Freescale Semiconductor

Order Number: MGT5100UM/D
Revision 0, April 2002

MGT5100 User Manual

G2 Communications Microcontroller

freescale™
semiconductor

This page intentionally left blank.

REVISION HISTORY

Release	Date	Author	Summary of Changes
0		Laurin Ashby Allan Chin	Initial Release of MGT5100 User Manual.

This page intentionally left blank.

TABLE OF CONTENTS

Paragraph Number | **Page Number**

SECTION 1 INTRODUCTION

1.1 Overview 1-1
1.1.1 Features 1-1
1.2 Architecture 1-3
1.2.1 Embedded G2 Core 1-6
1.2.2 SmartComm I/O Subsystem 1-7
1.2.2.1 Programmable Serial Controllers (PSCs) 1-7
1.2.2.2 10/100 Ethernet Controller 1-8
1.2.2.3 Universal Serial Bus (USB) 1-8
1.2.2.4 Infrared (IR) Support 1-8
1.2.2.5 Inter-Integrated Circuit (I^2C) 1-8
1.2.2.6 Serial Peripheral Interface (SPI) 1-8
1.2.3 Dual Motorola Scalable (MS) Controller Area Network (CAN) 1-8
1.2.4 System Level Interfaces 1-9
1.2.4.1 Chip Selects 1-9
1.2.4.2 Interrupt Controller 1-9
1.2.4.3 Timers 1-9
1.2.4.4 General Purpose Input/Outputs (GPIO) 1-10
1.2.4.5 Functional Pin MUXing 1-10
1.2.4.6 Real-Time Clock (RTC) 1-10
1.2.5 SDRAM Controller and Interface 1-11
1.2.6 Multi-Function External Bus 1-11
1.2.7 Power Management 1-11
1.2.8 Systems Debug and Test 1-12
1.2.9 Physical Characteristics 1-12

SECTION 2 SIGNAL DESCRIPTIONS

2.1 Overview 2-1
2.2 Pinout Tables 2-3
2.2.1 Functional Signal List with Pin/Pad Count 2-9
2.3 Multi-Function Pin Maps 2-21

SECTION 3 MEMORY MAP

3.1 Overview 3-1
3.2 Internal Register Memory Map 3-2
3.3 MGT5100 Memory Map 3-3
3.3.1 Memory Map Registers—MBAR + 0x0000 3-4
3.3.1.1 Module Base Address (0x0000)—MBAR 3-4
3.3.1.2 CS0 Start Address (0004–002C)—CS0STR 3-5
3.3.1.3 CS0 Stop Address (0008)—CS0STP 3-5
3.3.1.4 CS1 Start Address (000C)—CS1STR 3-5
3.3.1.5 CS1 Stop Address (0010)—CS1STP 3-6
3.3.1.6 CS2 Start Address (0014)—CS2STR 3-6
3.3.1.7 CS2 Stop Address (0018)—CS2STP 3-7
3.3.1.8 CS3 Start Address (001C)—CS3STR 3-7

Paragraph Number | **Page Number**

3.3.1.9 CS3 Stop Address (0020)—CS3STP . . . 3-7
3.3.1.10 CS4 Start Address (0024)—CS4STR . . . 3-8
3.3.1.11 CS4 Stop Address (0028)—CS4STP . . . 3-8
3.3.1.12 CS5 Start Address (002C)—CS5STR . . . 3-9
3.3.1.13 CS5 Stop Address (0030)—CS5STP . . . 3-9
3.3.1.14 SDRAM Start Address (0034)—SDRAMSTR . . . 3-9
3.3.1.15 SDRAM Stop Address (0038)—SDRAMSTP . . . 3-10
3.3.1.16 PCI1 Start Address (003C)—PCI1STR . . . 3-10
3.3.1.17 PCI1 Stop Address (0040)—PCI1STP . . . 3-11
3.3.1.18 PCI2 Start Address (0044)—PCI2STR . . . 3-11
3.3.1.19 PCI2 Stop Address (0048)—PCI2STP . . . 3-11
3.3.1.20 Boot Start Address (004C)—BOOTSTR . . . 3-12
3.3.1.21 Boot Stop Address (0050)—BOOTSTP. . . . 3-12
3.3.1.22 Address Space Enable (0054)—ADREN (Control Reg in RTL). . . . 3-13
3.4 Register Summaries . . . 3-14
3.4.1 PCI XLB Configuration Registers . . . 3-14
3.4.2 SmartComm DMA Registers . . . 3-16
3.4.3 PSC Registers—Quick Reference. . . . 3-19
3.4.4 IrDA Registers—Quick Reference. . . . 3-23
3.4.5 SPI Registers—Quick Reference . . . 3-26
3.5 SDRAM Memory Controller Register Map (MBAR + 0x0100) . . . 3-27
3.6 Slice Timers Register Map (MBAR + 0x0700) . . . 3-27

SECTION 4
RESETS AND RESET CONFIGURATION

4.1 Overview. . . . 4-1
4.2 Hard and Soft Reset Pins . . . 4-1
4.2.1 Power-ON Reset—PORESET. . . . 4-1
4.2.2 Hard Reset—HRESET . . . 4-2
4.2.3 Soft Reset—SRESET . . . 4-2
4.3 Reset Sequence . . . 4-3
4.4 Reset Operation . . . 4-4
4.5 Other Resets. . . . 4-5
4.6 Reset Configuration . . . 4-6

SECTION 5
CLOCKS AND POWER MANAGEMENT

5.1 Overview. . . . 5-1
5.2 Clock Distribution Module (CDM). . . . 5-1
5.3 MGT5100 Clock Domains. . . . 5-2
5.3.1 G2 Clock Domain . . . 5-4
5.3.2 Processor Bus or XL Bus Clock Domain . . . 5-6
5.3.3 SDRAM Memory Controller Clock Domain . . . 5-6
5.3.4 IP Bus Clock Domain. . . . 5-7
5.3.5 PCI Clock Domain . . . 5-7
5.4 Clock Relationships . . . 5-8
5.5 Power Management . . . 5-9
5.5.1 Full-Power Mode . . . 5-10
5.5.2 Power Conservation Modes . . . 5-10
5.5.3 G2 Processor Power Modes . . . 5-10
5.5.3.1 Dynamic Power Mode . . . 5-11

Paragraph Number | Page Number

5.5.3.2 Doze Mode 5-11
5.5.3.3 Nap Mode. 5-11
5.5.3.4 Sleep Mode 5-12
5.5.4 Deep-Sleep Mode 5-12
5.5.4.1 Entering Deep Sleep 5-13
5.5.4.2 Exiting Deep Sleep. 5-13
5.6 CDM Registers—MBAR+0x0200 5-14
5.6.1 JTAG ID Number (0200)—JTAGID. 5-14
5.6.2 Power ON Reset Configuration (0204)—PORCFG 5-15
5.6.3 Bread Crumb (0208)—BC. 5-15
5.6.4 Configuration (020C)—CFG 5-16
5.6.5 48MHz Fractional Divider Configuration (0210)—FDCFG 5-17
5.6.6 Clock Enable (0214)—CLKEN 5-18
5.6.7 System Oscillator Configuration (0218)—OSCCFG 5-20
5.6.8 Clock Control Sequencer Configuration (021C)—CCSCFG. 5-20
5.6.9 Soft Reset (0220)—SFTRST 5-21
5.6.10 System PLL Status (0224)—PLLSTA (33-27MHz). 5-21

SECTION 6 G2 PROCESSOR CORE

6.1 Overview. 6-1
6.2 MGT5100 G2 Processor Core 6-1
6.3 Arbiter Registers—MBAR+0x0080. 6-2
6.3.1 Configuration Register (00C0)—ACFG. 6-3
6.3.2 Version Register (00C4)—VER. 6-4
6.3.3 Status Register (00C8)—STA. 6-4
6.3.4 Interrupt Enable Register (00CC)—INTEN 6-5
6.3.5 Address Capture Register (00D0)—ADRCAP 6-6
6.3.6 Bus Signal Capture Register (00D4)—SIGCAP 6-6
6.3.7 Address Tenure Time-Out Register (00D8)—ADRTO 6-7
6.3.8 Data Tenure Time-Out Register (00DC)—DATTO 6-8
6.3.9 Bus Activity Time-Out Register (00E0)—BUSTO 6-8
6.3.10 Master Priority Enable Register (00E4)—PRIEN 6-9
6.3.11 Master Priority Register (00E8)—PRI 6-10
6.3.12 Base Address Register (00EC)—BA. 6-11
6.3.13 Reserved Registers (0080–00BF, 00EC, 00F0–00FC). 6-12

SECTION 7 SYSTEM INTERFACE UNIT (SIU)

7.1 Overview. 7-1
7.2 Interrupt Controller 7-1
7.2.1 Block Description. 7-1
7.2.1.1 Machine Check Pin—core_mcp 7-2
7.2.1.2 System Management Interrupt—core_smi 7-2
7.2.1.3 Standard Interrupt—core_int 7-3
7.2.2 IRQ[0:3] Interrupt Requests 7-4
7.2.3 Interface Description 7-5
7.2.4 Interrupt Controller Registers—MBAR + 0x0500. 7-6
7.2.4.1 Peripheral Interrupt Mask (0500) 7-6
7.2.4.2 Peripheral Priority and HI/LO Select 1 (0504). 7-7
7.2.4.3 Peripheral Priority and HI/LO Select 2 (0508). 7-8

Paragraph Number		Page Number
7.2.4.4	Peripheral Priority and HI/LO Select 3 (050C)	7-9
7.2.4.5	External Enable and External Types (0510)	7-9
7.2.4.6	Critical Priority and Main Interrupt Mask (0514)	7-11
7.2.4.7	Main Interrupt Priority and INT/SMI Select 1 (0518)	7-12
7.2.4.8	Main Interrupt Priority and INT/SMI Select 2 (051C)	7-13
7.2.4.9	PerStat, MainStat, CritStat Encoded (0524)	7-14
7.2.4.10	Critical Interrupt Status All (0528)	7-16
7.2.4.11	Main Interrupt Status All (052C)	7-16
7.2.4.12	Peripheral Interrupt Status All (0530)	7-17
7.2.4.13	Peripheral Interrupt Status All (0538)	7-18
7.3	General Purpose I/O (GPIO)	7-19
7.3.1	GPIO Pin Multiplexing	7-22
7.3.1.1	PSC1 (UART1/AC97/CODEC1)	7-23
7.3.1.2	PSC2 (UART2/AC97/CODEC2)	7-23
7.3.1.3	PSC3 (USB2/CODEC3/SPI/UART3)	7-23
7.3.1.4	USB1	7-23
7.3.1.5	Ethernet/USB2/RST_CONFIG	7-24
7.3.1.6	IRDA	7-24
7.3.1.7	I^{2}C	7-25
7.3.1.8	GPIO Timer Pins	7-25
7.3.1.9	Dedicated GPIO Port	7-26
7.3.2	GPIO Programmer's Model	7-26
7.3.2.1	GPIO Standard Registers—MBAR+0x0B00	7-26
7.3.2.1.1	Port Configuration Register (MBAR+0B00)—GPIOPCR	7-28
7.3.2.1.2	Simple GPIO Enables (0B04)—GPIOSEN	7-30
7.3.2.1.3	Simple GPIO Open Drain Type (0B08)—GPIOSOD	7-31
7.3.2.1.4	Simple GPIO Data Direction (0B0C)—GPIOSDD	7-32
7.3.2.1.5	Simple GPIO Data Output Values (0B10)—GPIOSDO	7-34
7.3.2.1.6	Simple GPIO Data Input Values (0B14)—GPIOSDI	7-35
7.3.2.1.7	Output-Only Enables (0B18)—GPIOOE	7-36
7.3.2.1.8	Output-Only Data Value Out (0B1C)—GPIOODO	7-37
7.3.2.1.9	Simple Interrupt Enables (0B20)—GPIOSIE	7-38
7.3.2.1.10	Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD	7-38
7.3.2.1.11	Simple Interrupt Data Direction (0B28)—GPIOSIDD	7-39
7.3.2.1.12	Simple Interrupt Data Value Out (0B2C)—GPIOSIDO	7-40
7.3.2.1.13	Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE	7-40
7.3.2.1.14	Simple Interrupt Interrupt Types (0B34)—GPIOSIIT	7-41
7.3.2.1.15	Simple Interrupt Master Enable (0B38)—GPIOSIME	7-42
7.3.2.1.16	Simple Interrupt Status (0B3C)—GPIOSIST	7-42
7.3.2.2	WakeUp GPIO Registers—MBAR+0x0C00	7-44
7.3.2.2.1	WakeUp GPIO Enables (0C00)—GPIOWE	7-44
7.3.2.2.2	WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD	7-45
7.3.2.2.3	WakeUp GPIO Data Direction (0C08)—GPIOWDD	7-46
7.3.2.2.4	WakeUp GPIO Data Value Out (0C0C)—GPIOWDO	7-46
7.3.2.2.5	WakeUp GPIO Interrupt Enable (0C10)—GPIOWUE	7-47
7.3.2.2.6	WakeUp GPIO Simple Interrupt Enable (0C14)—GPIOWSIE	7-48
7.3.2.2.7	WakeUp GPIO Interrupt Types (0C18)—GPIOWT	7-49
7.3.2.2.8	WakeUp GPIO Master Enables (0C1C)—GPIOWME	7-50
7.3.2.2.9	WakeUp GPIO Data Input Values (0C20)—GPIOWI	7-50
7.3.2.2.10	WakeUp GPIO Status Register (0C24)—GPIOWS	7-51
7.4	General Purpose Timers (GPT)	7-52
7.4.1	Timer Configuration Method	7-52
7.4.2	Mode Overview	7-53

7.4.3 Programming Notes . . . 7-53
7.4.4 GPT Registers—MBAR + 0x0600 . . . 7-54
7.4.4.1 GPT[0–7] Enable and Mode Select (0600–0670) . . . 7-54
7.4.4.2 GPT[0–7] Counter Input (0604–0674) . . . 7-57
7.4.4.3 GPT[0–7] PWM Configuration (0608–0678) . . . 7-58
7.4.4.4 GPT[0–7] Status (060C–067C) . . . 7-59
7.5 Slice Timers . . . 7-60
7.5.1 SLT Registers—MBAR + 0x0700 . . . 7-61
7.5.1.1 Terminal Count Register — TCNT0, 1 (0700, 0710) . . . 7-61
7.5.1.2 Control Register — CTL0, 1 (0704, 0714) . . . 7-62
7.5.1.3 Count Value Register — TMRCNT0, 1 (0708, 0718) . . . 7-62
7.5.1.4 Timer Status Register — TMRSTA0, 1 (070C, 071C) . . . 7-63
7.6 Real-Time Clock . . . 7-63
7.6.1 Real-Time Clock Signals . . . 7-64
7.6.2 Programming Note . . . 7-65
7.6.3 RTC Interface Registers—MBAR + 0x0800 . . . 7-65
7.6.3.1 Time Set (0800) . . . 7-66
7.6.3.2 Date Set (0804) . . . 7-67
7.6.3.3 New Year and Stopwatch (0808) . . . 7-68
7.6.3.4 Alarm and Interrupt Enable (080C) . . . 7-68
7.6.3.5 Current Time (0810) . . . 7-69
7.6.3.6 Current Date (0814) . . . 7-70
7.6.3.7 Alarm and Stopwatch Interrupt (0818) . . . 7-71
7.6.3.8 Periodic Interrupt and Bus Error (081C) . . . 7-72
7.6.3.9 Test Register/Divides . . . 7-72

SECTION 8
SDRAM MEMORY CONTROLLER

8.1 Overview . . . 8-1
8.2 Features . . . 8-2
8.3 Functional Description . . . 8-2
8.3.1 SDRAM Configurations . . . 8-2
8.3.2 Timing Considerations . . . 8-6
8.3.3 External Signals (SDRAM Side) . . . 8-6
8.4 Operation . . . 8-7
8.4.1 Block Diagram . . . 8-7
8.4.2 Address Multiplexing and Address Pipeline Blocks . . . 8-8
8.4.3 External Chip Select . . . 8-9
8.4.4 Power-Up Initialization . . . 8-10
8.4.5 Page Management . . . 8-11
8.4.6 Transfer Size . . . 8-12
8.4.7 Commands . . . 8-13
8.4.7.1 Mode and Extended Mode Register Set Command . . . 8-13
8.4.7.2 Precharge Select Bank and Precharge all Banks Commands . . . 8-14
8.4.7.3 Bank Active Command . . . 8-15
8.4.7.4 Read Command . . . 8-15
8.4.7.5 Write Command . . . 8-16
8.4.7.6 Burst Stop Command . . . 8-17
8.4.7.7 Auto-Refresh and Self-Refresh Commands . . . 8-17
8.5 Programming the SDRAM Memory Controller . . . 8-18
8.6 SDRAM Memory Controller Registers (MBAR + 0x0100) . . . 8-19
8.6.1 Mode Register (MOD) . . . 8-19

Paragraph Number		Page Number
8.6.2	Control Register (CTR)	8-21
8.6.3	Configuration Register 1 (CFG1)	8-23
8.6.4	Configuration Register 2	8-25
8.6.5	Address Select Register (ADRSEL)	8-26
8.7	Examples	8-27
8.7.1	Example—Physical Address Multiplexing	8-27
8.7.2	Example—Address Bus Mapping	8-28
8.7.3	Example—Setting Registers for Standard SDRAM	8-28
8.7.4	Example—Setting Registers for DDR SDRAM	8-31

SECTION 9
CS/LP BOOT ROM/SRAM CONTROLLER

9.1	Local Bus Overview	9-1
9.2	LocalPlus Bus (LP bus)	9-2
9.2.1	LocalPlus Features	9-2
9.2.2	LocalPlus Operation	9-3
9.2.2.1	Non-MUXed Mode	9-3
9.2.2.2	MUXed Mode	9-4
9.2.2.3	Boot Configuration	9-4
9.2.3	Chip Select Configuration Address Spaces	9-5
9.2.4	Reset Configuration	9-6
9.3	LocalPlus Controller (LPC) Design	9-7
9.3.1	Standard non-MUXed Mode	9-7
9.3.2	ALE Driven MUXed Mode	9-8
9.3.3	Signals	9-8
9.3.4	Interface Description	9-9
9.3.5	Physical Peripheral Connections	9-10
9.3.5.1	During Address Tenure	9-10
9.3.5.2	During Data Tenure	9-11
9.4	LPC Programmer’s Model	9-12
9.4.1	Interrupt and Bus Errors	9-13
9.4.2	Chip Select/LocalPlus Bus Registers—MBAR + 0x0300	9-13
9.4.2.1	CS Boot ROM, (0300)	9-13
9.4.2.2	CS Configuration (0304–0314)	9-16
9.4.2.3	CS Control (0318)	9-18
9.4.2.4	CS Status (031C)	9-19

Section 10
PCI Controller

10.1	Overview	10-1
10.2	PCI External Signals	10-2
10.3	PCI Interface	10-2
10.3.1	PCI XLB Initiator Interface	10-4
10.3.2	PCI XLB Target Interface	10-5
10.3.3	PCI XLB Configuration Interface	10-6
10.3.3.1	PCI XLB Configuration Registers—MBAR + 0x0D00	10-6
10.3.3.1.1	PCI Header: Device ID/Vendor ID (0D00)	10-6
10.3.3.1.2	PCI Header: Status/Command (0D04)	10-7
10.3.3.1.3	PCI Header: Class Code/Revision (0D08)	10-9
10.3.3.1.4	PCI Header: BIST/Type/Latency/Cache (0D0D)	10-10
10.3.3.1.5	PCI Header: BAR0 (0D10)	10-11

Paragraph Number | Page Number

10.3.3.1.6 PCI Header: BAR1 (0D14) 10-11
10.3.3.1.7 PCI Header: Reserved (0D18–0D27) 10-12
10.3.3.1.8 PCI Header: CardBus CIS Pointer (0D28) 10-12
10.3.3.1.9 PCI Header: Subsystem Vendor ID (0D2C) 10-12
10.3.3.1.10 PCI Header: Unused (0D30) 10-12
10.3.3.1.11 PCI Header: Unused (0D34–0D3B) 10-12
10.3.3.1.12 PCI Header: Max Lat, Min Grant, Interrupt (0D3C) 10-13
10.3.3.1.13 PCI Header: Unused (0D40–0D5F) 10-13
10.3.3.2 MGT5100 Application Interface Registers—MBAR + 0x0D00 10-13
10.3.3.2.1 PCI Interrupt Enable (0D60) 10-14
10.3.3.2.2 PCI Status (0D64) 10-14
10.3.3.2.3 PCI Control (0D68) 10-15
10.3.3.2.4 PCI Mask/Value Read (0D6C) 10-16
10.3.3.2.5 PCI Mask/Value Write (0D70) 10-17
10.3.3.2.6 PCI Subwindow 1 (0D74) 10-18
10.3.3.2.7 PCI Subwindow 2 (0D78) 10-18
10.3.3.2.8 PCI Window Command/Control (0D7C) 10-19
10.3.4 PCI SmartDMA Initiator Interface 10-20
10.3.4.1 PCI SmartDMA Transmit (Tx) Initiator Interface 10-20
10.3.4.2 PCI Transmit (Tx) Registers—MBAR + 0x3800 10-21
10.3.4.2.1 PCI Tx Packet Size (3800) 10-21
10.3.4.2.2 PCI Tx Start Address (3804) 10-22
10.3.4.2.3 PCI Tx Transaction Control (3808) 10-22
10.3.4.2.4 PCI Tx Enables (380C) 10-23
10.3.4.2.5 PCI Tx Next Address (3810) 10-25
10.3.4.2.6 PCI Tx Last Word (3814) 10-25
10.3.4.2.7 PCI Tx Done Counts (3818) 10-26
10.3.4.2.8 PCI Tx Status Bits (381C) 10-26
10.3.4.2.9 PCI Tx FIFO Data (3840) 10-28
10.3.4.2.10 PCI Tx FIFO Status (3844) 10-28
10.3.4.2.11 PCI Tx FIFO Control (3848) 10-29
10.3.4.2.12 PCI Tx Alarm (384E) 10-30
10.3.4.2.13 PCI Tx Read Pointer (3852) 10-30
10.3.4.2.14 PCI Tx Write Pointer (3856) 10-31
10.3.4.3 PCI SmartDMA Receive (Rx) Initiator Interface 10-31
10.3.4.4 PCI Receive (Rx) Registers—MBAR + 0x3880 10-32
10.3.4.4.1 PCI Rx Packet Size (3880) 10-32
10.3.4.4.2 PCI Rx Start Address (3884) 10-33
10.3.4.4.3 PCI Rx Transaction Command (3888) 10-33
10.3.4.4.4 PCI Rx Enables (388C) 10-34
10.3.4.4.5 PCI Rx Next Address (3890) 10-35
10.3.4.4.6 PCI Rx Done Counts (3898) 10-36
10.3.4.4.7 PCI Rx Status Bits (389C) 10-36
10.3.4.4.8 PCI Rx FIFO Data (38C0) 10-38
10.3.4.4.9 PCI Rx FIFO Status (38C4) 10-38
10.3.4.4.10 PCI Rx FIFO Control (38C8) 10-39
10.3.4.4.11 PCI Rx Alarm (38CC) 10-40
10.3.4.4.12 PCI Rx Read Pointer (38D0) 10-40
10.3.4.4.13 PCI Rx Write Pointer (38D4) 10-41

SECTION 11
ATA CONTROLLER

11.1 Overview 11-1
11.2 SmartComm Key Features 11-2
11.2.1 SmartComm Read 11-2
11.2.2 SmartComm Write 11-2
11.3 ATA Register Interface 11-2
11.3.1 ATA Host Registers—MBAR + 0x3A00 11-3
11.3.1.1 Host Configuration (3A00)—HCFG 11-3
11.3.1.2 Host Status (3A04)—HSR 11-4
11.3.1.3 PIO Timing 1 (3A08)—PIO1 11-4
11.3.1.4 PIO Timing 2 (3A0C)—PIO2 11-5
11.3.1.5 Multiword DMA Timing 1 (3A10)—DMA1 11-5
11.3.1.6 Multiword DMA Timing 2 (3A14)—DMA2 11-6
11.3.1.7 Ultra DMA Timing 1 (3A18)—UDMA1 11-6
11.3.1.8 Ultra DMA Timing 2 (3A1C)—UDMA2 11-7
11.3.1.9 Ultra DMA Timing 3 (3A20)—UDMA3 11-8
11.3.1.10 Ultra DMA Timing 4 (3A24)—UDMA4 11-8
11.3.1.11 Ultra DMA Timing 5 (3A28)—UDMA5 11-9
11.3.2 ATA FIFO Registers—MBAR + 0x3A00 11-9
11.3.2.1 Rx/Tx FIFO Data Word (3A3C)—RTFDWR 11-10
11.3.2.2 Rx/Tx FIFO Status (3A40)—RTFSR 11-10
11.3.2.3 Rx/Tx FIFO Control (3A44)—RTFCR 11-11
11.3.2.4 Rx/Tx Alarm (3A48)—RTFAR 11-12
11.3.2.5 Rx/Tx FIFO Read Pointer (3A4C)—RTFRPR 11-12
11.3.2.6 Rx/Tx FIFO Write Pointer (3A50)—RTFWPR 11-13
11.3.3 ATA Drive Registers—MBAR + 0x3A00 11-13
11.3.3.1 Device Control (3A5C)—DCTR 11-14
11.3.3.2 Drive Alternate Status (3A5C)—DASR 11-14
11.3.3.3 Drive Data (3A60)—DDR 11-15
11.3.3.4 Drive Features (3A64)—DFR 11-15
11.3.3.5 Drive Error (3A64)—DER 11-16
11.3.3.6 Drive Sector Count (3A68)—DSCR 11-16
11.3.3.7 Drive Sector Number (3A6C)—DSNR 11-17
11.3.3.8 Drive Cylinder Low (3A70)—DCLR 11-17
11.3.3.9 Drive Cylinder High (3A74)—DCHR 11-18
11.3.3.10 Drive Device/Head (3A78)—DDHR 11-18
11.3.3.11 Drive Command (3A7C)—DCR 11-19
11.3.3.12 Device Status (3A80)—DSR 11-20
11.4 ATA Host Controller Operation 11-21
11.4.1 PIO State Machine 11-22
11.4.2 DMA State Machine 11-24
11.4.2.1 Software Requirements 11-24
11.5 Signals and Connections 11-24
11.6 ATA Interface Description 11-27
11.7 ATA Bus Background 11-29
11.7.1 Terminology 11-29
11.7.2 ATA Modes 11-30
11.7.3 ATA Addressing 11-30
11.7.3.1 ATA Register Addressing 11-30
11.7.3.2 Drive Interrupt 11-31
11.7.3.3 Sector Addressing 11-31

Paragraph Number | Page Number

11.7.3.4 Physical/Logical Addressing Modes 11-32
11.7.4 ATA Transactions 11-33
11.7.4.1 PIO Mode Transactions 11-33
11.7.4.1.1 Class 1—PIO Read 11-34
11.7.4.1.1 Class 2—PIO Write 11-35
11.7.4.1.1 Class 3—Non-Data Command 11-36
11.7.4.2 DMA Protocol 11-36
11.7.4.3 Multiword DMA Transactions 11-40
11.7.4.3.1 Class 4—DMA Command 11-40
11.7.4.4 Ultra DMA Protocol 11-41
11.8 ATA RESET/Power-Up 11-42
11.8.1 Hardware Reset 11-42
11.8.2 Software Reset 11-42
11.9 ATA I/O Cable Specifications 11-43
11.10 ATA Electrical Characteristics 11-43
11.10.1 ATA Timing Diagrams 11-44

SECTION 12
UNIVERSAL SERIAL BUS (USB)

12.1 Overview 12-1
12.2 Data Transfer Types 12-2
12.3 Host Controller Interface 12-2
12.3.1 Communication Channels 12-2
12.3.2 Data Structures 12-3
12.4 Host Control (HC) Operational Registers 12-6
12.4.1 Control and Status Partition—MBAR + 0x1000 12-7
12.4.1.1 HC Revision (1000)—HcRevision 12-7
12.4.1.2 HC Control (1004)—HcControl 12-7
12.4.1.3 HC Command Status (1008)—HcCommandStatus 12-9
12.4.1.4 HC Interrupt Status (100C)—HcInterruptStatus 12-11
12.4.1.5 HC Interrupt Enable (1010)—HcInterruptEnable 12-12
12.4.1.6 HC Interrupt Disable (1014)—HcInterruptDisable 12-13
12.4.2 Memory Pointer Partition—MBAR + 0x1000 12-14
12.4.2.1 HC Communication (1018)—HcHCCA 12-14
12.4.2.2 HC Period Current ED (101C)—HcPeriodCurrentED 12-15
12.4.2.3 HC Control Head ED (1020)—HcControlHeadED 12-16
12.4.2.4 HC Control Current ED (1024) HcControlCurrentED 12-16
12.4.2.5 HC First Bulk Head List ED (1028)—HcBulkHeadED 12-17
12.4.2.6 HC Bulk List Current ED (102C)—HcBulkCurrentED 12-17
12.4.2.7 HC Last Completed Transfer (1030)—HcDoneHead 12-18
12.4.3 Frame Counter Partition—MBAR + 0x1000 12-19
12.4.3.1 HC Frame Interval (1034)—HcFmInterval 12-19
12.4.3.2 HC Frame Bit-time Remaining (1038)—HcFmRemaining 12-20
12.4.3.3 HC Timing Reference (103C)—HcFmNumber 12-20
12.4.3.4 HC Start Processing Periodic List (1040)—HcPeriodicStart 12-21
12.4.3.5 HC Commit Transfer (1044)—HcLSThreshold 12-22
12.4.4 Root Hub Partition—MBAR + 0x1000 12-22
12.4.4.1 HC Root Hub Descriptor A (1048)—HcRhDescriptorA 12-23
12.4.4.2 HC Root Hub Descriptor B (104C)—HcRhDescriptorB 12-24
12.4.4.3 HC Root Hub Status (1050)—HcRhStatus 12-25
12.4.4.4 HC RH Port Status (1054)—HcRhPortStatus[1:NDP] 12-26

Paragraph Number | Page Number

SECTION 13
SMARTCOMM/SMARTDMA

13.1 Overview . . . 13-1
13.2 SmartComm Functional Description . . . 13-2
13.3 SmartComm DMA Registers—MBAR+0x1200 . . . 13-3
13.3.1 Task Bar (1200)—taskBar . . . 13-4
13.3.2 Current Pointer (1204)—currentPointer . . . 13-4
13.3.3 End Pointer (1208)—endPointer . . . 13-4
13.3.4 Variable Pointer (120C)—variablePointer . . . 13-5
13.3.5 Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl . . . 13-5
13.3.6 Interrupt Pending (1214)—IntPend . . . 13-6
13.3.7 Interrupt Mask (1218)—IntMask . . . 13-6
13.3.8 Task Control 0, 1 (121C)—TCR0, TCR1 . . . 13-7
13.3.9 Task Control 2, 3 (1220)—TCR2, TCR3 . . . 13-7
13.3.10 Task Control 4, 5 (1224)—TCR4, TCR5 . . . 13-8
13.3.11 Task Control 6, 7 (1228)—TCR6, TCR7 . . . 13-8
13.3.12 Task Control 8, 9 (122C)—TCR8, TCR9 . . . 13-8
13.3.13 Task Control A, B (1230)—TCRA, TCRB . . . 13-9
13.3.14 Task Control C, D (1234)—TCRC, TCRD . . . 13-9
13.3.15 Task Control E, F (1238)—TCRE, TCRF . . . 13-10
13.3.16 Initiator Priority 0–3 (123C)—IPR0–3 . . . 13-10
13.3.17 Initiator Priority 4–7 (1240)—IPR4–7 . . . 13-10
13.3.18 Initiator Priority 8–11 (1244)—IPR8–11 . . . 13-11
13.3.19 Initiator Priority 12–15 (1248)—IPR12–15 . . . 13-11
13.3.20 Initiator Priority 16–19 (124C)—IPR16–19 . . . 13-12
13.3.21 Initiator Priority 20–23 (1250)—IPR20–23 . . . 13-12
13.3.22 Initiator Priority 24–27 (1254)—IPR24–27 . . . 13-13
13.3.23 Initiator Priority 28–31 (1258)—IPR28–31 . . . 13-13
13.3.24 Reserved Register 1 (125C)—res1 . . . 13-14
13.3.25 Reserved Register 2 (1260)—res2 . . . 13-14
13.3.26 Reserved Register 3 (1264)—res3 . . . 13-14
13.3.27 Reserved Register 4 (1268)—res4 . . . 13-15
13.3.28 Reserved Register 5 (126C)—res5 . . . 13-15
13.3.29 Debug Module Comparator 1 (1270)—Value1 . . . 13-16
13.3.30 Debug Module Comparator 2 (1274)—Value2 . . . 13-16
13.3.31 Debug Modulator Control (1278)—Control . . . 13-16
13.3.32 Debug Module Status (127C)—Status . . . 13-17
13.4 On-Chip SRAM . . . 13-17
13.5 SmartComm Timer Registers (SCTMR)—MBAR+0x0400 . . . 13-17

SECTION 14
FAST ETHERNET CONTROLLER (FEC)

14.1 Overview . . . 14-1
14.1.1 Features . . . 14-3
14.2 Modes of Operation . . . 14-3
14.2.1 Full- and Half-Duplex Operation . . . 14-3
14.2.2 10Mbps and 100Mbps MII Interface Operation . . . 14-3
14.2.3 10Mbps 7Wire Interface Operation . . . 14-4
14.2.4 Address Recognition Options . . . 14-4
14.2.5 Internal Loopback . . . 14-4
14.3 I/O Signal Overview . . . 14-4

Paragraph Number		Page Number
14.3.1	Detailed Signal Descriptions	14-5
14.3.1.1	MII Ethernet MAC-PHY Interface	14-5
14.3.1.2	MII Management Frame Structure	14-7
14.3.1.2.1	MII Management Register Set	14-8
14.4	FEC Memory Map and Registers	14-8
14.4.1	Top Level Module Memory Map	14-9
14.4.2	Control and Status (CSR) Memory Map	14-9
14.4.3	MIB Block Counters Memory Map	14-10
14.5	FEC Registers—MBAR + 0x3000	14-12
14.5.1	FEC ID (3000)—FEC_ID	14-13
14.5.2	Interrupt Event (3004)—IEVENT	14-14
14.5.3	Interrupt Enable (3008)—IMASK	14-16
14.5.4	Ethernet Control (3024)—ECNTRL	14-17
14.5.5	MII Management Frame (3040)—MII_DATA	14-18
14.5.6	MII Speed Control (3044)—MII_SPEED	14-20
14.5.7	MIB Control (3064)—MIB_CONTROL	14-21
14.5.8	Receive Control (3084)—R_CNTRL	14-22
14.5.9	Hash (3088)—R_HASH	14-23
14.5.10	Rx Destination Address Low (309C)—R_DA_LOW	14-24
14.5.11	Rx Destination Address High (30A0)—R_DA_HIGH	14-24
14.5.12	Tx Control (30C4)—X_CNTRL	14-25
14.5.13	Tx Status (30D0)—XMIT.X_STATUS	14-26
14.5.14	Physical Address Low (30E4)—PADDR1	14-27
14.5.15	Physical Address High (30E8)—PADDR2	14-27
14.5.16	Opcode/Pause Duration (30EC)—OP_PAUSE	14-28
14.5.17	Descriptor Individual Address 1 (3118)—IADDR1	14-28
14.5.18	Descriptor Individual Address 2 (311C)—IADDR2	14-29
14.5.19	Descriptor Group Address 1 (3120)—GADDR1	14-30
14.5.20	Descriptor Group Address 2 (3124)—GADDR2	14-30
14.5.21	Tx FIFO Watermark (3144)—X_WMRK	14-31
14.5.22	FIFO Interface	14-32
14.5.23	FIFO Data Register (3184,31A4)—TFIFO_DATA, RFIFO_DATA	14-32
14.5.24	FIFO Status Register(3188,31A8)—TFIFO_STATUS, RFIFO_STATUS	14-32
14.6	FIFO Control Register(318C,31AC)—TFIFO_CNTRL, RFIFO_CNTRL	14-34
14.7	FIFO Last Read Frame Pointer Register(3190,31B0)—TFIFO_LRF_PTR, RFIFO_LRF_PTR	14-35
14.8	FIFO Last Write Frame Pointer Register(3194,31B4)—TFIFO_LWF_PTR, RFIFO_LWF_PTR	14-35
14.9	FIFO Alarm Pointer Register(3198,31B8)—TFIFO_ALARM, RFIFO_ALARM	14-36
14.10	FIFO Read Pointer Register(319C,31BC)—TFIFO_RDPTR, RFIFO_RDPTR	14-37
14.11	FIFO Write Pointer Register(31A0,31C0)—TFIFO_WRPTR, RFIFO_WRPTR	14-38
14.12	Reset Control Register(31C4)—RESET_CNTRL	14-38
14.13	Transmit FSM Register(31C8)—XMIT_FSM	14-39
14.14	Initialization Sequence	14-39
14.14.1	Hardware Controlled Initialization	14-39
14.14.2	User Initialization (Prior to Asserting ETHER_EN)	14-40
14.14.2.1	Microcontroller Initialization	14-40
14.14.3	Frame Control/Status Words	14-41
14.14.3.1	Receive Frame Status Word	14-41
14.14.3.2	Transmit Frame Control Word	14-42
14.14.4	Network Interface Options	14-42
14.14.5	FEC Frame Reception	14-44
14.14.6	Ethernet Address Recognition	14-44
14.14.7	Full Duplex Flow Control	14-50
14.14.8	Inter Packet Gap Time	14-51

Paragraph Number | Page Number

14.14.9 Collision Handling 14-52
14.14.10 Internal and External Loopback 14-52
14.14.11 Ethernet Error-Handling Procedure 14-52
14.14.11.1 Transmission Errors 14-53
14.14.11.2 Reception Errors 14-53

SECTION 15 PROGRAMMABLE SERIAL CONTROLLERS (PSC)

