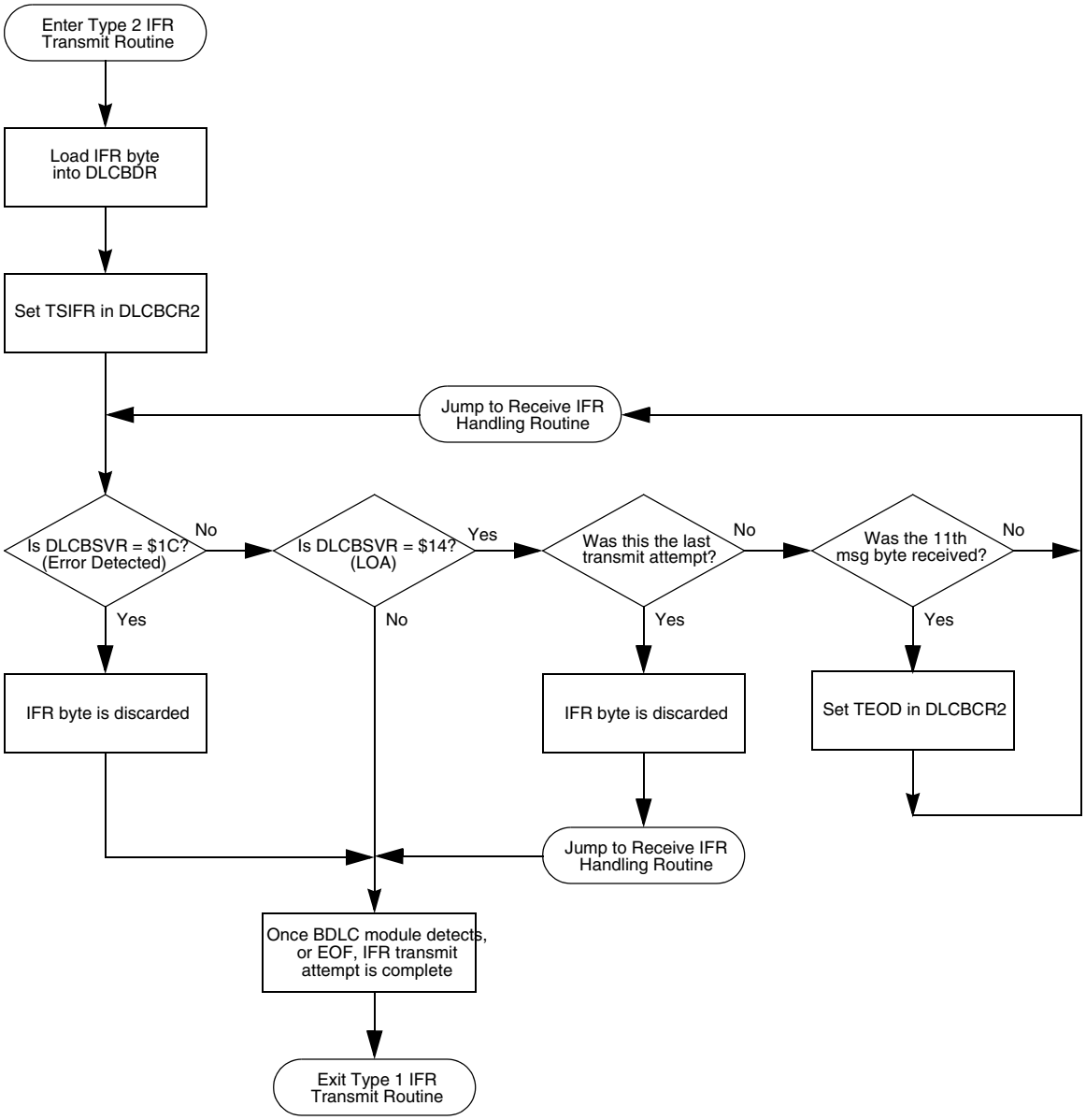


Figure 20-16. Transmitting A Type 2 IFR

- Transmitting a Type 3 IFR

  Transmitting a Type 3 IFR, with or without a CRC byte, is done in a fashion similar to transmitting a message frame. The user loads the first byte to be transmitted into the BDLC Data Register and then sets the appropriate TMIFR bit, depending upon whether a CRC byte is desired. When the last byte is written to the BDLC Data Register, the TEOD bit is set, and a CRC byte (if desired) and an EOD are then transmitted. Because the two versions of the Type 3 IFR are transmitted identically, the description which follows will discuss both. For an illustration of the Type 3 IFR transmit sequence, refer to Figure 20-17.

  — Step 1: Load the First IFR Byte into the BDLC Data Register

The user begins initiation of a Type 3 IFR, as with each of the other IFR types, by loading the desired IFR byte into the BDLC Data Register. If a byte has already been written into the BDLC Data Register for transmission as a new message, the user can simply write the first IFR byte to the BDLC Data Register, replacing the previously written byte. This must be done before the first EOD symbol is received.

— Step 2: Set the TMIFR Bit

The second step necessary for transmitting a Type 3 IFR is to set the desired TMIFR bit in BDLC Control Register 2, depending upon whether or not a CRC is desired. As previously described in Section 20.8.6.2, *BDLC IFR Transmit Control Bits*, the TMIFR1 bit should be set if the user requires a CRC byte to be appended following the last byte of the Type 3 IFR, and TMIFR0 if no CRC byte is required.

Setting the TMIFR1 or TMIFR0 bit will direct the BDLC module to transmit the byte in the BDLC Data Register as the first byte of a single or multi-byte IFR preceded by the appropriate Normalization Bit. Once this has occurred, the BDLC State Vector Register will reflect that the next byte of the IFR can be written to the BDLC Data Register (TDRE interrupt).

NOTE

The user must set the TMIFR1 or TMIFR0 bit before the EOD following the main part of the message frame is received, or no IFR transmit attempts will be made for the current message. If another node does transmit an IFR to this message or a reception error occurs, the TMIFR1 or TMIFR0 bit will be cleared. If not, the IFR will be transmitted after the EOD of the next received message.

— Step 3: When TDRE is Indicated, Write the Next IFR Byte into the BDLC Data Register

When a TDRE state is reflected in the BDLC State Vector Register, the CPU writes the next IFR byte to be transmitted into the BDLC Data Register, clearing the TDRE interrupt. This step is repeated until the last IFR byte to be transmitted is written to the BDLC Data Register.

NOTE

As when transmitting a message, when transmitting a Type 3 IFR the user may write two, or possibly even three of the bytes to be transmitted into the BDLC Data Register before the first RxIFR interrupt occurs. For this reason, the user should never use receive IFR byte interrupts to control the sequencing of IFR bytes to be transmitted.

— Step 4: Write the Last IFR Byte into the BDLC Data Register and Set TEOD

Once the last IFR byte to be transmitted is written to the BDLC Data Register, the CPU then sets the TEOD bit in BDLC Control Register 2. Once the TEOD bit is set, after the last IFR byte written to the BDLC Data Register is transmitted onto the bus, if the TMIFR1 bit has been set the BDLC module will begin transmitting the CRC byte, followed by an EOD. If the TMIFR0 bit has been set, the last IFR byte will immediately be followed by the transmission of an EOD. Following the EOD, and EOF will be recognized and the message will be complete.

If at any time during the transmission of a Type 3 IFR a loss of arbitration occurs, the TMIFR bit which is set and the TEOD bit (if set) will be cleared, any IFR byte being transmitted will be discarded and the loss of arbitration state will be reflected in the BDLC State Vector Register. Likewise, if an error is detected during the transmission of a Type 3 IFR the IFR control bits will be cleared, the byte being transmitted will be discarded and the BDLC State Vector Register will reflect the detected error.

NOTE

If the Type 3 IFR being transmitted is made up of a single byte, the appropriate TMIFR bit and the TEOD bit can be set at the same time. The BDLC module will then treat that byte as both the first and last IFR byte to be sent.

20.8.6.7 Transmitting IFR Exceptions

This basic IFR transmitting flow can be interrupted for the same reasons as a normal message transmission. The IFR transmit process can be adversely affected due to a loss of arbitration, an Invalid or Out of Range Symbol, or due to a transmitter underrun caused by the CPU failing to service a TDRE interrupt in a timely fashion. For a description of how these exceptions can affect the IFR transmit process, refer to Section 20.8.4.2, *Transmitting Exceptions*.

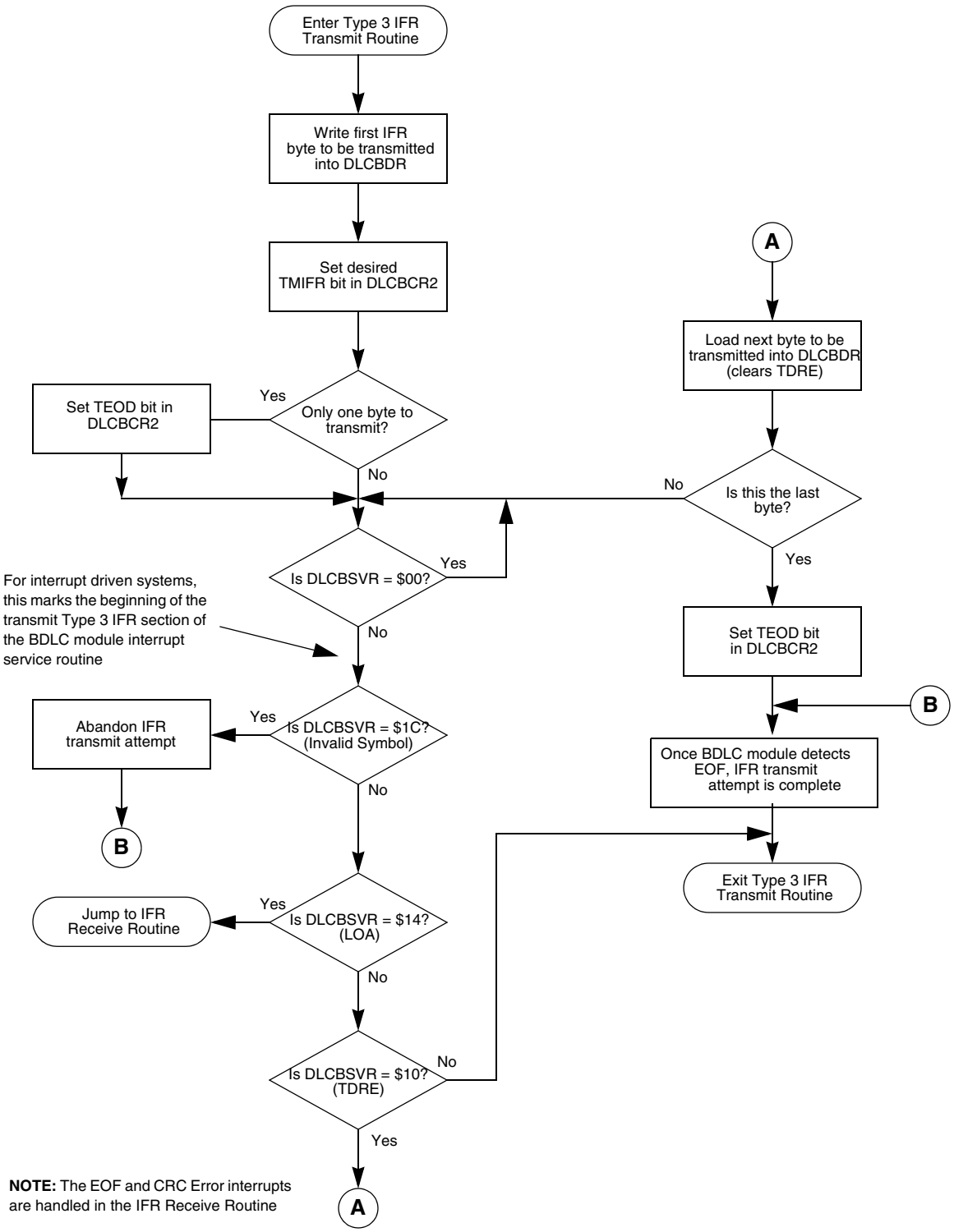


Figure 20-17. Transmitting A Type 3 IFR

20.8.7 Receiving An In-Frame Response (IFR)

Receiving an In-Frame Response with the BDLC module is very similar to receiving a message frame. As each byte of an IFR is received, the BDLC State Vector Register will indicate this to the CPU. An EOF indication in the BDLC State Vector Register indicates that the IFR (and message) is complete. Also, the IMSG bit can also be used to command the BDLC module to mask any further network activity from the CPU, including IFR bytes being received, until the next valid SOF is received.

NOTE

As with a message transmission, the IMSG bit should never be used to ignore the BDLC module's own IFR transmissions. This is again due to the BDLC State Vector Register bits being inhibited from updating until IMSG is cleared, preventing the CPU from detecting any IFR-related state changes which may be of interest.

20.8.7.1 Receiving an IFR with the BDLC module

Receiving an IFR from the SAE J1850 bus requires the same procedure that receiving a message does, except that as each byte is received the Received IFR Byte (RxIFR) state is indicated in the BDLC State Vector Register. All other actions are the same. For an illustration of the steps described below, refer to Figure 20-18.

- Step 1: When RxIFR Interrupt Occurs, Retrieve IFR Byte

  When the first byte of an IFR following a valid EOD symbol is received that byte is placed in the BDLC Data Register, and an RxIFR state is reflected in the BDLC State Vector Register. No indication of the EOD reception in made, since the RxIFR state will indicate that the main portion of the message has ended and the IFR portion has begun.

  The RxIFR interrupt is cleared when the received IFR byte is read from the BDLC Data Register. Once this is done, no further CPU intervention is necessary until the next IFR byte is received, and this step is repeated. As with a message reception, all bytes of the IFR, including the CRC byte, will be placed into the BDLC Data Register as they are received for the CPU to retrieve.

- When an EOF is Received, the IFR (and Message) is Complete

  Once all IFR bytes (including the possible CRC byte) have been received from the bus, the bus will again be idle for a time period equal to an EOD symbol. Following this, the BDLC module will determine whether or not the last byte of the IFR is a CRC byte, and if so verify that the CRC byte is correct. If the CRC byte is not correct, this will be reflected in the BDLC State Vector Register.

  After an additional period of time the EOD symbol will transition into an EOF symbol. When the EOF is received it will be reflected in the BDLC State Vector Register, indicating to the user that the IFR, and the message, is complete.

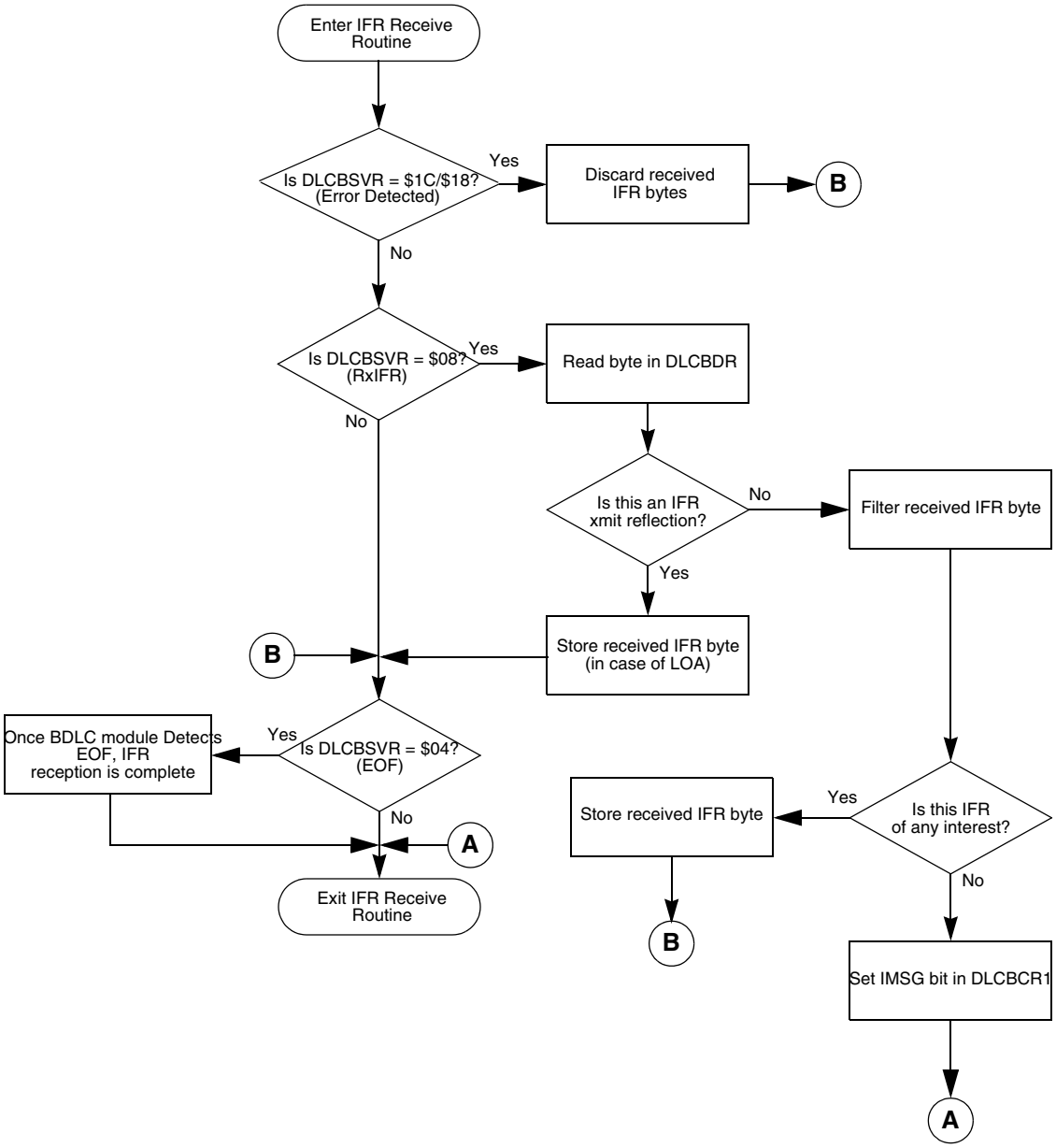


Figure 20-18. Receiving An IFR With the BDLC module

20.8.7.2 Receiving IFR Exceptions

This basic IFR receiving flow can be interrupted for the same reasons as a normal message reception. The IFR receiving process can be adversely affected due to a CRC error, an Invalid or Out of Range Symbol or due to a receiver overrun caused by the CPU failing to service an RxIFR interrupt in a timely fashion. For a description of how these exceptions can affect the IFR receiving process, refer to Section 20.8.5.4, *Receiving Exceptions*.

20.8.8 Special BDLC Module Operations

There are a few special operations which the BDLC module can perform. What follows is a brief description of each of these functions and when they might be used.

20.8.8.1 Transmitting Or Receiving A Block Mode Message

The BDLC module, because it handles each message on a byte-by-byte basis, has the inherent capability of handling messages any number of bytes in length. While during normal operation this requires the user to carefully monitor message lengths to ensure compliance with SAE J1850 message limits, often in a production or diagnostic environment messages which exceed the SAE J1850 limits can be beneficial. This is especially true when large amounts of configuration data need to be downloaded over the SAE J1850 network.

Because of the BDLC module's architecture, it can both transmit and receive messages of unlimited length. The CRC calculations, both for transmitting and receiving, are not limited to eight bytes, but will instead be calculated and verified using all bytes in the message, regardless of the number. All control bits, including TEOD and IMSG, also work in an identical manner, regardless of the length of the message.

To transmit or receive these "Block Mode" messages, no extra BDLC module control functions must be performed. The user simply transmits or receives as many bytes as desired in one message frame, and the BDLC module will operate just as if a message of normal length was being used.

20.8.8.2 Transmitting Or Receiving A Message In 4X Mode

In a diagnostic or production environment large amounts of data may need to be downloaded across the network to a component or module. This data is often sent in a large "Block Mode" message (see above) which violates the SAE J1850 limit for message length. In order to speed up the downloading of these large blocks of data, they are sometimes transmitted at four times (4X) the normal bit rate for the Variable Pulse Width modulation version of SAE J1850. This higher speed transmission, nominally 41.6kbps, allows these large blocks to be transmitted much more quickly.

The BDLC module is designed to receive and transmit messages at this higher speed. By setting the 4XE bit in BDLC Control Register 2, the user can command the BDLC module to transmit and receive any message over the network at a 4X rate.

If the BDLC module is placed in this 4X mode, messages transmitted at the normal bit rate will not be received correctly. Likewise, 4X messages transmitted on the SAE J1850 bus when the BDLC module is in normal mode will be interpreted as noise on the network by the BDLC module. For more information on the 4XE bit, refer to Section • 4X Mode.

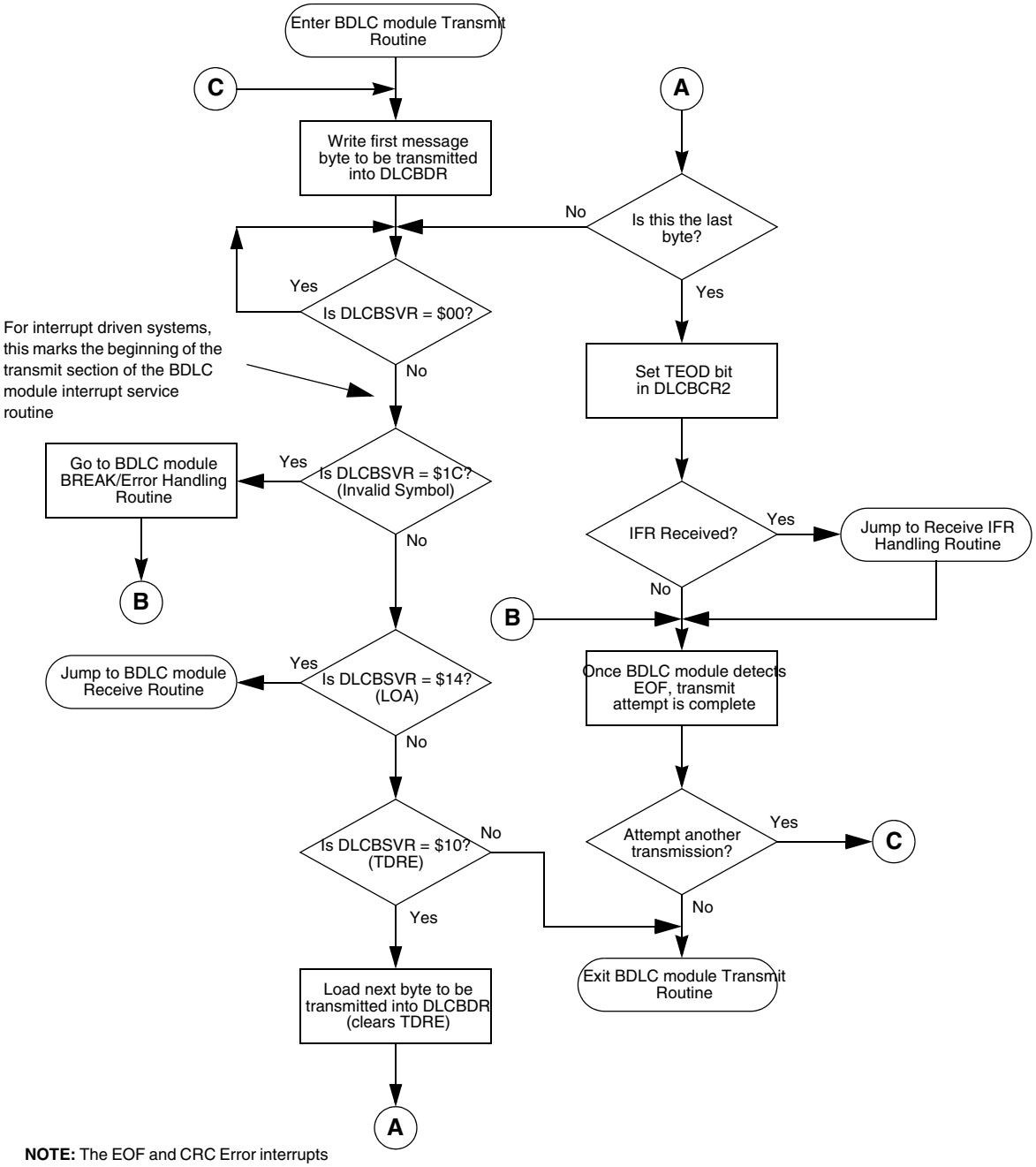


Figure 20-19. Basic BDLC Module Transmit Flowchart

20.8.9 BDLC Module Initialization

This section includes sample flows for initializing the BDLC module and using it to transmit and receive messages.