15.1 Overview 15-1
15.1.1 PSC1—MBAR + 0x2000 15-1
15.1.2 PSC2—MBAR + 0x2400 15-1
15.1.3 PSC3—MBAR + 0x2800 15-1
15.1.4 Features 15-3
15.2 PSC Registers—MBAR + 0x2000, 0x2400, 0x2800 15-3
15.2.1 Mode Register 1 (2x00)—MR1_[1, 2, 3] 15-5
15.2.2 Mode Register 2 (2x00)—MR2_[1, 2, 3] 15-6
15.2.3 Status Register (2x04)—SR[1, 2, 3] 15-8
15.2.4 Modem Mode (2x04)—SR[1, 2, 3] 15-9
15.2.5 Clock-Select Register (2x04)—CSR[1, 2, 3] 15-10
15.2.6 Command Register (2x08)—CR[1, 2, 3] 15-11
15.2.7 Rx Buffer Registers (2x0C)—RB[1, 2, 3] 15-14
15.2.8 Tx Buffer Registers (2x0C)—TB[1, 2, 3] 15-15
15.2.9 Input Port Change Register (2x10)—IPCR[1, 2, 3] 15-16
15.2.10 Auxiliary Control Register (2x10)—ACR[1, 2, 3] 15-17
15.2.11 Interrupt Status Register (2x14)—ISR[1, 2, 3] 15-17
15.2.12 Interrupt Mask Register (2x14)—IMR[1, 2, 3] 15-18
15.2.13 Counter Timer Upper Register (2x18)—CTUR[1, 2, 3] 15-19
15.2.14 Counter Timer Lower Register (2x1C)—CTLR[1, 2, 3] 15-20
15.2.15 Interrupt Vector Register (2x30)—IVR[1, 2, 3] 15-20
15.2.16 Input Port (2x34)—IP[1, 2, 3] 15-20
15.2.17 Output Port 1 Bit Set (2x38)—OP1_[1, 2, 3] 15-21
15.2.18 Output Port 0 Bit Reset (2x3C)—OP0_[1, 2, 3] 15-22
15.2.19 PSC Control Register (2x40)—SICR[1, 2, 3] 15-23
15.2.20 Rx FIFO Number of Data (2x58)—RFNUM[1, 2, 3] 15-24
15.2.21 Tx FIFO Number of Data (2x5C)—TFNUM[1, 2, 3] 15-24
15.2.22 Rx FIFO Data (2x60)—RFDATA[1, 2, 3] 15-25
15.2.23 Rx FIFO Status (2x64)—RFSTAT[1, 2, 3] 15-25
15.2.24 Rx FIFO Control (2x68)—RFCNTL[1, 2, 3] 15-25
15.2.25 Rx FIFO Alarm (2x6E)—RFALARM[1, 2, 3] 15-26
15.2.25.1 Rx FIFO 15-27
15.2.25.2 Tx FIFO 15-27
15.2.26 Rx FIFO Read Pointer (2x72)—RFRPTR[1, 2, 3] 15-28
15.2.27 Rx FIFO Write Pointer (2x76)—RFWPTR[1, 2, 3] 15-28
15.2.28 Rx FIFO Last Read Frame PTR (2x7A)—RFLRFPTR[1, 2, 3] 15-29
15.2.29 Rx FIFO Last Write Frame PTR (2x7C)—RFLWFPTR[1, 2, 3] 15-29
15.2.30 Tx FIFO Data (2x80)—TFDATA[1, 2, 3] 15-29
15.2.31 Tx FIFO Status (2x84)—TFSTAT[1, 2, 3] 15-29
15.2.32 Tx FIFO Control (2x88)—TFCNTL[1, 2, 3] 15-30
15.2.33 Tx FIFO Alarm (2x8E)—TFALARM[1, 2, 3] 15-31
15.2.34 Tx FIFO Read Pointer (2x92)—TFRPTR[1, 2, 3] 15-31
15.2.35 Tx FIFO Write Pointer (2x96)—TFWPTR[1, 2, 3] 15-31
15.2.36 Tx FIFO Last Read Frame PTR (2x9A)—TFLRFPTR[1, 2, 3] 15-32

Paragraph Number | Page Number

15.2.37 Tx FIFO Last Write Frame PTR (2x9C)—TFLWFPTR[1, 2, 3] 15-32
15.3 PSC Module Signal Definitions 15-33
15.4 PSC Operation 15-35
15.4.1 Transmitter/Receiver Clock Source 15-35
15.4.1.1 Programmable Divider 15-35
15.4.1.2 Calculating Baud Rates 15-36
15.4.1.3 CLKIN Baud Rates 15-36
15.4.1.4 External Clock 15-36
15.4.2 Transmitter and Receiver Operating Modes 15-36
15.4.2.1 Transmitting in UART Mode 15-37
15.4.2.2 Transmitter in Modem Mode 15-39
15.4.2.3 AC97 Low-Power Mode 15-40
15.4.2.4 Receiver in UART Mode 15-40
15.4.2.5 Receiver in Modem Mode 15-42
15.4.2.6 FIFO Stack 15-42
15.4.3 Looping Modes 15-43
15.4.3.1 Automatic Echo Mode 15-44
15.4.3.2 Local Loop-Back Mode 15-44
15.4.3.3 Remote Loop-Back Mode 15-45
15.4.4 Multidrop Mode 15-45
15.4.5 Bus Operation 15-47
15.4.5.1 Read Cycles 15-47
15.4.5.2 Write Cycles 15-47
15.4.6 Programming 15-48
15.4.6.1 PSC Module Initialization Sequence 15-48

SECTION 16
Infrared Data Association (IrDA) Interface

16.1 Overview 16-1
16.1.1 Features 16-1
16.2 IrDA Registers—MBAR + 0x2C00 16-1
16.2.1 Mode Register 1 (2C00)—MR1 16-2
16.2.2 Mode Register 2 (2C00)—MR2 16-3
16.2.3 SIR Status Register (2C04)—SR 16-4
16.2.4 MIR/FIR Status Register (2C04)—SR 16-5
16.2.5 Clock-Select Register (2C04)—CSR 16-6
16.2.6 Command Register (2C08)—CR 16-7
16.2.7 Rx Buffers (2C0C)—RB 16-9
16.2.8 Tx Buffers (2C0C)—TB 16-9
16.2.9 Input Port Change (2C10)—IPCR 16-10
16.2.10 Auxiliary Control (2C10)—ACR 16-10
16.2.11 Interrupt Status (2C14)—ISR 16-11
16.2.12 Interrupt Mask (2C14)—IMR 16-12
16.2.13 Counter Timer Upper Bytes (2C18)—CTUR 16-13
16.2.14 Counter Timer Lower Bytes (2C1C)—CTLR 16-13
16.2.15 Interrupt Vector (2C30)—IVR 16-14
16.2.16 Input Port (2C34)—IP 16-14
16.2.17 Output Port Bit Set (2C38)—OP1 16-15
16.2.18 Output Port Bit Reset (2C3C)—OP0 16-15
16.2.19 IrDA Control (2C40)—SICR 16-16
16.2.20 Infrared Control 1 (2C44)—IRCR1 16-17
16.2.21 Infrared Control 2 (2C48)—IRCR2 16-18

Paragraph Number | Page Number

16.2.22 Infrared SIR Divide (2C4C)—IRSDR 16-19
16.2.23 Infrared MIR Divide (2C50)—IRMDR 16-19
16.2.24 Infrared FIR Divide (2C54)—IRFDR 16-20
16.2.25 Rx FIFO Number of Data (2C58)—RFNUM 16-21
16.2.26 Tx FIFO Number of Data (2C5C)—TFNUM 16-22
16.2.27 Rx FIFO Data (2C60)—RFDATA 16-22
16.2.28 Rx FIFO Status (2C64)—RFSTAT 16-22
16.2.29 Rx FIFO Control (2C68)—RFCNTL 16-23
16.2.30 Rx FIFO Alarm (2C6E)—RFALARM 16-23
16.2.31 Rx FIFO Read Pointer (2C72)—RFRPTR 16-24
16.2.32 Rx FIFO Write Pointer (2C76)—RFWPTR 16-24
16.2.33 Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR 16-24
16.2.34 Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR 16-25
16.2.35 Tx FIFO Data (2C80)—TFDATA 16-25
16.2.36 Tx FIFO Status (2C84)—TFSTAT 16-25
16.2.37 Tx FIFO Control (2C88)—TFCNTL 16-26
16.2.38 Tx FIFO Alarm (2C8E)—TFALARM 16-26
16.2.39 Tx FIFO Read Pointer (2C92)—TFRPTR 16-27
16.2.40 Tx FIFO Write Pointer (2C96)—TFWPTR 16-27
16.2.41 Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR 16-27
16.2.42 Tx FIFO Last Write Frame PTR (2C9C)—TFLWFPTR 16-28

SECTION 17
SERIAL PERIPHERAL INTERFACE (SPI)

17.1 Overview 17-1
17.1.1 Features 17-1
17.1.2 Modes of Operation 17-1
17.2 SPI Signal Description 17-2
17.2.1 Master In/Slave Out (MISO) 17-2
17.2.2 Master Out/Slave In (MOSI) 17-3
17.2.3 Serial Clock (SCK) 17-3
17.2.4 Slave-Select (SS) 17-3
17.3 SPI Registers—MBAR + 0x0F00 17-4
17.3.1 Control Register 1 (0F00)—SPICR1 17-4
17.3.2 Control Register 2 (0F01)—SPICR2 17-5
17.3.3 Baud Rate Register (0F04)—SPIBR 17-6
17.3.4 Status Register (0F05)—SPISR 17-8
17.3.5 Data Register (0F09)—SPIDR 17-9
17.3.6 Port Reduced Drive/Pull-up Select (0F0C)—SPIPURD 17-9
17.3.7 Port Data Register (0F0D)—SPIPORT 17-9
17.3.8 Data Direction Register (0F90)—SPIDDR 17-10

SECTION 18
INTER-INTEGRATED CIRCUIT (I^2C)

18.1 Overview 18-1
18.1.1 Features 18-2
18.2 I2C Controller 1 18-3
18.2.1 START Signal 18-4
18.2.2 STOP Signal 18-4
18.2.2.1 Slave Address Transmission 18-4
18.2.2.2 Data Transfer 18-5

Paragraph Number | Page Number

18.2.2.3 Acknowledge . . . 18-5
18.2.2.4 Repeated Start . . . 18-6
18.2.2.5 Clock Synchronization and Arbitration . . . 18-7
18.3 I2C Interface Registers—MBAR + 0x3D00. . . . 18-8
18.3.1 Address Register (3D00, 3D40)—I2C[1,2] . . . 18-8
18.3.2 Frequency Divider Register (3D04, 3D44)—I2C[1,2] . . . 18-9
18.3.3 Control Register (3D08, 3D48)—I2C[1,2] . . . 18-11
18.3.4 Status Register (3D0C, 3D4C)—I2C[1,2] . . . 18-12
18.3.5 Data I/O Register (3D10, 3D50)—I2C[1,2]. . . . 18-13
18.3.6 Interrupt Control Register (3D20) . . . 18-14
18.4 Initialization Sequence . . . 18-15
18.5 Transfer Initiation and Interrupt. . . . 18-16
18.5.1 Post-Transfer Software Response . . . 18-16
18.5.2 Slave Mode . . . 18-17
18.5.3 Typical M-bus Interrupt Routine Flow-Chart . . . 18-18
18.6 External Signals . . . 18-19
18.7 Interface Description. . . . 18-19
18.8 I2C Controller 2. . . . 18-19

SECTION 19
INFRARED (IR) INTERFACE

19.1 Overview. . . . 19-1
19.2 Block Description . . . 19-2
19.3 Signals and Connections . . . 19-2
19.4 IR Blaster . . . 19-2
19.4.1 IR Registers—MBAR + 0x0E00. . . . 19-3
19.4.1.1 Enable Control Register 0 (0E00) . . . 19-4
19.4.1.2 Data Stream Control Register (0E04) . . . 19-5
19.4.1.3 Data Stream Format Register (0E08) . . . 19-6
19.4.1.4 Data Stream Format Register (0E0C) . . . 19-7
19.4.1.5 Data Stream Format Register (0E10) . . . 19-7
19.4.1.6 Preamble Register (0E14) . . . 19-8
19.4.1.7 Status Register (0E18). . . . 19-8
19.4.1.8 Dataword Transmitted Register (0E1C). . . . 19-9
19.5 Remote IR Receiver . . . 19-10
19.5.1 Remote IR Registers—MBAR + 0x0E00 . . . 19-11
19.5.1.1 Enable Control Register (0E00). . . . 19-11
19.5.1.2 Data Stream Control Register (0E04) . . . 19-12
19.5.1.3 Data Stream Format Register (0E08) . . . 19-13
19.5.1.4 Data Stream Format Register (0E0C) . . . 19-13
19.5.1.5 Data Stream Format Register (0E10) . . . 19-14
19.5.1.6 Data Compare/Mask Register (0E14) . . . 19-15
19.5.1.7 Data Compare/Mask Register (0E18) . . . 19-15
19.5.1.8 Data Stream Format Register (0E1C) . . . 19-15
19.5.1.9 Status Register (0E20). . . . 19-16
19.5.1.10 Dataword Received Register (0E24) . . . 19-17
19.6 IR Keyboard Receiver. . . . 19-17

SECTION 20
MOTOROLA SCALABLE CAN (MSCAN)

20.1 Overview . . . 20-1
20.2 Features . . . 20-2
20.3 MSCAN Registers—MBAR + 0900 . . . 20-2
20.3.1 Control Register 0 (0900, 0980)—CAN[1,2]CTL0 . . . 20-4
20.3.2 Control Register 1 (0901, 0981)—CAN[1,2]CTL1 . . . 20-5
20.3.3 Bus Timing Register 0 (0904, 0984)—CAN[1,2]BTR0 . . . 20-6
20.3.4 Bus Timing Register 1 (0905, 0985)—CAN[1,2]BTR1 . . . 20-7
20.3.5 Rx Flag (0908, 0988)—CAN[1,2]RFLG . . . 20-8
20.3.6 Rx Interrupt Enable (0909, 0989)—CAN[1,2]RIER . . . 20-10
20.3.7 Tx Flag (090C, 098C)—CAN[1,2]TFLG . . . 20-11
20.3.8 Tx Interrupt Enable (090D, 098D)—CAN[1,2]TIER . . . 20-11
20.3.9 Tx Message Abort Request (0910, 0990)—CAN[1,2]TARQ . . . 20-12
20.3.10 Tx Message Abort Ack (0911, 0991)—CAN[1,2]TAAK . . . 20-12
20.3.11 Tx Buffer Select (0914, 0994)—CAN[1,2]BSEL . . . 20-13
20.3.12 ID Acceptance Control (0915, 0995)—CAN[1,2]IDAC . . . 20-13
20.3.13 Rx Error (091C, 099C)—CAN[1,2]RXERR . . . 20-14
20.3.14 Tx Error (091D, 099D)—CAN[1,2]TXERR . . . 20-14
20.3.15 ID Acceptance (0920–09B5)—CAN[1,2]IDAR[0–7] . . . 20-15
20.3.16 ID Mask (0928–09BD)—CAN[1,2]IDMR[0–7] . . . 20-16
20.3.17 Rx ID Register 0 (0940, 09C0)—CAN[1,2]RXIDR0 . . . 20-17
20.3.18 Rx ID Register 1 (0941, 09C1)—CAN[1,2]RXIDR1 . . . 20-17
20.3.19 Rx ID Register 0 (0940, 09C0)—CAN[1,2]RXIDR0 . . . 20-18
20.3.20 Rx ID Register 1 (0941, 09C1)—CAN[1,2]RXIDR1 . . . 20-18
20.3.21 Rx ID Register 2 (0944, 09C4)—CAN[1,2]RXIDR2 . . . 20-18
20.3.22 Rx ID Register 3 (0945,09C5)—CAN[1,2]RXIDR3 . . . 20-19
20.3.23 Rx Data Segment (0948–09D5)—CAN[1,2]RXDSR[0–7] . . . 20-19
20.3.24 Rx Data Length (0958, 09D8)—CAN[1,2]RXDLR . . . 20-20
20.3.25 Rx Time Stamp High (095C, 09DC)—CAN[1,2]RXTIMH . . . 20-20
20.3.26 Rx Time Stamp Low (095D, 09DD)—CAN[1,2]RXTIML . . . 20-21
20.3.27 Tx Buffer Priority (0979, 09F9)—CAN[1,2]TXTBPR . . . 20-21
20.3.28 Tx Time Stamp High (097C, 09FC)—CAN[1,2]TXTIMH . . . 20-22
20.3.29 Tx Time Stamp Low (097D, 09FD)—CAN[1,2]TXTIML . . . 20-22
20.3.30 Tx ID Register 0 (0960, 09E0)—CAN[1,2]TXIDR0 . . . 20-23
20.3.31 Tx ID Register 1 (0961, 09E1)—CAN[1,2]TXIDR1 . . . 20-23
20.3.32 Tx ID Register 2 (0964, 09E4)—CAN[1,2]TXIDR2 . . . 20-24
20.3.33 Tx ID Register 3 (0965, 09E5)—CAN[1,2]TXIDR3 . . . 20-24
20.3.34 Tx Data Segment (0968–09F5)—CAN[1,2]TXDSR[0–7] . . . 20-25
20.3.35 Tx Data Length (0978, 09F8)—CAN[1,2]TXDLR . . . 20-25
20.4 External Pin Descriptions . . . 20-26

SECTION 21
DEBUG SUPPORT AND JTAG INTERFACE

21.1 Overview . . . 21-1
21.2 TAP Link Module (TLM) and Slave TAP Implementation . . . 21-1
21.3 TLM and TAP Signal Descriptions . . . 21-5
21.3.1 Test Reset (TRST) . . . 21-5
21.3.2 Test Clock (TCK) . . . 21-5
21.3.3 Test Mode Select (TMS) . . . 21-6
21.3.4 Test Data In (TDI) . . . 21-6

21.3.5 Test Data Out (TDO) . . . 21-6
21.4 Slave Test Reset (STRST) . . . 21-6
21.4.1 Enable Slave—ENA[0:n] . . . 21-6
21.4.2 Select DR Link—SEL[0:n] . . . 21-6
21.4.3 Slave Test Data Out—STDO[0:n] . . . 21-7
21.5 TAP State Machines . . . 21-7
21.6 Harpo Core JTAG/COP Serial Interface . . . 21-8
21.7 TLM Link DR Instructions . . . 21-9
21.7.1 TLM:TLMENA . . . 21-9
21.7.2 TLM:PPCENA . . . 21-9
21.8 TLM Test Instructions . . . 21-9
21.8.1 IDCODE . . . 21-10
21.8.1.1 Device ID Register . . . 21-10
21.8.2 BYPASS . . . 21-10
21.8.3 SAMPLE/PRELOAD . . . 21-10
21.8.4 EXTEST . . . 21-11
21.8.5 CLAMP . . . 21-11
21.8.6 HIGHZ . . . 21-11
21.9 HARPO COP/BDM Interface . . . 21-11

SECTION A
TIMING AND ELECTRICAL SPECIFICATIONS

A.1 AC Timing Quick Reference . . . A-1
A.2 AC Timing Specifications . . . A-2
A.2.1 Clock . . . A-2
A.2.2 Reset . . . A-3
A.2.3 SDRAM . . . A-3
A.2.3.1 Memory Interface Timing—DDR SDRAM Read Command . . . A-3
A.2.3.2 Memory Interface Timing—Standard SDRAM Write Command . . . A-5
A.2.3.3 Memory Interface Timing—Standard SDRAM Read Command . . . A-6
A.2.4 PCI . . . A-7
A.2.5 LP–CS . . . A-9
A.2.5.1 Chip Select Non-MUXed Timing—1:1 . . . A-9
A.2.5.2 Chip Select Non-MUXed Timing—2:1 Phase A . . . A-10
A.2.5.3 Chip Select Non-MUXed Timing—2:1 Phase B . . . A-11
A.2.5.4 Chip Select MUXed Timing—1:1 . . . A-12
A.2.5.5 Chip Select MUXed Timing—2:1 Phase A . . . A-14
A.2.5.6 Chip Select MUXed Timing—2:1 Phase B . . . A-16
A.2.6 ATA . . . A-17
A.2.7 Ethernet . . . A-21
A.2.8 IR . . . A-24
A.2.9 IrDA . . . A-25
A.2.10 JTAG . . . A-26
A.2.10.1 IEEE 1149.1 (JTAG) AC Timing Specification . . . A-26
A.2.11 USB . . . A-28
A.2.12 SPI . . . A-28
A.2.12.1 SPI Master AC Timing Specifications . . . A-28
A.2.12.2 SPI Slave AC Timing Specifications . . . A-29
A.2.13 I^{2}C . . . A-31
A.2.14 MSCAN . . . A-32
A.2.15 PSC . . . A-33
A.2.16 GPIOs and Timers . . . A-34

Paragraph Number | Page Number

A.3 DC Electrical Specifications A-35
A.3.1 Electrostatic Discharge A-37
A.3.2 Thermal Characteristics. A-37
A.3.3 Heat Dissipation A-38
A.3.4 Power Dissipation A-39

SECTION B
MECHANICAL SPECIFICATIONS

B.1 Overview. B-1
B.2 Case Diagrams B-2
B.2.1 272-Pin PBGA B-2

SECTION C
ADDENDUM

C.1 Overview. C-1

SECTION D
TROUBLESHOOTING

D.1 Solutions to Known Problems D-1

SECTION E
ACRONYMS AND TERMS

SECTION F
LIST OF REGISTERS

LIST OF FIGURES

Page Number

Figure 1-1 Simplified Block Diagram—MGT5100 1-4
Figure 1-2 MGT5100-Based System 1-6
Figure 2-1. 272-Pin PBGA Pin Detail 2-1
Figure 2-2 272-Pin PBGA—Top View 2-2
Figure 2-3 PSC Peripheral Multiplexing 2-22
Figure 2-4 PSC1 Port Map—5 Pins 2-22
Figure 2-5 PSC2 Port Map—5 Pins 2-23
Figure 2-6 PSC3 Port Map—10 Pins 2-24
Figure 2-7 Ethernet Output Port Map—8 Pins 2-25
Figure 2-8 Ethernet Input/Control Port Map—10 Pins 2-25
Figure 2-9 USB Port Map—10 Pins 2-26
Figure 2-10 Timer Port Map—8 Pins 2-26
Figure 2-11 I^{2}C Port Map—4 Pins (two pins each, for two I^{2}Cs) 2-27
Figure 2-12 IR/IrDA Port Map—4 Pins 2-27
Figure 4-1 Reset sequence 4-3
Figure 4-2 PORESET Assertion 4-4
Figure 4-3 Internal Hard Reset vs External HRESET Assertion 4-4
Figure 4-4 HRESET Asserted more than 1024 Reference Clock Cycles 4-5
Figure 5-1 Primary Synchronous Clock Domains 5-3
Figure 5-2 Timing Diagram—Clock Waveforms for SDRAM and DDR Memories 5-7
Figure 5-3 Bus Clock Ratios 5-8
Figure 7-1 Interrupt Sources and Core Interrupt Pins 7-4
Figure 7-2 Interrupt Controller Routing Scheme 7-5
Figure 7-3 GPIO/Generic MUX Cell 7-22
Figure 7-4 Diagram—Suggested Crystal Oscillator Circuit 7-65
Figure 8-1 Block Diagram—SDRAM Subsystem Example 8-3
Figure 8-2 Block Diagram—SDRAM Memory Controller 8-8
Figure 8-3 Example—Address Bus Mapping 8-28
Figure 9-1 Timing Diagram—Non-MUXed Access 9-4
Figure 9-2 LPC Concept Diagram 9-9
Figure 9-3 Using ALE Transactions 9-10
Figure 9-4 Timing Diagram—MUXed Transaction Type 9-12
Figure 10-1 Block Diagram—PCI Interface 10-3
Figure 11-1 ATA Controller Interface 11-2
Figure 11-2 Connections—Controller Cable, System Board, MGT5100 11-26
Figure 11-3 Pin Description—ATA Interface 11-28
Figure 11-4 ATA Sector Format 11-33
Figure 11-5 Timing Diagram—PIO Read Command (Class 1) 11-34
Figure 11-6 Timing Diagram—PIO Write Command (Class 2) 11-35
Figure 11-7 Timing Diagram—Non-Data Command (Class 3) 11-36
Figure 11-8 Flow Diagram—DMA Command Protocol 11-39
Figure 11-9 Timing Diagram—DMA Command (Class 4) 11-40
Figure 11-10 Timing Diagram—Reset Timing 11-42
Figure 11-11 Timing Diagram—PIO Mode 11-44
Figure 11-12 Timing Diagram—Multiword DMA 11-45
Figure 11-13 Timing Diagram—Initiating an Ultra DMA Data In Burst 11-46
Figure 11-14 Timing Diagram—Sustained Ultra DMA Data In Burst 11-48
Figure 11-15 Timing Diagram—Host Pausing an Ultra DMA Data In Burst 11-49
Figure 11-16 Timing Diagram—Drive Terminating Ultra DMA Data In Burst 11-49
Figure 11-17 Timing Diagram—Host Terminating Ultra DMA Data In Burst 11-50
Figure 11-18 Timing Diagram—Initiating an Ultra DMA Data Out Burst 11-50
Figure 11-19 Timing Diagram—Sustained Ultra DMA Data Out Burst 11-51

Page Number

Figure 11-20 Timing Diagram—Drive Pausing an Ultra DMA Data Out Burst 11-51
Figure 11-21 Timing Diagram—Host Terminating Ultra DMA Data Out Burst 11-52
Figure 11-22 Timing Diagram—Drive Terminating Ultra DMA Data Out Burst 11-52
Figure 12-1 USB Focus Areas 12-1
Figure 12-2 Communication Channels 12-3
Figure 12-3 Typical List Structure 12-4
Figure 12-4 Interrupt ED Structure 12-5
Figure 12-5 Sample Interrupt Endpoint Schedule 12-6
Figure 14-1 Block Diagram—FEC 14-2
Figure 14-2 Receive Frame Status Word Format....................14-41
Figure 14-3 Transmit Frame Control Word Format 14-42
Figure 14-4 Ethernet Address Recognition - receive block decisions 14-46
Figure 14-5 Ethernet Address Recognition - microcode decisions....................14-47
Figure 15-1 Simplified Block Diagram 15-2
Figure 15-2 Block Diagram—PSC....................15-33
Figure 15-3 PSC/RS-232 Interface....................15-34
Figure 15-4 PSC1/CODEC and PSC2/CODEC Interface 15-35
Figure 15-5 PSC1/AC97 and PSC2/AC97 Interface 15-35
Figure 15-6 Clocking Source Diagram 15-36
Figure 15-7 Functional Diagram—Tx and Rx....................15-37
Figure 15-8 Timing Diagram—Transmitter....................15-38
Figure 15-9 Timing Diagram—16-Bit CODEC Interface (lsb First)....................15-39
Figure 15-10 Timing Diagram—8-Bit CODEC Interface (msb First)....................15-39
Figure 15-11 Timing Diagram—AC97 Interface 15-39
Figure 15-12 Timing Diagram—Receiver....................15-41
Figure 15-13 Automatic Echo....................15-44
Figure 15-14 Local Loop-Back 15-44
Figure 15-15 Remote Loop-Back 15-45
Figure 15-16 Timing Diagram—Multidrop Mode 15-46
Figure 15-17 Programming Flowchart—UART Mode (sheet 1 of 5)....................15-49
Figure 15-18 Programming Flowchart—UART Mode (sheet 2 of 5)....................15-50
Figure 15-19 Programming Flowchart—UART Mode (sheet 3 of 5)....................15-51
Figure 15-20 Programming Flowchart—UART Mode (sheet 4 of 5)....................15-52
Figure 15-21 Programming Flowchart—UART Mode (sheet 5 of 5)....................15-53
Figure 17-1 Block Diagram—SPI 17-2
Figure 18-1 Block Diagram—I^{2}C 18-3
Figure 18-2 Timing Diagram—Start, Address Transfer and Stop Signal....................18-5
Figure 18-3 Timing Diagram—Data Transfer 18-5
Figure 18-4 Timing Diagram—Receiver Acknowledgement....................18-6
Figure 18-5 Data Transfer, Combined Format 18-7
Figure 18-6 Timing Diagram—Clock Synchronization....................18-7
Figure 18-7 Timing Diagram—Arbitration Procedure 18-8
Figure 18-8 Timing Diagram—SCL Period and SDA Hold Time 18-10
Figure 18-9 Flow Chart—M-bus Interrupt Routine 18-18
Figure 18-10 I^{2}C Controller1 External Connections....................18-19
Figure 19-1 IRDA Controller External Connections....................19-2
Figure 20-1 Block Diagram—MSCAN 20-1
Figure 20-2 The CAN System 20-27
Figure 21-1 Generic TLM/TAP Architecture Diagram....................21-3
Figure 21-2 Generic TAP Link Module (TLM) Diagram 21-4
Figure 21-3 Generic Slave TAP 21-5
Figure 21-4 State Diagram—TAP Controller....................21-7

Page
Number

Figure 21-5 Harpo Core JTAG/COP Serial Interface....21-8
Figure 21-6 COP Connector Diagram....21-13
Figure A-1 Timing Diagram—SYS_XTAL_IN.... A-2
Figure A-2 Timing Diagram—DDR SDRAM Memory Read Timing.... A-3
Figure A-3 Timing Diagram—Standard SDRAM Memory Write Timing.... A-5
Figure A-4 Timing Diagram—Standard SDRAM Memory Write Timing.... A-6
Figure A-5 PCI Timing Diagram—Basic Read/Write.... A-7
Figure A-6 Timing Diagram—Chip Select Access (non-MUXed).... A-9
Figure A-7 Timing Diagram—Chip Select Access (non-MUXed).... A-10
Figure A-8 Timing Diagram—Chip Select Access (non-MUXed).... A-11
Figure A-9 Timing Diagram—Chip Select Access (MUXed, 1:1).... A-12
Figure A-10 Timing Diagram—Chip Select (MUXed, 2:1 Phase A).... A-14
Figure A-11 Timing Diagram—Chip Select (MUXed, 2:1 Phase B).... A-16
Figure A-12 PIO Mode Timing.... A-18
Figure A-13 Multiword DMA Timing.... A-19
Figure A-14 Initiating an Ultra DMA Data In Burst.... A-20
Figure A-15 Ethernet Timing Diagram—Interrupt Events Tx Example.... A-21
Figure A-16 Ethernet Timing Diagram—Interrupt Events Rx Example.... A-21
Figure A-17 Ethernet Timing Diagram—MII Rx Signal.... A-22
Figure A-18 Ethernet Timing Diagram—MII Tx Signal.... A-22
Figure A-19 Ethernet Timing Diagram—MII Tx Signal.... A-23
Figure A-20 Ethernet Timing Diagram—MII Async.... A-23
Figure A-21 Ethernet Timing Diagram—MII Serial Management.... A-24
Figure A-22 IR Timing Diagram—SIP Waveform.... A-24
Figure A-23 IrDA Timing Diagram—Low-Speed Data Format.... A-25
Figure A-24 IrDA Timing Diagram—Middle-Speed Data Format.... A-25
Figure A-25 IrDA Timing Diagram—High-Speed Data Format.... A-25
Figure A-26 Timing Diagram—JTAG Clock Input.... A-26
Figure A-27 Timing Diagram—JTAG TRST.... A-26
Figure A-28 Timing Diagram—JTAG Boundary Scan.... A-26
Figure A-29 Timing Diagram—Test Access Port.... A-27
Figure A-30 Timing Diagram—USB.... A-28
Figure A-31 Timing Diagram—SPI Master (CPHA=0).... A-28
Figure A-32 Timing Diagram—SPI Master (CPHA=1).... A-29
Figure A-33 Timing Diagram—SPI Slave (CPHA=0).... A-30
Figure A-34 Timing Diagram—I^{2}C Input/Output.... A-31
Figure A-35 Timing Diagram—CAN.... A-32
Figure A-36 Timing Diagram—8- and 16-bit CODEC Mode.... A-33
Figure A-37 Timing Diagram—AC97 Mode.... A-33
Figure A-38 Timing Diagram—General-Purpose I/O.... A-34
Figure B-1 Case Diagram—272-Pin PBGA.... B-2

LIST OF TABLES

		Page Number
Table 2-1	Signals by Ball/Pin	2-3
Table 2-2	Signals by Signal Name	2-6
Table 2-3	Functional Signal List with Pin/Pad Count	2-9
Table 3-1	Internal Register Memory Map	3-2
Table 3-2	Module Base Address (0x0000)—MBAR	3-4
Table 3-3	CS0 Start Address (0004)—CS0STR	3-5
Table 3-4	CS0 Stop Address (0008)—CS0STP	3-5
Table 3-5	CS1 Start Address (000C)—CS1STR	3-5
Table 3-6	CS1 Stop Address (0010)—CS1STP	3-6
Table 3-7	CS2 Start Address (0014)—CS2STR	3-6
Table 3-8	CS2 Stop Address (0018)—CS2STP	3-7
Table 3-9	CS3 Start Address (001C)—CS3STR	3-7
Table 3-10	CS3 Stop Address (0020)—CS3STP	3-7
Table 3-11	CS4 Start Address (0024)—CS4STR	3-8
Table 3-12	CS4 Start Address (0028)—CS4STP	3-8
Table 3-13	CS5 Start Address (002C)—CS5STR	3-9
Table 3-14	CS5 Start Address (0030)—CS5STP	3-9
Table 3-15	SDRAM Start Address (0034)—SDRAMSTR	3-9
Table 3-16	SDRAM Stop Address (0038)—SDRAMSTP	3-10
Table 3-17	PCI1 Start Address (003C)—PCI1STR	3-10
Table 3-18	PCI1 Stop Address (0040)—PCI1STP	3-11
Table 3-19	PCI2 Start Address (0044)—PCI2STR	3-11
Table 3-20	PCI2 Stop Address (0048)—PCI2STP	3-11
Table 3-21	Boot Start Address (004C)—BOOTSTR	3-12
Table 3-22	Boot Stop Address (0050)—BOOTSTP	3-12
Table 3-23	Address Space Enable (0054)—ADREN	3-13
Table 3-24	Overview—PCI XLB Configuration Registers	3-14
Table 3-25	SmartComm DMA Registers	3-16
Table 3-26	PSC Memory Mapping	3-19
Table 3-27	IrDA Memory Mapping	3-23
Table 3-28	SPI Memory Mapping	3-26
Table 3-29	SDRAM Controller Register Map	3-27
Table 3-30	Slice Timers Register Map	3-27
Table 4-1	Module Specific Reset Signals	4-5
Table 4-2	POR Configuration Word Source Pins	4-6
Table 5-1	Clock Distribution Module	5-1
Table 5-2	G2 Frequencies vs xlb_clk Frequencies	5-4
Table 5-3	G2 APLL Configuration Options	5-5
Table 5-4	SDRAM Memory Controller Clock Domain	5-6
Table 5-5	Typical System Clock Frequencies with 33MHz Input	5-8
Table 5-6	Typical System Clock Frequencies with 27MHz Input	5-9
Table 5-7	CDM JTAG ID Number (0200)—JTAGID: 01C5301D hex	5-14
Table 5-8	CDM Power ON Reset Configuration (0204)—PORCFG	5-15
Table 5-9	CDM Bread Crumb (0208)—BC	5-16
Table 5-10	CDM Configuration (020C)—CFG	5-16
Table 5-11	CDM 48MHz Fractional Divider Configuration (0210)—FDCFG	5-17
Table 5-12	CDM Clock Enable (0214)—CLKEN	5-18
Table 5-13	CDM System Oscillator Configuration (0218)—OSCCFG	5-20
Table 5-14	CDM Clock Control Sequencer Configuration (021C)—CCSCFG	5-20
Table 5-15	CDM Soft Reset (0220)—SFTRST	5-21
Table 5-16	CDM System PLL Status (0224)—PLLSTA (33–27MHz)	5-22
Table 6-1	Arbiter Configuration (00C0)—ACFG	6-3
Table 6-2	Arbiter Version (00C4)—VER	6-4

Page Number

Table 6-3 Arbiter Status (00C8)—STA....6-4
Table 6-4 Arbiter Interrupt Enable (00CC)—INTEN....6-5
Table 6-5 Arbiter Address Capture (00D0)—ADRCAP....6-6
Table 6-6 Arbiter Bus Signal Capture (00D4)—SIGCAP....6-7
Table 6-7 Arbiter Address Tenure Time-Out (00D8)—ADRTO....6-7
Table 6-8 Arbiter Data Tenure Time-Out (00DC)—DATTO....6-8
Table 6-9 Arbiter Bus Activity Time-Out (00E0)—BUSTO....6-8
Table 6-10 Arbiter Master Priority Enable (00E4)—PRIEN....6-9
Table 6-11 Disabled Master Priority....6-9
Table 6-12 Arbiter Master Priority (00E8)—PRI....6-10
Table 6-13 Base Address (00EC)—BA....6-11
Table 6-14 High Order and Significant Bits....6-11
Table 6-15 Reserved Registers (0080–00BF, 00EC, 00F0–00FC)....6-12
Table 7-1 Interrupt Sources....7-1
Table 7-2 System Management Interrupt Pin Interrupts....7-2
Table 7-3 Core Interrupt Pins Summary....7-3
Table 7-4 Peripheral Interrupt Mask (0500)....7-6
Table 7-5 Peripheral Priority and HI/LO Select 1 (0504)....7-7
Table 7-6 Peripheral Priority and HI/LO Select 2 (0508)....7-8
Table 7-7 Peripheral Priority and HI/LO Select 3 (050C)....7-9
Table 7-8 External Enable and External Types (0510)....7-9
Table 7-9 Critical Priority and Main Interrupt Mask (0514)....7-11
Table 7-10 Main Interrupt Priority and INT/SMI Select 1 (0518)....7-12
Table 7-11 Main Interrupt Priority and INT/SMI Select 2 (051C)....7-13
Table 7-12 PerStat, MainStat, CritStat Encoded (0524)....7-14
Table 7-13 Critical Interrupt Status All (0528)....7-16
Table 7-14 Main Interrupt Status All (052C)....7-16
Table 7-15 Peripheral Interrupt Status All (0530)....7-17
Table 7-16 Peripheral Interrupt Status All (0538)....7-18
Table 7-17 GPIO Pin List....7-20
Table 7-18 GPIO Standard Register Types—MBAR+0x0B00....7-27
Table 7-19 Port Configuration Register (0B00)—GPIOPCR....7-28
Table 7-20 Simple GPIO Enables (0B04)—GPIOSEN....7-30
Table 7-21 Simple GPIO Open Drain Type (0B08)—GPIOSOD....7-31
Table 7-22 Simple GPIO Data Direction (0B0C)—GPIOSDD....7-32
Table 7-23 Simple GPIO Data Output Values (0B10)—GPIOSDO....7-34
Table 7-24 Simple GPIO Data Input Values (0B14)—GPIOSDI....7-35
Table 7-25 GPIO Output-Only Enables (0B18)—GPIOOE....7-36
Table 7-26 GPIO Output-Only Data Value Out (0B1C)—GPIOODO....7-37
Table 7-27 GPIO Simple Interrupt Enables (0B20)—GPIOSIE....7-38
Table 7-28 GPIO Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD....7-38
Table 7-29 GPIO Simple Interrupt Data Direction (0B28)—GPIOSIDD....7-39
Table 7-30 GPIO Simple Interrupt Data Value Out (0B2C)—GPIOSIDO....7-40
Table 7-31 GPIO Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE....7-40
Table 7-32 GPIO Simple Interrupt Interrupt Types (0B34)—GPIOSIIT....7-41
Table 7-33 GPIO Simple Interrupt Master Enable (0B38)—GPIOSIME....7-42
Table 7-34 GPIO Simple Interrupt Status (0B3C)—GPIOSIST....7-42
Table 7-35 WakeUp GPIO Enables (0C00)—GPIOWE....7-44
Table 7-36 WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD....7-45
Table 7-37 WakeUp GPIO Data Direction (0C08)—GPIOWDD....7-46
Table 7-38 WakeUp GPIO Data Value Out (0C0C)—GPIOWDO....7-46
Table 7-39 WakeUp GPIO Interrupt Enable (0C10)—GPIOWUE....7-47

Page Number

Table 7-40 WakeUp GPIO Individual Interrupt Enable (0C14)—GPIOWSIE 7-48
Table 7-41 WakeUp GPIO Interrupt Types (0C18)—GPIOWT 7-49
Table 7-42 WakeUp GPIO Master Enables (0C1C)—GPIOWME 7-50
Table 7-43 WakeUp GPIO Data Input Values (0C20)—GPIOWI 7-50
Table 7-44 WakeUp GPIO Status Register (0C24)—GPIOWS 7-51
Table 7-45 GPT[0–7] Enable and Mode Select (0600–0670) 7-54
Table 7-46 GPT[0–7] Counter Input (0604–0674) 7-57
Table 7-47 GPT[0–7] PWM Configuration (0608–0678) 7-58
Table 7-48 GPT[0–7] Status (060C–067C) 7-59
Table 7-49 Terminal Count Register — TCNT0, 1 (0700,0710) 7-61
Table 7-50 Control Register — CTL0, 1 (0704, 0714) 7-62
Table 7-51 Count Value Register — TMRCNT0, 1 (0708, 0718) 7-62
Table 7-52 Timer Status Register — TMRSTA0, 1 (070C, 071C) 7-63
Table 7-53 Real-Time Clock Signals 7-64
Table 7-54 RTC Time Set (0800) 7-66
Table 7-55 RTC Date Set (0804) 7-67
Table 7-56 RTC New Year and Stopwatch (0808) 7-68
Table 7-57 RTC Alarm and Interrupt Enable (080C) 7-68
Table 7-58 RTC Current Time (0810) 7-69
Table 7-59 RTC Current Date (0814) 7-70
Table 7-60 RTC Alarm and Stopwatch Interrupt (0818) 7-71
Table 7-61 RTC Periodic Interrupt and Bus Error (081C) 7-72
Table 7-62. Test Register/Divides (0820) 7-73
Table 8-1 Memory Configurations 8-5
Table 8-2 SDRAM External Signals 8-6
Table 8-3 SDRAM Address Multiplexing 8-9
Table 8-4 SDRAM Commands 8-13
Table 8-5 Mode Register 0 (0100)—MOD 8-19
Table 8-6 Extended Mode Register 0, DDR Only (0100)—MOD 8-20
Table 8-7 Control Register 1 (0104)—CTR 8-21
Table 8-8 Address Select 8-22
Table 8-9 Configuration Register 1 (0108)—CFG1 8-23
Table 8-10 Configuration Register 2 (010C)—CFG2 8-25
Table 8-11 Address Select Register (0110)—ADRSEL 8-26
Table 8-12 XL Bus Select 8-26
Table 8-13 Configuration Register 1 = 0xC222 2700 8-29
Table 8-14 Configuration Register 2 = 0x88B7 0004 8-29
Table 8-15 Control Register = 0xD858 000x1 8-30
Table 8-16 Mode Register = H008D 0000 8-30
Table 8-17 Load Sequence for Registers 8-30
Table 8-18 Configuration Register 1 = 0x7362 2830 8-31
Table 8-19 Configuration Register 2 = 0x4577 0004 8-31
Table 8-20 Control Register = 0xF051 0F00 8-31
Table 8-21 Mode Register = 0x008D 0000 and 0x4009 0000 8-32
Table 8-22 Load Sequence for Registers 8-32
Table 9-1 BOOT_CONFIG (RST_CONFIG) Settings 9-6
Table 9-2 LocalPlus External Signals 9-9
Table 9-3 CS Boot ROM, (0300) 9-13
Table 9-4 CS Configuration (0304–0314) 9-16
Table 9-5 CS Control (0318) 9-18
Table 9-6 CS Status (031C) 9-19
Table 10-1 PCI External Signals 10-2

		Page Number
Table 10-2	PCI Header: Device ID/Vendor ID (0D00)	10-6
Table 10-3	PCI Header: Status/Command (0D04)	10-7
Table 10-4	PCI Header: Class Code/Revision (0D08)	10-9
Table 10-5	PCI Header: BIST/Type/Latency/Cache (0D0C)	10-10
Table 10-6	PCI Header: BAR0 (0D10)	10-11
Table 10-7	PCI Header: BAR1 (0D14)	10-11
Table 10-8	PCI Header: Max Lat/Min Grant/Interrupt (0D3C)	10-13
Table 10-9	PCI Interrupt Enable (0D60)	10-14
Table 10-10	PCI Status (0D64)	10-14
Table 10-11	PCI Control (0D68)	10-15
Table 10-12	PCI Mask/Value Read (0D6C)	10-16
Table 10-13	PCI Mask/Value Write (0D70)	10-17
Table 10-14	PCI Subwindow 1 (0D74)	10-18
Table 10-15	PCI Subwindow 2 (0D78)	10-18
Table 10-16	PCI Window Command/Control (0D7C)	10-19
Table 10-17	PCI Tx Packet Size (3800)	10-21
Table 10-18	PCI Tx Start Address (3804)	10-22
Table 10-19	PCI Tx Transaction Control (3808)	10-22
Table 10-20	PCI Tx Enables (380C)	10-23
Table 10-21	PCI Tx Next Address (3810)	10-25
Table 10-22	PCI Tx Last Word (3814)	10-25
Table 10-23	PCI Tx Done Counts (3818)	10-26
Table 10-24	PCI Tx Status Bits (381C)	10-26
Table 10-25	PCI Tx FIFO Data (3840)	10-28
Table 10-26	PCI Tx FIFO Status (3844)	10-28
Table 10-27	PCI Tx FIFO Control (3848)	10-29
Table 10-28	PCI Tx Alarm (384E)	10-30
Table 10-29	PCI Tx Read Pointer (3852)	10-30
Table 10-30	PCI Tx Write Pointer (3856)	10-31
Table 10-31	PCI Rx Packet Size (3880)	10-32
Table 10-32	PCI Rx Start Address (3884)	10-33
Table 10-33	PCI Rx Transaction Command (3888)	10-33
Table 10-34	PCI Rx Enables (388C)	10-34
Table 10-35	PCI Rx Next Address (3890)	10-35
Table 10-36	PCI Rx Done Counts (3898)	10-36
Table 10-37	PCI Rx Status Bits (389C)	10-36
Table 10-38	PCI Rx FIFO Data (38C0)	10-38
Table 10-39	PCI Rx FIFO Status (38C4)	10-38
Table 10-40	PCI Rx FIFO Control (38C8)	10-39
Table 10-41	PCI Rx Alarm (38CC)	10-40
Table 10-42	PCI Rx Read Pointer (38D0)	10-40
Table 10-43	PCI Rx Write Pointer (38D4)	10-41
Table 11-1	ATA Host Configuration (3A00)—HCFG	11-3
Table 11-2	ATA Host Status (3A04)—HSR	11-4
Table 11-3	ATA PIO Timing 1 (3A08)—PIO1	11-4
Table 11-4	ATA PIO Timing 2 (3A0C)—PIO2	11-5
Table 11-5	ATA Multiword DMA Timing 1 (3A10)—DMA1	11-5
Table 11-6	ATA Multiword DMA Timing 2 (3A14)—DMA2	11-6
Table 11-7	ATA Ultra DMA Timing 1 (3A18)—UDMA1	11-6
Table 11-8	ATA Ultra DMA Timing 2 (3A1C)—UDMA2	11-7
Table 11-9	ATA Ultra DMA Timing 3 (3A20)—UDMA3	11-8
Table 11-10	ATA Ultra DMA Timing 4 (3A24)—UDMA4	11-8

Page Number

Table 11-11 ATA Ultra DMA Timing 5 (3A28)—UDMA5 11-9
Table 11-12 ATA Rx/Tx FIFO Data Word (3A3C)—RTFDWR 11-10
Table 11-13 ATA Rx/Tx FIFO Status (3A40)—RTFSR 11-10
Table 11-14 ATA Rx/Tx FIFO Control (3A44)—RTFCR 11-11
Table 11-15 ATA Rx/Tx Alarm (3A48)—RTFAR 11-12
Table 11-16 ATA Rx/Tx FIFO Read Pointer (3A4C)—RTFRPR 11-12
Table 11-17 ATA Rx/Tx FIFO Write Pointer (3A50)—RTFWPR 11-13
Table 11-18 ATA Drive Device Control (3A5C)—DCTR 11-14
Table 11-19 ATA Drive Alternate Status (3A5C)—DASR 11-14
Table 11-20 ATA Drive Data (3A60)—DDR 11-15
Table 11-21 ATA Drive Features (3A64)—DFR 11-15
Table 11-22 ATA Drive Error (3A64)—DER 11-16
Table 11-23 ATA Drive Sector Count (3A68)—DSCR 11-16
Table 11-24 ATA Drive Sector Number (3A6C)—DSNR 11-17
Table 11-25 ATA Drive Cylinder Low (3A70)—DCLR 11-17
Table 11-26 ATA Drive Cylinder High (3A74)—DCHR 11-18
Table 11-27 ATA Drive Device/Head (3A78)—DDHR 11-18
Table 11-28 ATA Drive Command (3A7C)—DCR 11-19
Table 11-29 ATA Drive Device Status (3A80)—DSR 11-20
Table 11-30 PIO Timing Requirements 11-23
Table 11-31 Multiword DMA Timing Requirements 11-24
Table 11-32 MGT5100 External Signals 11-24
Table 11-33 ATA Controller External Connections 11-27
Table 11-34 ATA Standards 11-29
Table 11-35 ATA Physical Level Modes 11-30
Table 11-36 ATA Register Address/Chip Select Decoding 11-30
Table 11-37 DMA Command Parameters 11-38
Table 11-38 Redefinition of Signal Lines for Ultra DMA Protocol 11-41
Table 11-39 Reset Timing Characteristics 11-43
Table 11-40 DC Electrical Characteristics 11-43
Table 11-41 AC Electrical Specifications 11-44
Table 11-42 PIO Mode Timing Specifications 11-44
Table 11-43 Multiword DMA Timing Specifications 11-46
Table 11-44 Ultra DMA Timing Specification 11-47
Table 12-1 HC Revision Register (1000)—HcRevision 12-7
Table 12-2 HC Control Register (1004)—HcControl 12-8
Table 12-3 HC Command Status Register (1008)—HcCommandStatus 12-10
Table 12-4 HC Interrupt Status Register (100C)—HcInterruptStatus 12-11
Table 12-5 HC Interrupt Enable Register (1010)—HcInterruptEnable 12-12
Table 12-6 HC Interrupt Disable Register (1014)—HcInterruptDisable 12-13
Table 12-7 HC Communication Register (1018)—HcHCCA 12-15
Table 12-8 HC Period Current ED Register (101C)—HcPeriodCurrentED 12-15
Table 12-9 HC Control Head ED Register (1020)—HcControlHeadED 12-16
Table 12-10 HC Control Current ED Register (1024) HcControlCurrentED 12-16
Table 12-11 HC First Bulk Head List ED Register (1028)—HcBulkHeadED 12-17
Table 12-12 HC Bulk List Current ED Register (102C)—HcBulkCurrentED 12-17
Table 12-13 HC Last Completed Transfer Register (1030)—HcDoneHead 12-18
Table 12-14 HC Frame Interval Register (1034)—HcFmInterval 12-19
Table 12-15 HC Frame Bit-time Remaining Register (1038)—HcFmRemaining 12-20
Table 12-16 HC Timing Reference Register (103C)—HcFmNumber 12-21
Table 12-17 HC Start Processing Periodic List Register (1040)—HcPeriodicStart 12-21
Table 12-18 HC Commit Transfer Register (1044)—HcLSThreshold 12-22

Page Number

Table 12-19 HC Root Hub Descriptor A Register (1048)—HcRhDescriptorA 12-23
Table 12-20 HC Root Hub Descriptor B Register (104C)—HcRhDescriptorB 12-24
Table 12-21 HC Root Hub Status Register (1050)—HcRhStatus 12-25
Table 12-22 HC RH Port Status Register (1054)—HcRhPortStatus[1:NDP].............................. 12-27
Table 13-1. Task Bar (1200)—taskBar 13-4
Table 13-2. Current Pointer (1204)—currentPointer.. 13-4
Table 13-3. End Pointer (1208)—endPointer 13-4
Table 13-4. Variable Pointer (120C)—variablePointer.. 13-5
Table 13-5. Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl 13-5
Table 13-6. Interrupt Pending (1214)—IntPend... 13-6
Table 13-7. Interrupt Mask (1218)—IntMask 13-6
Table 13-8. Task Control (121C)—TCR0, TCR1 13-7
Table 13-9. Task Control (1220)—TCR2, TCR3... 13-7
Table 13-10. Task Control (1224)—TCR4, TCR5... 13-8
Table 13-11. Task Control (1200)—TCR6, TCR7... 13-8
Table 13-12. Task Control (122C)—TCR8, TCR9 13-8
Table 13-13. Task Control (1200)—TCRA, TCRB... 13-9
Table 13-14. Task Control (1234)—TCRC, TCRD 13-9
Table 13-15. Task Control (1238)—TCRE, TCRF 13-10
Table 13-16. Initiator Priority (123C)—IPR0–3 13-10
Table 13-17. Initiator Priority (1240)—IPR4–7... 13-10
Table 13-18. Initiator Priority (1244)—IPR8–11 13-11
Table 13-19. Initiator Priority (1248)—IPR12–15... 13-11
Table 13-20. Initiator Priority (124C)—IPR16–19 13-12
Table 13-21. Initiator Priority (1250)—IPR20–23... 13-12
Table 13-22. Initiator Priority (1254)—IPR24–27... 13-13
Table 13-23. Initiator Priority (1258)—IPR28–31 13-13
Table 13-24. Reserved Register 1 (125C)—res1 13-14
Table 13-25. Reserved Register 2 (1260)—res2... 13-14
Table 13-26. Reserved Register 3 (1264)—res3... 13-14
Table 13-27. Reserved Register 4 (1268)—res4... 13-15
Table 13-28. Reserved Register 5 (126C)—res5 13-15
Table 13-29. Debug Module Comparator 1 (1270)—Value1 13-16
Table 13-30. Debug Module Comparator 2 (1274)—Value2.. 13-16
Table 13-31. Debug Modulator Control (1278)—Control 13-16
Table 13-32. Debug Module Status (127C)—Status 13-17
Table 13-33. SCTMR Register/Address—MBAR+0x0400 13-17
Table 14-1 Signal Properties 14-4
Table 14-2 MII: Valid Encoding of TxD, Tx_EN and Tx_ER 14-7
Table 14-3 MII: Valid Encoding of RxD, Rx_ER and Rx_DV 14-7
Table 14-4 MMI Format Definitions 14-7
Table 14-5 MII Management Register Set.. 14-8
Table 14-6 Module Memory Map.. 14-9
Table 14-7 CSR Counters 14-9
Table 14-8 MIB Counters... 14-11
Table 14-9 FEC ID Register (3000)—FEC_ID 14-13
Table 14-10 Interrupt Event Register (3004)—IEVENT 14-15
Table 14-11 Interrupt Enable Register (3008)—IMASK.. 14-16
Table 14-12 Ethernet Control Register (3024)—ECNTRL 14-17
Table 14-13 MII Management Frame Register (3040)—MII_DATA.. 14-18
Table 14-14 MII Speed Control Register (3044)—MII_SPEED... 14-20
Table 14-15 Programming Examples for MII_SPEED Register.. 14-21

Page
Number

Table 14-16 MIB Control Register (3064)—MIB_CONTROL........14-22
Table 14-17 Receive Control Register (3084)—R_CNTRL........14-22
Table 14-18 Hash Register (3088)—R_HASH........14-23
Table 14-19 Rx Destination Address Low Register (309C)—R_DA_LOW........14-24
Table 14-20 Rx Destination Address High Register (30A0)—R_DA_HIGH........14-25
Table 14-21 Tx Control Register (30C4)—X_CNTRL........14-25
Table 14-22 Tx Status Register (30D0)—XMIT.X_STATUS........14-26
Table 14-23 Physical Address Low Register (30E4)—PADDR1........14-27
Table 14-24 Physical Address High Register (30E8)—PADDR2........14-28
Table 14-25 Opcode/Pause Duration Register (30EC)—OP_PAUSE........14-28
Table 14-26 Descriptor Individual Address 1 Register (3118)—IADDR1........14-29
Table 14-27 Descriptor Individual Address 2 Register (311C)—IADDR2........14-29
Table 14-28 Descriptor Group Address 1 Register (3120)—GADDR1........14-30
Table 14-29 Descriptor Group Address 2 Register (3124)—GADDR2........14-30
Table 14-30 Tx FIFO Watermark Register (3144)—X_WMRK........14-31
Table 14-31 FIFO Interface Register Map........14-32
Table 14-32 Tx Rx FIFO Status Register (3188,31A8)—TFIFO_STATUS, RFIFO_STATUS.......14-33
Table 14-33 Tx Rx FIFO Control Register (318C,31AC)—TFIFO_CNTRL, RFIFO_CNTRL.........14-34
Table 14-34 Tx Rx FIFO Last Read Frame Pointer Register (3190,31B0)—TFIFO_LRF_PTR, RFIFO_LRF_PTR14-35
Table 14-35 Tx Rx FIFO Last Write Frame Pointer Register (3194,31B4)—TFIFO_LWF_PTR, RFIFO_LWF_PTR14-36
Table 14-36 Tx Rx FIFO Alarm Pointer Register (3198,31B8)—TFIFO_ALARM, RFIFO_ALARM14-37
Table 14-37 Tx Rx FIFO Read Pointer Register (319C,31BC)—TFIFO_RDPTR, RFIFO_RDPTR14-37
Table 14-38 Tx Rx FIFO Write Pointer Register (31A0,31C0)—TFIFO_WRPTR, RFIFO_WRPTR..14-38
Table 14-39 Reset Control Register (31C4)—RESET_CNTRL........14-38
Table 14-40 Transmit FSM Register (31C8)—XMIT_FSM........14-39
Table 14-41 ETHER_EN De-Assertion Affect on FEC........14-40
Table 14-42 User Initialization (Before ETHER_EN)........14-40
Table 14-43 Microcontroller Initialization (FEC)........14-41
Table 14-44 Destination Address to 6-Bit Hash........14-48
Table 14-45 PAUSE Frame Field Specification........14-50
Table 14-46 Transmit Pause Frame Registers........14-51
Table 15-1 UART Mode 1 (2000, 2400, 2800)—MR1_[1, 2, 3]........15-5
Table 15-2 Other Modes (2000, 2400, 2800)—MR1_[1, 2, 3]........15-5
Table 15-3 Parity Mode/Parity Type Definitions........15-6
Table 15-4 UART Mode 2 (2000, 2400, 2800)—MR2_[1, 2, 3]........15-6
Table 15-5 Other Modes (2000, 2400, 2800)—MR2_[1, 2, 3]........15-6
Table 15-6 Stop-Bit Lengths........15-7
Table 15-7 UART Mode (2004, 2404, 2804)—SR[1, 2, 3]........15-8
Table 15-8 Modem Status (2004, 2404, 2804)—SR[1, 2, 3]........15-9
Table 15-9 UART Mode (2004, 2404, 2804)—CSR[1, 2, 3]........15-10
Table 15-10 Other Modes (2004, 2404, 2804)—CSR[1, 2, 3]........15-10
Table 15-11 All Modes (2008, 2408, 2808)—CR[1, 2, 3]........15-11
Table 15-12 UART/Modem8&16 Rx Buffers (200C, 240C, 280C)—RB[1, 2, 3]........15-14
Table 15-13 AC97 Rx Buffers (200C, 240C)—RB[1, 2]........15-14
Table 15-14 UART/Modem8&16 Tx Buffers (200C, 240C, 280C)—TB[1, 2, 3]........15-15
Table 15-15 AC97 Tx Buffers (200C, 240C)—TB[1, 2]........15-15
Table 15-16 Input Port UART Change (2010, 2810, 2410)—IPCR[1, 3, 2]........15-16
Table 15-17 Modem Mode (2010, 2810, 2410)—IPCR[1, 3, 2]........15-16
Table 15-18 All Modes (2010, 2410, 2810)—ACR[1, 2, 3]........15-17

Page Number

Table 15-19 UART Mode (2014, 2414, 2814)—ISR[1, 2, 3] 15-17
Table 15-20 Modem Mode (2014, 2414, 2814)—ISR[1, 2, 3] 15-17
Table 15-21 Interrupt UART (2014, 2414, 2814)—IMR[1, 2, 3] 15-18
Table 15-22 Modem Mode (2014, 2414, 2814)—IMR[1, 2, 3] 15-18
Table 15-23 Counter Timer UART (2018, 2418, 2818)—CTUR[1, 2, 3] 15-19
Table 15-24 Other Modes (2018, 2418, 2818)—CTUR[1, 2, 3] 15-19
Table 15-25 Counter Timer UART (201C, 241C, 281C)—CTLR[1, 2, 3] 15-20
Table 15-26 Other Modes (201C, 241C, 281C)—CTLR[1, 2, 3] 15-20
Table 15-27 Interrupt Vector (2030, 2430, 2830)—IVR[1, 2, 3] 15-20
Table 15-28 Input UART (2034, 2434, 2834)—IP[1, 2, 3] 15-21
Table 15-29 Input Modem8&16 Mode (2034, 2434, 2834)—IP[1, 2, 3] 15-21
Table 15-30 Input AC97 Mode (2034, 2434)—IP[1, 2] 15-21
Table 15-31 UART/Modem8&16 Set (2038, 2438, 2838)—OP1_[1, 2, 3] 15-21
Table 15-32 AC97 Mode (2038, 2438)—OP1_[1, 2] 15-22
Table 15-33 UART/Modem8&16 Reset (203C, 243C, 284C)—OP0_[1, 2, 3] 15-22
Table 15-34 AC97Bit Reset (203C, 243C)—OP0_[1, 2] 15-22
Table 15-35 UART Mode (2040, 2440, 2840)—SICR[1, 2, 3] 15-23
Table 15-36 Modem8&16 Mode (2040, 2440, 2840)—SICR[1, 2, 3] 15-23
Table 15-37 AC97 Mode(2040, 2440, 2840)—SICR[1, 2] 15-23
Table 15-38 Rx FIFO Number of Data (2058, 2458, 2858)—RFNUM[1, 2, 3] 15-24
Table 15-39 Tx FIFO Number of Data (205C, 245C, 285C)—TFNUM[1, 2, 3] 15-24
Table 15-40 Rx FIFO Status (2064, 2464, 2864)—RFSTAT[1, 2, 3] 15-25
Table 15-41 Rx FIFO Control (2068, 2468, 2868)—RFCNTL[1, 2, 3] 15-25
Table 15-42 Rx FIFO Alarm (206E, 246E, 286E)—RFALARM[1, 2, 3] 15-26
Table 15-43 Rx FIFO Read Pointer (2072, 2472, 2872)—RFRPTR[1, 2, 3] 15-28
Table 15-44 Rx FIFO Write Pointer (2076, 2476, 2876)—RFWPTR[1, 2, 3] 15-28
Table 15-45 Rx FIFO Last Read Frame PTR (207A, 247A, 287A)—RFLRFPTR[1, 2, 3] 15-29
Table 15-46 Rx FIFO Last Write Frame PTR (207C, 247C, 287C)—RFLWFPTR[1, 2, 3] 15-29
Table 15-47 Tx FIFO Status (2084, 2484, 2884)—TFSTAT[1, 2, 3] 15-29
Table 15-48 Tx FIFO Control (2088, 2488, 2888)—TFCNTL[1, 2, 3] 15-30
Table 15-49 Tx FIFO Alarm (208E, 248E, 288E)—TFALARM[1, 2, 3] 15-31
Table 15-50 Tx FIFO Read Pointer (2092, 2492, 2892)—TFRPTR[1, 2, 3] 15-31
Table 15-51 Tx FIFO Write Pointer (2096, 2496, 2896)—TFWPTR[1, 2, 3] 15-31
Table 15-52 Tx FIFO Last Read Frame PTR (209A, 249A, 289A)—TFLRFPTR[1, 2, 3] 15-32
Table 15-53 Tx FIFO Last Write Frame PTR (209C, 249C, 289C)—TFLWFPTR[1, 2, 3] 15-32
Table 15-54 PSC Module Signals 15-34
Table 15-55 PSC Module Initialization Sequence 15-48
Table 16-1 SIR Mode 1 (2C00)—MR1 16-2
Table 16-2 MIR/FIR Modes (2C00)—MR1 16-2
Table 16-3 SIR Mode 2 (2C00)—MR2 16-3
Table 16-4 MIR/FIR Modes (2C00)—MR2 16-3
Table 16-5 SIR Status (2C04)—SR 16-4
Table 16-6 MIR/FIR Status (2C04)—SR 16-5
Table 16-7 SIR Mode (2C04)—CSR 16-6
Table 16-8 Other Modes (2C04)—CSR 16-6
Table 16-9 Command Register, All Modes (2C08)—CR 16-7
Table 16-10 SIR/MIR/FIR Rx Buffers (2C0C)—RB 16-9
Table 16-11 SIR/MIR/FIR Tx Buffers (2C0C)—TB 16-9
Table 16-12 Input Port SIR/MIR/FIR Change (2C10)—IPCR 16-10
Table 16-13 Auxiliary Control (2C10)—ACR 16-10
Table 16-14 Interrupt SIR Status (2C14)—ISR 16-11
Table 16-15 Interrupt MIR/FIR Status (2C14)—ISR 16-11

Page
Number

Table 16-16 Interrupt SIR Mask (2C14)—IMR 16-12
Table 16-17 Interrupt MIR/FIR Mask (2C14)—IMR 16-12
Table 16-18 Counter Timer SIR Upper Bytes (2C18)—CTUR 16-13
Table 16-19 Counter Timer MIR/FIR Upper Bytes (2C18)—CTUR 16-13
Table 16-20 Counter Timer SIR Lower Bytes (2C1C)—CTLR 16-14
Table 16-21 Counter Timer MIR/FIR Lower Bytes (2C1C)—CTLR 16-14
Table 16-22 Interrupt Vector (2C30)—IVR 16-14
Table 16-23 IrDA Input Port (2C34)—IP 16-14
Table 16-24 IrDA Output Port Bit Set (2C38)—OP1 16-15
Table 16-25 IrDA Output Port Bit Reset (2C3C)—OP0 16-15
Table 16-26 IrDA SIR Control (2C40)—SICR 16-16
Table 16-27 IrDA MIR/FIR Control (2C40)—SICR 16-16
Table 16-28 Infrared SIR Control 1 (2C44)—IRCR1 16-17
Table 16-29 Infrared MIR/FIR Control 1 (2C44)—IRCR1 16-17
Table 16-30 Infrared SIR Control 2 (2C48)—IRCR2 16-18
Table 16-31 Infrared MIR/FIR Control 2 (2C48)—IRCR2 16-18
Table 16-32 Infrared SIR Divide (2C4C)—IRSDR 16-19
Table 16-33 Infrared SIR Other Divide (2C4C)—IRSDR 16-19
Table 16-34 Infrared MIR Divide (2C50)—IRMDR 16-19
Table 16-35 Infrared MIR Other Divide (2C50)—IRMDR 16-19
Table 16-36 Frequency Selection in MIR Mode 16-20
Table 16-37 Infrared FIR Divide (2C54)—IRFDR 16-20
Table 16-38 Infrared FIR Other Divide (2C54)—IRFDR 16-20
Table 16-39 Frequency Selection in MIR Mode 16-21
Table 16-40 Rx FIFO Number of Data (2C58)—RFNUM 16-21
Table 16-41 Tx FIFO Number of Data (2C5C)—TFNUM 16-22
Table 16-42 Rx FIFO Status (2C64)—RFSTAT 16-22
Table 16-43 Rx FIFO Control (2C68)—RFCNTL 16-23
Table 16-44 Rx FIFO Alarm (2C6E)—RFALARM 16-23
Table 16-45 Rx FIFO Read Pointer (2C72)—RFRPTR 16-24
Table 16-46 Rx FIFO Write Pointer (2C76)—RFWPTR 16-24
Table 16-47 Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR 16-24
Table 16-48 Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR 16-25
Table 16-49 Tx FIFO Status (2C84)—TFSTAT 16-25
Table 16-50 Tx FIFO Control (2C88)—TFCNTL 16-26
Table 16-51 Tx FIFO Alarm (2C8E)—TFALARM 16-26
Table 16-52 Tx FIFO Read Pointer (2C92)—TFRPTR 16-27
Table 16-53 Tx FIFO Write Pointer (2C96)—TFWPTR 16-27
Table 16-54 Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR 16-27
Table 16-55 Tx FIFO Last Write Frame Pointer (2C9C)—TFLWFPTR 16-28
Table 17-1 SPI External Signal Descriptions 17-2
Table 17-2 SPI Control 1 (0F00)—SPICR1 17-4
Table 17-3 SS Input/Output Selection 17-5
Table 17-4 SPI Control 2 (0F01)—SPICR2 17-5
Table 17-5 Bidirectional Pin Configurations 17-5
Table 17-6 SPI Baud Rate (0F04)—SPIBR 17-6
Table 17-7 SPI Baud Rate Selection—40MHz Module Clock 17-6
Table 17-8 SPI Status (0F05)—SPISR 17-8
Table 17-9 SPI Data (0F09)—SPIDR 17-9
Table 17-10 SPI Port Reduced Drive/Pull-up Select (0F0C)—SPIPURD 17-9
Table 17-11 SPI Port Data (0F0D)—SPIPORT 17-9
Table 17-12 SPI Data Direction (0F90)—SPIDDR 17-10

		Page Number
Table 18-1	I^{2}C Terminology	18-4
Table 18-2	I^{2}C Address Register (3D00, 3D40)—I2C[1,2]	18-8
Table 18-3	I^{2}C Frequency Divider Register (3D04, 3D44)—I2C[1,2]	18-9
Table 18-4	I^{2}C Tap and Prescale Values	18-9
Table 18-5	I^{2}C Control Register (3D08, 3D48)—I2C[1,2]	18-11
Table 18-6	I^{2}C Status Register (3D0C, 3D4C)—I2C[1,2]	18-12
Table 18-7	I^{2}C Data I/O Register (3D10, 3D50)—I2C[1,2]	18-13
Table 18-8	I^{2}C Interrupt Control Register (3D20)	18-14
Table 18-9	I^{2}C External Signals	18-19
Table 19-1	Infrared Interface Port External Signals	19-2
Table 19-2	IR Enable Control Register (0E00)	19-4
Table 19-3	IR Data Stream Control Register (0E04)	19-5
Table 19-4	IR Data Stream Format Register (0E08)	19-6
Table 19-5	IR Data Stream Format Register (0E0C)	19-7
Table 19-6	IR Data Stream Format Register (0E10)	19-7
Table 19-7	IR Preamble Register (0E14)	19-8
Table 19-8	IR Status Register (0E18)	19-8
Table 19-9	IR Dataword Transmitted Register (0E1C)	19-9
Table 19-10	Remote IR Enable Control Register (0E00)	19-11
Table 19-11	Remote IR Data Stream Control Register (0E04)	19-12
Table 19-12	Remote IR Data Stream Format Register (0E08)	19-13
Table 19-13	Remote IR Data Stream Format Register (0E0C)	19-13
Table 19-14	Remote IR Data Stream Format Register (0E10)	19-14
Table 19-15	Remote IR Data Compare/Mask Register (0E14)	19-15
Table 19-16	Remote IR Data Compare/Mask Register (0E18)	19-15
Table 19-17	Remote IR Data Stream Format Register (0E1C)	19-15
Table 19-18	Remote IR Status Register (0E20)	19-16
Table 19-19	Remote IR Dataword Received Register (0E24)	19-17
Table 20-1.	MSCAN Control Register 0 (0900, 0980)—CAN[1,2]CTL0	20-4
Table 20-2.	MSCAN Control Register 1 (0901, 0981)—CAN[1,2]CTL1	20-5
Table 20-3.	MSCAN Bus Timing Register 0 (0904, 0984)—CAN[1,2]BTR0	20-6
Table 20-4.	Baud Rate Prescaler	20-7
Table 20-5.	MSCAN Bus Timing Register 1 (0905, 0985)—CAN[1,2]BTR1	20-7
Table 20-6.	Time Segment 1 Values	20-8
Table 20-7.	Time Segment 2 Values	20-8
Table 20-8.	MSCAN Rx Flag (0908, 0988)—CAN[1,2]RFLG	20-8
Table 20-9.	MSCAN Rx Interrupt Enable (0909, 0989)—CAN[1,2]RIER	20-10
Table 20-10.	MSCAN Tx Flag (090C, 098C)—CAN[1,2]TFLG	20-11
Table 20-11.	MSCAN Tx Interrupt Enable (090D, 098D)—CAN[1,2]TIER	20-11
Table 20-12.	MSCAN Tx Message Abort Request (0910, 0990)—CAN[1,2]TARQ	20-12
Table 20-13.	MSCAN Tx Message Abort Ack (0911, 0991)—CAN[1,2]TAAK	20-12
Table 20-14.	MSCAN Tx Buffer Select (0914, 0994)—CAN[1,2]BSEL	20-13
Table 20-15.	MSCAN ID Acceptance Control (0915, 0995)—CAN[1,2]IDAC	20-13
Table 20-16.	Identifier Acceptance Hit Indication	20-13
Table 20-17.	Identifier Acceptance Mode Settings	20-14
Table 20-18.	MSCAN Rx Error (091C, 099C)—CAN[1,2]RXERR	20-14
Table 20-19.	MSCAN Tx Error (091D, 099D)—CAN[1,2]TXERR	20-15
Table 20-20.	MSCAN ID Acceptance (0920–0935)—CAN[1,2]IDAR[1–7]	20-15
Table 20-21.	MSCAN ID Mask (0928–09BD)—CAN[1,2]IDMR[0–7]	20-16
Table 20-22.	MSCAN Rx ID Register 0 (0940–09C0)—CAN[1,2]RXIDR0	20-17
Table 20-23.	MSCAN Rx ID Register 1 (0941–09C1)—CAN[1,2]RXIDR1	20-17
Table 20-24.	MSCAN Rx ID Register 0 (0940, 09C0)—CAN[1,2]RXIDR0	20-18

Page Number

Table 20-25. MSCAN Rx ID Register 1 (0941, 09C1)—CAN[1,2]RXIDR1 20-18
Table 20-26. MSCAN Rx ID Register 2 (0944, 09C4)—CAN[1,2]RXIDR2 20-19
Table 20-27. MSCAN Rx ID Register 3 (0945,09C5)—CAN[1,2]RXIDR3 20-19
Table 20-28. MSCAN Rx Data Segment (0948–09D5)—CAN[1,2]RXDSR[0–7]............................20-20
Table 20-29. MSCAN Rx Data Length (0958, 09D8)—CAN[1,2]RXDLR 20-20
Table 20-30. MSCAN Rx Time Stamp High (095C, 09DC)—CAN[1,2]RXTIMH 20-20
Table 20-31. MSCAN Rx Time Stamp Low (095D, 09DD)—CAN[1,2]RXTIML.............................20-21
Table 20-32. MSCAN Tx Buffer Priority (0979, 09F9)—CAN[1,2]TXTBPR 20-21
Table 20-33. MSCAN Tx Time Stamp High (097C, 09FC)—CAN[1,2]TXTIMH.............................20-22
Table 20-34. MSCAN Tx Time Stamp Low (097D, 09FD)—CAN[1,2]TXTIML 20-22
Table 20-35. MSCAN Tx ID Register 0 (0960, 09E0)—CAN[1,2]TXIDR0 20-23
Table 20-36. MSCAN Tx ID Register 1 (0961, 09E1)—CAN[1,2]TXIDR1 20-23
Table 20-37. MSCAN Tx ID Register 2 (0964, 09E4)—CAN[1,2]TXIDR2 20-24
Table 20-38. MSCAN Tx ID Register 3 (0965, 09E5)—CAN[1,2]TXIDR3 20-24
Table 20-39. MSCAN Tx Data Segment (0968–09F5)—CAN[1,2]TXDSR[0–7] 20-25
Table 20-40. MSCAN Tx Data Length (0978, 09F8)—CAN[1,2]TXDLR.......................................20-25
Table 20-41. Data Length Codes...20-26
Table 21-1 TLM Link-DR Instructions 21-9
Table 21-2 TLM Test Instruction Encoding 21-10
Table 21-3 Device ID Register = 01C5301D hex...21-10
Table 21-4 COP/BDM Interface Signals 21-11
Table A-1 SYS_XTAL_IN Timing A-2
Table A-2 Clock Frequencies... A-2
Table A-3 Power Consumption A-2
Table A-4 DDR SDRAM Memory Read Timing... A-4
Table A-5 Standard SDRAM Memory Write Timing A-5
Table A-6 Standard SDRAM Memory Write Timing A-6
Table A-7 PCI Electrical Characteristics A-8
Table A-8 Chip Select Access Electrical Characteristics A-9
Table A-9 Chip Select Access Electrical Characteristics A-10
Table A-10 Chip Select Access Electrical Characteristics A-11
Table A-11 CS MUXed Electrical Characteristics—1:1 .... A-13
Table A-12 CS MUXed Electrical Characteristics—2:1 Phase A.. A-15
Table A-13 CS MUXed Electrical Characteristics—2:1 Phase B.. A-17
Table A-14 PIO Mode Timing Specifications A-19
Table A-15 Multiword DMA Timing Specifications A-20
Table A-16 MII Rx Signal Timing A-22
Table A-17 MII Tx Signal Timing... A-22
Table A-18 MII Tx Signal Timing... A-23
Table A-19 MII Async Signal Timing A-23
Table A-20 MII Serial Management Channel Signal Timing A-24
Table A-21 JTAG Timing Specification A-27
Table A-22 USB Timing Specifications A-28
Table A-23 SPI Master AC Timing Specifications.. A-29
Table A-24 SPI Slave AC Timing Specifications.. A-30
Table A-25 I^{2}C Input Timing Specifications—SCL and SDA.. A-31
Table A-26 I^{2}C Output Timing Specifications—SCL and SDA.. A-31
Table A-27 CAN Timing Specifications... A-32
Table A-28 PSC Module AC Timing Specifications... A-34
Table A-29 General-Purpose I/O Timing Specifications A-34
Table A-30 Absolute Maximum Ratings[1].. A-35
Table A-31 Recommended Operating Conditions A-35

Page Number

Table A-32 DC Electrical Specifications A-36
Table A-33 Electrostatic Discharge A-37
Table A-34 Package Thermal Characteristics A-37
Table A-35 Power Dissipation A-39
Table D-1 Solutions or Work-Arounds D-1

SECTION 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. MGT5100 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*".