20.8.9.1 Initialization Sequence

To initialize the BDLC module, the user should first write the desired data to the configuration bits. The BDLC module should then be taken out of digital and analog loopback mode and enabled. Exiting from loopback mode will entail change of state indications in the BDLC State Vector Register which must be dealt with. Once this is complete, CPU interrupts can be enabled (if desired), and then the BDLC module is capable of SAE J1850 serial network communication. For an illustration of the sequence necessary for initializing the BDLC module, refer to Figure 20-20.

20.8.9.2 Initializing the Configuration Bits

The first step necessary for initializing the BDLC module following an MCU reset is to write the desired values to each of the BDLC module control registers. This is best done by storing predetermined initialization values directly into these registers. The following description outlines a basic flow for initializing the BDLC module. This basic flow does not detail more elaborate initialization routines, such as performing digital and analog loopback tests before enabling the BDLC module for SAE J1850 communication. However, from the following descriptions and the BDLC module specification, the user should be able to develop routines for performing various diagnostic procedures such as loopback tests.

- Step 1 - Initialize BDLC Analog Round Trip Delay Register

  Begin initialization of the configuration bits by writing the desired analog transceiver configuration data into the BDLC Analog Round Trip Delay Register. Following this write to BDLC Analog Round Trip Delay Register, all of these bits will become read only.

- Step 2- Initialize BDLC Baud Rate Select Register

  The next step in BDLC module initialization is to write the desired bus clock divisor minus one into the BDLC Baud Rate Select RegisterBDLC Baud Rate Select Register. The divisor should be chosen to generate a 1 MHz or 1.048576 MHz mux interface clock (f_{bdlc}). Following this write to BDLC Baud Rate Select Register, all of these bits will become read only.

- Step 3- Initialize BDLC Control Register 2

  The next step in BDLC module initialization should be writing the configuration bits into the BDLC Control Register 2 register. This initialization description assumes that the BDLC module will be put into normal mode (not 4X mode), and that the BDLC module should not yet exit either digital or analog loopback mode. Therefore, this step should write SMRST and DLOOP as logic ones, 4XE as a logic zero, write NBFS to the desired level, and write TEOD, TSIFR, TMIFR1 and TMIFR0 as logic zeros. These last four bits MUST be written as logic zeros in order to prevent undesired operation of the BDLC module.

- Step 4- Initialize BDLC Control Register 1

  The next step in BDLC module initialization is to write the configuration bits in BDLC Control Register 1. The CLKS bit should be written to its desired values at this time, following which it will become read-only. The IE bit should be written as a logic zero at this time so BDLC module interrupts of the CPU will remain masked for the time being. The IMSG bit should be written as a logic one to prevent any receive events from setting the BDLC State Vector Register until a valid SOF (or BREAK) symbol has been received by the BDLC module.

20.8.9.3 Exiting Loopback Mode and Enabling the BDLC module

Once the configuration bits have been written to the desired values, the BDLC module should be taken out of loopback and connected to the SAE J1850 bus. This is done by clearing the DLOOP bit and then setting the BDLCE bit in the BDLC Control Register.

- Step 5- Perform Loopback Tests (optional)

  Once the BDLC module is configured for desired operation, the user may wish to perform digital and/or analog loopback tests to determine the integrity of the link to the SAE J1850 network. This would involve leaving the DLOOP bit (BDLC Control Register 2) set, setting the BDLCE bit, preforming the desired loopback tests and finally exiting digital loopback mode by clearing DLOOP in the BDLC Control Register 2.

- Step 6- Exit Loopback Mode and enable the BDLC module

  If loopback mode tests are not to be preformed the BDLC module can be removed from digital loopback mode by clearing the DLOOP bit. The BDLC module can then be enabled by setting the BDLCE bit in the BDLC Control Register.

  Once DLOOP is cleared and BDLCE is set, the BDLC module is ready for SAE J1850 communication. However, to ensure that the BDLC module does not attempt to receive a message already in progress or to transmit a message while another device is transmitting, the BDLC module must first observe an EOF symbol on the bus before the receiver will be activated. To activate the transmitter, the BDLC module will need to observe an Inter-Frame Separator symbol.

20.8.9.4 Enabling BDLC Interrupts

The final step in readying the BDLC module for proper communication is to clear any pending interrupt sources and then, if desired, enable BDLC module interrupts of the CPU.

- Step 7- Clear Pending BDLC Interrupts

  In order to ensure that the BDLC module does not immediately generate a CPU interrupt when interrupts are enabled, the user should read the BDLC State Vector Register to determine if any BDLC module interrupt sources are pending before setting the IE bit in the BDLC Control Register 1. If the BDLC State Vector Register reads as a %00000000, no interrupts are pending and the user is free to enable BDLC interrupts, if desired.

  If the BDLC State Vector Register indicates that an interrupt is pending, the user should perform whatever actions are necessary to clear the interrupt source before enabling the interrupts. Whether any interrupts are pending will depend primarily upon how much

time passes between the exit from loopback modes and enabling the BDLC module and the enabling of interrupts. It is a good practice to always clear any source of interrupts before enabling interrupts on any MCU subsystem.

If any interrupts are pending (BDLC State Vector Register not %00000000), then each interrupt source should be dealt with accordingly. Once all of the interrupt sources have been dealt with, the BDLC State Vector Register should read %00000000, and the user is then free to enable BDLC interrupts.

- Step 8- Enable BDLC Interrupts

  The last step in initializing the BDLC module is to enable interrupts to the CPU, if so desired. This is done by simply setting the IE bit in the BDLC Control Register 1. Following this, the BDLC module is ready for operating in interrupt mode. If the user chooses not to enable interrupts, the BDLC State Vector Register must be polled periodically to ensure that state changes in the BDLC module are detected and dealt with appropriately.

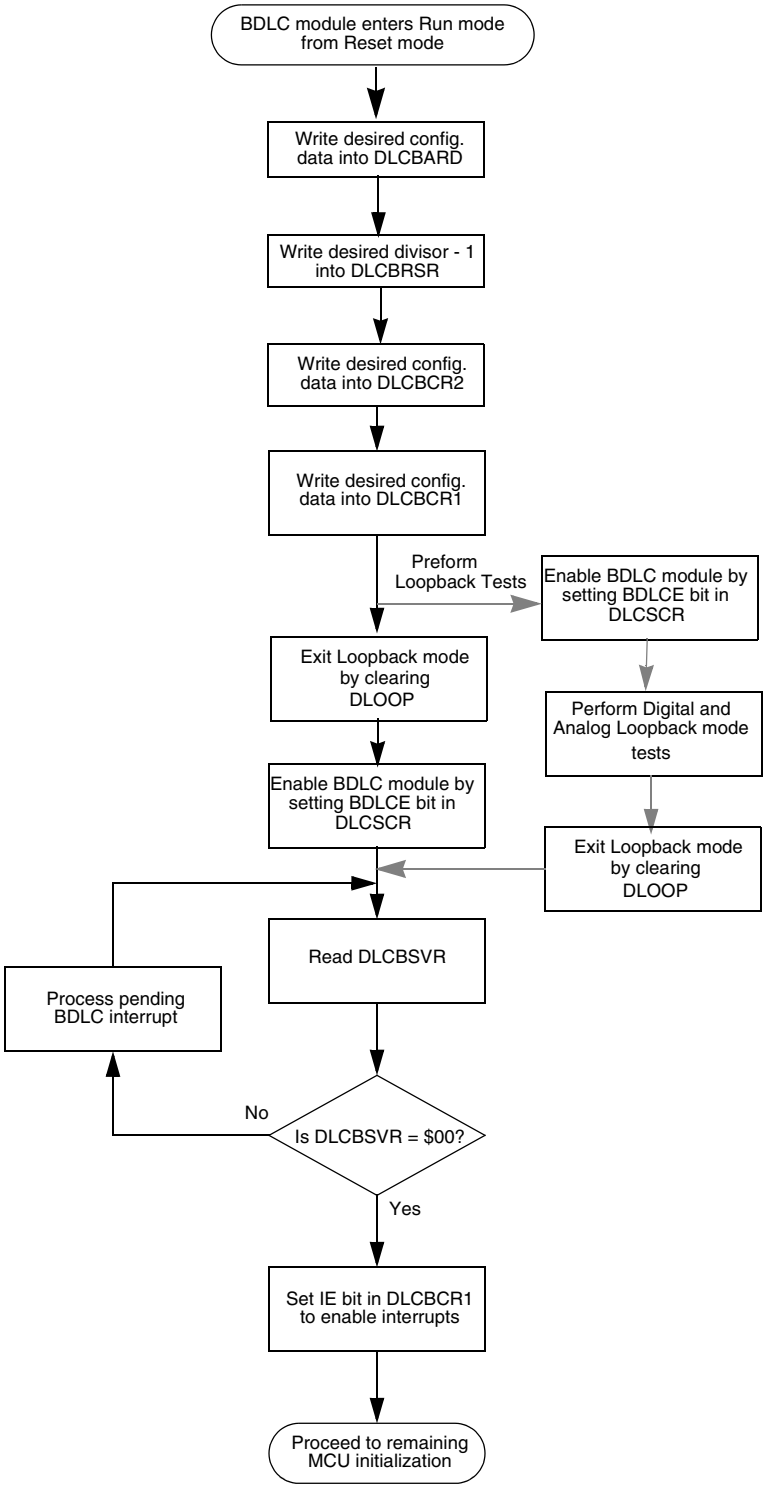


Figure 20-20. Basic BDLC Module Initialization Flowchart

20.9 Resets

20.9.1 General

The reset state of each individual bit is listed within Section 20.7, *Memory Map and Registers* which details the registers and their bit-fields.

Chapter 21
Debug Support and JTAG Interface

21.1 Overview

The following sections are contained in this document:

The MPC5200 provides the user an IEEE 1149.1 JTAG interface to facilitate board/system testing. It also provides a Common On-Chip Processor (COP) Interface, which shares the IEEE 1149.1 JTAG port. The COP Interface provides access to the MPC5200's imbedded Freescale MPC603e G2_LE processor. This interface provides a means for executing test routines and for performing software development & debug functions.

21.2 TAP Link Module (TLM) and Slave TAP Implementation

The MPC5200 debug and development logic consists of:

- a master TAP Link Module (TLM), which implements the mandatory instructions of the IEEE JTAG 1149.1 standard.
- a slave JTAG TAP block dedicated to microprocessor debug functions, which are contained within the imbedded G2_LE microprocessor.

The master/slave TLM/TAP architecture is not yet an approved extension of the IEEE standard. It is, however, interface-compatible with the standard.

The TLM state machine is active at all times.

The TLM and slave TAP blocks each consist of:

- a TAP Controller state machine
- Instruction Register (IR)
- instruction decode
- various Data Registers (DR)

There is no inherent limit to the number of slave TAP blocks. However, no more than one slave TAP Controller state machine, designated by its asserted Enable, is active at any time. The slave TAP state machines have an Enable input and a Select output not present in the IEEE standard.

- All slave Enable signals are generated by the TLM block. No more than one Enable signal is ever asserted at one time.
- All slave Select signals are inputs to the TLM block. Any number of Select signals may be asserted at any time.

The TLM block contains a Link DR that determines which, if any, slave TAP block is active.