MGT5100 integrates a high performance MPC603e series G2 core with a rich set of peripheral functions focused on communications and systems integration. The G2 core design is based on the PowerPC core architecture. MGT5100 incorporates an innovative I/O subsystem, which isolates routine maintenance of peripheral functions from the embedded G2 core.

MGT5100 supports a dual external bus architecture. It has a high speed SDRAM/DDR bus interface that connects directly to the G2. In addition, it has a Peripheral Component Interconnect (PCI) compatible bus interface, which is used as a generalized interface to system level peripheral devices and debug environments.

1.1.1 Features

Key features are shown below.

- MPC603e series G2 core
  - Superscalar architecture
  - 0–231 MHz static operation
  - 0-250 MHz static operation (depending on operating conditions)
  - 423 Mips at 300 MHz
  - 280 Mips at 200MHz
  - 16k Instruction cache, 16k data cache
  - Double precision FPU
  - Instruction and Data MMU
  - Standard & Critical interrupt capability
- High speed SDRAM memory interfacen
  - SDRAM & DDR SDRAM support
  - 256-MByte addressing range
  - 32-bit data bus
  - Built-in initialization and refresh

- Flexible multi-function external bus
  - Supports PCI, ATA and SRAM interfaces
  - Version 2.2 PCI master compatibility
    - 32-bit MUXed address/data
    - 0–66MHz operation
    - 0–33MHz operation (with SmartComm enabled)
    - Limited PCI target capability
  - Version 4 ATA compatible external interface—IDE Disk Drive connectivity
  - ROM/SRAM/Flash interface—Boot ROM, external peripheral connectivity
- SmartComm I/O subsystem
  - Intelligent virtual DMA Controller
  - Dedicated DMA channels to control peripheral reception and transmission
  - 3 Programmable Serial Controllers (PSCx)
    - UART or RS232 interface
    - External full function modem interface
    - 1200 baud POTS, V.34, V.90
    - CODEC interface for Soft Modem
  - 10/100 Ethernet MAC
  - USB Version 1.1 Master—Support for two independent USB slave ports
    - 12Mbps transfers
  - Infrared (IR) data port
    - Two IR receive ports (Standard IR and IrDA)
    - IRBlaster
    - IrDA 1.0 SIR mode to 115.2Kbps
  - I^2C Controller(s) to 485Kbps (with a 33MHz crystal frequency)
  - Support for two independent I^2C ports
  - SPI Controller to 20Mbps
- Dual MSCAN 2.0 A/B Controller Modules
  - Motorola Scalable Controller Area Network (MSCAN) architecture
  - Implementation of version 2.0A/B CAN protocol
  - Standard and extended data frames
  - Programmable bit-rate up to 1Mbps
- Systems level features
  - 6 programmable chip selects
  - Interrupt Controller
    - 4 external interrupt request lines supporting standard and/or critical interrupt
    - Support for all other internally generated interrupt sources

- GPIO/Timer functions
    - 1 dedicated GPIO pin supporting WakeUp capability
    - 8 GPIO pins with timer capability supporting input capture, output compare and pulse width modulation functions
    - Up to 56 total GPIO pins (depending on functional MUXing selections) that support a variety of interrupt/WakeUp capabilities.
- Systems Protection (watch dog timer, bus monitor)
- Real-time Clock with 1 second resolution
- Power management
    - Nap, Doze, Sleep, Deep Sleep modes
    - Individual control of functional block clock sources
    - support of WakeUp from low power modes by different sources

- Test/Debug features
    - JTAG (IEEE 1149.1 test access port)
    - Common On-Chip Processor (COP) debug port
- On-board PLL and clock generation
- Software
    - VXWorks
    - Linux
    - Software Modem capable
    - JAVA

1.2 Architecture

The following areas comprise the MGT5100 system architecture:

- Embedded G2 Core
- SmartComm I/O Subsystem
- Dual Motorola Scalable (MS) Controller Area Network (CAN)
- System Level Interfaces
- SDRAM Controller and Interface
- Multi-Function External 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 MGT5100 block diagram.

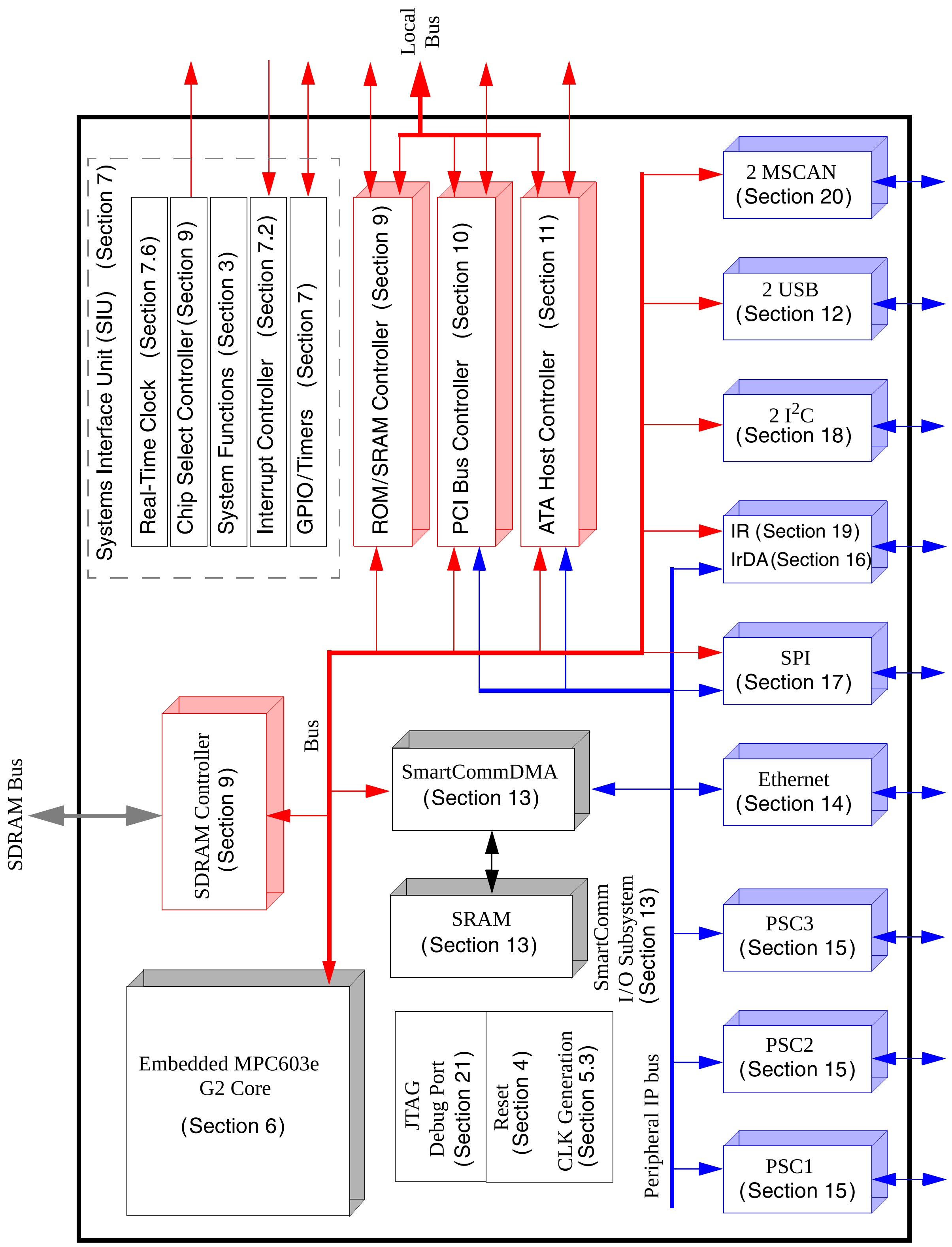

Local Bus
Systems Interface Unit (SIU) (Section 7)
Real-Time Clock (Section 7.6)
Chip Select Controller (Section 9)
System Functions (Section 3)
Interrupt Controller (Section 7.2)
GPIO/Timers (Section 7)
ROM/SRAM Controller (Section 9)
PCI Bus Controller (Section 10)
ATA Host Controller (Section 11)
2 MSCAN (Section 20)
2 USB (Section 12)
2 I2C (Section 18)
IR (Section 19)
IrDA (Section 16)
SPI (Section 17)
Ethernet (Section 14)
PSC3 (Section 15)
PSC2 (Section 15)
PSC1 (Section 15)
SDRAM Bus
SDRAM Controller (Section 9)
Bus
SmartCommDMA (Section 13)
SRAM (Section 13)
SmartComm I/O Subsystem (Section 13)
Embedded MPC603e G2 Core (Section 6)
JTAG Debug Port (Section 21)
Reset (Section 4)
CLK Generation (Section 5.3)
Peripheral IP bus

Figure 1-1 Simplified Block Diagram—MGT5100

MGT5100 supports a dual external bus architecture consisting of:

1. a Memory Controller interface bus
2. a multi-function Local bus

The Memory Controller has an SDRAM interface, which supports standard SDRAM and Dual Data Rate (DDR) SDRAM devices. The Memory Controller has a 13-bit Memory Address (MA) and a 32-bit data bus. 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 16KByte instruction and 16KByte data caches, enables performance hungry applications such as Java and soft modems. Still, plenty of processing power remains for peripheral management and system control tasks.

The Local bus allows connection of external peripheral devices, disk storage, and slower speed memory. The Local bus supports:

- PCI compliant devices
- ATA compliant devices (IDE disk drives)
- an external Boot ROM/FLASH/SRAM interface

MGT5100 integrates a high performance MPC603e series G2 core with an I/O subsystem containing an intelligent Direct Memory Access (DMA) unit. MGT5100 is capable of:

- responding to peripheral interrupts, independent of the G2 core.
- providing low level peripheral management, protocol processing, and peripheral data movement functions.

MGT5100 has an optimized peripheral mix to support today’s embedded automotive and telematics requirements.

Figure 1-2 shows an MGT5100-based system.

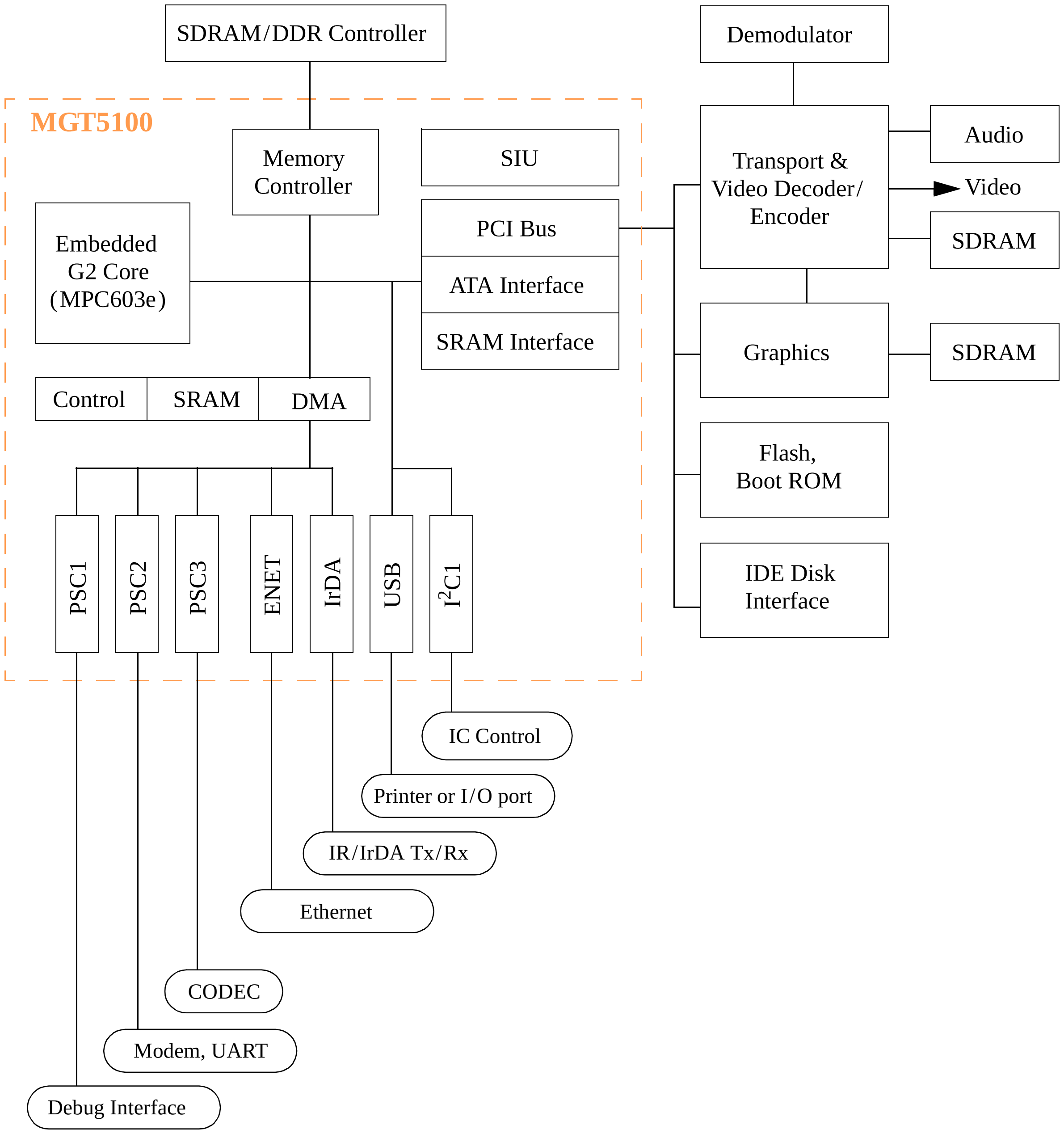


Figure 1-2 MGT5100-Based System

1.2.1 Embedded G2 Core

The MGT5100 embedded G2 core is derived from Motorola's MPC603e family of Reduced Instruction Set Computer (RISC) microprocessors. The G2 core is a high-performance low-power implementation of the PowerPC superscalar architecture. The MGT5100 G2 core contains:

- 16 KBytes of instruction cache
- 16 KBytes 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) and two Memory Management Units (MMU), one for each cache. The core implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses and integer data types of 8-, 16-, and 32-bits.

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

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

G2 core performance for SPEC95 benchmark integer operations, ranges between 4.4 and 5.1 at 200MHz. In Dhrystone 2.1MIPS, the G2 core is 280MIPS at 200MHz.

1.2.2 SmartComm I/O Subsystem

SmartComm 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 G2 core free for higher level activities. The concurrent operation enables a significant boost in overall systems performance.

SmartComm 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

SmartComm uses internal buffers to prefetch reads and post writes. Bursting is used whenever possible. This optimizes both internal and external bus activity. SmartComm also contains:

- four independent baud rate generators
- four 16-bit or two 32-bit timers

1.2.2.1 Programmable Serial Controllers (PSCs)

The MGT5100 supports three 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.

Both 8-bit and 16-bit data widths are supported. PSCs can be configured to support 1200 baud POTS modem, 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 MGT5100 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 (IR) Support

MGT5100 supports two independent infrared inputs and one output. One input can be programmed to recognize consumer IR signals from remote controls. The other input recognizes the IrDA format. The output supports IR blasting or IrDA TX operations. 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 MGT5100 supports two I^2C channels. Both master and slave interfaces can be controlled directly by the processor or can use the DMA Controller to buffer Tx/Rx data when the I^2C interrupt frequency is high.

1.2.2.6 Serial Peripheral Interface (SPI)

The SPI module allows full-duplex, synchronous, serial communication between the MGT5100 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 Dual Motorola Scalable (MS) Controller Area Network (CAN)

The MGT5100 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, fiber optic cable or an unshielded twisted pair.

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 maskable identifier filters

1.2.4 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 MUXing
- Real-Time Clock (RTC)

1.2.4.1 Chip Selects

MGT5100 integrates the most common system integration interfaces and signals. There are 6 fully programmable external chip selects, which are independent of the SDRAM interface. 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.

1.2.4.2 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 G2 core interrupts outside operating system boundaries, for critical functions such as real-time packet processing.

1.2.4.3 Timers

MGT5100 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.

1.2.4.4 General Purpose Input/Outputs (GPIO)

A total of 56 pins on the MGT5100 can be programmed as GPIOs.

- 8 pins can interrupt the processor.
- 8 pins can support a “WakeUp” capability that lets the MGT5100 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 types depends on the peripheral functionality required. See pin descriptions and I/O port maps below for more information.

1.2.4.5 Functional Pin MUXing

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.

1.2.4.6 Real-Time Clock (RTC)

An RTC is included on the MGT5100. The RTC provides a 2-pin interface to an external 32.768KHz crystal. This allows internal time-of-day/calendar tracking, as well as clock based periodic interrupts.

1.2.5 SDRAM Controller and Interface

The MGT5100 high speed SDRAM Controller supports both standard SDRAM and Dual-Data Rate (DDR) SDRAM devices. It supports up to 256 MBytes per chip select with a 32-bit interface. 64-Mbit, 128-Mbit, 256-Mbit and 512-Mbit memories are also supported.

1.2.6 Multi-Function External Bus

MGT5100 supports a multi-function external local bus to allow connections to PCI and ATA compliant devices, as well as external ROM/SRAM.

MGT5100 integrates a 3.3V, PCI V2.2 compatible external bus controller and interface. This bus is a 32-bit multiplexed address/data bus.

The external 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 MGT5100 recognizes an external 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 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 datar
- 16 bits of address and 16 bits of data.

MGT5100 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.

1.2.7 Power Management

The MGT5100 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 WakeUp capability is supported by CAN, RTC, several GPIOs, the interrupt lines and the IR interface. Therefore, the MGT5100 can be shut down to a low-power standby mode, then re-enabled by one of the WakeUp inputs.

1.2.8 Systems Debug and Test

MGT5100 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.

MGT5100 also supports a JTAG IEEE 1149.1 controller and test access port (TAP).

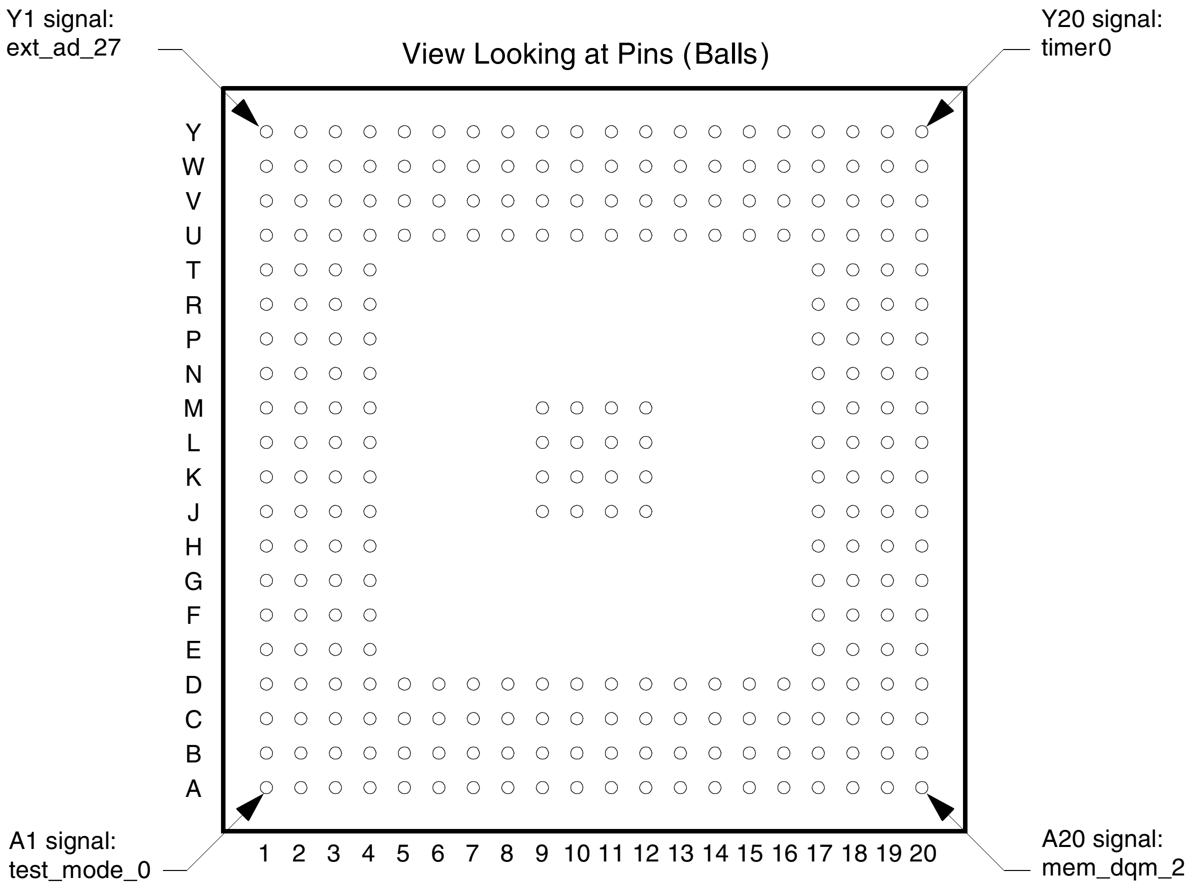
1.2.9 Physical Characteristics

- 1.8V internal, 3.3V external operation (2.5v for DDR interface)
- TTL compatible I/O pins
- 272-pin Plastic Ball Grid Array (PBGA)

SECTION 2
SIGNAL DESCRIPTIONS

2.1 Overview

MGT5100 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 MGT5100 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.

A01 TEST_MODE1	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 IR_TX	A13 PORRESET	A14 SRESET	A15 SYS_XTAL_IN	A16 MEM_MA[1]	A17 MEM_MBA[1]	A18 MEM_RAS	A19 MEM_WE	A20 MEM_DQM2
B01 TEST_SEL0	B02 TEST_MODE0	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 IR_RX	B13 HRESET	B14 SYS_PLL_AVDD	B15 SYS_PLL_TPA	B16 MEM_MA[2]	B17 MEM_MA[10]	B18 MEM_CS	B19 MEM_CAS	B20 MEM_MA[4]
C01 RTC_XTAL_OUT	C02 RTC_XTAL_IN	C03 TEST_SEL1	C04 PSC3_9	C05 PSC3_6	C06 PSC3_3	C07 PSC3_0	C08 HARPO_PLL_AVDD	C09 PSC2_0	C10 PSC1_2	C11 IRDA_RX	C12 GPIO_WKUP7	C13 IR_USB_CLK	C14 SYS_PLL_AVSS	C15 GPIO_WKUP6	C16 MEM_MA[3]	C17 MEM_MA[0]	C18 MEM_MBA0	C19 MEM_MA[5]	C20 MEM_MA[6]
D01 TIMER4	D02 TIMER3	D03 TIMER2	D04 VSS	D05 VDD_CORE	D06 VDD_IO	D07 VDD_CORE	D08 (no connection)	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 TIMER7	E02 TIMER6	E03 TIMER5	E04 VDD_IO													E17 VDD_MEM_IO	E18 MEM_MDQ[16]	E19 MEM_MA[9]	E20 MEM_MA[11]
F01 USB1_7	F02 USB1_8	F03 USB1_9	F04 VDD_IO													F17 VDD_MEM_IO	F18 MEM_MDQ[17]	F19 MEM_MA[12]	F20 MEM_CLK_EN
G01 USB1_3	G02 USB1_4	G03 USB1_5	G04 USB1_6													G17 MEM_MDQ[18]	G18 MEM_MDQ[19]	G19 MEM_MEMCLK	G20 MEM_MEMCLK
H01 USB1_0	H02 USB1_1	H03 USB1_2	H04 VDD_IO													H17 d VDD_MEM_IO	H18 203 MEM_MDQ[20]	H19 204 MEM_DQM1	H20 206 MEM_MDQS[1]
J01 ETH3	J02 ETH4	J03 ETH10	J04 ETH17					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 ETH0	K02 ETH1	K03 ETH2	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 ETH9	L02 ETH16	L03 ETH5	L04 ETH11					L09 VSS	L10 VSS	L1 VSS	L12 VSS					L17 MEM_DQM3	L18 MEM_MDQS[3]	L19 MEM_MDQ[12]	L20 MEM_MDQ[13]
M01 ETH13	M02 ETH12	M03 ETH8	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 ETH7	N02 ETH6	N03 ETH15	N04 ETH14													N17 MEM_MDQ[25]	N18 MEM_MDQ[26]	N19 MEM_DQM0	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 CS2	V16 CS5	V17 ATA_DRQ	V18 TIMER1	V19 I2C1_CLK	V20 I2C2_CLK
W01 EXT_AD[29]	W02 EXT_AD[25]	W03 EXT_AD[23]	W04 EXT_AD[16]	W05 PCI_TRDY	W06 PCI_CBE2	W07 PCI_DEVSEL	W08 PCI_SERR	W09 EXT_AD[14]	W10 PCI_CBE0	W11 EXT_AD[8]	W12 EXT_AD[5]	W13 EXT_AD[1]	W14 CS0	W15 CS3	W16 LP_RWB	W17 ATA_IOW	W18 ATA_IOCHRDY	W19 I2C1_IO	W20 I2C2_IO
Y01 EXT_AD[27]	Y02 PCI_CBE3	Y03 EXT_AD[21]	Y04 EXT_AD[19]	Y05 EXT_AD[17]	Y06 PCI_IRDY	Y07 PCI_PERR	Y08 PCI_CBE1	Y09 EXT_AD[12]	Y10 EXT_AD[10]	Y11 EXT_AD[7]	Y12 EXT_AD[3]	Y13 LP_TS	Y14 CS1	Y15 CS4	Y16 ATA_ISOLATION	Y17 ATA_IOR	Y18 ATA_DACK	Y19 ATA_INTRQ	Y20 TIMER0

Key for IO Balls:

A6	<– Ball
PSC3_5	<– Signal Name

Key for PWR/GND Balls:

VSS	Core and IO VSS
VDD_CORE	1.8V Core VDD
VDD IO	3.3V IO VDD
VDD_MEM_IO	Memory VDD

Figure 2-2 272-Pin PBGA—Top View

2.2 Pinout Tables

Table 2-1 Signals by Ball/Pin

Ball/Pin	Pin Name
A01	TEST_MODE1
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	IR_TX
A13	$\overline{\text{PORRESET}}$
A14	$\overline{\text{SRESET}}$
A15	SYS_XTAL_IN
A16	MEM_MA[1]
A17	MEM_MBA1
A18	$\overline{\text{MEM\_RAS}}$
A19	$\overline{\text{MEM\_WE}}$
A20	MEM_DQM2
B01	TEST_SEL0
B02	TEST_MODE0
B03	$\overline{\text{JTAG\_TRST}}$
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	IR_RX
B13	$\overline{\text{HRESET}}$
B14	SYS_PLL_AVDD
B15	SYS_PLL_TPA
B16	MEM_MA[2]
B17	MEM_MA[10]
B18	$\overline{\text{MEM\_CS0}}$
B19	$\overline{\text{MEM\_CAS}}$
B20	MEM_MA[4]
C01	RTC_XTAL_OUT
C02	RTC_XTAL_IN
C03	TEST_SEL1
C04	PSC3_9
C05	PSC3_6
C06	PSC3_3
C07	PSC3_0
C08	HARPO_PLL_AVDD
C09	PSC2_0
C10	PSC1_2
C11	IRDA_RX
C12	GPIO_WKUP7
C13	IR_USB_CLK
C14	SYS_PLL_AVSS
C15	GPIO_WKUP6
C16	MEM_MA[3]
C17	MEM_MA[0]
C18	MEM_MBA0
C19	MEM_MA[5]
C20	MEM_MA[6]
D01	TIMER4
D02	TIMER3
D03	TIMER2
D08	NC (no connection)
D14	SYS_XTAL_OUT
D18	MEM_MDQS[2]
D19	MEM_MA[7]
D20	MEM_MA[8]
E01	TIMER7
E02	TIMER6
E03	TIMER5
E18	MEM_MDQ[16]
E19	MEM_MA[9]

Table 2-1 Signals by Ball/Pin

Ball/Pin	Pin Name
E20	MEM_MA[11]
F01	USB1_7
F02	USB1_8
F03	USB1_9
F18	MEM_MDQ[17]
F19	MEM_MA[12]
F20	MEM_CLK_EN
G01	USB1_3
G02	USB1_4
G03	USB1_5
G04	USB1_6
G17	MEM_MDQ[18]
G18	MEM_MDQ[19]
G19	MEM_MEMCLK
G20	$\overline{\text{MEM\_MEMCLK}}$
H01	USB1_0
H02	USB1_1
H03	USB1_2
H18	MEM_MDQ[20]
H19	MEM_DQM1
H20	MEM_MDQS[1]
J01	ETH3
J02	ETH4
J03	ETH10
J04	ETH17
J17	MEM_MDQ[22]
J18	MEM_MDQ[21]
J19	MEM_MDQ[8]
J20	MEM_MDQ[9]
K01	ETH0
K02	ETH1
K03	ETH2
K18	MEM_MDQ[23]
K19	MEM_MDQ[10]
K20	MEM_MDQ[11]
L01	ETH9
L02	ETH16

Table 2-1 Signals by Ball/Pin

Ball/Pin	Pin Name
L03	ETH5
L04	ETH11
L17	MEM_DQM3
L18	MEM_MDQS[3]
L19	MEM_MDQ[12]
L20	MEM_MDQ[13]
M01	ETH13
M02	ETH12
M03	ETH8
M18	MEM_MDQ[24]
M19	MEM_MDQ[14]
M20	MEM_MDQ[15]
N01	ETH7
N02	ETH6
N03	ETH15
N04	ETH14
N17	MEM_MDQ[25]
N18	MEM_MDQ[26]
N19	MEM_DQM0
N20	MEM_MDQS[0]
P01	IRQ1
P02	IRQ2
P03	IRQ0
P18	MEM_MDQ[27]
P19	MEM_MDQ[7]
P20	MEM_MDQ[6]
R01	IRQ3
R02	$\overline{\text{PCI\_RESET}}$
R03	EXT_AD[30]
R04	$\overline{\text{PCI\_GNT}}$
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]

Table 2-1 Signals by Ball/Pin

Ball/Pin	Pin Name
T18	MEM_MDQ[30]
T19	MEM_MDQ[3]
T20	MEM_MDQ[2]
U01	$\overline{\text{PCI\_REQ}}$
U02	PCI_IDSEL
U03	EXT_AD[24]
U08	EXT_AD[15]
U11	EXT_AD[6]
U14	LP_ACK
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	$\overline{\text{PCI\_FRAME}}$
V06	$\overline{\text{PCI\_STOP}}$
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	$\overline{\text{LP\_ALE}}$
V15	$\overline{\text{CS2}}$
V16	$\overline{\text{CS5}}$
V17	ATA_DRQ
V18	TIMER1
V19	I2C1_CLK
V20	I2C2_CLK
W01	EXT_AD[29]
W02	EXT_AD[25]
W03	EXT_AD[23]
W04	EXT_AD[16]
W05	$\overline{\text{PCI\_TRDY}}$
W06	$\overline{\text{PCI\_CBE2}}$
W07	$\overline{\text{PCI\_DEVSEL}}$
W08	$\overline{\text{PCI\_SERR}}$
W09	EXT_AD[14]
W10	$\overline{\text{PCI\_CBE0}}$
W11	EXT_AD[8]
W12	EXT_AD[5]
W13	EXT_AD[1]
W14	$\overline{\text{CS0}}$
W15	$\overline{\text{CS3}}$
W16	LP_RWB
W17	$\overline{\text{ATA\_IOW}}$
W18	ATA_IOCHRDY
W19	I2C1_IO
W20	I2C2_IO
Y01	EXT_AD[27]
Y02	$\overline{\text{PCI\_CBE3}}$
Y03	EXT_AD[21]
Y04	EXT_AD[19]
Y05	EXT_AD[17]
Y06	$\overline{\text{PCI\_IRDY}}$
Y07	$\overline{\text{PCI\_PERR}}$
Y08	$\overline{\text{PCI\_CBE1}}$
Y09	EXT_AD[12]
Y10	EXT_AD[10]
Y11	EXT_AD[7]
Y12	EXT_AD[3]
Y13	$\overline{\text{LP\_TS}}$
Y14	$\overline{\text{CS1}}$
Y15	$\overline{\text{CS4}}$
Y16	ATA_ISOLATION
Y17	$\overline{\text{ATA\_IOR}}$
Y18	$\overline{\text{ATA\_DACK}}$
Y19	ATA_INTRQ
Y20	TIMER0

Table 2-2 Signals by Signal Name

Signal Name	Ball/Pin
$\overline{\text{ATA\_DACK}}$	Y18
ATA_DRQ	V17
ATA_INTRQ	Y19
ATA_IOCHRDY	W18
$\overline{\text{ATA\_IOR}}$	Y17
$\overline{\text{ATA\_IOW}}$	W17
ATA_ISOLATION	Y16
$\overline{\text{CS0}}$	W14
$\overline{\text{CS1}}$	Y14
$\overline{\text{CS2}}$	V15
$\overline{\text{CS3}}$	W15
$\overline{\text{CS4}}$	Y15
$\overline{\text{CS5}}$	V16
ETH0	K01
ETH1	K02
ETH2	K03
ETH3	J01
ETH4	J02
ETH5	L03
ETH6	N02
ETH7	N01
ETH8	M03
ETH9	L01
ETH10	J03
ETH11	L04
ETH12	M02
ETH13	M01
ETH14	N04
ETH15	N03
ETH16	L02
ETH17	J04
EXT_AD[0]	V13
EXT_AD[1]	W13
EXT_AD[2]	V12
EXT_AD[3]	Y12
EXT_AD[4]	V11
EXT_AD[5]	W12
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_WKUP6	C15
GPIO_WKUP7	C12
HARPO_PLL_AVDD	C08
HARPO_PLL_AVSS	NC (no connection)
$\overline{\text{HRESET}}$	B13
I2C1_CLK	V19
I2C1_IO	W19
I2C2_CLK	V20
I2C2_IO	W20
IR_RX	B12
IR_TX	A12

Table 2-2 Signals by Signal Name

Table 2-2 Signals by Signal Name

Signal Name	Ball/Pin
IR_USB_CLK	C13
IRDA_RX	C11
IRQ0	P03
IRQ1	P01
IRQ2	P02
IRQ3	R01
JTAG_TCK	B04
JTAG_TDI	A03
JTAG_TDO	A02
JTAG_TMS	A04
$\overline{\text{JTAG\_TRST}}$	B03
LP_ACK	U14
$\overline{\text{LP\_ALE}}$	V14
LP_RWB	W16
$\overline{\text{LP\_TS}}$	Y13
$\overline{\text{MEM\_CAS}}$	B19
MEM_CLK_EN	F20
$\overline{\text{MEM\_CS0}}$	B18
MEM_DQM0	N19
MEM_DQM1	H19
MEM_DQM2	A20
MEM_DQM3	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
MEM_MA[10]	B17
MEM_MA[11]	E20
MEM_MA[12]	F19
MEM_MBA0	C18
MEM_MBA1	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
MEM_MDQS[0]	N20
MEM_MDQS[1]	H20
MEM_MDQS[2]	D18
MEM_MDQS[3]	L18
MEM_MEMCLK	G19

Table 2-2 Signals by Signal Name

Signal Name	Ball/Pin
$\overline{\text{MEM\_MEMCLK}}$	G20
$\overline{\text{MEM\_RAS}}$	A18
$\overline{\text{MEM\_WE}}$	A19
$\overline{\text{PCI\_CBE0}}$	W10
$\overline{\text{PCI\_CBE1}}$	Y08
$\overline{\text{PCI\_CBE2}}$	W06
$\overline{\text{PCI\_CBE3}}$	Y02
PCI_CLOCK	T01
$\overline{\text{PCI\_DEVSEL}}$	W07
$\overline{\text{PCI\_FRAME}}$	V05
$\overline{\text{PCI\_GNT}}$	R04
PCI_IDSEL	U02
$\overline{\text{PCI\_IRDY}}$	Y06
PCI_PAR	V07
$\overline{\text{PCI\_PERR}}$	Y07
$\overline{\text{PCI\_REQ}}$	U01
$\overline{\text{PCI\_RESET}}$	R02
$\overline{\text{PCI\_SERR}}$	W08
$\overline{\text{PCI\_STOP}}$	V06
$\overline{\text{PCI\_TRDY}}$	W05
$\overline{\text{PORRESET}}$	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
PSC2_4	A08
PSC3_0	C07
PSC3_1	B07
PSC3_2	A07
PSC3_3	C06
PSC3_4	B06
PSC3_5	A06
PSC3_6	C05
PSC3_7	B05
PSC3_8	A05
PSC3_9	C04
RTC_XTAL_IN	C02
RTC_XTAL_OUT	C01
$\overline{\text{SRESET}}$	A14
SYS_PLL_AVDD	B14
SYS_PLL_AVSS	C14
SYS_PLL_TPA	B15
SYS_XTAL_IN	A15
SYS_XTAL_OUT	D14
TEST_MODE0	B02
TEST_MODE1	A01
TEST_SEL0	B01
TEST_SEL1	C03
TIMER0	Y20
TIMER1	V18
TIMER2	D03
TIMER3	D02
TIMER4	D01
TIMER5	E03
TIMER6	E02
TIMER7	E01
USB1_0	H01
USB1_1	H02
USB1_2	H03
USB1_3	G01
USB1_4	G02
USB1_5	G03
USB1_6	G04
USB1_7	F01
USB1_8	F02
USB1_9	F03

2.2.1 Functional Signal List with Pin/Pad Count

Table 2-3 contains a listing of MGT5100 I/O signals in functional groups. I/Os that can have multiple functions are grouped together and each function is described separately. The first name in these groupings is the actual pin name used in the signal/ball number reference above. The pin name is not necessarily the default function of the pin after reset. Throughout this document, the default after reset is designated by the symbol “⊕“. The "Reset Value" column indicates the electrical state of the signal during reset.

NOTE: The following note applies to Table 2-3.

1. The external local bus default after a reset is LocalPlus. The LocalPlus configuration depends on the reset configuration pins sampled during reset. See the LocalPlus section for more information on pin definitions.