- When a slave TAP block is inactive, its TAP Controller state machine is locked in the RunTestIdle state, preventing its IR and DRs from shifting.
- When a slave TAP block is active, the TLM IR and DRs (except the TLM Link DR described below) are disabled. However, the TLM state machine continues to respond to the TAP interface signals TRST, TCK, and TMS.

The TLM Link DR can be shifted while a slave TAP is active. This is done by loading the slave IR with an instruction that activates the Select signal, then performing a DR scan operation. This only affects the TLM Link DR, because the TLM IR selected the Link DR to enable a slave in the first place, and the TLM IR cannot change while a slave is active.

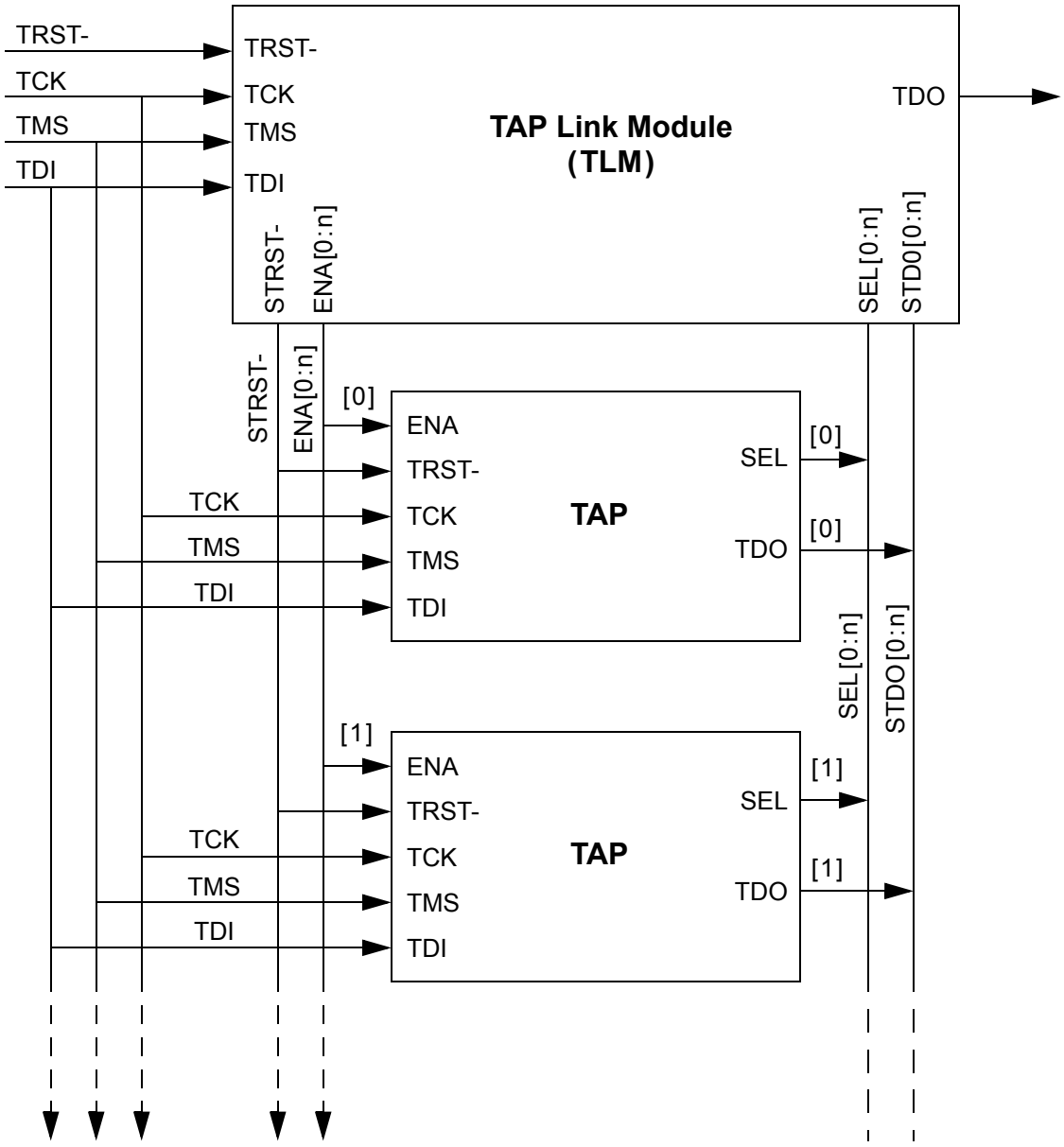


Figure 21-1. Generic TLM/TAP Architecture Diagram

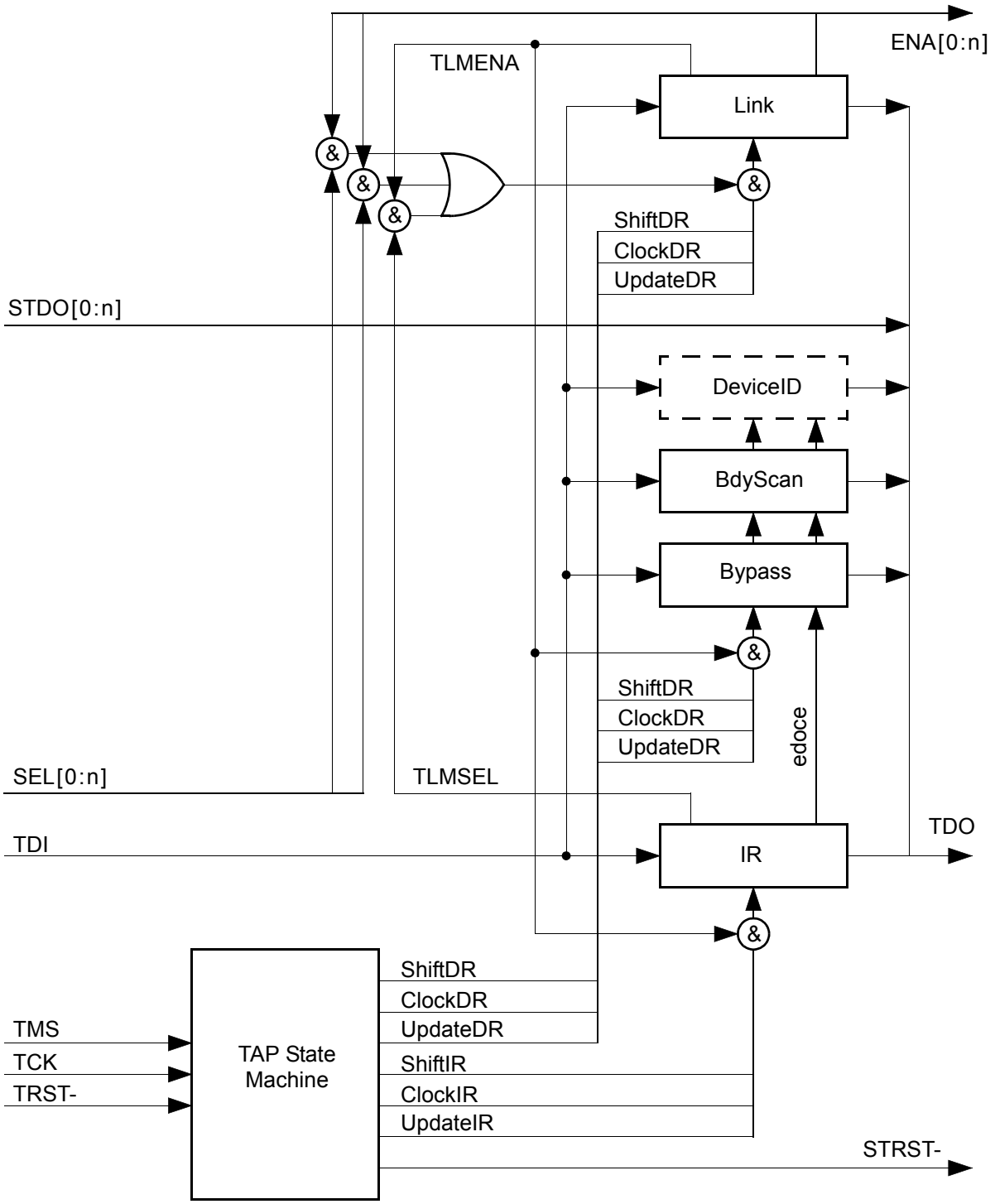


Figure 21-2. Generic TAP Link Module (TLM) Diagram

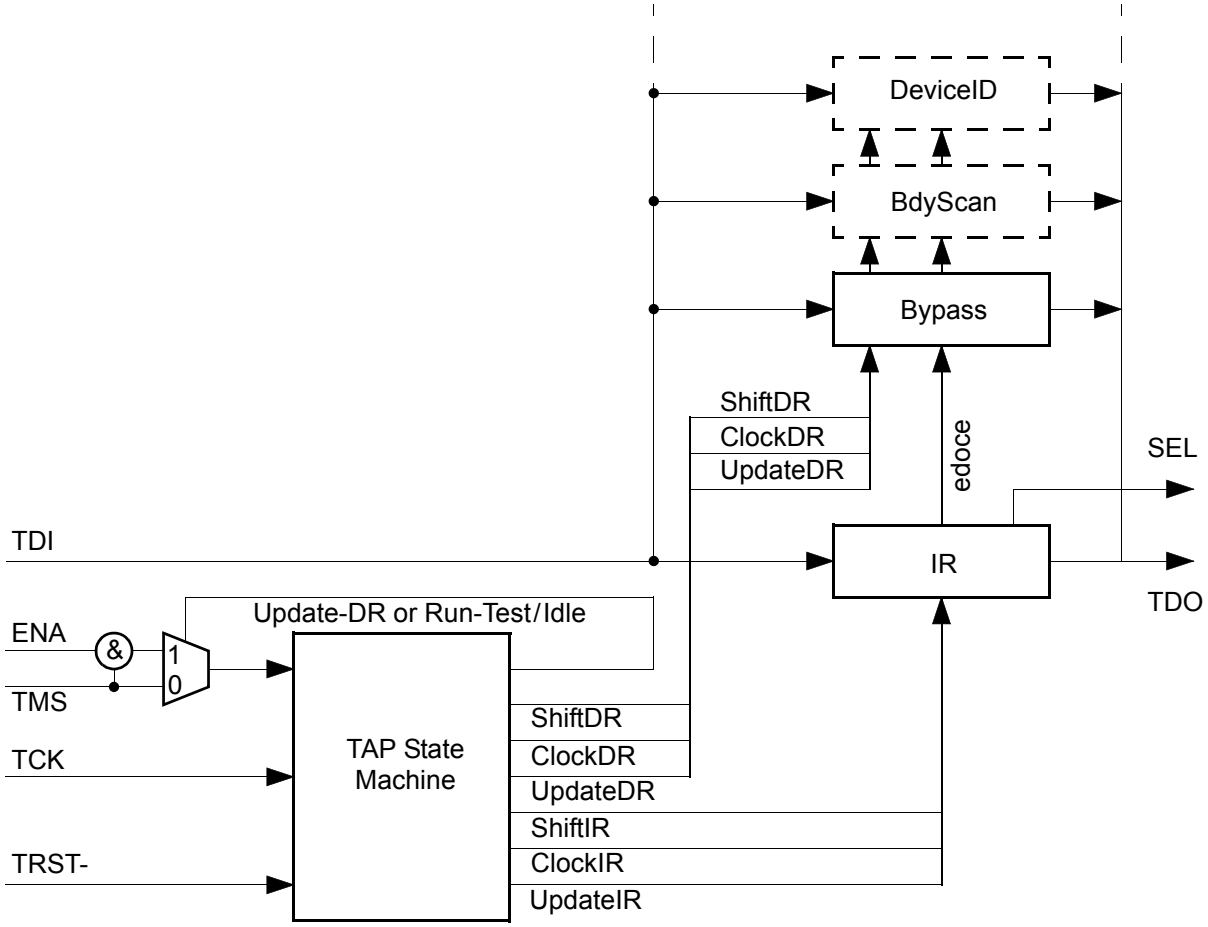


Figure 21-3. Generic Slave TAP

21.3 TLM and TAP Signal Descriptions

21.3.1 Test Reset (TRST)

JTAG reset, active low. When asserted, any on-going JTAG operation is immediately aborted. All TAP state machines, including the TLM, immediately enter the Test-Logic-Reset state. Other JTAG input signals (TCK, TMS, and TDI) have no effect while TRST is asserted. TDO is immediately tri-stated.