Table 2-3 Functional Signal List with Pin/Pad Count

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
SDRAM/Memory Bus—62 Pins Total					
$\overline{\text{MEM\_RAS}}$	O	1	15mA	0	SDRAM: Row Address Select
$\overline{\text{MEM\_CAS}}$	O	1	15mA	0	SDRAM: Column Address Select
$\overline{\text{MEM\_WE}}$	O	1	15mA	0	SDRAM: Write Enable
$\overline{\text{MEM\_CS0}}$	O	1	15mA	1	SDRAM: Chip Select, MEM_CS0 always available
MEM_CS1	I/O	1	15mA	0	SDRAM: Second chip select optionally available Not pinned out in MGT5100
MEM_CLK_EN	O	1	15mA	0	SDRAM: Clock Enable
MEM_MEMCLK	O	1	20mA	0	SDRAM: Memory Clock
$\overline{\text{MEM\_MEMCLK}}$	O	1	20mA	1	SDRAM: Inverted MEMCLK, required for DDR SDRAM
MEM_MBA[1:0]	O	2	15mA	0	SDRAM: Memory Chip Bank Address
MEM_MDQS[3:0]	I/O	4	15mA	hi-z	SDRAM: Bidirectional Data Strobe, required for DDR SDRAM
MEM_DQM[3:0]	O	4	15mA	1	SDRAM: Data Mask, Output enable
MEM_MA[12:0]	O	13	15mA	0	SDRAM: Memory Address
MEM_MDQ[31:0]	I/O	32	15mA	hi-z	SDRAM: Data
MEM_RDCLK	I/O	0	15mA	clk	Memory read clock I/O delay element. NOT PINNED OUT ON PKG

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
External Bus—PCI, ATA, LocalPlus—32 Pins Total Shared					
EXT_AD[31]	—	1	20mA	hi-z	External Address/Data Bus bit 31
PCI_AD31	O,I/O				PCI bus: MUXed 32-bit address and 8-, 16-, and 32-bit data
LPdata15 or LPadd7	I/O				LP: Data or address for non-MUXed peripheral. ⊕ See Note 1 above.
LP_AD31	I/O				LP: Address/data signal for MUXed peripheral connection. ⊕ See Note 1 above.
EXT_AD[30:19]	—	12	20mA	hi-z	External Address/Data Bus bits 19–30
PCI_AD[30:19]	O,I/O				PCI bus: MUXed 32-bit address & 8-, 16-, and 32-bit data
LPdata[14:3] or LPdata[6:0], LPadd[23:19]	I/O				LP: Data and/or address for non-MUXed peripheral. ⊕ See Note 1 above.
LP_AD[30:19]	I/O				LP: Address/data signal for MUXed peripheral connection. ⊕ See Note 1 above.
EXT_AD[18:16]	—	3	20mA	hi-z	External Address/Data Bus bits 16–18
PCI_AD[18:16]	O,I/O				PCI bus: MUXed 32-bit address and 8-, 16-, and 32-bit data
ATA_SA[2:0]	O				ATA: Address 0–2
LPdata[2:0] or LPadd[18:16]	I/O				LP: Data or address for non-MUXed peripheral. ⊕ See Note 1 above.
LP_AD[18:16]	I/O				LP: Address/data signal for MUXed peripheral connection. ⊕ See Note 1 above.
EXT_AD[15:8]	—	8	20mA	hi-z	External Address/Data Bus bits 8–15
PCI_AD[15:8]	O,I/O				PCI bus: MUXed 32-bit address and 8-, 16-, and 32-bit data
ATA_DATA[15:8]	I/O				ATA: valid for 16-bit data transfer only
LPadd[15:8] or LPdata[15:8]	I/O				LP: Addressor data for non-MUXed peripheral. ⊕ See Note 1 above.
LP_AD[15:8]					LP: Address/data signal for MUXed peripheral connection. ⊕ See Note 1 above.

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
EXT_AD[7:0]	—	8	20mA	hi-z	External Address/Data Bus bit 0,1,2
PCI_AD[7:0]	I/O				PCI bus: MUXed 32-bit address and 8-, 16-, and 32-bit data
ATA_DATA[7:0]	I/O				ATA: 8-, 16-bit data
LPADD[7:0] or LPADD[7:0]	I/O				LP: Address for non-MUXed peripheral. ⊕ See Note 1 above.
LP_AD[7:0]	I/O				LP: Address/data signal for MUXed peripheral connection. ⊕ See Note 1 above.
PCI Dedicated Signals—17 Pins Total					
PCI_PAR	I/O	1		1	PCI bus: Parity
PCI_CBE0	I/O	4	20mA	1	PCI bus Command/Byte enable 0–3
PCI_CBE1					
PCI_CBE2					
PCI_CBE3					
PCI_TRDY	I/O	1	20mA	1	PCI bus: Target (slave) asserts (0) to signify ready
PCI_IRDY	I/O	1	20mA	1	PCI bus: Initiator (host) ready
PCI_STOP	I/O	1	20mA	1	PCI bus: Transaction stop
PCI_DEVSEL	I/O	1	20mA	1	PCI bus: Device select
PCI_FRAME	I/O	1	20mA	1	PCI bus: Frame start
PCI_SERR	OD	1	20mA	hi-z	PCI bus: System error (open drain with internal pull up). Not active
PCI_PERR	I/O	1	20mA	1	PCI bus: Parity error
PCI_IDSEL	I	1		1	PCI bus: Initial device select
PCI_REQ	I	1		1	PCI bus: Bus request
PCI_GNT	O	1	5mA	1	PCI bus: Bus grant
PCI_CLOCK	O	1	20mA	clk	PCI bus: Clock
PCI_RESET	O	1		0	PCI bus: Reset Output, open drain
ATA Dedicated Signals—7 Pins Total					
ATA_DRQ	I	1	—	0	ATA: DMA Request
ATA_DACK	O	1	5mA	1	ATA: Active low DMA acknowledge
RST_CFG0	I				Reset Config input 0
ATA_IOR	O	1	5mA	1	ATA_DIOR
RST_CFG1	I				Reset Config input 1
ATA_IOW	O	1	5mA	1	ATA_DIOW
RST_CFG2	I				Reset Config input 2

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
ATA_IOCHDRY	I	1		1	ATA: Negated (0) to extend host transfer
ATA_INTRQ	I	1		0	ATA: Interrupt Request
ATA_ISOLATION	O	1	5mA	1	ATA: on-board transceiver control (Isolate ATA signals from ATA/PCI shared bus)
LocalPlus Dedicated Signals—4 Pins Total					
LP_RWB	O	1	10mA	1	LP R/W signal for STD peripheral connection
RST_CFG3	I				Reset Config input 3
LP_ALE	O	1	10mA	1	LP Address Latch Enable for MUXed transactions
RST_CFG4	I				Reset Config input 4
LP_ACK	I	1		1	LP ACK signal for special peripheral connection
LP_TS	O	1	10mA	1	LP Transfer Start, for support of interfaces
RST_CFG5	I				Reset Config input 5
Systems Integration Unit (SIU)—12 Pins Total					
CS0	O	1	5mA	1	LP: External device select (Boot ROM)
CS[1:3]	O	3	5mA	1	LP: configurable chip select
CS4	O	1	5mA	1	LP: External device select
ATA_CS0	O				ATA: Chip-select 0
CS5	O	1	5mA	1	LocalPlus: External device select
ATA_CS1	O				ATA: Chip-select 1
IRQ0	I	1		0	INT: Critical interrupt or External interrupt
IRQ1	I	1		0	INT: External interrupt
IRQ2	I	1		0	INT: External interrupt
IRQ3	I	1		0	INT: External interrupt
RTC_XTAL_IN	I	1		clk	RTC: 32.768KHz watch crystal input, or external Clock input
RTC_XTAL_OUT	O	1		clk	RTC: 32.768KHz watch crystal output
PSC1—CODEC1/AC971/UART1—5 Pins Total (Figure 2-4 shows details)					
PSC1_0	—	1		0	
UART1_TXD	O				UART1: Transmit Data
CODEC1_TXD	O				CODEC1: Transmit Data
AC97_1_SDATA_OUT	O				AC97: Transmit Data
⊕ GPIO_PSC1_0	I/O				simple GPIO (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
PSC1_1	—	1		0	
UART1_RXD	I				UART1: Receive Data
CODEC1_RXD	I				CODEC1: Receive Data
AC97_1_SDATA_IN	I				AC91: Receive Data
⊕ GPIO_PSC1_1	I/O				simple GPIO (default at reset)
PSC1_2	—	1		0	
UART1_$\overline{\text{RTS}}$	O				UART1: Ready to Send
AC97_1_SYNC	O				AC971: Frame Sync
⊕ GPIO_PSC1_2	I/O				simple GPIO (default at reset)
PSC1_3	—	1		0	
UART1_$\overline{\text{CTS}}$	I				UART1: Clear to Send
CODEC1_CLK	I				CODEC1: bit-clock
AC97_1_BITCLK	I				AC971: bit-clock
⊕ GPIO_PSC1_3	I/O				simple GPIO (default at reset)
PSC1_4	—	1		0	
UART1_CD	I				UART1: Carrier Detect (for UARTe only)
CODEC1_FRAME	I				CODEC1: Frame Sync
$\overline{\text{AC97\_1\_RES}}$	O				AC97: Reset
⊕ GPIO_WKUP0	I/O				simple GPIO with WakUp (default at reset)
PSC2—CODEC2/AC972/UART2—5 Pins Total (Figure 2-5 shows details)					
PSC2_0	—	1		1	
UART2_TXD	O				UART2: Transmit Data
CODEC2_TXD	O				CODEC2: Transmit Data
AC97_2_SDATA_OUT	O				AC972: Transmit Data
CAN1_TXD	O				CAN: Transmit pin for module 1
⊕ GPIO_PSC2_0	I/O				simple GPIO (default at reset)
PSC2_1	—	1		1	
UART2_RXD	I				UART2: Receive Data
CODEC2_RXD	I				CODEC2: Receive Data
AC97_2_SDATA_IN	I				AC972: Receive Data
CAN1_RX	I				CAN: Receive pin for module 1
⊕ GPIO_PSC2_1	I/O				simple GPIO (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
PSC2_2	—	1		1	
UART2_RTS	O				UART2: Ready to Send
AC97_2_SYNC	O				AC972: Frame Sync
CAN2_TX	O				CAN: Transmit pin for module 2
⊕ GPIO_PSC2_2	I/O				simple GPIO (default at reset)
PSC2_3	—	1		1	
UART2_CTS	I				UART2: Clear to Send
CODEC2_CLK	I				CODEC2: bit-clock
AC97_2_BITCLK	I				AC97: bit-clock
CAN2_RX	I				CAN: Receive pin for module 2
⊕ GPIO_PSC2_3	I/O				simple GPIO (default at reset)
PSC2_4	—	1		hi-z	
UART2_CD	I				UART2: Carrier Detect (for UARTe only)
CODEC2_FRAME	I				CODEC2: Frame Sync
AC97_1_RES	O				AC97: Reset
⊕ GPIO_WKUP1	I/O				simple GPIO with WakUp (default at reset)
PSC3—USB2/SPI/CODEC3/UART3/modem/rs232—10 Pins Total (Figure 2-6 shows details)					
PSC3_0	—	1		hi-z	
USB2_OE	O				USB2: Output Enable
CODEC3_TXD	O				CODEC3: Transmit Data
UART3_TXD	O				UART3: Transmit Data
⊕ GPIO_PSC3_0	I/O				simple GPIO (default at reset)
PSC3_1	—	1		hi-z	
USB2_TXN	O				USB2: Transmit Data Negative
CODEC3_RXD	I				CODEC3: Receive Data
UART3_RXD	I				UART3: Receive Data
⊕ GPIO_PSC3_1	I/O				simple GPIO (default at reset)
PSC3_2	—	1		0	
USB2_TXP	O				USB2: Transmit Data Positive
CODEC3_CLK	I				CODEC3: bit-clock
UART3_RTS	O				UART3: RTS
⊕ GPIO_PSC3_2	I/O				simple GPIO (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
PSC3_3	—	1		0	
USB2_RXD	I				USB2: USBRXD
CODEC3_FRAME	I				CODEC3: Frame Sync
UART3_CTS	I				UART3: CTS
⊕ GPIO_PSC3_3	I/O				simple GPIO (default at reset)
PSC3_4	—	1		hi-z	
USB2_RXP	I				USB2: Receive Data Positive
UART3_CD	I				UART3: Carrier Detect
⊕ GPIO_SINT0	I/O				simple GPIO with Interrupt (default at reset)
PSC3_5	—	1		hi-z	
USB2_RXN	I				USB2: Receive Data Negative
⊕ GPIO_SINT1	I/O				simple GPIO with Interrupt (default at reset)
PSC3_6	—	1		1	
USB2_PORTPWR	O				USB2: Port Power Indicator (1=On, 0=Off)
SPI_MOSI	I/O				SPI: MOSI
⊕ GPIO_PSC3_4	I/O				simple GPIO (default at reset)
PSC3_7	—	1		1	
USB2_SPEED	O				USB2: Speed
SPI_MISO	I/O				SPI: MISO
⊕ GPIO_PSC3_5	I/O				simple GPIO (default at reset)
PSC3_8	—	1		1	
USB2_SUSPEND	O				USB2: Suspend
SPI_SS	I/O				SPI: Slave Select
⊕ GPIO_SINT2	I/O				simple GPIO with Interrupt (default at reset)
PSC3_9	—	1		1	
USB2_OVRCURRENT	I				USB2: Over Current
SPI_CLK	I/O				SPI: Clock
⊕ GPIO_WKUP2	I/O				simple GPIO with WakUp (default at reset)
USB1—Primary USB Port—10 Pins Total (Figure 2-9 shows details)					
USB1_0	—	1		hi-z	
USB1_OE	O				USB1: Output enable for TX
⊕ GPIO_USB0	I/O				simple GPIO (default at reset)
USB1_1	—	1		0	
USB1_TXN	O				USB1: Transmit Data Negative
⊕ RST_CFG6	I				Reset Config input 6 (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
USB1_2	—	1		1	
USB1_TXP	O				USB1: Transmit Data Positive
⊕ RST_CFG7	I				Reset Config input 7 (default at reset)
USB1_3	—	1		hi-z	
USB1_RXD	I				USB1: Receive Data Differential
USB1_4	—	1		hi-z	
USB1_RXP	I				USB1: Receive Data Positive
USB1_5	—	1		hi-z	
USB1_RXN	I				USB1: Receive Data Negative
USB1_6	—	1		hi-z	
USB1_PORTPWR	O				USB1: Port Power Indicator (1=ON, 0=OFF)
⊕ GPIO_USB1	I/O				simple GPIO (default at reset)
USB1_7	—	1		hi-z	
USB1_SPEED	O				USB1: Speed
⊕ GPIO_USB2	I/O				simple GPIO (default at reset)
USB1_8	—	1		hi-z	
USB1_SUSPEND	O				USB1: Suspend
⊕ GPIO_USB3	I/O				simple GPIO (default at reset)
USB1_9	—	1		hi-z	
USB1_OVRCURRENT	I				USB1: Over Current
⊕ GPIO_SINT3	I/O				GPIO with Interrupt (default at reset)
Ethernet Port—Ethernet/USB2/RST_CFGx—18 Pins Total (Figure 2-7 and Figure 2-8 shows details)					
ETH0	—	1	5mA	0	
ETH_TXEN	O				Ethernet: Transmit enable output
RST_CFG8	I				RESET: Reset Config input 8
⊕ GPIO_ETH0	O				GPIO: Output Only (default at reset)
ETH1	—	1	5mA	0	
ETH_TXD0	O				Ethernet: Transmit data output
RST_CFG9	I				RESET: Reset Config input 9
⊕ GPIO_ETH1	O				GPIO: Output Only (default at reset)
ETH2	—	1	5mA	1	
ETH_TXD1	O				Ethernet: Transmit data output, not used if 7-wire only
RST_CFG10	I				RESET: Reset Config input 10
USB2_TXP	O				USB2: Transmit (positive)
⊕ GPIO_ETH2	O				GPIO: Output Only (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
ETH3	—	1	5mA	1	
ETH_TXD2	O				ETHERNET: TRANSMIT DATA OUTPUT, NOT USED IF 7-WIRE ONLY
RST_CFG11	I				RESET: RESET CONFIG INPUT 11
USB2_PORTPWR	O				USB2: PORT POWER INDICATOR
⊕ GPIO_ETH3	O				GPIO: OUTPUT ONLY (DEFAULT AT RESET)
ETH4	—	1	5mA	0	
ETH_TXD3	O				ETHERNET: TRANSMIT DATA OUTPUT, NOT USED IF 7-WIRE ONLY
RST_CFG12	I				RESET: RESET CONFIG INPUT 12
USB2_SPEED	O				USB2: SPEED
⊕ GPIO_ETH4	O				GPIO: OUTPUT ONLY (DEFAULT AT RESET)
ETH5	—	1	5mA	1	
ETH_TXERR	O				ETHERNET: TRANSMIT ERROR OUTPUT, NOT USED IF 7-WIRE ONLY
RST_CFG13	I				RESET: RESET CONFIG INPUT 13
USB2_SUSPEND	O				USB2: SUSPEND
⊕ GPIO_ETH5	O				GPIO: OUTPUT ONLY (DEFAULT AT RESET)
ETH6	—	1	5mA	1	
ETH_MDC	O				ETHERNET: MANAGEMENT DATA CLOCK OUTPUT, OPTIONAL AND NOT 7-WIRE
USB2_OE	O				USB2: OUTPUT ENABLE FOR TX
RST_CFG14	I				RESET: RESET CONFIG INPUT 14
⊕ GPIO_ETH6	O				GPIO: OUTPUT ONLY (DEFAULT AT RESET)
ETH7	—	1	5mA	hi-z	
ETH_MDIO	I/O				ETHERNET: MANAGEMENT DATA I/O, OPTIONAL AND NOT 7-WIRE
USB2_TXN	O				USB2: TRANSMIT (NEGATIVE)
RST_CFG15	I				RESET: RESET CONFIG INPUT 15
⊕ GPIO_ETH7	O				GPIO: OUTPUT ONLY (DEFAULT AT RESET)
ETH8	—	1	5mA	hi-z	
ETH_RXDV_CD	I				ETHERNET: RECEIVE DATA VALID OR CARRIER DETECT INPUT
⊕ GPIO_ETH10	I/O				GPIO: SIMPLE (DEFAULT AT RESET)
ETH9	—	1	5mA	hi-z	
ETH_RXCLK	I				ETHERNET: RECEIVE CLOCK INPUT
⊕ GPIO_ETH11	I/O				GPIO: SIMPLE (DEFAULT AT RESET)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
ETH10	—	1	5mA	hi-z	
ETH_COL	I				ETHERNET: COLLISION DETECT INPUT
⊕ GPIO_ETH12	I/O				GPIO: SIMPLE (DEFAULT AT RESET)
ETH11	—	1	5mA	hi-z	
ETH_TXCLK	I				ETHERNET: TRANSMIT CLOCK INPUT
⊕ GPIO_ETH13	I/O				GPIO: SIMPLE (DEFAULT AT RESET)
ETH12	—	1	5mA	hi-z	
ETH_RXD0	I				ETHERNET: RECEIVE DATA INPUT
ETH13	—	1	5mA	hi-z	
ETH_RXD1	I				ETHERNET: RECEIVE DATA INPUT, NOT USED IF 7-WIRE ONLY
USB2_RXD	I/O				USB2: RECEIVE (DIFFERENTIAL)
⊕ GPIO_SINT4					GPIO: SIMPLE WITH INTERRUPT (DEFAULT AT RESET)
ETH14	—	1	5mA	hi-z	
ETH_RXD2	I				ETHERNET: RECEIVE DATA INPUT, NOT USED IF 7-WIRE ONLY
USB2_RXP	I				USB2: RECEIVE (POSITIVE)
⊕ GPIO_SINT5	I/O				GPIO: SIMPLE WITH INTERRUPT (DEFAULT AT RESET)
ETH15	—	1	5mA	hi-z	
ETH_RXD3	I				ETHERNET: RECEIVE DATA INPUT, NOT USED IF 7-WIRE ONLY
USB2_RXN	I				USB2: RECEIVE (NEGATIVE)
⊕ GPIO_SINT6	I/O				GPIO: SIMPLE WITH INTERRUPT (DEFAULT AT RESET)
ETH16	—	1	5mA	hi-z	
ETH_RXERR	I				ETHERNET: RECEIVE ERROR INPUT, NOT USED IF 7-WIRE ONLY
USB2_OVERCURRENT	I				USB2: OVER CURRENT
⊕ GPIO_SINT7	I/O				GPIO: SIMPLE WITH INTERRUPT (DEFAULT AT RESET)
ETH17		1	5mA	hi-z	
ETH_CRS	I				ETHERNET: CARRIER SENSE INPUT, NOT USED IF 7-WIRE ONLY
⊕ GPIO _WKUP3	I/O				GPIO: WAKUP CAPABLE (DEFAULT AT RESET)
Infrared—IR/IrDA—4 Pins Total (Figure 2-12 shows details)					
IR_RX	—	1	5mA	hi-z	
IR_RX	I				IR: RECEIVE DATA FOR CONSUMER IR CONTROLLER
⊕ GPIO_WKUP4	I/O				WAKUP GPIO (DEFAULT AT RESET)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
IRDA_RX	—	1	5mA	hi-z	
IRDA_RX	I				IrDA: Receive Data for IrDA Controller
⊕ GPIO_WKUP5	I/O				WakUp GPIO (default at reset)
IR_TX	—	1	5mA	hi-z	
IRDA_TX	O				IrDA: Transmit Data from IrDA Controller
IR_TX	O				IR: Transmit Data (Blaster) from consumer IR Controller
⊕ GPIO_IRDA0	I/O				simple GPIO (default at reset)
IR_USB_CLK	I	1	5mA	hi-z	48MHz IR and/or USB clock input
⊕ GPIO_IRDA1	I/O				simple GPIO (default at reset)
TIMERS—IC/OC/PWM—8 Pins Total (Figure 2-10 shows details)					
TIMER0	I/O	1	5mA	1	Full function Timer
ATA_CS0	O				ATA: Chip Select 0
CAN2_TX	O				CAN: Transmit pin for module 2
⊕ GPIO_TMR0	I/O				simple GPIO (default at reset)
TIMER1	I/O	1	5mA	1	Full function Timer
ATA_CS1	O				ATA: Chip select 1
CAN2_RX	I				CAN: Receive pin for module 2
⊕ GPIO_TMR1	I/O				simple GPIO (default at reset)
TIMER2	I/O	1	5mA	1	Full function Timer
SPI_MOSI	I/O				SPI: Master out, Slave in data pin
⊕ GPIO_TMR2	I/O				simple GPIO (default at reset)
TIMER3	I/O	1	5mA	1	Full function Timer
SPI_MISO	I/O				SPI: Master in, Slave out data pin
⊕ GPIO_TMR3	I/O				simple GPIO (default at reset)
TIMER4	I/O	1	5mA	1	Full function Timer
SPI_SS	I/O				SPI: Slave Select pin
⊕ GPIO_TMR4	I/O				simple GPIO (default at reset)
TIMER5	I/O	1	5mA	1	Full function Timer
SPI_CLK	I/O				SPI: Clock pin
⊕ GPIO_TMR5	I/O				simple GPIO (default at reset)
TIMER6	I/O	1	5mA	0	Full function Timer/WakUp capable (as Input Capture)
⊕ GPIO_TMR6	I/O				simple GPIO (default at reset)

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
TIMER7	I/O	1	5mA	0	Full function Timer/WakUp capable (as Input Capture)
⊕ GPIO_TMR7	I/O				simple GPIO (default at reset)
Dedicated GPIO—2 Pins Total					
⊕ GPIO_WKUP6	I/O	1	15mA	0	WakUp GPIO. Async, usable during deep sleep (default at reset)
GPIO_WKUP7	I/O	1	5mA	hi-z	WakUp GPIO. Async, usable during deep sleep
I^{2}C Controller1—2 Pins Total (Figure 2-11 shows details)					
⊕ I2C1_CLK	I/OD	1		1	I^{2}C: Clock—Schmitt Trigger input (default at reset)
CAN1_TX	O				CAN: Transmit pin for module 1
⊕ I2C1_IO	I/OD	1		1	I^{2}C: Data in/out. Open Drain, Schmitt Trigger input (default at reset)
CAN1_RX	I				CAN: Receive pin for module 1
I^{2}C Controller2—2 Pins Total (Figure 2-11 shows details)					
⊕ I2C2_CLK	I/O	1		1	I^{2}C: Clock—Schmitt Trigger input (default at reset)
ATA_CS0	O				ATA: Chip Select 0
⊕ I2C2_IO	OD	1		1	I^{2}C: Data in/out. Open Drain, Schmitt Trigger input (default at reset)
ATA_CS1	O				ATA: Chip Select 1
Clock/Reset—6 Pins Total					
PORRESET	I	1		1	RESET: Put microprocessor in Power-On Reset state. Schmitt input
HRESET	OD	1	10mA	1	RESET: Hard reset microprocessor. Schmitt input
SRESET	OD	1	10mA	1	RESET: Soft reset microprocessor. Schmitt input
SYS_XTAL_IN	I	1			APLL: Chip clock crystal APLL: External Clock
SYS_XTAL_OUT	O	1		clk	APLL: Chip clock crystal
SYS_PLL_TPA	AO	1		x	MGT5100 System Test PLL Output (analog output)
JTAG Test Access Port—9 Pins Total					
JTAG_TCK	I	1		0	JTAG: Test clock
DSCK	I				Debug: Debug serial clock
JTAG_TMS	I	1		0	JTAG: Test Mode Select

Table 2-3 Functional Signal List with Pin/Pad Count (continued)

Pin Name (⊕ indicates default)	I/O	# of Pins	Drive (mA)	Reset Value	Description
JTAG_TDI	I	1		0	JTAG: Serial data in
DSDI	I				Debug: Serial data in
JTAG_TDO	O	1	5mA	hi-z	JTAG: Serial data out
DSDO	O				Debug: Serial data out
JTAG_TRST	I	1		0	JTAG: Active low reset
TEST_MODE0	I	1		0	Test Mode. One of four test modes
TEST_MODE1	I	1		0	Test Mode. One of four test modes
TEST_SEL0	I/O	1	5mA	1	Test inputs: Scan_en, PLL_Bypass. CK_STOP output
TEST_SEL1	I	1		0	ENID input in test mode, or CK_STOP output in functional mode
		215			Total Functional Pins
Power—57 Pins Total					
VDD_CORE	P	10			Core: Power (1.8V)
VSS_CORE	P	20			Core Ground
VSS_IO	P				I/O Ground
VDD_IO	P	12			I/O Power (3.3V)
VDD_MEM_IO	P	11			I/O Power (2.5 or 3.3V) for SDRAM interface
AVDD1	P	1			APLL: Power—MGT5100 PLL
AGND1	P	1			APLL: Ground—MGT5100 PLL
AVDD2	P	1			APLL: Power—Harpo PLL
GND	P	1			Not Used. Tie to ground. Use to be Harpo APLL ground.
		272			Total Chip Pin (ball) Count

2.3 Multi-Function Pin Maps

This section gives the options available for multi-function I/O signals. The port maps below can help application designers select an implementable peripheral set and determine available GPIO pins and types for the peripheral set.

This section focuses on the peripheral pin MUXing options, not the Local Bus. Figure 2-3 shows a graphical summary of the peripheral functions available on a particular I/O port.

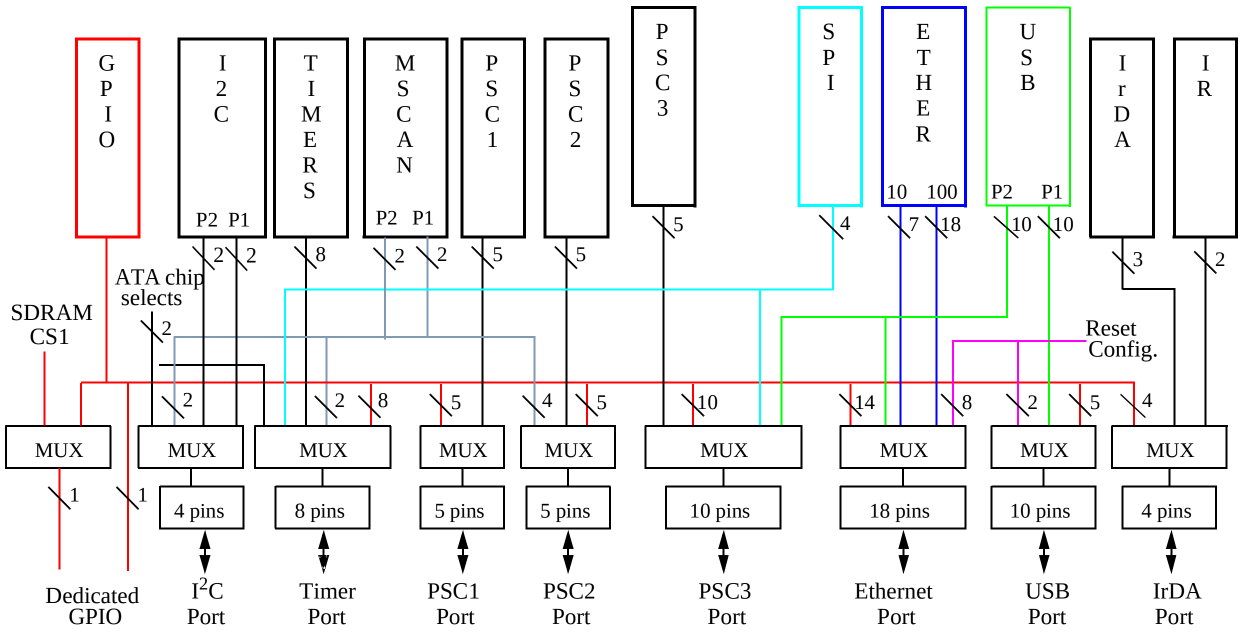
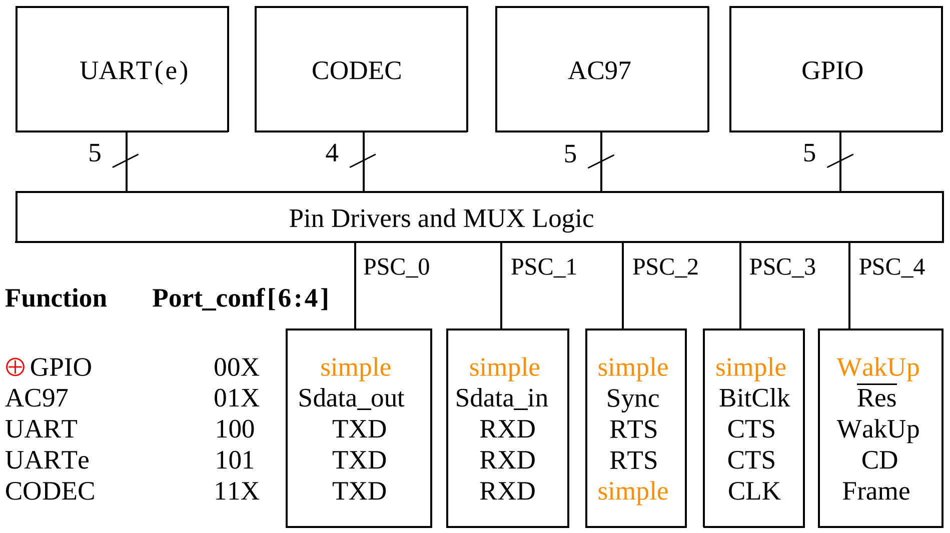


Figure 2-3 PSC Peripheral Multiplexing

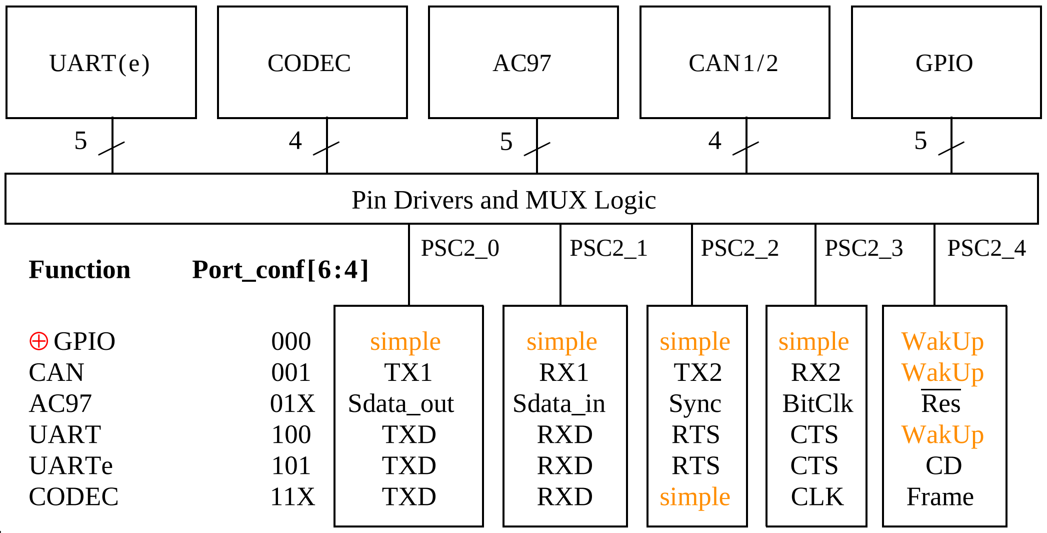


Function	Port_conf[6:4]	PSC_0	PSC_1	PSC_2	PSC_3	PSC_4
⊕ GPIO	00X	simple	simple	simple	simple	WakUp
AC97	01X	Sdata_out	Sdata_in	Sync	BitClk	$\overline{\text{Res}}$
UART	100	TXD	RXD	RTS	CTS	WakUp
UARTe	101	TXD	RXD	RTS	CTS	CD
CODEC	11X	TXD	RXD	simple	CLK	Frame

NOTES:

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 either PSC1 or PSC2, but not both.

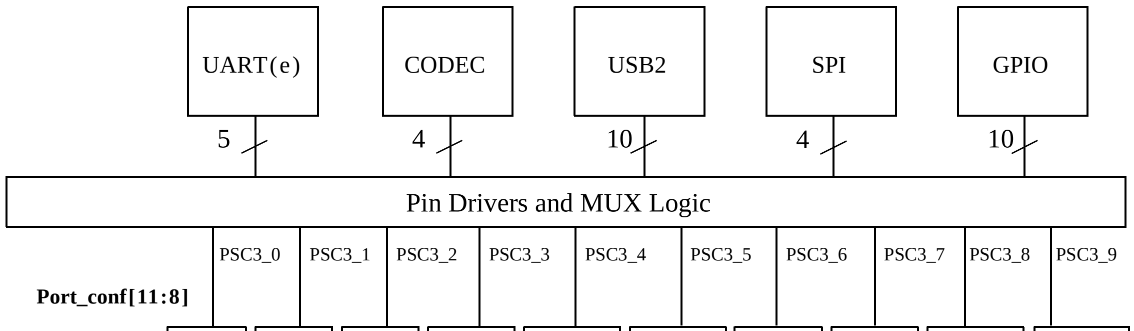
Figure 2-4 PSC1 Port Map—5 Pins


UART(e)
CODEC
AC97
CAN1/2
GPIO
5
4
5
4
5
Pin Drivers and MUX Logic
PSC2_0
PSC2_1
PSC2_2
PSC2_3
PSC2_4
Function
Port_conf[6:4]
⊕GPIO 000 simple simple simple simple WakUp
CAN 001 TX1 RX1 TX2 RX2 WakUp
AC97 01X Sdata_out Sdata_in Sync BitClk Res
UART 100 TXD RXD RTS CTS WakUp
UARTe 101 TXD RXD RTS CTS CD
CODEC 11X TXD RXD simple CLK Frame

NOTES:

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 either PSC1 or PSC2, but not both.
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

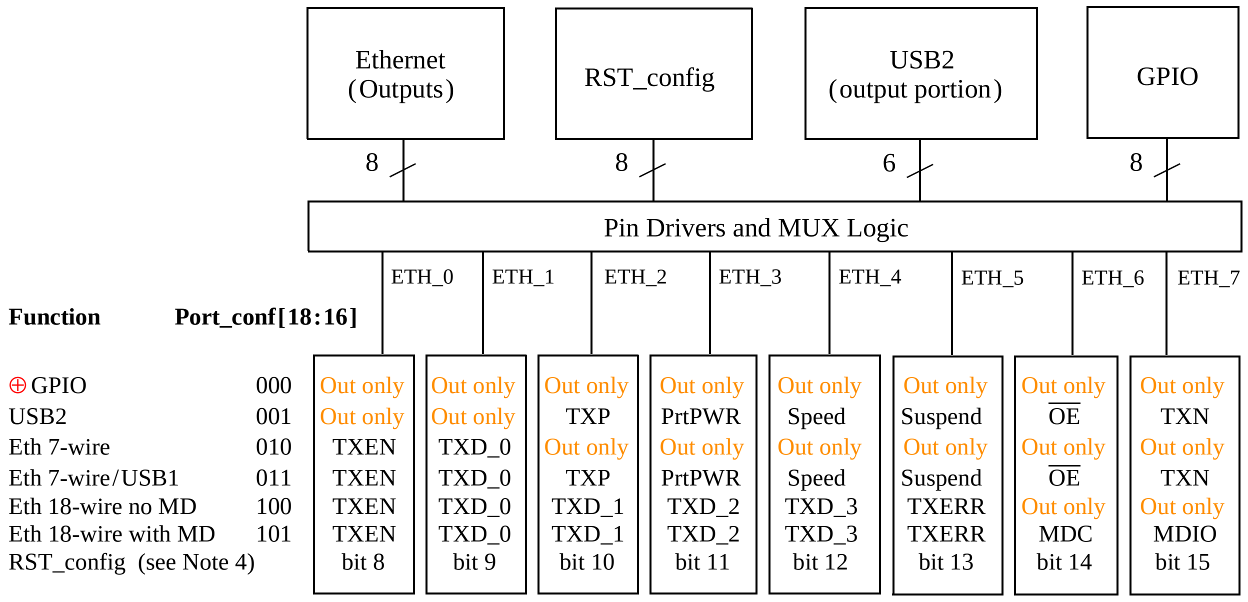


Function	**Port_conf[11:8]**	PSC3_0	PSC3_1	PSC3_2	PSC3_3	PSC3_4	PSC3_5	PSC3_6	PSC3_7	PSC3_8	PSC3_9
⊕ GPIO	0000	simple	simple	simple	simple	Interrupt	Interrupt	simple	simple	Interrupt	WakUp
USB2	0001	$\overline{\text{OE}}$	TXN	TXP	RXD	RXP	RXN	PrtPWR	Speed	Suspend	OvrCrnt
Reserved	0010										
Reserved	0011										
UART	0100	TXD	RXD	RTS	CTS	Interrupt	Interrupt	simple	simple	Interrupt	WakUp
UARTe	0101	TXD	RXD	RTS	CTS	CD	Interrupt	simple	simple	Interrupt	WakUp
CODEC	011X	TXD	RXD	CLK	Frame	Interrupt	Interrupt	simple	simple	Interrupt	WakUp
SPI	100X	simple	simple	simple	simple	Interrupt	Interrupt	MOSI	MISO	SS	CLK
Reserved	1010										
Reserved	1011										
UART/SPI	1100	TXD	RXD	RTS	CTS	Interrupt	Interrupt	MOSI	MISO	SS	CLK
UARTe/SPI	1101	TXD	RXD	RTS	CTS	CD	Interrupt	MOSI	MISO	SS	CLK
CODEC/SPI	111X	TXD	RXD	CLK	Frame	Interrupt	Interrupt	MOSI	MISO	SS	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.

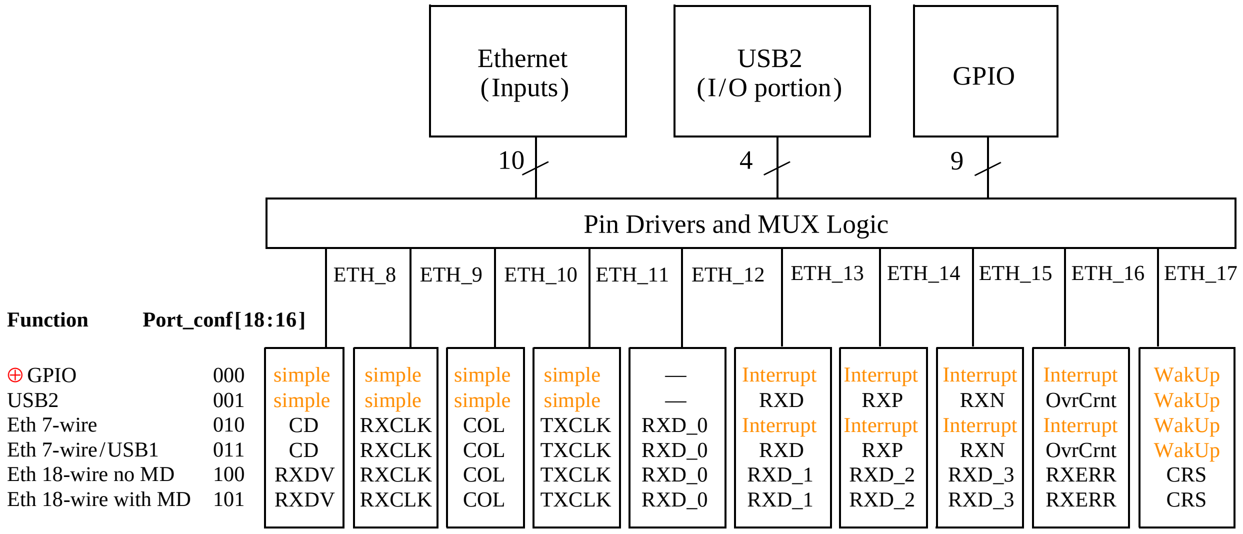
Figure 2-6 PSC3 Port Map—10 Pins



Function	Port_conf[18:16]	ETH_0	ETH_1	ETH_2	ETH_3	ETH_4	ETH_5	ETH_6	ETH_7
⊕GPIO	000	Out only	Out only	Out only	Out only	Out only	Out only	Out only	Out only
USB2	001	Out only	Out only	TXP	PrtPWR	Speed	Suspend	$\overline{\text{OE}}$	TXN
Eth 7-wire	010	TXEN	TXD_0	Out only	Out only	Out only	Out only	Out only	Out only
Eth 7-wire/USB1	011	TXEN	TXD_0	TXP	PrtPWR	Speed	Suspend	$\overline{\text{OE}}$	TXN
Eth 18-wire no MD	100	TXEN	TXD_0	TXD_1	TXD_2	TXD_3	TXERR	Out only	Out only
Eth 18-wire with MD	101	TXEN	TXD_0	TXD_1	TXD_2	TXD_3	TXERR	MDC	MDIO
RST_config (see Note 4)		bit 8	bit 9	bit 10	bit 11	bit 12	bit 13	bit 14	bit 15

NOTES:
1. The port continues on next map (Ethernet In—Ethi).
2. MDC and MDIO are optional for MII operation and if not used can be Output only GPIO
3. Second USB port (USB2) can be configured on PSC3 or on the Ethernet port, but not both locations.
4. RST_config is an input function on these pins, but only during reset.

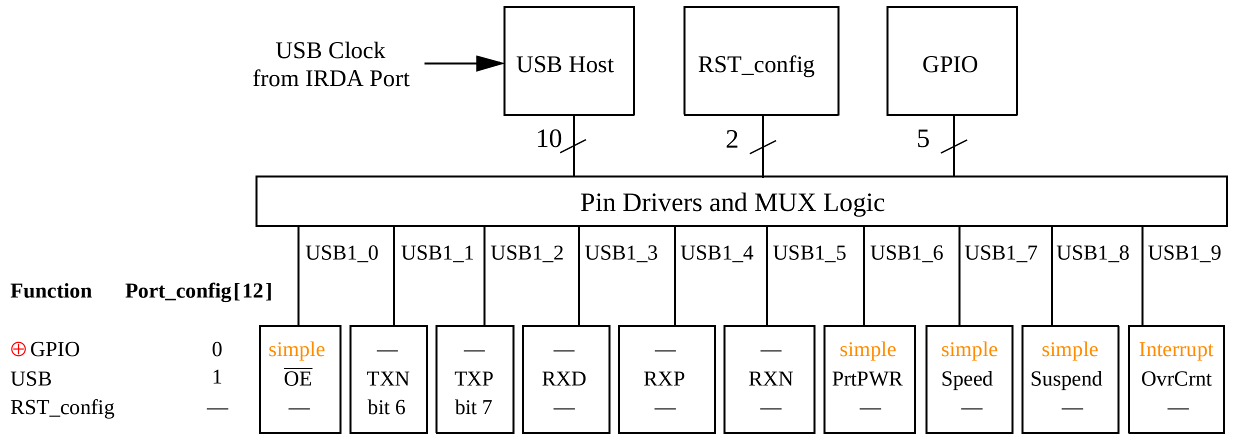
Figure 2-7 Ethernet Output Port Map—8 Pins



Function	Port_conf[18:16]	ETH_8	ETH_9	ETH_10	ETH_11	ETH_12	ETH_13	ETH_14	ETH_15	ETH_16	ETH_17
⊕GPIO	000	simple	simple	simple	simple	—	Interrupt	Interrupt	Interrupt	Interrupt	WakUp
USB2	001	simple	simple	simple	simple	—	RXD	RXP	RXN	OvrCrnt	WakUp
Eth 7-wire	010	CD	RXCLK	COL	TXCLK	RXD_0	Interrupt	Interrupt	Interrupt	Interrupt	WakUp
Eth 7-wire/USB1	011	CD	RXCLK	COL	TXCLK	RXD_0	RXD	RXP	RXN	OvrCrnt	WakUp
Eth 18-wire no MD	100	RXDV	RXCLK	COL	TXCLK	RXD_0	RXD_1	RXD_2	RXD_3	RXERR	CRS
Eth 18-wire with MD	101	RXDV	RXCLK	COL	TXCLK	RXD_0	RXD_1	RXD_2	RXD_3	RXERR	CRS

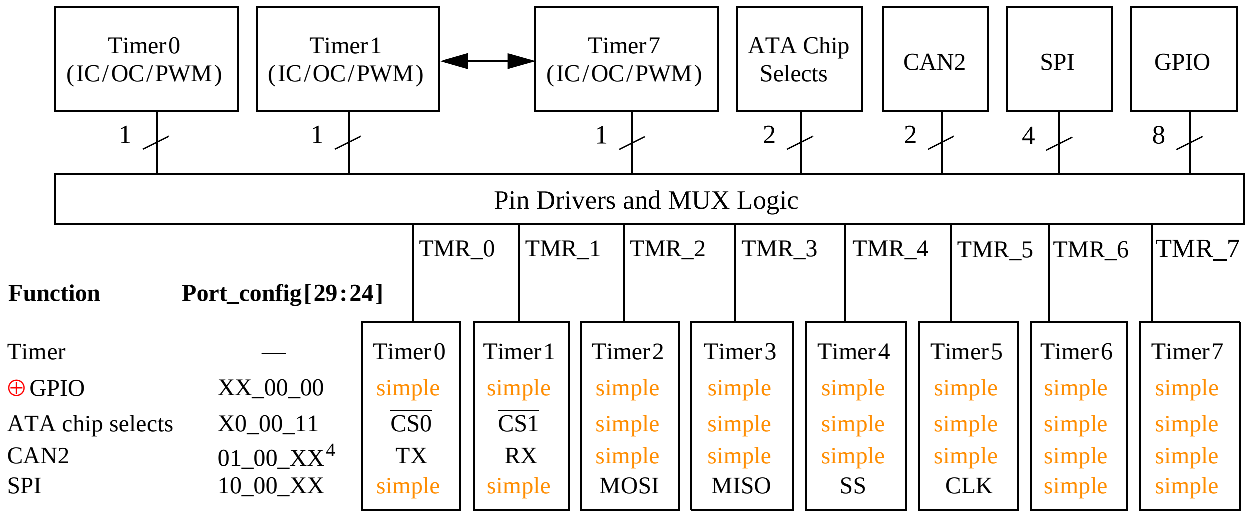
NOTE:
1. This port can serve as a GPIO resource when Ethernet and secondary USB are not needed. It contains 4 Simple, 4 Interrupt GPIO, and 1 WakUp GPIO.

Figure 2-8 Ethernet Input/Control Port Map—10 Pins



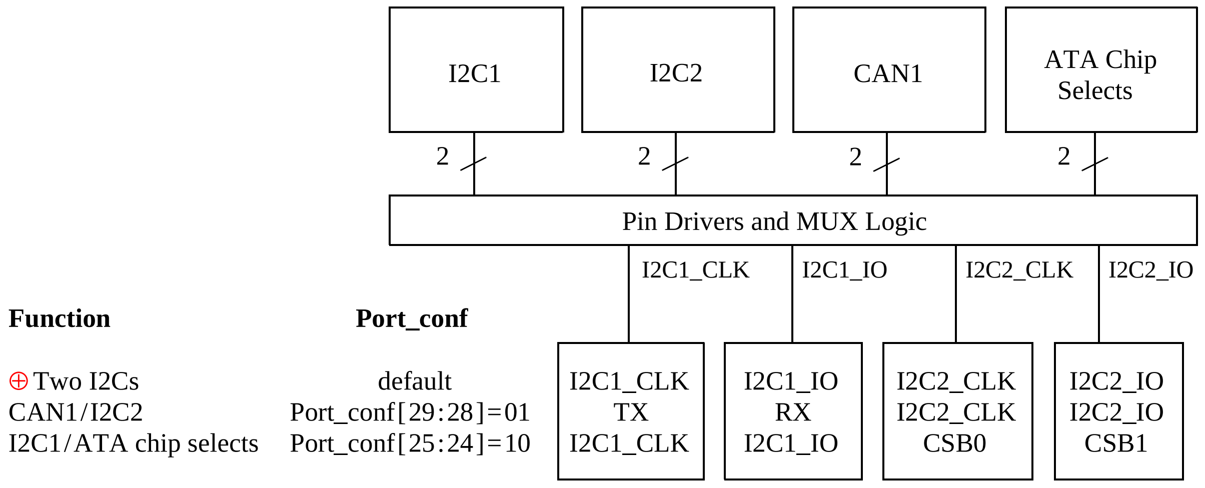
NOTE:
1. If not used for USB, this port is available as a GPIO resource.
2. USB clock source can be generated internally or sourced from IR_USB_CLK input.
3. Pins 3–5 are not mapped to any function other than USB.
4. RST_config bits are sampled only during Reset.

Figure 2-9 USB Port Map—10 Pins


Timer0 (IC/OC/PWM)
Timer1 (IC/OC/PWM)
Timer7 (IC/OC/PWM)
ATA Chip Selects
CAN2
SPI
GPIO
1
1
1
2
2
4
8
Pin Drivers and MUX Logic
TMR_0
TMR_1
TMR_2
TMR_3
TMR_4
TMR_5
TMR_6
TMR_7
Function
Port_config[29:24]
Timer —
⊕GPIO XX_00_00
ATA chip selects X0_00_11
CAN2 01_00_XX4
SPI 10_00_XX
Timer0 simple CS0 TX simple
Timer1 simple CS1 RX simple
Timer2 simple simple simple MOSI
Timer3 simple simple simple MISO
Timer4 simple simple simple SS
Timer5 simple simple simple CLK
Timer6 simple simple simple simple
Timer7 simple simple simple simple

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). 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. Bits 25:24 can not be set to “11” in this case.
5. CAN RX input supports WakeUp functionality.

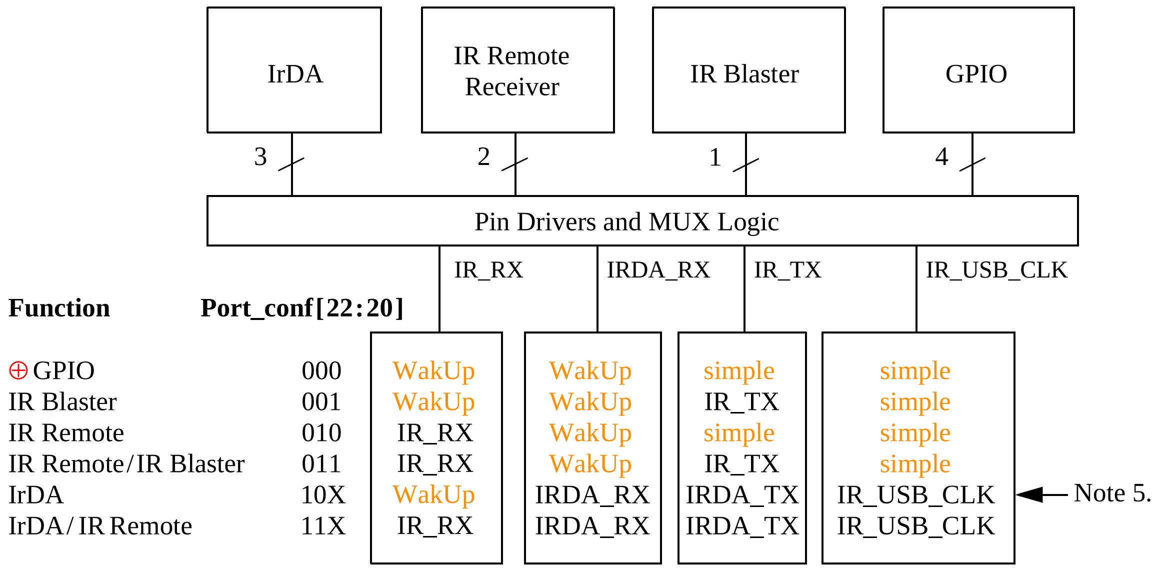
Figure 2-10 Timer Port Map—8 Pins



Function	Port_conf	I2C1_CLK	I2C1_IO	I2C2_CLK	I2C2_IO
⊕ Two I2Cs	default	I2C1_CLK	I2C1_IO	I2C2_CLK	I2C2_IO
CAN1/I2C2	Port_conf[29:28]=01	TX	RX	I2C2_CLK	I2C2_IO
I2C1/ATA chip selects	Port_conf[25:24]=10	I2C1_CLK	I2C1_IO	CSB0	CSB1

NOTE:
1. CAN RX input supports WakeUp functionality.

Figure 2-11 I^{2}C Port Map—4 Pins (two pins each, for two I^{2}Cs)



Function	Port_conf[22:20]	IR_RX	IRDA_RX	IR_TX	IR_USB_CLK
⊕ GPIO	000	WakUp	WakUp	simple	simple
IR Blaster	001	WakUp	WakUp	IR_TX	simple
IR Remote	010	IR_RX	WakUp	simple	simple
IR Remote/IR Blaster	011	IR_RX	WakUp	IR_TX	simple
IrDA	10X	WakUp	IRDA_RX	IRDA_TX	IR_USB_CLK
IrDA/IR Remote	11X	IR_RX	IRDA_RX	IRDA_TX	IR_USB_CLK

NOTES:
1. Because the transmit functions share the same pin, only IrDA or IR Blaster can be transmitted.
2. Both IrDA and IR Remote can be received at the same time.
3. IR_USB_CLK is 48MHz clock (same as USB) and may be used from here for USB regardless of IR/IRDA use.
4. IR Remote receiver input has built-in WakeUp functionality, independent of GPIO.
5. USB port may use this clock pin whether used by IR/IrDA or not.

Figure 2-12 IR/IrDA Port Map—4 Pins

SECTION 3
MEMORY MAP

3.1 Overview

The following sections are contained in this document:

- Internal Register Memory Map
- MGT5100 Memory Map, includes:
  - Memory Map Registers—MBAR + 0x0000
- Register Summaries, includes:
  - PCI XLB Configuration Registers
  - SmartComm DMA Registers
  - PSC Registers—Quick Reference
  - IrDA Registers—Quick Reference
  - SPI Registers—Quick Reference

— CAUTION —

The user should be aware that any unused or reserved register fields in the MGT5100 should not be used by software programmers. These fields may be used in future versions of the MGT5100 chip.

3.2 Internal Register Memory Map

Table 3-1 Internal Register Memory Map

Address	Name	Description	Reference
MBAR + 0x0000	MM	Memory Map registers.	Section 3.3.1
MBAR + 0x0080	ARB	Arbiter register (processor bus).	Section 6.3
MBAR + 0x0100	MC	SDRAM Memory Controller registers.	Section 8.6
MBAR + 0x0200	CDM	Clock Distribution Module registers.	Section 5.6
MBAR + 0x0300	CSC	Chip Select Controller registers.	Section 9.4.2
MBAR + 0x0400	SCT	SmartComm Timer registers.	Section 13.5
MBAR + 0x0500	ICTL	Interrupt Controller registers.	Section 7.2.4
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 20.3
MBAR + 0x0B00	GPS	GPIO Standard registers	Section 7.3.2.1
MBAR + 0x0C00	GPW	GPIO Wakeup registers.	Section 7.3.2.2
MBAR + 0x0D00	PPCI	PPC PCI registers	Section 10.3.4.1 (Rx) Section 10.3.4.3 (Tx)
MBAR + 0x0E00	IR	Consumer Infra-Red registers.	Section 19.4.1 Section 19.5.1
MBAR + 0x0F00	SPI	Serial Peripheral Interface registers.	Section 17.3
MBAR + 0x1000	USB	Universal Serial Bus registers.	Section 12.4.1 Section 12.4.2 Section 12.4.3 Section 12.4.4
MBAR + 0x1200	SDMA	SmartComm DMA registers.	Section 13.3
MBAR + 0x2000	PSC1	Programmable Serial Controller 1 registers.	Section 15.2
MBAR + 0x2400	PSC2	Programmable Serial Controller 2 registers.	Section 15.2
MBAR + 0x2800	PSC3	Programmable Serial Controller 3 registers.	Section 15.2
MBAR + 0x2C00	IRDA	Infra-Red Data Association registers.	Section 16.2
MBAR + 0x3000	ETH	Ethernet registers.	Section 14.5
MBAR + 0x3800 MBAR + 0x3880	PCI	SmartComm DMA PCI registers.	Section 10.3.4.1 (Rx) Section 10.3.4.3 (Tx)
MBAR + 0x3A00	ATA	Advanced Technology Attachment registers.	Section 11.3.1 Section 11.3.2 Section 11.3.3
MBAR + 0x3D00	I^2C	Inter-Integrated Circuit registers.	Section 18.3
MBAR + 0x4000	SRAM	On-chip Static RAM memory locations.	Section 13.4

3.3 MGT5100 Memory Map

The MGT5100 memory map has the following main regions:

- MGT5100 internal register space
- External Chip Selects 0–5
- SDRAM space
- PCI1 and PCI2 space
- Boot space

The Memory Base Address Register (MBAR) is the first register in this module. The value in MBAR determines the base address for all MGT5100 register space (including the MBAR itself). If the MBAR is rewritten, the new value must be “remembered” by software as the location of the register will move and it can only be read at its new address.

All the address related registers in this module are in the form of Start/Stop pairs. An address appearing on XLB 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 (see following paragraph).

Address registers (and the MBAR itself) have only 17 significant bits. Although these bits are right-justified in the registers they are actually interpreted as the most significant bits of the address for comparison tests. For this reason, software must right shift an absolute address by 15 before writing it as a value into the desired register. The same is true when reading values from these registers.

Start/Stop comparisons are enabled only if the corresponding enable bit in the Address Enable register (ADREN) 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.

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

Boot Space and CS0 space should **never** be simultaneously enabled, both spaces use the external Chip Select 0 pin. When enabled as Boot Space, only reads are possible, but up to the full 64-bits needed to support instruction fetches (burst reads are also supported). When enabled as CS0 space, normal chip select reads and writes are supported, but only up to 32-bits (and no bursting).

NOTE: The Memory Map module may be referred to as the IPBI module in other sections of this manual. No distinction should be inferred by one usage or the other.

3.3.1 Memory Map Registers—MBAR + 0x0000

There are 22 32-bit Memory Map registers. These registers are located at an offset from the Module Base Address Register (MBAR). Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0000 + register address**

Hyperlinks to the Memory Map registers are provided below:

- Module Base Address (0x0000)—MBAR
- CS0 Start Address (0004)—CS0STR
- CS0 Stop Address (0008)—CS0STP
- CS1 Start Address (000C)—CS1STR
- CS1 Stop Address (0010)—CS1STP
- CS2 Start Address (0014)—CS2STR
- CS2 Stop Address (0018)—CS2STP
- CS3 Start Address (001C)—CS3STR
- CS3 Stop Address (0020)—CS3STP
- CS4 Start Address (0024)—CS4STR
- CS4 Start Address (0028)—CS4STP
- CS5 Start Address (002C)—CS5STR
- CS5 Start Address (0030)—CS5STP
- SDRAM Start Address (0034)—SDRAMSTR
- SDRAM Stop Address (0038)—SDRAMSTP
- PCI1 Start Address (003C)—PCI1STR
- PCI1 Stop Address (0040)—PCI1STP
- PCI2 Start Address (0044)—PCI2STR
- PCI2 Stop Address (0048)—PCI2STP
- Boot Start Address (004C)—BOOTSTR
- Boot Stop Address (0050)—BOOTSTP
- Address Space Enable (0054)—ADREN

3.3.1.1 Module Base Address (0x0000)—MBAR

Table 3-2 Module Base Address (0x0000)—MBAR

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

Bits	Name	Description
0:14	—	Reserved
15:31	MBAR	Module Base Address Register—defines the base address location for all internal registers accessed via the IP bus. This register constitutes the upper 17 bits of the address. when a new value is written to this register the position of all MGT5100 internal registers changes relative to the new value. After reset, the effective MBAR is 0x8000_0000 and all MGT5100 registers are accessed relative to this base address.

3.3.1.2 CS0 Start Address (0004–002C)—CS0STR

Table 3-3 CS0 Start Address (0004)—CS0STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS0 STR
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	CS0STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS0STR	Chip select 0 start address register—defines the starting address of chip select 0.

3.3.1.3 CS0 Stop Address (0008)—CS0STP

Table 3-4 CS0 Stop Address (0008)—CS0STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS0 STP
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	CS0STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS0STP	Chip select 0 stop address register—defines stopping address of chip select 0.

3.3.1.4 CS1 Start Address (000C)—CS1STR

Table 3-5 CS1 Start Address (000C)—CS1STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS1 STR
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	CS1STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS1STR	Chip select 1 start address register—defines starting address of chip select 1.

3.3.1.5 CS1 Stop Address (0010)—CS1STP

Table 3-6 CS1 Stop Address (0010)—CS1STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS1 STP
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	CS1STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS1STP	Chip select 1 stop address register—defines stopping address of chip select 1.

3.3.1.6 CS2 Start Address (0014)—CS2STR

Table 3-7 CS2 Start Address (0014)—CS2STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS2 STR
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	CS2STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS2STR	Chip select 2 start address register—defines starting address of chip select 2.

3.3.1.7 CS2 Stop Address (0018)—CS2STP

Table 3-8 CS2 Stop Address (0018)—CS2STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS2 STP
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	CS2STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS2STR	Chip select 2 stop address register—defines stopping address of chip select 2.

3.3.1.8 CS3 Start Address (001C)—CS3STR

Table 3-9 CS3 Start Address (001C)—CS3STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS3 STR
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	CS3STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS3STR	Chip select 3 start address register—defines starting address of chip select 3.

3.3.1.9 CS3 Stop Address (0020)—CS3STP

Table 3-10 CS3 Stop Address (0020)—CS3STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS3 STP
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 / W	CS3STP															
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS3STR	Chip select 3 stop address register—defines stopping address of chip select 3.

3.3.1.10 CS4 Start Address (0024)—CS4STR

Table 3-11 CS4 Start Address (0024)—CS4STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved															CS4 STR
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 / W	CS4STR															
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS4STR	Chip select 4 start address register—defines starting address of chip select 4.

3.3.1.11 CS4 Stop Address (0028)—CS4STP

Table 3-12 CS4 Start Address (0028)—CS4STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved															CS4 STP
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 / W	CS4STP															
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS4STP	Chip select 4 stop address register—defines stopping address of chip select 4.

3.3.1.12 CS5 Start Address (002C)—CS5STR

Table 3-13 CS5 Start Address (002C)—CS5STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS5 STR
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	CS5STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS5STR	Chip select 5 start address register—defines starting address of chip select 5.

3.3.1.13 CS5 Stop Address (0030)—CS5STP

Table 3-14 CS5 Start Address (0030)—CS5STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															CS5 STP
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	CS5STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	CS5STP	Chip select 5 stop address register—defines stopping address of chip select 5.

3.3.1.14 SDRAM Start Address (0034)—SDRAMSTR

Table 3-15 SDRAM Start Address (0034)—SDRAMSTR

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

Bits	Name	Description
0:14	—	Reserved
15:31	SDRAMSTR	SDRAM start address register—defines starting address of external SDRAM.

3.3.1.15 SDRAM Stop Address (0038)—SDRAMSTP

Table 3-16 SDRAM Stop Address (0038)—SDRAMSTP

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

Bits	Name	Description
0:14	—	Reserved
15:31	SDRAMSTP	SDRAM stop address register—defines stopping address of external SDRAM.

3.3.1.16 PCI1 Start Address (003C)—PCI1STR

Table 3-17 PCI1 Start Address (003C)—PCI1STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															PCI1 STR
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	PCI1STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	PCI1STR	PCI1 start address register—defines starting address of the first PCI space. Relates to PCI Window 1 as described in PCI module sections.

3.3.1.17 PCI1 Stop Address (0040)—PCI1STP

Table 3-18 PCI1 Stop Address (0040)—PCI1STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															PCI1 STP
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	PCI1STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	PCI1STP	PCI1 stop address register—defines stopping address of the first PCI space. Relates to PCI Window 1 as described in PCI module sections.

3.3.1.18 PCI2 Start Address (0044)—PCI2STR

Table 3-19 PCI2 Start Address (0044)—PCI2STR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															PCI2 STR
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	PCI2STR															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	PCI2STR	PCI2 start address register—defines starting address of the second PCI space. Relates to PCI Window 2 as described in PCI module sections.

3.3.1.19 PCI2 Stop Address (0048)—PCI2STP

Table 3-20 PCI2 Stop Address (0048)—PCI2STP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															PCI2 STP
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	PCI2STP															
W																
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description
0:14	—	Reserved
15:31	PCI2STP	PCI2 stop address register—defines stopping address of the second PCI space. Relates to PCI Window 2 as described in PCI module sections.

3.3.1.20 Boot Start Address (004C)—BOOTSTR

Table 3-21 Boot Start Address (004C)—BOOTSTR

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

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

Bits	Name	Description
0:14	—	Reserved
15:31	BOOTSTR	Boot start address register—defines starting address of the boot code space. The value after reset is dependant on a reset configuration bit. There are 2 possible boot start values, either low or high. The effective low value is 0. The effective high value is 0xFFF0 0000. See Section 4.6, Reset Configuration.

3.3.1.21 Boot Stop Address (0050)—BOOTSTP

Table 3-22 Boot Stop Address (0050)—BOOTSTP

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

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

Bits	Name	Description
0:14	—	Reserved
15:31	BOOTSTP	Boot stop address register—defines starting address of boot code space. The value after reset is dependant on a reset configuration bit. There are 2 possible boot stop values, either low or high. The effective low value is 0. The effective high value is 0x0007 FFFF. See Section 4.6, Reset Configuration. The default boot space size in either case is 512KBytes.

3.3.1.22 Address Space Enable (0054)—ADREN (Control Reg in RTL)

Table 3-23 Address Space Enable (0054)—ADREN

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved						Boot _en	PCI1 _en	PCI2 _en	SDRAM _en	CS5 _en	CS4 _en	CS3 _en	CS2 _en	CS1 _en	CS0 _en
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 / W	Reserved															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:5	—	Reserved
6	Boot_en	When set, Boot address space is enabled.
7	PCI1_en	When set, PCI1 address space is enabled.
8	PCI2_en	When set, PCI2 address space is enabled.
9	SDRAM_en	When set, SDRAM address map is enabled.
10	CS5_en	When set, CS5 address map is enabled.
11	CS4_en	When set, CS4 address map is enabled.
12	CS3_en	When set, CS3 address map is enabled.
13	CS2_en	When set, CS2 address map is enabled.
14	CS1_en	When set, CS1 address map is enabled.
15	CS0_en	When set, CS0 address map is enabled.
16:31	—	Reserved

3.4 Register Summaries

3.4.1 PCI XLB Configuration Registers

Table 3-24 Overview—PCI XLB Configuration Registers

Address $MBAR +	Name	#	Bits	Description
0xxx00	PHIDR	0	[0:15]	Device ID
			[16:31]	Vendor ID
0xxx04	PHSCR	1	[0:15]	Status
			[16:31]	Command
0xxx08	PHCCR	2	[0:23]	Class Code
			[24:31]	Revision ID
0xxx0C	PHCR1	3	[0:7]	BIST
			[8:15]	Header Type
			[16:23]	Latency Timer
			[24:31]	Cache Line Size
0xxx10	PHBAR0	4	[0:31]	BAR0
xx14	PHBAR1	5	[0:31]	BAR1
0xxx18–xx24	—	6–9	[0:31]	Reserved
0xxx28	PHCPR	10	[0:31]	CardBus CIS Pointer
0xxx2C	PHSSR	11	[0:15]	Subsystem ID
			[16:31]	Subsystem Vendor ID
0xxx30	—	12	[0:31]	Expansion ROM Base Address
0xxx34	—	13	[0:23]	Reserved
			[24:31]	Cap_Ptr
0xxx38	—	14	[0:31]	Reserved
0xxx3C	PHMIR	15	[0:7]	Max_Lat
			[8:15]	Min_Gnt
			[16:23]	Int Pin
			[24:31]	Int Line
0xxx40–xx5C	—	16–23	[0:31]	Reserved
0xxx60	PIER	24	[0:22]	Reserved
			[23]	IE
			[24:31]	Max Retrys
0xxx64	PSR	25	[0:4]	Reserved
			[5]	RC
			[6]	TA
			[7]	IA
			[8:31]	Reserved

Table 3-24 Overview—PCI XLB Configuration Registers (continued)

Address $MBAR +	Name	#	Bits	Description
0xxx68	PCR	26	[0:3]	Reserved
			[4]	PR
			[5]	CM
			[6]	DP
			[7]	LD
			[8:31]	Reserved
0xxx6C	PMVRrd	27	[0:7]	Win 1 Mask
			[8:15]	Win 1 Value
			[16:23]	Win 2 Mask
			[24:31]	Win 2 Value
0xxx70	PMVRwr	28	[0:7]	Win 1 Mask
			[8:15]	Win 1 Value
			[16:23]	Win 2 Mask
			[24:31]	Win 2 Value
0xxx74	PS1R	29	[0:15]	Subwindow 1 Start
			[16:31]	Subwindow 1 Stop
0xxx78	PS2R	30	[0:15]	Subwindow 2 Start
			[16:31]	Subwindow 2 Stop
0xxx7C	PWCR	31	[0:2]	Win1 CMD
			[3]	Reserved
			[4:7]	Win1 CTL
			[8:10]	Win2 CMD
			[11]	Reserved
			[12:15]	Win2 CTL
			[16:18]	Sub1 CMD
			[19]	Reserved
			[20:23]	Sub1 CTL
			[24:26]	Sub2 CMD
			[27]	Reserved
			[28:31]	Sub2 CTL

3.4.2 SmartComm DMA Registers

Table 3-25 SmartComm DMA Registers

Address $MBAR +	Byte Lane	Name	Bits	Description
0x1200	ips_byte_31_24	taskBar	[31:0]	Base address register for SmartComm tasks
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1204	ips_byte_31_24	current Pointer	[31:0]	Pointer to current task instruction (LCD or DRD)
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1208	ips_byte_31_24	end Pointer	[31:0]	Pointer to last instruction of current task
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x120C	ips_byte_31_24	variable Pointer	[31:0]	Pointer to current task's variable table
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1210	ips_byte_31_24	IntVect1	[7:0]	Interrupt vector1 (not used in MGT5100)
	ips_byte_23_16	IntVect2	[7:0]	Interrupt vector2 (not used in MGT5100)
	ips_byte_15_8	PtdCntrl	[16:0]	PTD control register (not used)
	ips_byte_7_0			
0x1214	ips_byte_31_24	IntPend	[31:0]	Interrupt Pending
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1218	ips_byte_31_24	IntMask	[31:0]	Interrupt Mask
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x121C	ips_byte_31_24	TCR0	[16:0]	Task control register for task 0
	ips_byte_23_16			
	ips_byte_15_8	TCR1	[16:0]	Task control register for task 1
	ips_byte_7_0			
0x1220	ips_byte_31_24	TCR2	[16:0]	Task control register for task 2
	ips_byte_23_16			
	ips_byte_15_8	TCR3	[16:0]	Task control register for task 3
	ips_byte_7_0			

Table 3-25 SmartComm DMA Registers (continued)

Address $MBAR +	Byte Lane	Name	Bits	Description
0x1224	ips_byte_31_24	TCR4	[16:0]	Task control register for task 4
	ips_byte_23_16			
	ips_byte_15_8	TCR5	[16:0]	Task control register for task 5
	ips_byte_7_0			
0x1228	ips_byte_31_24	TCR6	[16:0]	Task control register for task 6
	ips_byte_23_16			
	ips_byte_15_8	TCR7	[16:0]	Task control register for task 7
	ips_byte_7_0			
0x122C	ips_byte_31_24	TCR8	[16:0]	Task control register for task 8
	ips_byte_23_16			
	ips_byte_15_8	TCR9	[16:0]	Task control register for task 9
	ips_byte_7_0			
0x1230	ips_byte_31_24	TCRA	[16:0]	Task control register for task 10
	ips_byte_23_16			
	ips_byte_15_8	TCRB	[16:0]	Task control register for task 11
	ips_byte_7_0			
0x1234	ips_byte_31_24	TCRC	[16:0]	Task control register for task 12
	ips_byte_23_16			
	ips_byte_15_8	TCRD	[16:0]	Task control register for task 13
	ips_byte_7_0			
0x1238	ips_byte_31_24	TCRE	[16:0]	Task control register for task 14
	ips_byte_23_16			
	ips_byte_15_8	TCRF	[16:0]	Task control register for task 15
	ips_byte_7_0			
0x123C	ips_byte_31_24	IPR0	[7:0]	Initiator priority register for initiator 0
	ips_byte_23_16	IPR1	[7:0]	Initiator priority register for initiator 1
	ips_byte_15_8	IPR2	[7:0]	Initiator priority register for initiator 2
	ips_byte_7_0	IPR3	[7:0]	Initiator priority register for initiator 3
0x1240	ips_byte_31_24	IPR4	[7:0]	Initiator priority register for initiator 4
	ips_byte_23_16	IPR5	[7:0]	Initiator priority register for initiator 5
	ips_byte_15_8	IPR6	[7:0]	Initiator priority register for initiator 6
	ips_byte_7_0	IPR7	[7:0]	Initiator priority register for initiator 7
0x1244	ips_byte_31_24	IPR8	[7:0]	Initiator priority register for initiator 8
	ips_byte_23_16	IPR9	[7:0]	Initiator priority register for initiator 9
	ips_byte_15_8	IPR10	[7:0]	Initiator priority register for initiator 10
	ips_byte_7_0	IPR11	[7:0]	Initiator priority register for initiator 11

Table 3-25 SmartComm DMA Registers (continued)

Address $MBAR +	Byte Lane	Name	Bits	Description
0x1248	ips_byte_31_24	IPR12	[7:0]	Initiator priority register for initiator 12
	ips_byte_23_16	IPR13	[7:0]	Initiator priority register for initiator 13
	ips_byte_15_8	IPR14	[7:0]	Initiator priority register for initiator 14
	ips_byte_7_0	IPR15	[7:0]	Initiator priority register for initiator 15
0x124C	ips_byte_31_24	IPR16	[7:0]	Initiator priority register for initiator 16
	ips_byte_23_16	IPR17	[7:0]	Initiator priority register for initiator 17
	ips_byte_15_8	IPR18	[7:0]	Initiator priority register for initiator 18
	ips_byte_7_0	IPR19	[7:0]	Initiator priority register for initiator 19
0x1250	ips_byte_31_24	IPR20	[7:0]	Initiator priority register for initiator 20
	ips_byte_23_16	IPR21	[7:0]	Initiator priority register for initiator 21
	ips_byte_15_8	IPR22	[7:0]	Initiator priority register for initiator 22
	ips_byte_7_0	IPR23	[7:0]	Initiator priority register for initiator 23
0x1254	ips_byte_31_24	IPR24	[7:0]	Initiator priority register for initiator 24
	ips_byte_23_16	IPR25	[7:0]	Initiator priority register for initiator 25
	ips_byte_15_8	IPR26	[7:0]	Initiator priority register for initiator 26
	ips_byte_7_0	IPR27	[7:0]	Initiator priority register for initiator 27
0x1258	ips_byte_31_24	IPR28	[7:0]	Initiator priority register for initiator 28
	ips_byte_23_16	IPR29	[7:0]	Initiator priority register for initiator 29
	ips_byte_15_8	IPR30	[7:0]	Initiator priority register for initiator 30
	ips_byte_7_0	IPR31	[7:0]	Initiator priority register for initiator 31
0x125C	ips_byte_31_24	res1	[31:0]	Reserved
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1260	ips_byte_31_24	res2	[31:0]	Reserved
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1264	ips_byte_31_24	res3	[31:0]	Reserved
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1268	ips_byte_31_24	res4	[31:0]	Reserved
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			

Table 3-25 SmartComm DMA Registers (continued)

Address $MBAR +	Byte Lane	Name	Bits	Description
0x126C	ips_byte_31_24	res5	[31:0]	Reserved
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1270	ips_byte_31_24	Value1	[31:0]	Debug Module Comparator 1 Value
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1274	ips_byte_31_24	Value2	[31:0]	Debug Module Comparator 2 Value
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x1278	ips_byte_31_24	Control	[31:0]	Debug Module Control Register
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			
0x127C	ips_byte_31_24	Status	[31:0]	Debug Module Status Register
	ips_byte_23_16			
	ips_byte_15_8			
	ips_byte_7_0			

3.4.3 PSC Registers—Quick Reference

Table 3-26 provides hyperlinks to the 32-bit PSC registers described in Section 15.2.