21.3.2 Test Clock (TCK)

This is the JTAG clock. The (non-reset) behavior of the active TAP and TLM state machines is governed by the TMS value at the TCK rising edge. TDI value is sampled at the TCK rising edge for all shift operations. All TDO non-reset transitions (including impedance) occur at the TCK falling edge. All shift register capture operations occur at the TCK rising edge. All shift register Update operations occur at the TCK falling edge.

21.3.3 Test Mode Select (TMS)

TAP state machine control, including TLM. The state of TMS at rising edges of TCK uniquely determines the state sequence of the TLM and the active TAP state machines. See Figure 21-4. Inactive TAPs ignore TMS completely.

21.3.4 Test Data In (TDI)

Serial test data input can be routed to any IR or DR, as determined by the state of the active TAP state machine and the contents of the active IR. TDI is sampled at the TCK rising edge while the active TAP state machine is in either the Shift-IR or Shift-DR state.

21.3.5 Test Data Out (TDO)

Serial test data output is routed from the active shift register to this pin. To ensure setup and hold time for TDO when connected to TDI (of another device), TDO switches at the TCK falling edge. TDO is driven while the TLM state machine is in the Shift-IR or Shift-DR states only; it is tri-stated in all other TAP states. Except, for the first half clock after exiting the shift state, because of its falling edge timing.

21.4 Slave Test Reset (STRST)

STRST is the active-low reset from the TLM to all slave TAP blocks. STRST is asserted whenever the TLM state machine is in the Test Logic Reset state. This is a result of TRST being asserted, or the TMS sequence.

21.4.1 Enable Slave—ENA[0:n]

Enable signals are decoded from the contents of the TLM:Link DR. There is one Enable signal for the TLM and one for each slave TAP block. No more than one Enable signal can be asserted at one time. Each slave TAP block gates (logical AND) TMS with a unique Enable signal. Any number of TLM:Link DR codes may activate any Enable signal. MPC5200 implements one TLM:Link DR code for each Enable signal.

21.4.2 Select DR Link—SEL[0:n]

Each slave TAP block generates one Select signal; its value is decoded from the contents of its IR. Any number of Select signals may be asserted at any time; the TLM ignores all SEL[0:n] signals except from the active slave TAP (if any). Instruction codes that activate Select may be different in each slave TAP block. Any number of instruction codes may activate Select, but the mandatory BYPASS, EXTEST, and SAMPLE:PRELOAD instructions (and IDCODE, if implemented) must not. However, a slave is allowed to implement additional instructions that behave identically to any of these instructions, except that Select is asserted and the normal DR is disabled.

21.4.3 Slave Test Data Out—STDO[0:n]

Each slave TAP block provides a serial test data output. Just like TDO, all transitions of STDO[0:n] must occur on the falling edge of TCK. ENA[0:n] and SEL[0:n] select either the active slave serial output data or the TLM serial output data to appear at the TDO pin.

21.5 TAP State Machines

All TAP state machines are the same, including the TLM, except for the single control signal. The TLM receives the external TMS signal unmodified; all other (slave) TAP Controllers respond to unique versions of TMS combined with their unique Enable signal.

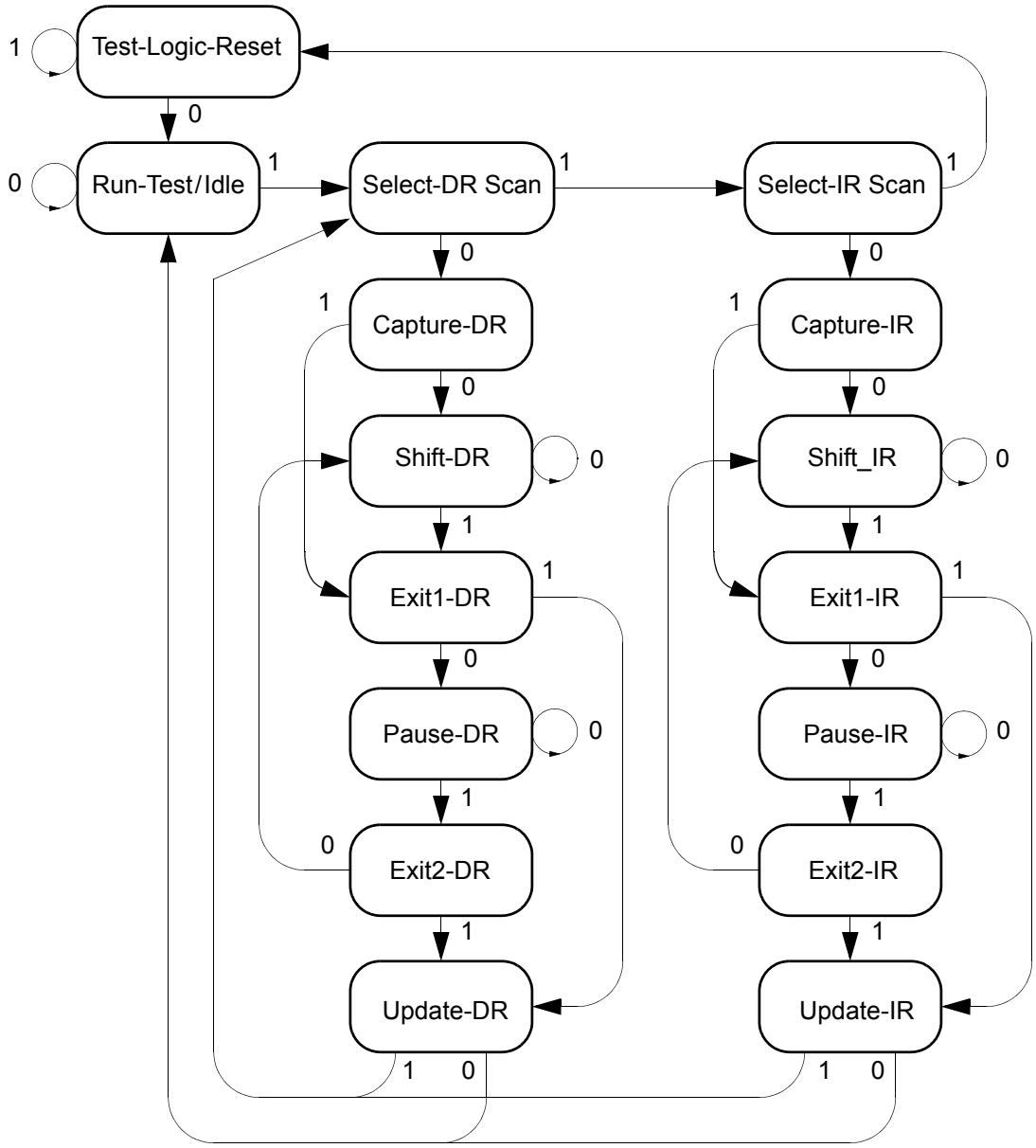


Figure 21-4. State Diagram—TAP Controller

Instructions are loaded by stepping the state machine to the Shift-IR state by applying an appropriate sequence of values on TMS at successive rising edges of TCK. Once in the Shift-IR state, TMS is held low and appropriate values are applied at TDI (lsb-first) at successive rising edges of TCK. As the last (ms) bit is applied at TDI, TMS is set high and the state machine is advanced through the Exit1-IR and Update-IR states. The instruction becomes effective at the falling edge of TCK in the Update-IR state.

Data registers are loaded by first selecting the desired data register with an appropriate instruction, then stepping the state machine to the Shift-DR state. Once in the Shift-DR state, TMS is held low and appropriate values are applied at TDI (lsb-first) at successive rising edges of TCK. As the last (ms) bit is applied at TDI, TMS is set high and the state machine is advanced through the Exit1-DR and Update-DR states. The data becomes effective at the falling edge of TCK in the Update-DR state.

21.6 G2_LE Core JTAG/COP Serial Interface

The Common On-chip Processor (COP) external interface adheres to the IEEE 1149.1 serial protocol. The COP uses the JTAG interface which includes a TAP Controller, a COP Controller, input and output multiplexors, registers, several shift register latches (SRLs) and a counter (RunN) which controls clock execution. All IEEE 1149.1 public instructions are implemented (SAMPLE_PRELOAD, BYPASS, and EXTEST). Figure 21-5 shows the components that make up the microprocessor JTAG/COP serial interface.

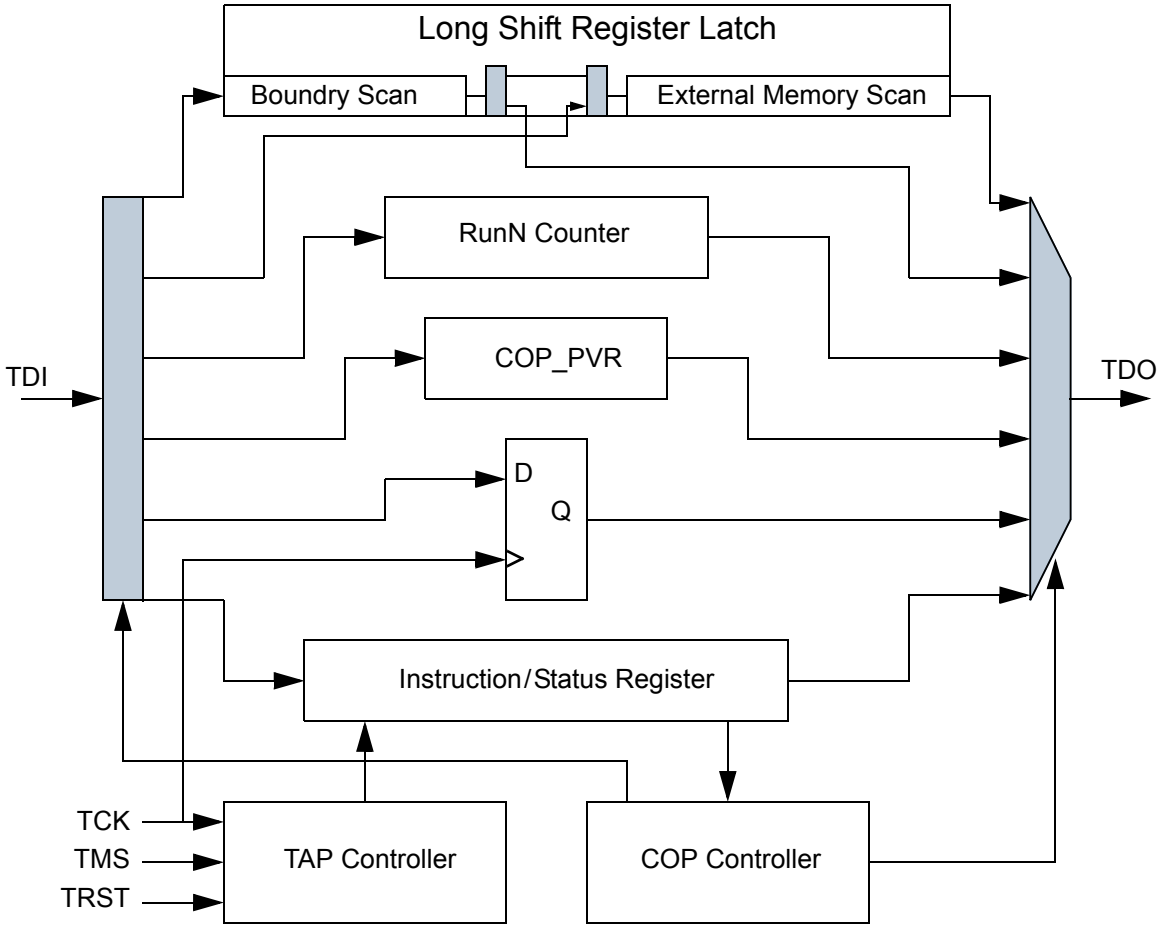


Figure 21-5. G2_LE Core JTAG/COP Serial Interface

21.7 TLM Link DR Instructions

— CAUTION —

1. For the following registers, only the instruction codes listed should be used. All other codes must be considered private and potentially damaging.
2. "Persistent" means an instruction's effect(s) persist even after it is overwritten in the register.
3. The reset value shown is the update register reset value. Per the JTAG standard, the raw IR shift register reset value is irrelevant.