NOTE: Register bits 16–31 are not shown except where used.

Table 3-26 PSC Memory Mapping

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	UART Mode 1 (2000, 2400, 2800)—MR1_[1, 2, 3]								Unused							
W	Other Modes (2000, 2400, 2800)—MR1_[1, 2, 3]								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	UART Mode 2 (2000, 2400, 2800)—MR2_[1, 2, 3]								Unused							
W	Other Modes (2000, 2400, 2800)—MR2_[1, 2, 3]								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Unused							UART Mode (2004, 2404, 2804)—SR[1, 2, 3]								
W	Unused															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Unused															
W	UART Mode (2004, 2404, 2804)—CSR[1, 2, 3] Other Modes (2004, 2404, 2804)—CSR[1, 2, 3]															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	**Do Not Access**							Unused								
W	All Modes (2008, 2408, 2808)—CR[1, 2, 3]							Unused								

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UART/Modem8&16 Rx Buffers (200C, 240C, 280C)—RB[1, 2, 3] AC97 Rx Buffers (200C, 240C)—RB[1, 2]															
W	UART/Modem8&16 Tx Buffers (200C, 240C, 280C)—TB[1, 2, 3] AC97 Tx Buffers (200C, 240C)—TB[1, 2]															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	UART/Modem8&16 Rx Buffers (200C, 240C, 280C)—RB[1, 2, 3] AC97 Rx Buffers (200C, 240C)—RB[1, 2]															
W	UART/Modem8&16 Tx Buffers (200C, 240C, 280C)—TB[1, 2, 3] AC97 Tx Buffers (200C, 240C)—TB[1, 2]															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Input Port UART Change (2010, 2810, 2410)—IPCR[1, 3, 2] Modem Mode (2010, 2810, 2410)—IPCR[1, 3, 2]								Unused							
W	All Modes (2010, 2410, 2810)—ACR[1, 2, 3]								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	UART Mode (2014, 2414, 2814)—ISR[1, 2, 3] Modem Mode (2014, 2414, 2814)—ISR[1, 2, 3]															
W	Interrupt UART (2014, 2414, 2814)—IMR[1, 2, 3] Modem Mode (2014, 2414, 2814)—IMR[1, 2, 3]															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Counter Timer UART (2018, 2418, 2818)—CTUR[1, 2, 3] Other Modes (2018, 2418, 2818)—CTUR[1, 2, 3]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Counter Timer UART (201C, 241C, 281C)—CTLR[1, 2, 3] Other Modes (201C, 241C, 281C)—CTLR[1, 2, 3]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Interrupt Vector (2030, 2430, 2830)—IVR[1, 2, 3]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Input UART (2034, 2434, 2834)—IP[1, 2, 3] Input Modem8&16 Mode (2034, 2434, 2834)—IP[1, 2, 3] Input AC97 Mode (2034, 2434)—IP[1, 2]								Unused							
W	**Do Not Access**								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	**Do Not Access**								Unused							
W	UART/Modem8&16 Set (2038, 2438, 2838)—OP1_[1, 2, 3] AC97 Mode (2038, 2438)—OP1_[1, 2]								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	**Do Not Access**								Unused							
W	UART/Modem8&16 Reset (203C, 243C, 284C)— OP0_[1, 2, 3] AC97Bit Reset (203C, 243C)—OP0_[1, 2]								Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	UART Mode (2040, 2440, 2840)—SICR[1, 2, 3] Modem8&16 Mode (2040, 2440, 2840)—SICR[1, 2, 3] AC97 Mode(2040, 2440, 2840)—SICR[1, 2]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Number of Data (2058, 2458, 2858)—RFNUM[1, 2, 3]									Unused						
W	Unused															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Number of Data (205C, 245C, 285C)—TFNUM[1, 2, 3]															
W	Unused															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Data (2x60)—RFDATA[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Status (2064, 2464, 2864)—RFSTAT[1, 2, 3]															
W	Unused															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Control (2068, 2468, 2868)—RFCNTL[1, 2, 3]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Alarm (206E, 246E, 286E)—RFALARM[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Read Pointer (2072, 2472, 2872)—RFRPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Write Pointer (2076, 2476, 2876)—RFWPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Last Read Frame PTR (207A, 247A, 287A)—RFLRFPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Last Write Frame PTR (207C, 247C, 287C)—RFLWFPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Data (2x80)—TFDATA[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Status (2084, 2484, 2884)—TFSTAT[1, 2, 3]															
W	Unused															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Control (2088, 2488, 2888)—TFCNTL[1, 2, 3]								Unused							
W									Unused							

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Alarm (208E, 248E, 288E)—TFALARM[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Read Pointer (2092, 2492, 2892)—TFRPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Write Pointer (2096, 2496, 2896)—TFWPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Last Read Frame PTR (209A, 249A, 289A)—TFLRFPTR[1, 2, 3]															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R W	Tx FIFO Last Write Frame PTR (209C, 249C, 289C)—TFLWFPTR[1, 2, 3]															

3.4.4 IrDA Registers—Quick Reference

Table 3-26 provides hyperlinks to the IrDA registers described in Section 16.2.

Table 3-27 IrDA Memory Mapping

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SIR Mode 1 (2C00)—MR1 MIR/FIR Modes (2C00)—MR1 SIR Mode 2 (2C00)—MR2 MIR/FIR Modes (2C00)—MR2															
W									▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SIR Status (2C04)—SR MIR/FIR Status (2C04)—SR															
W	SIR Mode (2C04)—CSR Other Modes (2C04)—CSR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SIR/MIR/FIR Rx Buffers (2C0C)—RB															
W	SIR/MIR/FIR Tx Buffers (2C0C)—TB															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	SIR/MIR/FIR Rx Buffers (2C0C)—RB															
W	SIR/MIR/FIR Tx Buffers (2C0C)—TB															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Input Port SIR/MIR/FIR Change (2C10)—IPCR															
W	Auxiliary Control (2C10)—ACR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Interrupt SIR Status (2C14)—ISR Interrupt MIR/FIR Status (2C14)—ISR															
W	Interrupt SIR Mask (2C14)—IMR Interrupt MIR/FIR Mask (2C14)—IMR															

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Counter Timer SIR Upper Bytes (2C18)—CTUR Counter Timer MIR/FIR Upper Bytes (2C18)—CTUR															
W									▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Counter Timer SIR Lower Bytes (2C1C)—CTLR															
W	Counter Timer MIR/FIR Lower Bytes (2C1C)—CTLR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Interrupt Vector (2C30)—IVR															
W									▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IrDA Input Port (2C34)—IP															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R																
W	IrDA Output Port Bit Set (2C38)—OP1								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R																
W	IrDA Output Port Bit Reset (2C3C)—OP0								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IrDA SIR Control (2C40)—SICR															
W	IrDA MIR/FIR Control (2C40)—SICR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Infrared SIR Control 1 (2C44)—IRCR1															
W	Infrared MIR/FIR Control 1 (2C44)—IRCR1								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Infrared SIR Control 2 (2C48)—IRCR2															
W	Infrared MIR/FIR Control 2 (2C48)—IRCR2								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Infrared SIR Divide (2C4C)—IRSDR															
W	Infrared SIR Other Divide (2C4C)—IRSDR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Infrared MIR Divide (2C50)—IRMDR															
W	Infrared MIR Other Divide (2C50)—IRMDR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Infrared FIR Divide (2C54)—IRFDR															
W	Infrared FIR Other Divide (2C54)—IRFDR								▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Number of Data (2C58)—RFNUM															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Number of Data (2C5C)—TFNUM															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Status (2C64)—RFSTAT															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Control (2C68)—RFCNTL															
W									▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Alarm (2C6E)—RFALARM															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Read Pointer (2C72)—RFRPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Write Pointer (2C76)—RFWPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Status (2C84)—TFSTAT															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Control (2C88)—TFCNTL															
W									▨	▨	▨	▨	▨	▨	▨	▨

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Alarm (2C8E)—TFALARM															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Read Pointer (2C92)—TFRPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Write Pointer (2C96)—TFWPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR															
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Tx FIFO Last Write Frame Pointer (2C9C)—TFLWFPTR															
W																

3.4.5 SPI Registers—Quick Reference

Table 3-28 provides hyperlinks to the SPI registers described in Section 17.3.

Table 3-28 SPI Memory Mapping

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SPI Control 1 (0F00)—SPICR1								SPI Control 2 (0F01)—SPICR2							
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SPI Baud Rate (0F04)—SPIBR								SPI Status (0F05)—SPISR							
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SPI Data (0F09)—SPIDR								Unused							
W																

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	SPI Port Data (0F0D)—SPIPORT								SPI Data Direction (0F90)—SPIDDR							
W																

3.5 SDRAM Memory Controller Register Map (MBAR + 0x0100)

The SDRAM Controller has 5 registers, which are mapped at addresses calculated by (MBAR+0x0100+Offset). Table 3-29 shows the offset value. Shaded bits are reserved and should be written and read as 0.

Table 3-29 SDRAM Controller Register Map

Offset	Name	Byte 0 (Subadd _00)								Byte 1 (Subadd _01)								Byte 2 (Subadd _10)								Byte 3 (Subadd _11)							
00	MOD	Mode & Extended Mode																▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
04	CTR						▨											▨	▨	▨	▨					▨	▨	▨		▨	▨	▨	▨
08	CFG1					▨							▨					▨								▨				▨			▨
0c	CFG2																	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
40	ADRSEL	▨	▨	▨	▨	▨	▨			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

3.6 Slice Timers Register Map (MBAR + 0x0700)

Table 3-30 Slice Timers Register Map

Offset	Register	Byte 0 (Subadd _00)	Byte 1 (Subadd _01)	Byte 2 (Subadd _10)	Byte 3 (Subadd _11)
00	0	Reserved	Terminal Count (Slice Timer 0)		
04	1	▨ RW IE TE	Reserved		
08	2	Reserved	Count Value (Slice Timer 0)		
0C	3	Reserved ST	Reserved		
10	4	Reserved	Terminal Count (Slice Timer 1)		
14	5	▨ RW IE TE	Reserved		
18	6	Reserved	Count Value (Slice Timer 1)		
1C	7	Reserved ST	Reserved		
20	8	Reserved			
—	—				
7C	1F				

SECTION 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

MGT5100 has three primary reset pins, which are implemented as open drain I/Os:

- Power-ON Reset—$\overline{\text{PORESET}}$
- Hard Reset—$\overline{\text{HRESET}}$
- Soft Reset—$\overline{\text{SRESET}}$

$\overline{\text{PORESET}}$ is a Power-ON Reset (POR) input. It is asserted by an external source and held active for a specified period of time until power is stable to the MGT5100.

$\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ can be asserted by an external source or they can be asserted by reset generation logic internal to MGT5100.

Internal reset logic analyzes all internal and external reset sources and asserts internal and external reset signals appropriately. When a hard or soft reset is detected, reset logic counters hold internal and external hard/soft reset asserted for 1024 reference clock cycles.

4.2.1 Power-ON Reset—$\overline{\text{PORESET}}$

$\overline{\text{PORESET}}$ must be asserted externally when power is applied to the system for a required period of time. When $\overline{\text{PORESET}}$ is asserted, internal logic forces $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ active. During $\overline{\text{PORESET}}$ assertion, the MGT5100 system oscillator begins oscillation and the system APLL establishes a locked condition. During $\overline{\text{PORESET}}$ the reset configuration word is sampled to establish the initial state of various vital internal MGT5100 functions. The reset configuration word is latched internally when $\overline{\text{PORESET}}$ is de-asserted.

After $\overline{\text{PORESET}}$ is deasserted MGT5100 continues to assert $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ I/O pins, and internal hard and soft resets for 1024 reference clock cycles. After negation of external and internal resets, reset logic waits 16 clock cycles before recognizing another external $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$.

Sources of Power-ON Reset:

- External, board level reset source (push button, reset control logic etc.)

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

External $\overline{\text{HRESET}}$ is an open drain (implemented as a tri-state with data driving the enable and the input grounded) signal. $\overline{\text{HRESET}}$ 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 1024 reference clock cycles.

$\overline{\text{HRESET}}$ 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:

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

4.2.3 Soft Reset—$\overline{\text{SRESET}}$

External $\overline{\text{SRESET}}$ is an open drain signal. It is implemented as a tri-state with data driving the enable and the input grounded. $\overline{\text{SRESET}}$ requires an external pull-up. Assertion of $\overline{\text{SRESET}}$ causes assertion of an external $\overline{\text{SRESET}}$ and internal soft reset for 1024 reference clock cycles.

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

Sources of soft reset:

- $\overline{\text{PORESET}}$, or $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ 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

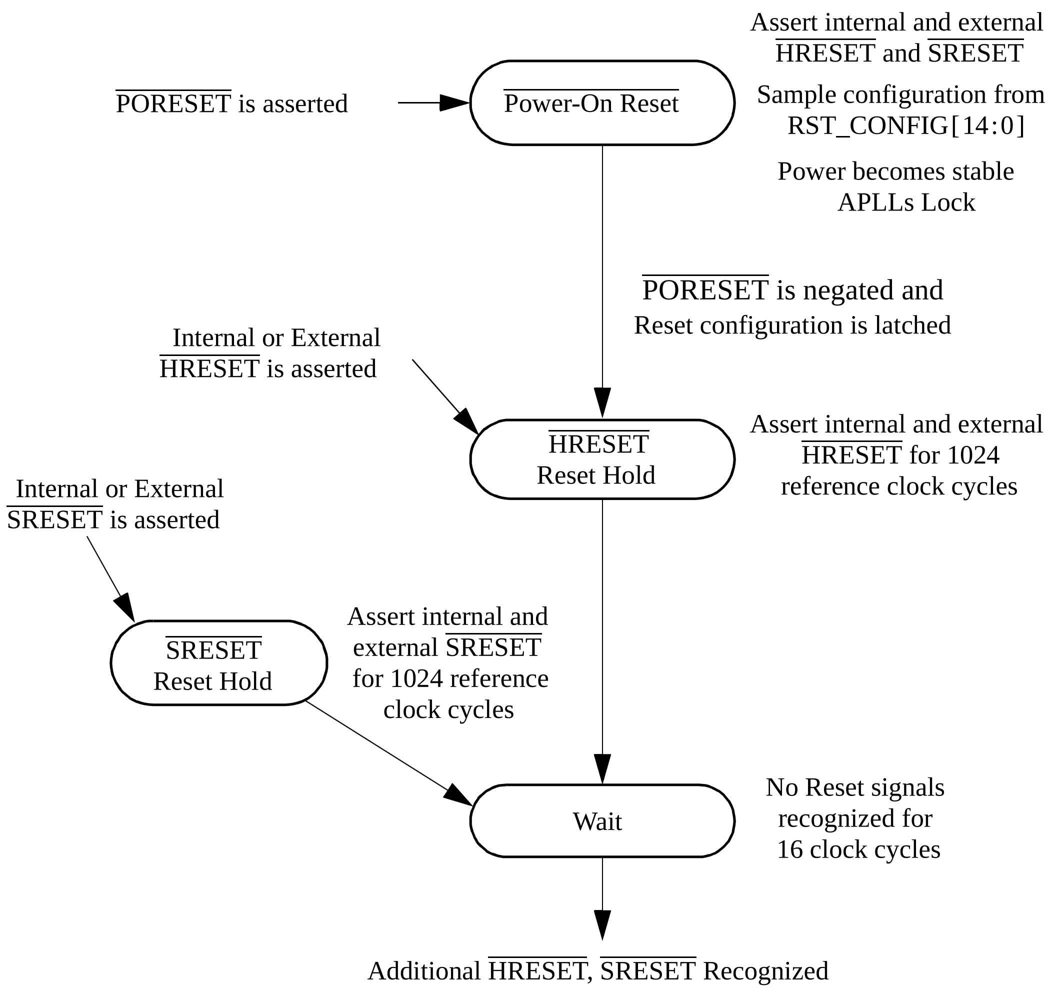

PORESET is asserted
Power-On Reset
Assert internal and external HRESET and SRESET
Sample configuration from RST_CONFIG[14:0]
Power becomes stable
APLLs Lock
PORESET is negated and Reset configuration is latched
Internal or External HRESET is asserted
HRESET Reset Hold
Assert internal and external HRESET for 1024 reference clock cycles
Internal or External SRESET is asserted
SRESET Reset Hold
Assert internal and external SRESET for 1024 reference clock cycles
Wait
No Reset signals recognized for 16 clock cycles
Additional HRESET, SRESET Recognized

Figure 4-1 Reset sequence

4.4 Reset Operation

$\overline{\text{PORESET}}$ must be asserted for at least TBD us. Following deassertion of $\overline{\text{Power-ON Reset}}$, $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ remain low for 1024 reference clock cycles.

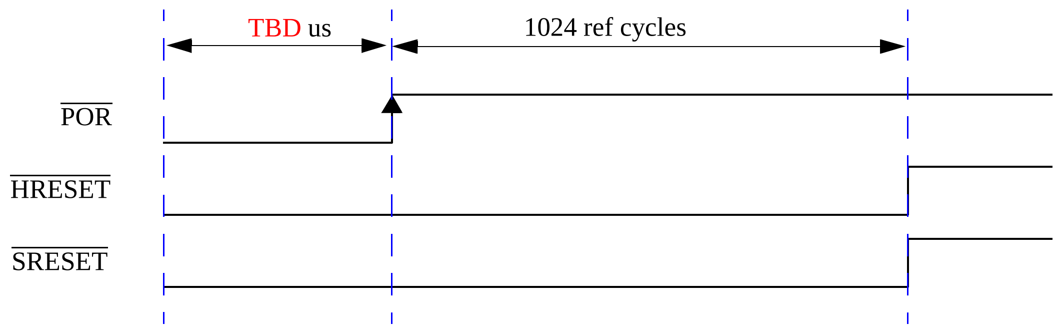
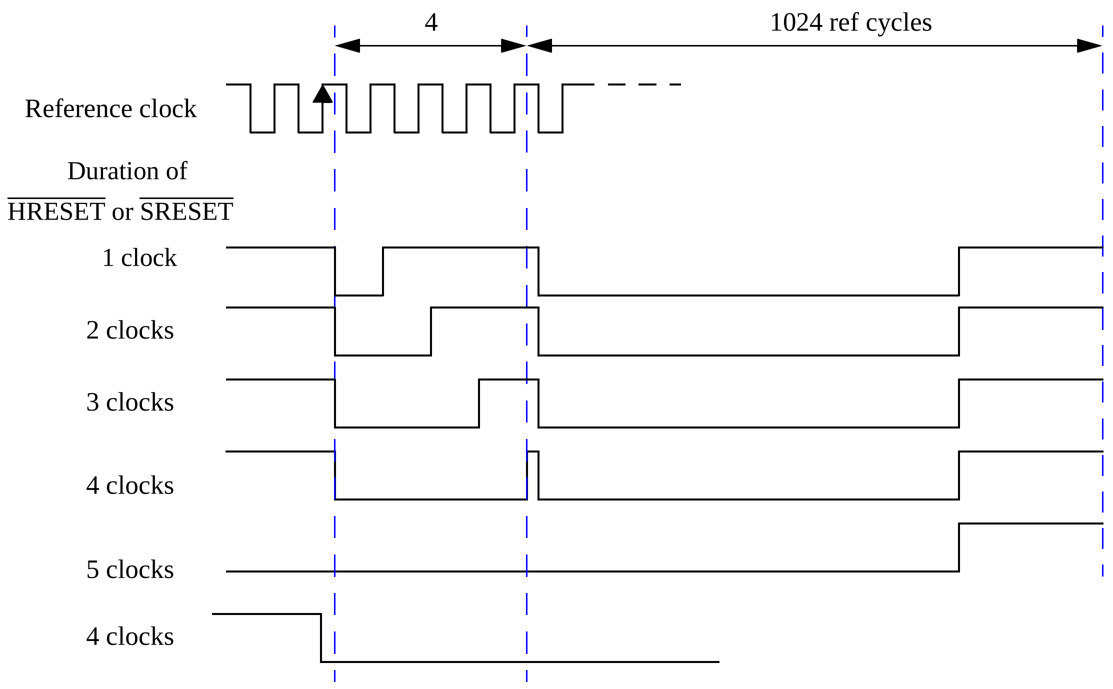


Figure 4-2 $\overline{\text{PORESET}}$ Assertion

When external $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ is asserted, internal reset logic catches the reset signal held low and asserts internal resets for 1024 reference clock cycles. It is recommended the external reset signal be held low for 4 reference clock cycles (must catch 4 rising edges of reference clock) to prevent transitions on internal reset signals.



NOTE: If $\overline{\text{Hreset}}$ assertion coincides with a rising-edge of reference clock, $\overline{\text{Hreset}}$ of 4 clock cycles duration is sufficient for the signal to avoid being pulled-up.

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

If $\overline{\text{HRESET}}$ is held low externally for more than 1024 cycles, internal Hard reset, and soft reset are driven low for just 1024 clock cycles. External $\overline{\text{SRESET}}$ is also held low for the same duration. If $\overline{\text{SRESET}}$ is held low externally for more than 1024 cycles, internal soft reset is driven low for just 1024 clock cycles.

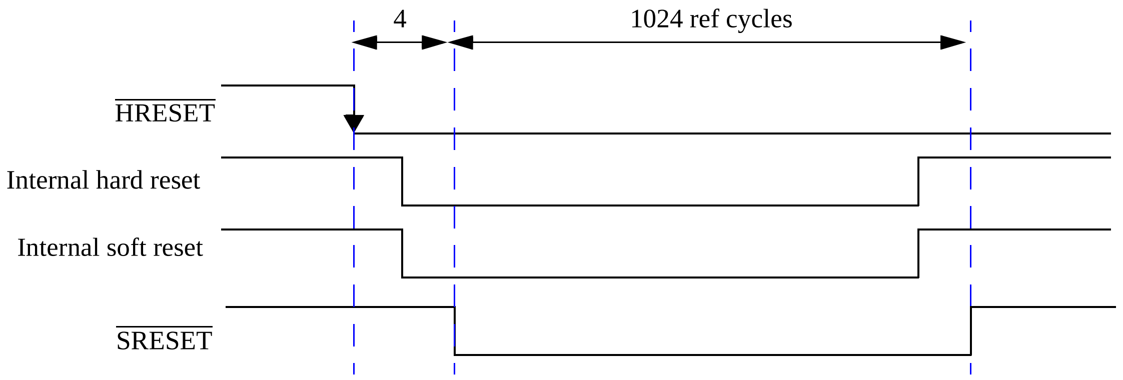


Figure 4-4 $\overline{\text{HRESET}}$ Asserted more than 1024 Reference Clock Cycles

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 for 1024 reference clock cycles.

4.5 Other Resets

MGT5100 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

Bits	Definition
$\overline{\text{PCI\_RESET}}$	PCI bus reset output. Generated by processor write to a PCI register
$\overline{\text{AC'971\_RES}}$	AC97 reset output. Generated from the AC'97 / PSC module
$\overline{\text{JTAG\_TRST}}$	JTAG reset input. Generated externally from JTAG or debug control logic. This input only resets the JTAG logic. Other system resets ($\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, and $\overline{\text{SRESET}}$) do not reset the JTAG logic.
$\overline{\text{ATA Reset}}$	This is NOT a reset pin on MGT5100. 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 MGT5100 is initialized by sampling values found on specific device pins during Power-ON-Reset ($\overline{\text{PORESET}}$). These pins are outputs in normal operation, but are sampled as inputs during POR. External pull-up or pull-down resistors on the board are used to force a value on these pins during POR. These values are latched into the CDM reset configuration register at the end of POR, then distributed to various modules in the design. After POR, these outputs overdrive the external pull-up or pull-down resistors and behave as functional outputs. Only during POR are these pins inputs.

Table 4-2 gives the POR configuration inputs.

Table 4-2 POR Configuration Word Source Pins

Pkg Ball	Reset Config Pin	I/O Signal Name	CDM Reset Config Register Bit	Config Signal from CDM	Description
$\overline{\text{Y18}}$	$\overline{\text{RST\_CFG0}}$	$\overline{\text{ATA\_DACK}}$	$\overline{\text{PORCFG[31]}}$	$\overline{\text{ppc\_pll\_cfg\_4}}$	MGT5100 G2 Harpo Core PLL Configuration Word
$\overline{\text{Y17}}$	$\overline{\text{RST\_CFG1}}$	$\overline{\text{ATA\_IOR}}$	$\overline{\text{PORCFG[30]}}$	$\overline{\text{ppc\_pll\_cfg\_3}}$	
$\overline{\text{W17}}$	$\overline{\text{RST\_CFG2}}$	$\overline{\text{ATA\_IOW}}$	$\overline{\text{PORCFG[29]}}$	$\overline{\text{ppc\_pll\_cfg\_2}}$	
$\overline{\text{W16}}$	$\overline{\text{RST\_CFG3}}$	$\overline{\text{LP\_RWB}}$	$\overline{\text{PORCFG[28]}}$	$\overline{\text{ppc\_pll\_cfg\_1}}$	
$\overline{\text{V14}}$	$\overline{\text{RST\_CFG4}}$	$\overline{\text{LP\_ALE}}$	$\overline{\text{PORCFG[27]}}$	$\overline{\text{ppc\_pll\_cfg\_0}}$	
$\overline{\text{Y13}}$	$\overline{\text{RST\_CFG5}}$	$\overline{\text{LP\_TS}}$	$\overline{\text{PORCFG[26]}}$	xlb_clk_sel	bit=0:XLB_CLK=SYS_PLL FVCO/4 bit=1:XLB_CLK=SYS_PLL FVCO/8
$\overline{\text{H02}}$	$\overline{\text{RST\_CFG6}}$	$\overline{\text{USB1\_1}}$	$\overline{\text{PORCFG[25]}}$	$\overline{\text{sys\_pll\_cfg\_0}}$	bit=0:SYS_PLL FVCO=16x SYS_PLL_FREF bit=1:SYS_PLL FVCO=12x SYS_PLL_FREF
$\overline{\text{H03}}$	$\overline{\text{RST\_CFG7}}$	$\overline{\text{USB1\_2}}$	$\overline{\text{PORCFG[24]}}$	$\overline{\text{sys\_pll\_cfg\_1}}$	bit reserved, pull low
$\overline{\text{K01}}$	$\overline{\text{RST\_CFG8}}$	ETH0	$\overline{\text{PORCFG[23]}}$	$\overline{\text{ipbi\_rst\_cfg}}$	bit reserved, pull low
$\overline{\text{K02}}$	$\overline{\text{RST\_CFG9}}$	ETH1	$\overline{\text{PORCFG[22]}}$	$\overline{\text{ppc\_tle}}$	bit reserved, pull low
$\overline{\text{K03}}$	$\overline{\text{RST\_CFG10}}$	ETH2	$\overline{\text{PORCFG[21]}}$	$\overline{\text{ppc\_msrip}}$	microprocessor Boot Address/Exception table location. bit=0:0000_0100 (hex) bit=1:FFF0_0100 (hex)
$\overline{\text{J01}}$	$\overline{\text{RST\_CFG11}}$	$\overline{\text{ETH3}}$	$\overline{\text{PORCFG[20]}}$	$\overline{\text{boot\_rom\_wait}}$	bit=0:4 IP bus clocks of wait state* bit=1:48 IP bus clocks of wait state*
$\overline{\text{J02}}$	$\overline{\text{RST\_CFG12}}$	$\overline{\text{ETH4}}$	$\overline{\text{PORCFG[19]}}$	$\overline{\text{boot\_rom\_swap}}$	bit=0:no byte lane swap, same endian ROM image bit=1:byte lane swap, different endian ROM image

Table 4-2 POR Configuration Word Source Pins (continued)

Pkg Ball	Reset Config Pin	I/O Signal Name	CDM Reset Config Register Bit	Config Signal from CDM	Description
$\overline{\text{L03}}$	$\overline{\text{RST\_CFG13}}$	$\overline{\text{ETH5}}$	$\overline{\text{PORCFG[18]}}$	$\overline{\text{boot\_ram\_size}}$	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
$\overline{\text{N02}}$	$\overline{\text{RST\_CFG14}}$	$\overline{\text{ETH6}}$	$\overline{\text{PORCFG[17]}}$	$\overline{\text{boot\_ram\_type}}$	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.
$\overline{\text{N01}}$	$\overline{\text{RST\_CFG15}}$	$\overline{\text{ETH7}}$	$\overline{\text{PORCFG[16]}}$	$\overline{\text{Reserved}}$	Reserved

NOTE:
1. The external bus clock (pci_clk) is 1/2 the frequency of the internal bus clock (ipb_clk) at power-up. Therefore, 4 IP bus wait states translate to as little as 1 external wait state (i.e. peripheral must respond within 2 external clocks). The "slow" setting represents 48 IP bus 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 $\overline{\text{ALE}}$ & $\overline{\text{TS}}$ is 2 IP bus clocks (i.e. 1 external clock). This represents the "wide $\overline{\text{ALE}}$" 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 $\overline{\text{ALE}}$ width setting (in LPC) *after* adjusting the clock relationship. Any fetches to the boot device between these two settings results in $\overline{\text{ALE}}$ and TS being 2 external clocks wide.
3. Another feature for muxed boot ROM types, if they have the ability to generate an $\overline{\text{ACK}}$ signal, is $\overline{\text{ACK}}$ can be used to shorten the number of wait states. If not using $\overline{\text{ACK}}$, the signal to MGT5100 MUST be in the high state. Note the use of $\overline{\text{ACK}}$ can only shorten the Chip Select low period, *not* extend it

SECTION 5
CLOCKS AND POWER MANAGEMENT

5.1 Overview

The following sections are contained in this document:

- Clock Distribution Module (CDM)
- MGT5100 Clock Domains
- Clock Relationships
- Power Management
- CDM Registers—MBAR+0x0200

5.2 Clock Distribution Module (CDM)

The CDM is the source of all internally generated clocks and reset signals. The MGT5100 clock generation uses two APLL blocks. The first 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

Bits	Description
xlb_clk	Microprocessor on-chip 64-bit XL bus clock. This is the fundamental MGT5100 frequency.
mem_clk	SDRAM Controller memory clock supplied to external SDRAM devices. Max frequency is 132MHz.
mem_2x_clk	XL bus clock times 2 for SDRAM Controller.
$\overline{\text{mem\_2x\_clk}}$	XL bus clock times 2 for SDRAM Controller inverted.
ipb_clk	Intellectual Property Bus (IPB) clock.
pci_clk	PCI Controller clock.
48mhz_clk	48MHz clock for USB and IrDA. This clock can be sourced from the internal CDM or from an external source via the IrDA_USB_CLK pin.

The second APLL is contained within the G2 core processor. The second APLL takes the processor bus clock (xlb_clk) and generates the G2 processor core clock, core_clk.

5.3 MGT5100 Clock Domains

The MGT5100 has 5 major clock domains, which are listed below. Details are given in the sections that follow.

- G2 Clock Domain—internal processor core frequency
- Processor Bus or XL Bus Clock Domain—internal G2 processor bus
- SDRAM Memory Controller Clock Domain
- IP Bus 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 MGT5100:

Ethernet—The Ethernet Controller requires a 25MHz or 50MHz Tx/Rx clock. Both clocks are inputs to MGT5100, supplied from the Ethernet physical device (ETH_RXCLK, ETH_TXCLK pins). The Ethernet Controller Tx/Rx portion on MGT5100 is asynchronous to the rest of MGT5100.

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 is 48MHz. When the clock source is externally supplied, the USB module Tx/Rx portion is asynchronous to the rest of MGT5100.

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, clock frequency is 48MHz.
- When generated externally, the frequency can be different.
- When the clock source is supplied externally, the IrDA module Tx/Rx portion is asynchronous to the rest of MGT5100.

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

PSC—The Programmable Serial Controller module is instantiated in the MGT5100 4 times (3 PSCs and 1 IrDA module). 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). PSC logic is therefore asynchronous to the rest of the chip.

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 Inter-Integrated Circuit modules on MGT5100. Both have input source clocks (I^{2}Cx_CLK) and therefore asynchronous clock domains.

RTC—The Real-Time Clock has its own clock domain, which is clocked by an external 32-KHz oscillator. The two oscillator pins are RTC_XTAL_IN and RTC_XTL_OUT. There is an asynchronous boundary between this clock domain and the IP bus register interface.

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

The following peripheral functions use clocks generated from CDM.

IR—The Infrared module uses a clock created by a baud rate generator in the IR block. The source clock is ipb_clk. The resultant clock samples the incoming IR data stream and generates the outgoing IR data stream (IR Blast).

MSCAN—The Motorola Scalable Controller Area Network module uses a clock created by an internal baud rate generator. The source clock is ipb_clk. The resultant clock samples an incoming CAN data stream and generates an outgoing data stream.

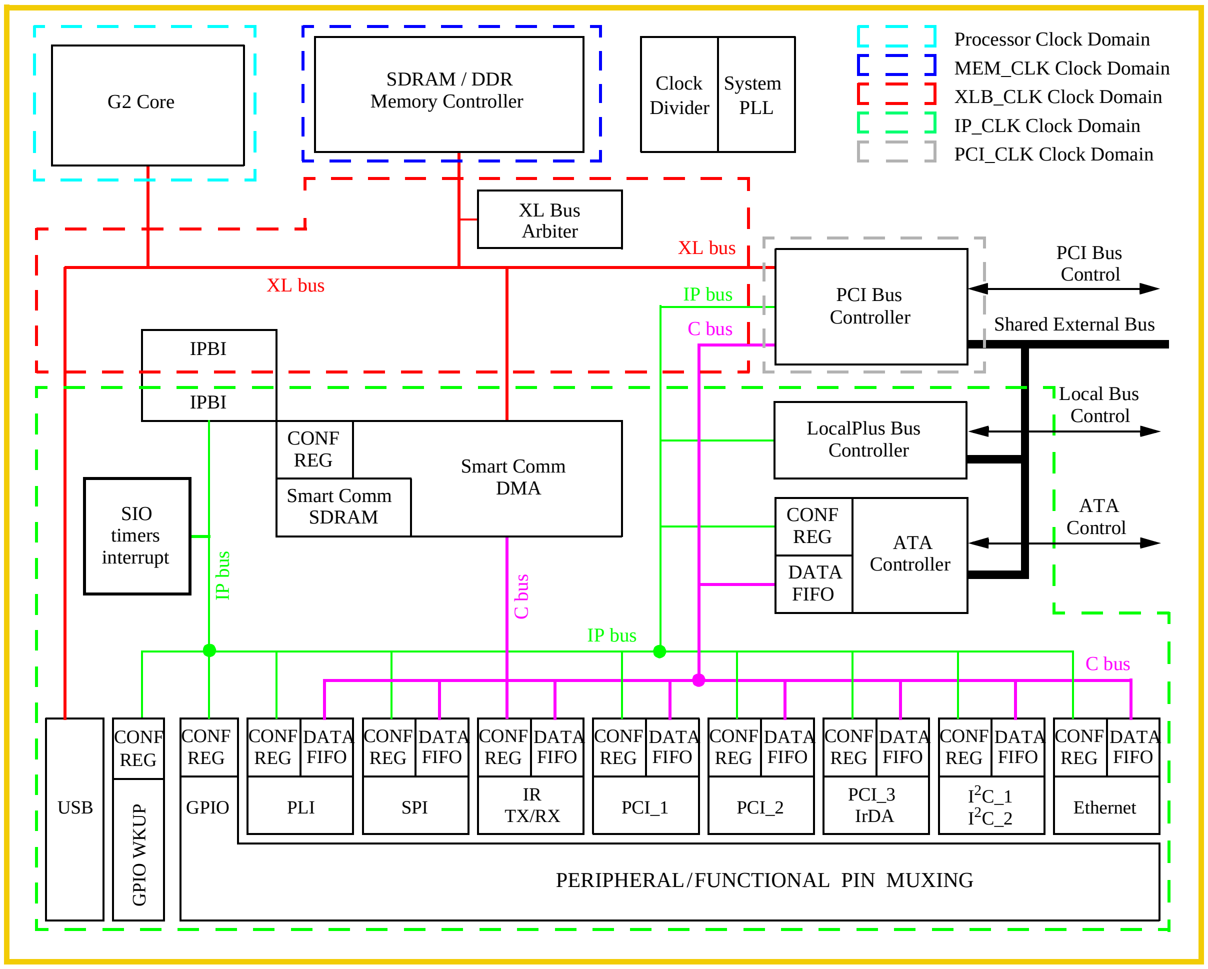


Figure 5-1 Primary Synchronous Clock Domains

5.3.1 G2 Clock Domain

G2 has its own APLL and clock domain, which is separate from the rest of the chip. The reference for processor APLL is xlb_clk. G2 can run at multiples of xlb_clk (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 300 MHz. Table 5-2 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 5-2 G2 Frequencies vs xlb_clk Frequencies

XL Bus Clock (MHz)		132	108	99	81	66	54	49.5	40.5	33	27
HARPO PPL Bus to Core Multiplier	**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		324	297	243	198	162	148.4	121.5	99	81
	x3.5			346.5	283.5	231	189	173.3	141.8	115.5	94.5
	x4				324	264	216	198	162	132	108
	x4.5					297	243	222.8	182.3	148.5	121.5
	x5					330	270	247.5	202.5	165	135
	x5.5						297	272.3	222.8	181.5	148.5
	x6						324	297	243	198	162
	x6.5							321.8	263.3	214.5	175.5
	x7								283.5	231	189
	x7.5								303.8	247.5	202.5
	x8								324	264	216

NOTE: 1x and 1.5x multiply ratios are not available in this version of the MGT5100.

Table 5-3 gives the G2 APLL and operating frequency options compared to the xlb_clk reference input (shown in Figure 5-3). The selection of a G2 frequency is made at Power-ON Reset (POR) via the reset configuration inputs. For more information see Reset Configuration, Section 4.6.

Frequency ranges indicated in Table 5-3 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 Section A.2, AC Timing Specifications.