Table 21-1. TLM Link-DR Instructions

Instruction	Encoding (ENA[1:0])	Persistent
TLM:TLMENA	01	N3
TLM:PPCENA	10	N

Note:

1. Reset = TLM:TLMENA
2. Capture = Current Value
3. Link Pseudo-instructions are persistent with respect to the enabled IR, but not with respect to the contents of the TLM:Link DR itself.

Link pseudo-instructions are loaded into the 2-bit TLM:Link DR when it is selected by instructions of the TLM or slave TAP blocks. The value shifted into the TLM:Link DR determines which IR will be active after the Update-DR state. The selection remains in effect until the TLM:Link register is selected again, and modified.

21.7.1 TLM:TLMENA

The TLM:TLMENA pseudo-instruction selects the 6-bit TLM IR.

21.7.2 TLM:PPCENA

The TLM:PPCENA pseudo-instruction selects the 8-bit microprocessor CPU test IR.

21.8 TLM Test Instructions

The TLM IR activates device-level functions, including the mandatory JTAG instructions and private device test data registers.

Table 21-2. TLM Test Instruction Encoding

Instruction	Encoding	Persistent	TLM Register
IDCODE	011101	N	Device_ID
BYPASS	111111	N	Bypass
SAMPLE/PRELOAD	100000	N	Boundary
EXTEST	000000	N	Boundary
CLAMP	100001	N	Bypass
HIGHZ	011111	N	Bypass
RESET: IDCODE Capture: IDCODE 0 1 1 1 0 1			

21.8.1 IDCODE

The IDCODE instruction selects the 32-bit DeviceID DR to be logically connected between TDI and TDO during DR shift operations. The capture value of the DeviceID DR identifies the manufacturer (Freescale), device type (MPC5200), and device revision level.

21.8.1.1 Device ID Register

Table 21-3. Device ID Register = 0001101D hex

Version	Device (MPC5200 – Initial Release)	Manufacturer (Freescale)	
0000	0000 0000 0001 0001	0000 0001 110	1

21.8.2 BYPASS

The BYPASS instruction selects the 1-bit Bypass DR to be logically connected between TDI and TDO during DR shift operations. It performs no testing function. The Bypass register provides for a minimum-length serial datapath from TDI to TDO. This allows more rapid test data movement to and from other JTAG scan chain components. The Bypass register capture value is 0. The Bypass register update value has no effect.

21.8.3 SAMPLE/PRELOAD

The SAMPLE/PRELOAD instruction selects the Boundary Scan DR to be logically connected between TDI and TDO during DR shift operations. As the name implies, the SAMPLE/PRELOAD instruction has two distinct uses:

Sample: To capture and examine the device pins state without disturbing normal system operation. Signals are captured at the TCK rising edge in the Capture-DR state, and examined by shifting out. For captured values to be meaningful, TCK may need to be synchronized to the normal system clock. The update value has no effect.

Preload: To shift an initial value into the boundary scan register prior to loading the EXTEST or CLAMP instruction into the Instruction register. Capture value may be examined or ignored. Update value has no effect until EXTEST/CLAMP instruction is loaded. It is then presented at the device pins.

21.8.4 EXTEST

The EXTEST instruction selects the Boundary Scan DR to be logically connected between TDI and TDO during DR shift operations. It also forces the Boundary Scan register contents to appear at the pins of the device. The state of all pins is captured at the TCK rising edge in the Capture-DR TAP Controller state. The update value appears on the pins at the TCK falling edge in the Update-DR state. EXTEST does not affect on-chip pull-up or pulldown resistors.

21.8.5 CLAMP

The CLAMP instruction forces the contents of the Boundary Scan DR to appear at the boundary of the microprocessor block, just like the EXTEST instruction, but selects the 1-bit Bypass DR to be logically connected between TDI and TDO during DR shift operations. This allows a static data pattern to be driven onto the device pins, while at the same time minimizing the length of shifts to access test data registers on other devices in the JTAG scan chain. CLAMP does not affect on-chip pull-up or pull-down resistors.

21.8.6 HIGHZ

The HIGHZ instruction selects the 1-bit Bypass DR to be logically connected between TDI and TDO during DR shift operations, and also forces all output and bidirectional pins of the device into a non-driving state. Input pins, and the input portion of bidirectional pins, are not affected.

21.9 G2_LE COP/BDM Interface

The MPC5200 functional pin interface and internal logic provides access to the embedded G2_LE processer core through the Freescale standard COP/BDM interface. For information on the connection between COP connector and MPC5200 refer to the MPC5200 Hardware Specifications.

Notes

Appendix A
Acronyms and Terms

This section contains an alphabetical list of terms, phrases, acronyms, and abbreviations used in this book. Some terms and definitions included are reprinted from *IEEE Std. 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*, copyright ©1985 by the Institute of Electrical and Electronics Engineers, Inc. with permission of the IEEE.

A

AAL . . . . . ATM Adaptation Layer

ABR . . . . . Available Bit-Rate. *See also* CBR and UBR.

ACR . . . . . Allowed Cell Rate

addr, adr . . . . . address

alm . . . . . alarm

ALE. . . . . Address Latch Enable

ALU . . . . . Arithmetic Logic Unit

APC. . . . . ATM Pace Control unit

ARB . . . . . Microprocessor Arbitor

Architecture . . . . . A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Asynchronous exception. . . . *Exceptions* that are caused by events external to the processor's execution. In this document, the term 'asynchronous exception' is used interchangeably with the word *interrupt*.

AT . . . . . Address Types

ATA. . . . . Advanced Technology Attachment—a standard interface used with storage devices such as hard disk drives. ATA drives are also referred to as Integrated Drive Electronics (IDE) drives.

ATAPI . . . . . ATA Packet Interface

ATM . . . . . Asynchronous Transfer Mode

Atomic access . . . . . 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 term refers to the fact that the transactions are indivisible). The PowerPC architecture implements atomic access through the **lwarx**/**stwcx** instruction pair.

Autobaud. . . . . The process of determining a serial data rate by timing the width of a single bit.

B

BAT. . . . . Block Address Translation

BB. . . . . Bus Busy

BD. . . . . Buffer Descriptor

BG. . . . . Bus Grant

BI . . . . . Burst Inhibit

Big-Endian (BE). . . . . A byte-ordering method in memory where the address *n* of a word corresponds to the *Most-Significant Byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the Most-Significant Byte. *See also* Little-Endian.

. . . . . In Big-Endian architectures, the leftmost bytes (those with a lower address) are most significant. For example, consider the number 1025 stored in a 4Byte integer as shown in the table below.

00000000 00000000 00000100 00000001		
Addr	**Big-Endian**	**Little-Endian**
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

BIP . . . . . Bit Interleaved Parity

BIST . . . . . Built-In Self Test

BISYNC . . . . . Binary Synchronous communication

Blockage . . . . . A pipeline stall that occurs when an instruction occupies an execution unit and prevents a subsequent instruction from being dispatched.

Boundedly undefined . . . . . A characteristic of certain operations results not rigidly prescribed by the PowerPC architecture. Boundedly undefined results for a given operation may vary among implementations, and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be boundedly undefined are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

bps. . . . . bits per second

BPU. . . . . Branch Processing Unit

BR . . . . . Bus Request

BRC . . . . . Backward Reporting Cells

Breakpoint. . . . . A programmable event that forces the core to take a breakpoint exception.

BT . . . . . Burst Tolerance

BUID. . . . . Bus Unit ID

Burst . . . . . A bus transfer whose data phase consists of a sequence of transfers. For example, on a 64-bit bus, a four-beat burst can transfer four, 64-bit double words.

Bus parking . . . . . A feature that optimizes bus usage by letting a device retain bus mastership without having to rearbitrate.

C

Cache . . . . . High-speed memory component containing recently accessed data and/or instructions (subset of main memory).

Cache coherency . . . . . An attribute in which an accurate and common view of memory is provided to all devices that share a memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache.

Cache flush . . . . . An operation that removes from a cache any data from a specified address range. This operation ensures any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited . . . . . A memory update policy in which the *cache* is bypassed and the load or store is done to or from main memory.

CAM . . . . . Content Addressable Memory

CAN . . . . . Controller Area Network

Cast-outs . . . . . *Cache blocks* that must be written to memory when a cache miss causes a cache block to be replaced.

CBR . . . . . Constant Bit-Rate. *See also* UBR and ABR.

CD. . . . . Carrier Detect

CDM . . . . . Clock Distribution Module

CDV . . . . . Cell Delay Variation

CEPT. . . . . Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations).

CES . . . . . Circuit Emulation Service

cfg . . . . . configuration

Changed bit. . . . . One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. *See also* Page access history bits and Referenced bit.

C/I. . . . . Condition/Indication (channel used in GCI protocol)

Clear . . . . . To cause a bit or bit field to register a value of 0. The opposite of *set*.

CLP . . . . . Cell Loss Priority

cmd . . . . . command

cnt . . . . . count

CODEC. . . . . COder/DECoder, or COmpression/DECommpression

Context synchronization . . . . An operation that ensures:
- all instructions in execution complete past the point where they can produce an *exception*
- all instructions in execution complete in the context in which they began execution
- all subsequent instructions are *fetched* and executed in the new context.

. . . . . Context synchronization may result from executing specific instructions (such as isync or rfi) or when certain events occur (such as an exception).

COP. . . . . Common On-chip Processor

Copy-back. . . . . An operation in which modified data in a *cache block* is copied back to memory.

CP . . . . . Communications Processor

CPI . . . . . Common Part Indicators

CPM . . . . . Communications Processor Module

CPS . . . . . Cells Per Slot

CQ. . . . . Completion Queue

CR . . . . . Condition Register

CRC . . . . . Cyclic Redundancy Check—Error detecting codes that generate a parity check.

Critical-data first. . . . . An aspect of *burst* access that lets requested data (typically a word or double word) in a *cache block* be transferred first.

CS . . . . . Chip Select, or Convergence Sublayer

CSC. . . . . Chip Select Controller—LocalPlus Controller

CSMA. . . . . Carrier Sense Multiple Access

CT . . . . . Connection Table

CTL, ctl. . . . . Control

CTR. . . . . Count Register

CUMB. . . . . Check Unused Mask Bits

D

DABR . . . . . Data Address Breakpoint Register

DAR . . . . . Data Address Register

DDR . . . . . Dual-Data Rate

DEC . . . . . Decrementer (register)

Denormalized number. . . . . A non-zero floating-point number whose *exponent* has:

a reserved value, usually the format's minimum, and

whose explicit or implicit leading significant bit is 0.

Direct-mapped cache . . . . . A cache in which each main memory address can appear in only one location within the cache, operates more quickly when the memory request is a cache hit.

Direct-store . . . . . Interface available only on microprocessors that use the PowerPC architecture; supports direct-store devices from the POWER architecture. When the T-bit of a *segment descriptor* is set, the descriptor defines the region of memory to be used as a direct-store segment.

This facility is being phased out of the architecture and is not likely be supported in future devices. Therefore, software should not depend on it and new software should not use it.

DMA. . . . . Direct Memory Access

DPLL . . . . . Digital Phase-Locked Loop

DPR. . . . . Dual-Port RAM

DR. . . . . Data Register

DRAM . . . . . Dynamic Random Access Memory

DSI . . . . . Data Storage Interrupt

DSISR. . . . . DSI Source Register—a register used for determining the source of a DSI exception.

DTLB . . . . . Data Translation Lookaside Buffer

DTV . . . . . Digital TV

DWPCI . . . . . designware PCI—synopsys designware component

E

EA . . . . . **Effective Address**—The 32- or 64-bit address specified for a load, store, or instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* address or an I/O address.

ED . . . . . Endpoint Descriptor

EEST . . . . . Enhanced Ethernet Serial Transceiver

en. . . . . enable

EPROM. . . . . Erasable Programmable Read-Only Memory

err . . . . . error

ESAR . . . . . Enhanced Segmentation And Reassembly

ETH. . . . . Ethernet

Exception . . . . . A condition encountered by the processor that requires special, supervisor-level processing.

Exception handler . . . . . A software routine that executes when an exception is taken. Normally, the exception handler corrects the condition that caused the exception, or performs some other meaningful task, which may include aborting the program that caused the exception. The address for each exception handler is identified by an exception vector offset defined by the architecture and a prefix selected by the MSR.

Extended opcode. . . . . A secondary opcode field generally located in instruction bits 21–30, that further defines the instruction type. All instructions are one word in length. The most significant 6 bits of the instruction are the *primary opcode*, identifying the type of instruction. *See also* Primary opcode.

Execution synchronization. . . A mechanism by which all instructions in execution are architecturally complete before beginning execution (appearing to begin execution) of the next instruction. Similar to context synchronization, but doesn't force contents of the instruction buffers to be deleted and refetched.

Exponent . . . . . In a floating-point number binary representation, the exponent is the component that signifies the integer power to which the value two is raised in determining the value of the represented number. *See also* Biased exponent.

EXTAL . . . . . External Crystal. *See also* XTAL.

F

FBP . . . . . Free Buffer Pool

FEC . . . . . Fast Ethernet Controller

Fetch . . . . . Retrieving instructions from either the cache or main memory and placing them into the instruction queue.

FIFO . . . . . First-In-First-Out (buffer)

FIR . . . . . Fast Infrared. *See also* MIR and SIR.

FMC . . . . . Forward Monitor Cells

FPR . . . . . Floating Point Register

FPSCR . . . . . Floating Point Status and Control Register

FPU . . . . . Floating Point Unit

FRM . . . . . Forward Resource Management

flg . . . . . flag

FLT . . . . . First-Level Table. *See also* SLT.

Fully-associative . . . . . Addressing scheme where every cache location (every byte) can have any possible address.

G

Gb, Gbit . . . . . Gigabit (written with lowercase b; 1024 megabits)

GB, GByte . . . . . Giga-Byte (written with upper case B; 1024 MegaBytes)

GCI . . . . . General Circuit Interface

GCRA . . . . . Generic Cell Rate Algorithm (leaky bucket)

GFC. . . . . Generic Flow Control

GPCM . . . . . General-Purpose Chip-select Machine

GPIO . . . . . General Purpose Input Output (standard)

GPR. . . . . General-Purpose Register—Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.

GPTMR. . . . . General Purpose Timer

GUI . . . . . Graphical User Interface

H

Harvard architecture . . . . . An architectural model featuring separate caches for instruction and data.

HC, Hc . . . . . Host Controller

HCD . . . . . Host Controller Driver

HDLC . . . . . High-level Data Link Control—a transmission protocol used at the data link layer (layer 2) of the OSI seven layer model for data communications. The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors.

. . . . . HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM.

HEC . . . . . Header Error Control

I

ICTL . . . . . Interrupt Controller

IEEE . . . . . Institute of Electrical and Electronics Engineers

IEEE 754 . . . . . A standard, written by the Institute of Electrical and Electronics Engineers, which defines operations and representations of binary floating-point arithmetic.

I^2C. . . . . Inter-Integrated Circuit

IC . . . . . Input Capture. *Also see* OC and PWM.

IDE . . . . . Integrated Drive Electronics—Interface for connecting additional hard drives to a computer.

IDL . . . . . Inter-chip Digital Link

IDMA . . . . . Internal Direct Memory Access

Illegal instructions. . . . . A class of instructions not implemented for a particular microprocessor. These include instructions not defined by the PowerPC architecture. In addition:

. . . . . For 32-bit implementations, instructions defined for 64-bit implementations only are considered illegal instructions.

. . . . . For 64-bit implementations, instructions defined for 32-bit implementations only are considered illegal instructions.

Implementation . . . . . A particular processor that conforms to the PowerPC architecture, but may differ from other architecture-compliant implementations; for example, in design, feature set, and implementation of *optional* features. The PowerPC architecture has many different implementations.

Implementation-dependent . . An aspect of a feature in a processor's design that is defined by a processor's design specifications, rather than by the PowerPC architecture.

Implementation-specific . . . . An aspect of a feature in a processor's design that is not required by the PowerPC architecture, but for which the PowerPC architecture may provide concessions to ensure processors implementing the feature do so consistently.

Imprecise exception . . . . . A type of *synchronous exception* that is allowed not to adhere to the precise exception model. *See also* Precise exception. The PowerPC architecture lets only floating-point exceptions be handled imprecisely.

individual serial controllers . SCC, SMC, SPI, I^2C, and USB—these individual serial controllers request service from the CPM.

Internal bus . . . . . bus that connects the core and System Interface Unit (SIU).

Instruction latency. . . . . The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

int . . . . . interrupt

Interrupt . . . . . An *asynchronous exception*—on processors that use the PowerPC architecture, interrupts are a special case of exceptions. *See also* asynchronous exception.

IP. . . . . Intellectual Property—a unique number that identifies a particular computer in a network of computers. The IP part of TCP/IP; a protocol used to route a data packet from its source to its destination.

IPBI, IP bus. . . . . IP Bus Interface—the Intellectual Property Bus Interface

IR . . . . . Infrared

IR . . . . . Instruction Register

IrDA, IRDA . . . . . Infrared Data Association

IRQ . . . . . Interrupt Request

ISI . . . . . Instruction Storage Interrupt

ITLB . . . . . Instruction Translation Lookaside Buffer

IU . . . . . Integer Unit

J

JAVA™ . . . . . From Sun Microsystems, Inc.—a robust and versatile programming language that enables developers to:

- Write software on one platform and run it on another.
- Create programs to run within a web browser.
- Develop server-side applications for online forums, stores, polls, processing HTML forms, and more.
- Write applications for cell phones, two-way pagers, and other consumer devices.

JTAG . . . . . Joint Test Action Group

K

Kbps . . . . . thousand (K) bits per second

Kb, Kbit . . . . . Kilobit (written with lowercase b; 1024 Bytes)

KB, KByte . . . . . KiloByte (written with uppercase B; 1024 bits)

L

LAN . . . . . Local Area Network—A computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a Wide-Area Network (WAN).

LANs are capable of transmitting data at fast rates, much faster than data can be transmitted over a telephone line. However, distances are limited. There is also a limit on the number of computers that can be attached to a single LAN.

Latency . . . . . The time an operation requires. For example:

- execution latency is the number of processor clocks an instruction takes to execute.
- memory latency is the number of bus clocks needed to perform a memory operation.

ld . . . . . load

LIFO . . . . . Last-In-First-Out (buffer)

Little-Endian (LE) . . . . . A byte-ordering method in memory where the address *n* of a word corresponds to the *Least-Significant Byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *Most-Significant Byte*. *See also* Big-Endian.

. . . . . In Little-Endian architectures, the rightmost bytes (those with a higher address) are most significant. For example, consider the number 1025 stored in a 4Byte integer as shown in the table below.

00000000 00000000 00000100 00000001		
Addr	**Big-Endian**	**Little-Endian**
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

LP . . . . . LocalPlus

LR . . . . . Link Register

LRU . . . . . Least Recently Used

lsb . . . . . **least significant bit**—the *bit* of least value in an address, register, data element, or instruction encoding.

LSB . . . . . **Least Significant Byte**—the *Byte* of least value in an address, register, data element, or instruction encoding.

LSU . . . . . Load/Store Unit

M

MA . . . . . Memory Address

MAC . . . . . Media Access Control

MAC/PHY . . . . . Multiply-and-ACcumulate/Physical Layer Device
Master . . . . . Name given to a bus device granted control, or mastership, of the bus.
MBAR. . . . . Module Base Address Register
Mb, Mbit . . . . . Megabit (written with lowercase b; 1024 Kilobits)
MB, MByte . . . . . MegaByte (written with uppercase B; 1024 KiloBytes)
Mbps . . . . . Million bits per second
MBS . . . . . Maximum Burst Size
MC . . . . . Memory Controller
MEMCTL . . . . . SDRAM Controller
Memory access ordering . . . . The specific order in which the processor performs load and store memory access and the order in which those accesses complete.
Memory Controller . . . . . A unit whose primary function is to control the external bus memories and I/O devices.
Memory coherency . . . . . An aspect of caching in which it is ensured an accurate view of memory is provided to all devices sharing system memory.
Memory consistency . . . . . Refers to agreement of levels of memory with respect to a single processor and system memory. For example, on-chip cache, secondary cache, and system memory.
Microarchitecture . . . . . Hardware details of a microprocessor's design. Such details are not defined by the PowerPC architecture.
MII . . . . . Media-Independent Interface
mips. . . . . million instructions per second
MIR. . . . . Medium Infrared. *See also* FIR and SIR.
MMAP . . . . . Memory Map
MMU . . . . . **Memory Management Unit**—a functional unit capable of translating an *effective* (logical) *address* to a physical address, providing protection mechanisms, and defining caching methods.
Mnemonic . . . . . The abbreviated name of an instruction used for coding.
mod . . . . . mode
Modified state . . . . . When a cache block is in the modified state, it has been modified by the processor since it was copied from memory. *See also* MESI.
MPC603e . . . . . a microprocessor—a low-power implementation of the PowerPC Reduced Instruction Set Computer (RISC) architecture. The MPC603e microprocessor offers workstation-level performance packed into a low-power, low-cost design ideal for desktop computers, notebooks and battery-powered systems, as well as printer and imaging equipment, telecommunications systems, networking and communications infrastructure, industrial controls, and home entertainment and educational devices.
Munging . . . . . A modification performed on an *effective address* that allows it to appear to the processor that individual aligned scalars are stored as Little-Endian values, when in fact it is stored in Big-Endian order, but at different byte addresses within double words.

Note: Munging affects only the effective address and not the byte order. The PowerPC architecture does not use this term.

MS. . . . . Motorola Scalable
msb . . . . . **most significant bit**—highest-order *bit* in an address, register, data element, or instruction encoding.
MSB . . . . . **Most Significant Byte**—highest-order *Byte* in an address, register, data element, or instruction encoding.
MSCAN . . . . . Motorola Scalable CAN (Controller Area Network)
MSR . . . . . Main Shift Register, or Machine State Register

N

NaN. . . . . Not a Number
NCITS. . . . . Number of Cells In Time Slot
NIC . . . . . Network Interface Card
NMI. . . . . Non-Maskable Interrupt
NMSI . . . . . Non-Multiplexed Serial Interface
No-op . . . . . No-operation—a single-cycle operation that does not affect registers or generate bus activity
NRT. . . . . Non-Real Time

O

OC . . . . . Output Compare

OE . . . . . Output Enable signal

OEA . . . . . **Operating Environment Architecture—**the level of PowerPC architecture that describes memory management model, supervisor-level registers, synchronization requirements, and the exception model. It also defines the time-base feature from a supervisor-level perspective.

OHCI . . . . . Open Host Controller Interface—an "Open Host" standard.

Option, Optional . . . . . A feature, such as an instruction, register, or exception, defined by the PowerPC architecture, but not required to be implemented.

OSI . . . . . Open Systems Interconnection

Out-of-order . . . . . **An** aspect of an operation that lets it be performed ahead of one that may have preceded it in the sequential model. For example, speculative operations. An operation is said to be performed out-of-order if, at the time it is performed, it is not known to be required by the sequential execution model. *See also* In-order.

Out-of-order execution . . . . . A technique that lets instructions be issued and completed in an order that differs from their sequence in the instruction stream.

Overflow . . . . . An error condition that occurs during arithmetic operations when the result cannot be stored accurately in the destination register(s). For example, if two 32-bit numbers are multiplied, the result may not be representable in 32 bits.

P

Pace control . . . . . Controls the data flow rate between a master and slave.

Page . . . . . A region in memory. The OEA defines a page as a 4KByte area of memory, aligned on a 4KByte boundary.

Page fault . . . . . A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory.* On microprocessors that use the PowerPC architecture, a page fault exception condition occurs when a matching, valid *page table entry* (PTE[V]=1) cannot be located.

PCI . . . . . Peripheral Component Interconnect

PCMCIA . . . . . Personal Computer Memory Card International Association

PCR . . . . . Peak Cell Rate

PDU . . . . . Protocol Data Unit

PHY . . . . . Physical Layer Device

Physical memory . . . . . The actual memory that can be accessed through the system memory bus.

PIP . . . . . Parallel Interface Port

Pipelining . . . . . A technique that breaks operations (such as instruction processing or bus transactions) into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

PIT . . . . . Periodic Interrupt Timer

PLL . . . . . Phase-Locked Loop

PM . . . . . Performance Monitors

PMD . . . . . Physical Media-Dependent

POTS . . . . . Plain Old Telephone Service—refers to the standard telephone service that most homes use. The main distinctions between POTS and non-POTS services are speed and bandwidth. POTS is generally restricted to about 52Kbps. The POTS network is also called the public switched telephone network (PSTN).

PPC . . . . . Port Power Control

PPM . . . . . Pulse-Position Modulation

Precise exceptions . . . . . A category of exception for which the pipeline can be stopped so that instructions preceding the faulting instruction can complete. Subsequent instructions can then be flushed and redispatched after exception handling has completed. *See also* Imprecise exceptions.

pri . . . . . priority

Primary opcode . . . . . The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction. S*ee also* Secondary opcode.

Protection boundary . . . . . A boundary between *protection domains*.

Protection domain . . . . . a segment, virtual page, BAT area, or range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

PSC . . . . . Programmable Serial Controller

PTE . . . . . Page Table Entry
PTI. . . . . Payload Type Identifier
PTP . . . . . Port-To-Port switching
PTR . . . . . Program Trace
PVR. . . . . Processor Version Register
PWM. . . . . Pulse Width Modulator

Q

QNX . . . . . From QNX Software Systems—a hybrid realtime platform that represents a cross between a realtime operating system and a platform OS. The first integrated, self-hosted, graphical platform for embedded developers.
Quad word. . . . . A group of 16 contiguous locations starting at an address divisible by 16.

R

rA . . . . . The rA instruction field specifies a GPR used as a source or destination.
rB . . . . . The rB instruction field specifies a GPR used as a source.
rD . . . . . The rD instruction field specifies a GPR used as a destination.
rS. . . . . The rS instruction field specifies a GPR used as a source.
RCT. . . . . Receive Connection Table
RD. . . . . Read
Real address mode . . . . . An MMU mode when no address translation is done and the *effective address* specified is the same as the physical address. The processor's MMU is operating in real address mode if its ability to perform address translation has been disabled through the MSR registers IR and/or DR bits.
Record bit . . . . . Bit 31 (or the Rc bit) in the instruction encoding. When set, it updates the condition register (CR) to reflect the result of the operation.
Registers . . . . . See Section XXX
Register indirect addressing. . A form of addressing that specifies one GPR that contains the address for the load or store.
Register indirect with . . . . . A form of addressing that specifies an immediate value to
immediate index addressing. . be added to the contents of a specified GPR to form the target address for the load or store.
Register indirect with . . . . . A form of addressing that specifies that the contents of two
index addressing . . . . . GPRs be added together to yield the target address for the load or store.
Reservation . . . . . The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.
Reserved field . . . . . In a register, a reserved field is one not assigned a function. A reserved field may be a single bit. The handling of reserved bits is *implementation-dependent*. Software is allowed to write any value to such a bit. A subsequent reading of the bit returns 0 if the value last written to the bit was 0; otherwise, it returns an undefined value (0 or 1).
RISC . . . . . **Reduced Instruction Set Computing**—an *architecture* characterized by fixed-length instructions with non-overlapping functionality and a separate set of load and store instructions that perform memory access.
RM . . . . . Resource Management
rst . . . . . reset
RSV. . . . . Reservation
RT . . . . . Real Time
RTC. . . . . Real-Time Clock
RTOS. . . . . Real-Time Operating System
R/W . . . . . Read/Write
rwc . . . . . read-write-clear
RWITM. . . . . Read With Intent To Modify
Rx, RX . . . . . Receive

S

SAR. . . . . Segment And Reassemble

Scalability . . . . . The capability of an architecture to generate *implementations* specific for a wide range of purposes, and in particular implementations of significantly greater performance and/or functionality than at present, while maintaining compatibility with current implementations.

Scan chain . . . . . The peripheral buffers of a device, linked in JTAG test mode, that are addressed in a shift-register fashion.

SCC . . . . . Serial Communication Controller

SCP . . . . . Serial Control Port

SCR . . . . . Sustained Cell Rate

SDLC . . . . . Synchronous Data Link Control

SDRAM . . . . . Synchronous Dynamic RAM—a faster version of DRAM. SDRAM is generally synchronized with the clock speed for which the microprocessor is optimized. This tends to increase the number of instructions the processor can perform in a given time. The speed of SDRAM is rated in MHz rather than in nanoseconds (ns). This makes it easier to compare the bus speed and the RAM chip speed. You can convert the RAM clock speed to nanoseconds by dividing the chip speed into 1 billion ns (which is one second). For example, an 83 MHz RAM would be equivalent to 12 ns.

sel . . . . . select

Set (*v*) . . . . . To write a non-zero value to a bit or bit field; the opposite of *clear*. The term "set" may also be used to generally describe the updating of a bit or bit field.

Set (*n*) . . . . . A subdivision of a *cache*. Cacheable data can be stored in a given location in any one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was least recently used (LRU). *See also* Set-associative.

Set-associative . . . . . Aspect of cache organization in which cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

Signals . . . . . See Section XXX

Significand . . . . . The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.

SI . . . . . Serial Interface

SIM . . . . . System Integration Module

SIMM . . . . . Signed IMMediate Value, or Single In-line Memory Module

SIP . . . . . Serial Infrared Interaction Pulse

SIR . . . . . Slow Infrared. *See also* FIR and MIR.

SIU . . . . . Systems Interface Unit

Slave . . . . . A device that responds to the master's address. A slave receives data on a write cycle and gives data to the master on a read cycle.

SLT . . . . . Second-Level Tables. *See also* FLT.

SLTMR . . . . . Slice Timer

SMC . . . . . Serial Management Controllers

SNA . . . . . Systems Network Architecture

SPI . . . . . Serial Peripheral Interface—the SPI channel supports the out-of-band control channel to external physical chips. The SPI module allows full-duplex, synchronous, serial communication between the MPC5200 and peripheral devices. It supports master and slave mode, double-buffered operation and can operate in a polling or interrupt driven environment.

SPR . . . . . Special-Purpose Register

SR . . . . . Segment Register

SRAM . . . . . Static Random Access Memory—a type of memory that is faster and more reliable than the more common DRAM (Dynamic RAM). The term "static" is derived from the fact that it does not need to be refreshed like DRAM.

SRR0 . . . . . machine Status save/Restore Register 0

SRR1 . . . . . machine Status save/Restore Register 1

SRTS . . . . . Synchronous Residual Time Stamp

SRU . . . . . System Register Unit

sta . . . . . status

Static branch prediction . . . . Mechanism by which software (for example, compilers) can give a hint to the machine hardware about the direction a branch is likely to take.

STB . . . . . Set-Top Box

Sticky bit . . . . . A bit that when *set* must be cleared explicitly.

stp . . . . . stop
str . . . . . start
STS . . . . . Special Transfer Start
Superscalar machine . . . . . A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.
Supervisor mode . . . . . The privileged operation state of a processor. In supervisor mode, software (typically the operating system) can access all control registers and the supervisor memory space, among other privileged operations.
SWT . . . . . Software Watchdog Timer
Synchronization . . . . . A process to ensure operations occur *in order*. *See also* Context synchronization and Execution synchronization.
Synchronous exception . . . . . An *exception* generated by the execution of a particular instruction or instruction sequence. There are two types of synchronous exceptions, *precise* and *imprecise*.
System memory . . . . . The physical memory available to a processor.

T

TA . . . . . Transfer Acknowledge
TAP . . . . . Test Access Port
TB . . . . . Time Base (register)
TC . . . . . Transmission Convergence
TCT . . . . . Transmit Connection Table
TDM . . . . . Time-Division Multiplex—a single serial channel used by several channels taking turns.
TE . . . . . Terminal Endpoint
TEA . . . . . Transfer Error Acknowledge
Throughput . . . . . A measure of the number of instructions processed per clock cycle.
TLB . . . . . **Translation Lookaside Buffer—**A cache that holds recently-used *page table entries*.
TLE . . . . . True Little-Endian
TMR, tmr . . . . . Timer
TO, to . . . . . Timeout
TS . . . . . Transfer Start
TSA . . . . . Time-Slot Assigner
tst . . . . . test
TSIZ . . . . . Transfer Size
Tx, TX . . . . . Transmit

U

UART . . . . . Universal Asynchronous Receiver-Transmitter—a component that handles asynchronous serial communication.
UARTe . . . . . UART enhanced (simple UART with carrier detect input)
UBR . . . . . Unspecified Bit-Rate. *See also* CBR and ABR.
UBR+ . . . . . Unspecified Bit-Rat**e with minimum cell rate guarantee**
UIMM . . . . . Unsigned IMMediate value
UISA . . . . . **User Instruction Set Architecture—**the level of the architecture to which user-level software should conform. The UISA defines the base user-level instruction set, user-level registers, data types, floating-point memory conventions and exception model as seen by user programs, and the memory and programming models.
UPM . . . . . User-Programmable Machine
USART . . . . . Universal Synchronous/Asynchronous Rx/Tx
USB . . . . . Universal Serial Bus—a new external bus standard that supports data transfer rates of 12 Mbps.
User mode . . . . . The unprivileged operating state of a processor used typically by application software. In user mode, software can only access certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.
UTOPIA . . . . . Universal Test and Operations Physical Interface for ATM

V

VA . . . . . Virtual Address—an intermediate address used in translation of an *effective address* to a physical address.

VBR . . . . . Variable Bit-Rate

VC . . . . . Virtual Channel, Circuit, Call, or Connection

VCC . . . . . Virtual Channel Connection

VCI . . . . . Virtual Circuit Identifier

VCO . . . . . Voltage-Controlled Oscillator

VEA . . . . . **Virtual Environment Architecture—**the level of the *architecture* that describes the memory model for an environment in which multiple devices can access memory. VEA can:

- define aspects of the cache model
- define cache control instructions
- define the time-base facility from a user perspective.

ver . . . . . version

VM . . . . . Virtual Memory—the address space created using the memory management facilities of the processor. Program access to virtual memory is possible only when it coincides with *physical memory.*

VP . . . . . Virtual Path

VPC . . . . . Virtual Path Connection

VPI . . . . . Virtual Path Identifier

W

WAN . . . . . Wide Area Network—A computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LANs).

Watchpoint . . . . . A reported event, but does not change machine timing.

WE . . . . . Write Enable signals

WKIO . . . . . GPIO WakeUp

Word . . . . . A 32-bit data element.

Note: Other processors may have a different word size.

WR . . . . . Write

Write-back . . . . . A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly. For example, when a modified cache block is *cast out* to make room for newer data.

Write-through . . . . . A cache memory update policy in which all processor write cycles are written to both cache and memory.

X

XCPCI . . . . . PCI_CFG （PCI configuration）

XER . . . . . Register used primarily for indicating conditions such as carries and overflows for integer operations.

XFC . . . . . External Filter Capacitor

XTAL . . . . . Crystal. *See also* EXTAL.

VxWorks . . . . . From Wind River Systems, is a networked real-time operating system designed to be used in a distributed environment.

Appendix B
List of Registers