Table 5-3 G2 APLL Configuration Options

PPC Harpo Core Internal PLL Configuration					
hex	**ppc_pll_cfg [0:1:2:3:4]**	**Bus to Core Multiplier**	**G2 FVCO Divider**	**xlb_clk Frequency Range**	**G2 core_clk Frequency Range**
0x00	00000	—	—	—	—
0x01	00001	—	—	—	—
0x02	00010	—	—	—	—
0x03	00011	PLL off/bypassed		PLL off, BUS_CLK clocks core directly, 1x bus-to-core defaulted	
0x04	00100	2.0x	2	50–150MHz	100–300MHz
0x05	00101	2.0x	4	25–75MHz	50–150MHz
0x06	00110	2.5x	2	40–120MHz	100–300MHz
0x07	00111	4.5x	2	22–65MHz	100–300MHz
0x08	01000	3.0x	2	33–100MHz	100–300MHz
0x09	01001	5.5x	2	18–55MHz	100–300MHz
0x0A	01010	4.0x	2	25–75MHz	100–300MHz
0x0B	01011	5.0x	2	20–60MHz	100–300MHz
0x0C	01100	—	—	—	—
0x0D	01101	6.0x	2	30–85MHz	100–300MHz
0x0E	01110	3.5x	2	30–85MHz	100–300MHz
0x0F	01111	PLL off		PLL off, no core clocking occurs	
0x10	10000	3.0x	4	16–50MHz	50–150MHz
0x11	10001	2.5x	4	20–60MHz	50–150MHz
0x12	10010	6.5x	2	15–45MHz	100–300MHz
0x13	10011	PLL off/bypassed		PLL off, BUS_CLK clocks core directly, 1x bus-to-core defaulted	
0x14	10100	7.0x	2	14–43MHz	100–300MHz
0x15	10101	2.0x	4	25–75MHz	50–150MHz
0x16	1-11-	7.4x	2	13–40MHz	100–300MHz
0x17	10111	4.5x	2	22–65MHz	100–300MHz
0x18	11000	—	—	—	—
0x19	11001	5.5x	2	18–55MHz	100–300MHz
0x1A	11010	4.0x	2	25–75MHz	100–300MHz
0x1B	11011	5.0x	2	20–60MHz	100–300MHz
0x1C	11100	8.0x	2	12–38MHz	100–300MHz
0x1D	11101	6.0x	2	16–50MHz	100–300MHz
0x1E	11110	3.5x	2	30–85MHz	100–300MHz
0x1F	11111	PLL off		PLL off, no core clocking occurs.	
	NOTE: Shading implies same mode can be configured with ppc_pll_cfg[4]=0				

5.3.2 Processor Bus or XL Bus Clock Domain

The XL bus (xlb_clk) is the fundamental MGT5100 clock frequency. The following operate at this frequency:

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

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

5.3.3 SDRAM Memory Controller Clock Domain

The Memory Controller uses the clocks shown in Table 5-4.

Table 5-4 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 clocks are twice the frequency of xlb_clk and are used to add more resolution to SDRAMC control signals
mem2x1x_clk	This is the memory read clock. It is phase shifted to allow more set-up time in the memory read path. It is also fed through an I/O buffer (on-chip) to facilitate phase alignment with the external SDRAM control and data signals. This clock is the same frequency as xlb_clk for SDR memories and the same frequency as mem_2x_clk for DDR memories.

Figure 5-2 shows the clock relationships for the SDRAM Controller.

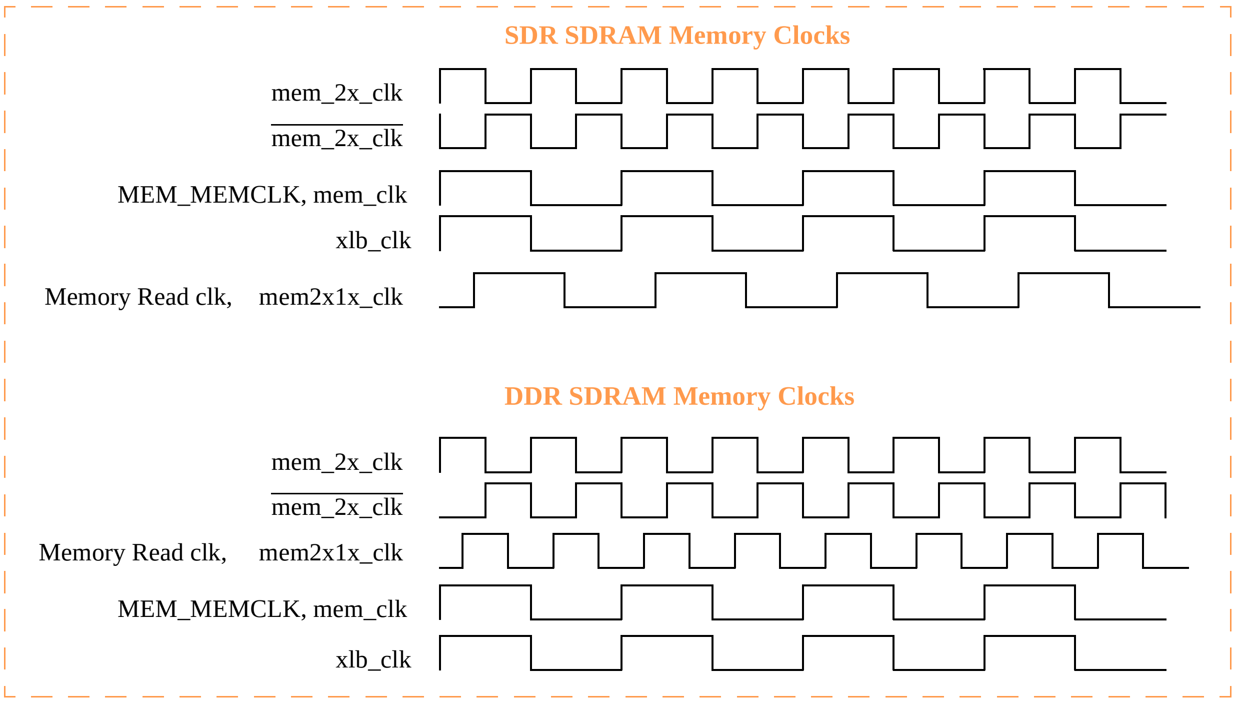

SDR SDRAM Memory Clocks
mem_2x_clk
$\overline{\text{mem\_2x\_clk}}$
MEM_MEMCLK, mem_clk
xlb_clk
Memory Read clk, mem2x1x_clk
DDR SDRAM Memory Clocks
mem_2x_clk
$\overline{\text{mem\_2x\_clk}}$
Memory Read clk, mem2x1x_clk
MEM_MEMCLK, mem_clk
xlb_clk

Figure 5-2 Timing Diagram—Clock Waveforms for SDRAM and DDR Memories

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

MGT5100 supplies 2 external memory clocks as part of the SDRAM interface:

- MEM_MEMCLK
- $\overline{\text{MEM\_MEMCLK}}$

These 2 clocks are always the same frequency as xlb_clk.

5.3.4 IP Bus Clock Domain

IP bus clock (ipb_clk) can run at the same frequency or 1/2 the frequency of xlb_clk. SmartComm DMA runs at the ipb_clk frequency as does all IP bus control register access logic.

5.3.5 PCI Clock Domain

The PCI bus clock (pci_clk) is the fundamental frequency of the PCI bus interface. pci_clk can run at the xlb_clk frequency, at 1/2 the xlb_clk frequency or at 1/4 the xlb_clk frequency.

5.4 Clock Relationships

Figure 5-3 shows the CDM clock divide circuitry.

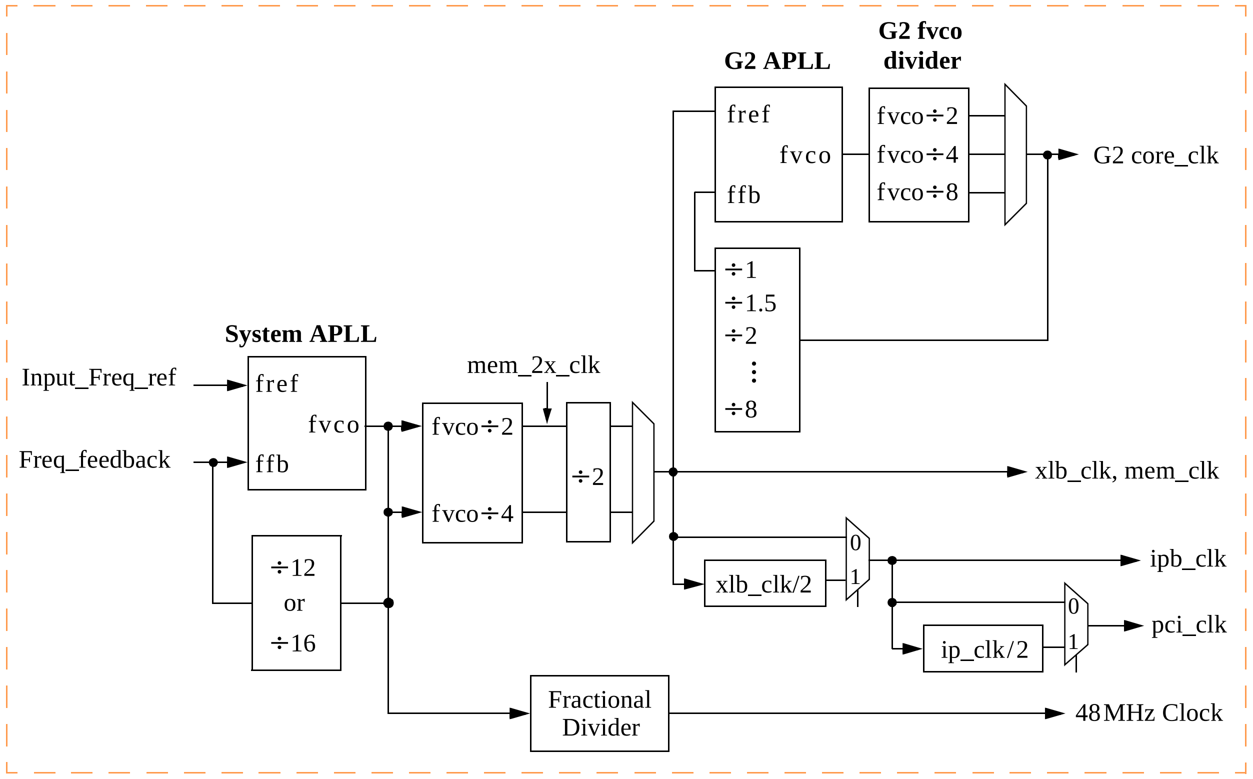


Figure 5-3 Bus Clock Ratios

Table 5-5 shows possible internal frequencies, given a reference of 33MHz.

NOTE: Frequencies ranges indicated in Table 5-5 and Table 5-6 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.

Table 5-5 Typical System Clock Frequencies with 33MHz Input

Input Freq_ref	System PLL FVCO	System PLL fb Divide	2x Mem Clock (mem_2x_clk)	Mem Clock (mem_clk)	XLB Clock (xlb_clk)	IP Clock (ipb_clk)	PCI Clock (pci_clk)	IrDA/USB 48MHz	Description
33	528	16	264 (÷2)	132	132	66	66÷33	fvco÷11	Fast SDRAM, Fast PCI
33	528	16	132 (÷4)	66	66	66	66÷33	fvco÷11	Slow SDRAM, Fast SDMA

Table 5-5 Typical System Clock Frequencies with 33 MHz Input (continued)

Input Freq_ref	System PLL FVCO	System PLL fb Divide	2x Mem Clock (mem_2x_clk)	Mem Clock (mem_clk)	XLB Clock (xlb_clk)	IP Clock (lpb_clk)	PCI Clock (pci_clk)	IrDA/USB 48MHz	Description
33	528	16	132 (÷4)	66	66	33	33÷16.5	fvco÷11	Slow SDRAM, Slow SDMA, PCI
33	396	12	198 (÷2)	99	99	49.5	49.5÷24.75	fvco÷8.25	Medium SDRAM, SDMA, PCI

Table 5-6 shows possible internal frequencies, given a reference of 27 MHz.

Table 5-6 Typical System Clock Frequencies with 27 MHz Input

Input Freq_ref	System PLL FVCO	System PLL fb Divide	2x Mem Clock (mem_2x_clk)	Mem Clock (mem_clk)	XLB Clock (xlb_clk)	IP Clock (lpb_clk)	PCI Clock (pci_clk)	IrDA/USB 48MHz	Description
27	432	16	216 (÷2)	108	108	54	54÷27	fvco÷9	Fast SDRAM, Fast PCI
27	432	16	108 (÷4)	54	54	54	54÷27	fvco÷9	Slow SDRAM, Fast SDMA
27	432	16	108 (÷4)	54	54	27	27÷13.5	fvco÷9	Slow SDRAM, Slow SDMA, PCI
27	324	12	162 (÷2)	81	81	40.5	40.5÷20.25	fvco÷6.75	Medium SDRAM, SDMA, PCI

5.5 Power Management

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

- Full-Power Mode
- Power Conservation Modes

The MGT5100 design is equipped with many power conservation features, which are supported in the peripherals and system logic. The G2 processor has its own power-down modes:

- nap
- doze
- sleep

Individual peripheral functions can be disabled by stopping the module clock. Clock control sequencer (CCS) logic, sequences the MGT5100 clock system to enter and exit a deep-sleep power mode. This limits power consumption to device leakage levels.

The MGT5100 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 G2 core uses the XL bus 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 power domain and drives the RTC module.

5.5.1 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 MGT5100 system. In this mode, the G2 core may use the Dynamic Power Mode (DPM). If this occurs, logic not required for instruction execution is not activated. This results in a power reduction over a design that would be fully clocked during normal operation.

Performance in not decreased in Dynamic Mode. This means it should never be disabled (although it is possible) when running the core at full speed.

MGT5100 peripherals can be individually enabled based on what functionality is required by the application running and the external stimulus presented to MGT5100. Peripherals not required, can be powered-down through a write to an MGT5100 system control register. This disables the peripheral and gates the peripheral clock.

5.5.2 Power Conservation Modes

Sleep modes in the MGT5100 design can be exercised through microprocessor sleep mode control or powering-down peripherals. Since the OSC, system PLL and microprocessor PLL remain locked, the response time to WakeUP interrupts is faster than in the deep-sleep mode (see Deep-Sleep Mode, Section 5.5.4). Since clocks are still running in the MGT5100 chip, any interrupt normally present in the MGT5100 design can be used to wake up the power-down logic. See Section 5.6.6, Clock Enable register.

5.5.3 G2 Processor Power Modes

The G2 processor 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 G2 control registers. These modes only apply to the G2 processor. Logic outside the G2 core remains active. In any of these modes, peripherals can be enabled or disabled by writing to an MGT5100 system control register.

5.5.3.1 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

5.5.3.2 Doze Mode

All functional G2 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 brings the core into the 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 external clock input (SYSCLK). Transition to Full-Power Mode takes only a few processor clock cycles.

5.5.3.3 Nap Mode

The Nap Mode further reduces G2 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 Mode:

- 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.

5.5.3.4 Sleep Mode

Sleep Mode reduces G2 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 G2 core PLL, further reducing power. this requires special sequencing logic external to the G2 core and is discussed in Section 5.5.4.

5.5.4 Deep-Sleep Mode

The MGT5100 system provides a very low power consumption mode where the 27/33MHz system oscillator, system PLL and G2 Processor PLL are powered down and disabled. Once MGT5100 is sequenced into this mode and clocks are static, the current draw of the device 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 a 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 MGT5100 using a RTC interrupt. However, no clock is required to trigger the wake up process in the case of the external interrupt or the MSCAN module interrupt. This means the RTC clock does not have to be present to use Deep Sleep mode. The G2 Processor must enable the deep sleep process in the CDM module, then put itself into sleep mode before the G2 Processor PLL can be disabled.

Since MGT5100 clocks are stopped in Deep Sleep mode, the wake-up time is longer than in the G2 Processor-only power down modes. A power-on sequence must occur which relocks both the MGT5100 system and processor PLLs.

The sequence of events to enter and exit Deep Sleep mode are initiated by the G2 Processor under software control and then sequenced in hardware by the Clock Control Sequencer (CCS) in CDM.

5.5.4.1 Entering Deep Sleep

When entering Deep Sleep mode, the following occurs:

- G2 Processor 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.
- G2 Processor finishes instructions in execution pipeline.
- G2 Processor software enables the Deep Sleep mode with a write to a MGT5100 control register.
- G2 Processor software writes sleep mode configuration to G2 Processor control register.
- G2 Processor asserts the QREQ signal indicating that it would like to enter sleep mode.
- CCS waits for G2 Processor sleep (initiated by QREQ, since QACK is always asserted in MGT5100).
- CCS disables interrupts.
- CCS waits for the G2 Processor to enter the sleep mode.
- CCS disables the OSC, system PLL, G2 Processor PLL and gates the system clocks.

5.5.4.2 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 G2 Processor PLL and waits for the PLL to lock to the system PLL clock.
- CCS enables the SDRAM controller and waits for SDRAM system to power up.
- CCS enables interrupts, which triggers a wakeup interrupt to the G2 Processor (from the WakeUp source).
- G2 Processor wakes up and puts MGT5100 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 MGT5100 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.

5.6 CDM Registers—MBAR+0x0200

CDM uses 10 32-bit registers. All registers are located at an offset from the Module Base Address Register (MBAR). MBAR is 0x0000. The CDM offset is 0x0200. Register addresses are relative to the MBAR offset. Therefore, the actual register address is:
MBAR + 0x0200 + register address

Hyperlinks to the CDM registers are provided below:

- CDM JTAG ID Number (0200)—JTAGID: 01C5301D hex, read-only
- CDM Power ON Reset Configuration (0204)—PORCFG, read-only
- CDM Bread Crumb (0208)—BC, never reset
- CDM Configuration (020C)—CFG, R/W
- CDM 48MHz Fractional Divider Configuration (0210)—FDCFG, R/W
- CDM Clock Enable (0214)—CLKEN, R/W
- CDM System Oscillator Configuration (0218)—OSCCFG, R/W
- CDM Clock Control Sequencer Configuration (021C)—CCSCFG, R/W
- CDM Soft Reset (0220)—SFTRST, R/W
- CDM System PLL Status (0224)—PLLSTA (33–27MHz), R/W

5.6.1 JTAG ID Number (0200)—JTAGID

The CDM JTAG ID Number Register is a read-only register that contains the JTAG Identification number identifying MGT5100. The value is hard coded (01C5301D hex) and cannot be modified.

Table 5-7 CDM JTAG ID Number (0200)—JTAGID: 01C5301D hex

	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	1	1	1	0	0	0	1	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	1	1	0	0	0	0	0	0	0	1	1	1	0	1

Device I.D. Register = 01C5301D hex

Version	Device (MGT5100 – Initial Release)	Manufacturer (Motorola)	
000 2	0001 1100 0101 0011	0000 0001 110	1

5.6.2 Power ON Reset Configuration (0204)—PORCFG

This is a read-only register containing the configuration value latched at POR.

Table 5-8 CDM Power ON Reset Configuration (0204)—PORCFG

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W	Unused															
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	boot_ ram_type	boot_ ram_size	boot_ ram_swap	boot_ ram_wait	ppc_msrip	ppc_tle	Reserved	Reserved	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	Unused															
RESET:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Bit	Name	Description
0–16	—	Reserved
17	boot_ram_type	
18	boot_ram_size	
19	boot_ram_swap	
20	boot_ram_wait	
21	ppc_msrip	
22	ppc_tle	
23–24	—	Reserved
25	sys_pll_cfg_0	
26	xlb_clk_sel	
27	ppc_pll_cfg_0	
28	ppc_pll_cfg_1	
29	ppc_pll_cfg_2	
30	ppc_pll_cfg_3	
31	ppc_pll_cfg_4	

5.6.3 Bread Crumb (0208)—BC

The CDM Bread Crumb Register is a 32-bit register that is not reset. Its purpose is to let firmware designers to 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 5-9 CDM Bread Crumb (0208)—BC

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

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

5.6.4 Configuration (020C)—CFG

The CDM Configuration Register contains 2bits that set IPB_CLK and PCI_CLK ratios.

Table 5-10 CDM Configuration (020C)—CFG

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved Program 0							ddr_mode	Reserved Program 0							xlb_ clk_sel
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	Reserved Program 0							ipb_ clk_sel	Reserved Program 0							pci_ clk_sel
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. Program 0.
7	ddr_mode	SDRAM Controller DDR memory mode, read-only. bit 0=configures SDRAM Controller for SDR SDRAM (single data rate) bit 1=configures SDRAM Controller for DDR SDRAM (double data rate) This register location is a read-only status bit. The controlling register is in the SDRAM Controller register map. In the CDM this bit determines the frequency and phase of mem_2x1x_clk (memory read clock)
8–14	—	Reserved for future use. Program 0.

Bit	Name	Description
15	xlb_clk_sel	XLB Clock Frequency bit 0 –XLB_CLK = CDM_SYS_PLL_FVCO/4 bit 1 –XLB_CLK = CDM_SYS_PLL_FVCO/8 This register location is a read-only status bit. The controlling register is the POR Configuration register - cdm_reg1 [26]. In the CDM this bit determines if MEM_2X_CLK is FVCO/2 or FVCO/4. NOTE: This bit is currently incorrect due to an rtl coding error. It reflects cdm_reg1[27] (ppc_pll_cfg[0]), instead of cdm_reg1[26] (xlb_clk_sel).
16–22	—	Reserved for future use. Program 0.
23	ipb_clk_sel	IPB Clock Select bit 0 –IPB_CLK = XLB_CLK bit 1 –IPB_CLK = XLB_CLK/2
24–30	—	Reserved for future use. Program 0.
31	pci_clk_sel	IPB Clock Select bit 0 –PCI_CLK = IPB_CLK bit 1 –PCI_CLK = IPB_CLK/2

5.6.5 48 MHz Fractional Divider Configuration (0210)—FDCFG

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

Table 5-11 CDM 48 MHz Fractional Divider Configuration (0210)—FDCFG

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved Program 0						ext_usb_ 48mhz_en	ext_irda_ 48mhz_en	Reserved Program 0							fd_en
RESET:	0	0	0	0	0	0	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 / W	fd_phase_3_count				fd_phase_2_count				fd_phase_1_count				fc_phase_0_count			
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. Program 0.
6	ext_usb_48MHz_en	USB External 48 MHz Clock Select Setting bit to 1 drives 48 MHz clock tree for IRDA and USB with external clock from GPIO. Setting bit to 0 drives 48 MHz clock tree for IRDA and USB from CDM Fractional Divider.

Bit	Name	Description
7	ext_irda_48MHz_en	IrDA External 48MHz Clock Select Setting bit to 1 drives 48MHz clock tree for IRDA and USB with external clock from GPIO. Setting bit to 0 drives 48MHz clock tree for IRDA and USB from CDM Fractional Divider.
8–14	—	Reserved for future use. Program 0.
15	fd_en	CDM 48MHz Fractional Divider Enable Setting bit to 1 enables Fractional Divide Circuitry. Setting bit to 0 disables Fractional Divide Circuitry.
16–19	cgfd_p3_cnt[3:0]	These fields hold 4 phase divide ratios used by the fractional divider. This field is incompletely decoded; bit3 is unused and fvco/11 is obtained with 6 values. bits X110=fractional counter divide ration of fvco_clk/6 bits X111=fractional counter divide ration of fvco_clk/7 bits X000=fractional counter divide ration of fvco_clk/8 bits X001=fractional counter divide ration of fvco_clk/9 bits X010=fractional counter divide ration of fvco_clk/10 bits X011=fractional counter divide ration of fvco_clk/11 bits X10X=fractional counter divide ration of fvco_clk/11
20–23	cgfd_p2_cnt[3:0]	
24–27	cgfd_p1_cnt[1:0]	
28–31	cgfd_p0_cnt[1:0]	

5.6.6 Clock Enable (0214)—CLKEN

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 5-12 CDM Clock Enable (0214)—CLKEN

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved Program 0												mem_ clk_en	pci_ clk_en	lpc_ clk_en	slt_ clk_en
RESET:	0	0	0	0	0	0	0	0	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 W	scom_ clk_en	ata_ clk_en	eth_ clk_en	usb_ clk_en	spi_ clk_en	pli_ clk_en	irrx_ clk_en	irtx_ clk_en	psc3_ clk_en	psc2_ clk_en	psc1_ clk_en	irda_ clk_en	mscan_ clk_en	i2c_ clk_en	timer_ clk_en	gpio_ clk_en
RESET:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	Name	Description
0–11	—	Reserved for future use. Program 0.
12	mem_clk_en	Memory Clock Enable—controls clocks going to the SDRAM Controller module: mem_clk, mem_2x1x_clk, mem_2x_clk, mem_2x_clkb NOTE: Memory Controller IPB_CLK “mem_ipb_clk”, not controlled by mem_clk_en.

Bit	Name	Description
13	pci_clk_en	PCI Bus Clock Enable—controls clocks going to the PCI bus control module: pci_xlb_clk, pci_ipb_clk, pci_clk
14	lpc_clk_en	Local Plus Bus Clock Enable—controls IPB_CLK clock going to the LP bus control module: lpc_clk
15	slt_clk_en	Slice Timer Clock Enable—controls IPB_CLK clock going to the slice timer module; slt_clk
16	scom_clk_en	Smart Comm Clock Enable—controls IPB_CLK clock going to the smart comm module: scom_clk
17	ata_clk_en	ATA Clock Enable—controls IPB_CLK clock going to the ATA disk drive control module: ata_clk
18	eth_clk_en	Ethernet Clock Enable—controls IPB_CLK clock going to the Ethernet Controller module: eth_clk
19	usb_clk_en	Universal Serial Bus Clock Enable—controls IPB_CLK clock going to the USB module: usb_clk
20	spi_clk_en	SPI Clock Enable—controls IPB_CLK clock going to the SPI module: spi_clk
21	pli_clk_en	PLI Clock Enable—controls IPB_CLK clock going to the PLI module: pli_clk
22	irrx_clk_en	Infrared Receive Clock Enable—controls IPB_CLK clock going to the IRRX module: irrx_clk
23	irtx_clk_en	Infrared Transmit Clock Enable—controls IPB_CLK clock going to the IRTX module: irtx_clk
24	psc3_clk_en	PCI #3 Clock Enable—controls IPB_CLK clock going to the #3 PSC module: psc3_clk
25	psc2_clk_en	PCI #2 Clock Enable—controls IPB_CLK clock going to the #2 PSC module: psc2_clk
26	psc1_clk_en	PCI #1 Clock Enable—controls IPB_CLK clock going to the #1 PSC module: psc1_clk
27	irda_clk_en	IRDA Clock Enable—controls clocks going to the IRDA module: irda_clk (IPB_CLK), irda_48mhz_clk (48MHz clock).
28	mscan_clk_en	MSCAN Clock Enable—controls two IPB_CLK clocks going to the MSCAN module: mscan_clk, mscan_clk_b (inverted version of mscan_clk)
29	i2c_clk_en	I2C Clock Enable—controls IPB_CLK clock going to the I^2C module: i2c_clk.
30	timer_clk_en	Timer Clock Enable—controls IPB_CLK clock going to the timer module: timer_clk TIME_CLK_EN runs at IP_CLK frequency, partial disable of timer block. 2 timers for wake-up mode do not have gated clocks.
31	gpio_clk_en	GPIO Clock Enable—controls IPB_CLK clock going to some GPIO modules: gpio_clk GPIO wake-up mode circuitry uses free running IPB_CLK: ipb_clk.
NOTE: Enable value 1, enables the corresponding clock. Enable value 0, disables corresponding clock.		

5.6.7 System Oscillator Configuration (0218)—OSCCFG

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 system oscillator is disabled to reduce power consumption (~6 mW for system oscillator).

Table 5-13 CDM System Oscillator Configuration (0218)—OSCCFG

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved Program 0							sys_osc_disable	Reserved Program 0							
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 Program 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. Program 0.
7	sys_osc_disable	CDM System Oscillator Disable bit 1 = System Oscillator is disabled. External clock oscillator source is being used. bit 0 = System Oscillator is enabled. 27–33 MHz crystal is being used.
8–31	—	Reserved for future use. Program 0.

5.6.8 Clock Control Sequencer Configuration (021C)—CCSCFG

This register contains the configuration that controls the CCS module. The CCS module lets MGT5100 enter deep sleep power down mode (all clocks stopped).

Table 5-14 CDM Clock Control Sequencer Configuration (021C)—CCSCFG

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved Program 0							ccs_sleep_en	Reserved Program 0							ccs_osc_sleep_en
RESET:	0	0	0	0	0	0	0	1	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 / W	Reserved Program 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. Program 0.
7	ccs_sleep_en	CCS Module Enable bit 1=CCS enabled. G2 Harpo Core QREQ signal triggers deep sleep cycle. bit 0=CCS disabled and inactive. No deep sleep mode possible.
8–14	—	Reserved for future use. Program 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. Program 0.
31	ccs_qreq_test	CCS Test bit—Used in CCS module functional simulation to simulate a QREQ signal. bit 0=OREQ input to CCS forced active. bit 1=QREQ input to CCS comes directly from G2 Harpo Core.

5.6.9 Soft Reset (0220)—SFTRST

This register contains a soft reset register bit. Writing 1 causes a soft reset. The resulting soft reset condition then resets this bit.

Table 5-15 CDM Soft Reset (0220)—SFTRST

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Program 0							cdm_ soft_reset	Reserved Program 0							
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 Program 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. Program 0.
7	cdm_soft_reset	CDM Soft Reset bit. bit 0=requests CDM soft reset. bit 1=CDM soft reset request inactive.
8–31	—	Reserved for future use. Program 0.

5.6.10 System PLL Status (0224)—PLLSTA (33-27MHz)

This register contains control bits for the CDM PLL lock detect module.

Table 5-16 CDM System PLL Status (0224)—PLLSTA (33–27 MHz)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved Program 0							pll_lock	Reserved Program 0							pll_small_ lock_window
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	Reserved Program 0							pll_small_ lock_window	Reserved Program 0							
W																
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. Program 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. Program 0.
15	pll_loose_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. Program 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 FVCO clock periods. FFB period 12 or 16 FVCO clks. bit 0 = lock window pulse width 4 FVCO clock periods. FFB period 12 or 16 FVCO clks.
24–31	—	Reserved for future use. Program 0.

NOTE:
1. System PLL Lock Condition—1024 System PLL FREF clock rising edges within PLL_Lock_Window (System PLL FFB rising edge). In PLL bypass mode, Lock is active after 1024 System Oscillator clock rising edges.
2. In current MGT5100 CDM the PLL Lock Circuitry is for information only. CDM does not wait for PLL lock to start clocks or use PLL_LOOSE_LOCK as an interrupt source.

SECTION 6
G2 PROCESSOR CORE

6.1 Overview

The following sections are contained in this document:

- MGT5100 G2 Processor Core
- Arbiter Registers—MBAR+0x0080

6.2 MGT5100 G2 Processor Core

The MGT5100 integrates a G2 processor core based on, and compatible with, the 603e which is a PowerPC compliant microprocessor. The G2 core is completely embedded. For example, its address, data and control signals are not visible external to MGT5100. The G2 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 core, refer to 603e user documentation.

The G2 processor has a 32-bit address/64-bit data bus refered to as the XL bus. This bus is the main system connecting all internal mastering modules. In addition to the G2 core, the USB host controller, PCI controller (as target) and SmartComm DMA controller can master the XL bus.

The G2 core fetches 64-bit instructions. After power-on reset, initial boot instructions are fetched from the local bus, with CS0 active. The processor can execute code from the local bus, (only during boot) or from the SDRAM controller. To facilitate high speed execution, boot code is typically copied from a Flash or ROM device attached to the local bus, to SDRAM. The G2 core cannot execute code from the on-chip SRAM.

The G2 core has memory mapped access to all MGT5100 resources including:

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

When a master device wants access to the XL bus, a request is made to the XL bus arbiter. When access is granted, the mastering device controls the XL bus in an address tenure and a data tenure.

Bursting is supported on the XL bus. Critical word first protocol is employed when the G2 core attempts to fill its address and data caches from SDRAM.

6.3 Arbiter Registers—MBAR+0x0080

The Arbiter uses 12 32-bit active registers. The other registers are reserved. All registers are located at an offset from the Module Base Address Register (MBAR). MBAR is 0x0000. The Arbiter offset is 0x0080. Register addresses are relative to the MBAR offset. Therefore, the actual register address is: **MBAR + 0x0080 + 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 and read 0.
- Registers may be accessed on the following aligned boundaries:
  - 1 byte
  - 2 byte
  - word (32 bit)
  - double-word (64 bit)

Registers are functionally 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

Hyperlinks to the Arbiter registers are provided below:

- Arbiter Configuration (00C0)—ACFG, R/W
- Arbiter Version (00C4)—VER, read-only
- Arbiter Status (00C8)—STA, R/W
- Arbiter Interrupt Enable (00CC)—INTEN, R/W
- Arbiter Address Capture (00D0)—ADRCAP, read-only
- Arbiter Bus Signal Capture (00D4)—SIGCAP, R/W
- Arbiter Address Tenure Time-Out (00D8)—ADRTO, R/W
- Arbiter Data Tenure Time-Out (00DC)—DATTO, R/W
- Arbiter Bus Activity Time-Out (00E0)—BUSTO, R/W
- Arbiter Master Priority Enable (00E4)—PRIEN, R/W
- Arbiter Master Priority (00E8)—PRI, R/W
- Base Address (00EC)—BA, R/W
- Reserved Registers (0080–00BF, 00EC, 00F0–00FC), read-only

6.3.1 Configuration Register (00C0)—ACFG

This read/write register is used to enable watchdog and arbiter protocol functions.

Table 6-1 Arbiter Configuration (00C0)—ACFG

	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												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
0:27	—	Reserved
28	BA	Bus Activity Time-Out Enable—If enabled, the arbiter sets the Bus Activity Time-Out Status bit (Arbiter Status Register bit 29) when the Data Tenure 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 Transfer Error Acknowledge (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 sets the Data Tenure Time-Out Status bit (Arbiter Status Register bit 30). Setting this bit enables the Address Tenure Time-Out. This is required to ensure a data time-out does not occur before an address acknowledge.

Bit	Name	Description
30	AT	Address Tenure Time-Out Enable—If enabled, the arbiter will Address Acknowledge (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. The arbiter also sets the Address Tenure Time-Out Status bit (Arbiter Status Register bit 31). Address Tenure Time-Out is enabled by the DT bit.
31	—	Reserved

6.3.2 Version Register (00C4)—VER

This read-only register holds the silicon version value for the Arbiter hardware.

Table 6-2 Arbiter Version (00C4)—VER

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VER															
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	VER															
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.

6.3.3 Status Register (00C8)—STA

This read/write 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 0 into that bit position. Even if the causal condition is removed, the bit remains set until cleared.

Table 6-3 Arbiter Status (00C8)—STA

	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								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
0:23	—	Reserved
24	MM	Multiple Masters—at priority 0. If more than 1 master is recognized at priority 0, this bit is set. Once set, this bit remains set until cleared. The Arbiter recognizes priority by the mNpri signals or, if enabled, the Arbiter Master N Priority Register. This bit is intended to help with tuning dynamic priority algorithm development.
25	TTA	TT Address—The Arbiter automatically AACKs for address only TT codes. This bit is set when this occurs.
26	TTR	TT Reserved—The Arbiter automatically AACKs for reserved TT codes. This bit is set when this occurs.
27	ECW	External Control Word Read/Write—operations are not supported on the XL bus. If either occur, the arbiter AACKs and TEAs and set this bit.
28	TTM	TBST/TSIZ Mismatch—set when an illegal/reserved TBST and TSIZ[0:2] combination occurs. These combinations are TBST asserted and TSIZ[0:2]=000, 001, 011, or 1xx (x is 0 or 1).
29	BA	Bus Activity Tenure Time-Out—set when bus activity time-out counter expires.
30	DT	Data Tenure Time-Out—set when data tenure time-out counter expires.
31	AT	Address Tenure Time-Out—set when address tenure time-out counter expires.

6.3.4 Interrupt Enable Register (00CC)—INTEN

This read/write 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 Arbiter Interrupt Enable Register, the Arbiter asserts 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 a interrupt. Disabling an interrupt by writing 0 to a bit in this register does not clear the status bit in the Arbiter Status Register.

Table 6-4 Arbiter Interrupt Enable (00CC)—INTEN

	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								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
0:23	—	Reserved
24	MME	Multiple Masters at priority 0 interrupt Enable.
25	TTAE	TT Address-only interrupt Enable.
26	TTRE	TT Reserved interrupt Enable.

Bit	Name	Description
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.

6.3.5 Address Capture Register (00D0)—ADRCAP

This read-only register captures the address for a tenure that has:

- 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 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 it's contents.

Table 6-5 Arbiter Address Capture (00D0)—ADRCAP

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADRCAP															
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	ADRCAP															
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 that is captured when a bus error occurs. this happens on an address time-out, data time-out, or any TEA.

6.3.6 Bus Signal Capture Register (00D4)—SIGCAP

This read-only register captures TT, TBST, GBL, and TSIZ for a tenure that has:

- an address time-out
- a data time-out, or
- any TEA

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 of the following Arbiter Status Register bits:

- bit 30 (Data Tenure Time-out Status)
- bit 31 (Address Tenure Time-Out Status)

Unlocking this register does not clear the contents.

Important bus signals are captured when a bus error occurs. This happens on a address time-out, data time-out, or any TEA.

Table 6-6 Arbiter Bus Signal Capture (00D4)—SIGCAP

	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						TSIZ			GBL	TBST	TT				
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	
25	GBL	
26	TBST	
27:31	TT	

6.3.7 Address Tenure Time-Out Register (00D8)—ADRTO

Table 6-7 Arbiter Address Tenure Time-Out (00D8)—ADRTO

	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											ADRTO				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bit	Name	Description
0:26	—	Reserved
27:31	ADRTO	Upper 5 bits of the Address Time-Out Counter. Values represent increments of 16. Default value is 0x1F.

6.3.8 Data Tenure Time-Out Register (00DC)—DATTO

Table 6-8 Arbiter Data Tenure Time-Out (00DC)—DATTO

	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											DATTO				
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Bit	Name	Description
0:26	—	Reserved
27:31	DATTO	Offset added to the upper 5 bits of the Address Time-Out Counter to derive the Data Tenure Counter. Values represent increments of 16. Default value is 0x1F.

6.3.9 Bus Activity Time-Out Register (00E0)—BUSTO

Table 6-9 Arbiter Bus Activity Time-Out (00E0)—BUSTO

	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	BUSTO[0:15]															
W																
RESET:	0xFFFF															

Bit	Name	Description
0:15	—	Reserved
16:31	BUSTO[0:15]	Set the value of the Bus Activity Counter. Values represent increments of 1. Default value is 0xFFFF.

6.3.10 Master Priority Enable Register (00E4)—PRIEN

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 used at the same time for different masters. This register may be written at any time. The change becomes effective 1-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 master's priority is determined by the hardware mNpri signals, as shown in Table 6-11.

Table 6-10 Arbiter Master Priority Enable (00E4)—PRIEN

	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								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 6-11 Disabled Master Priority

Master	Priority	Description
M7–M4	—	Unused
M3	0	PCI Target Interface
M2	1	SmartComm
M1	2	USB
M0	7	603e Core

6.3.11 Master Priority Register (00E8)—PRI

The Arbiter Master N Priority Register is used to set the priority of each master if the corresponding Arbiter Master Priority Enable register bit is enabled. In conjunction with the Arbiter Master Priority Enable register, this register lets master priorities be set ignoring the mNpri signals. This register may be written at any time.

Changes to this register become effective 1-clock after the register is written. Valid values are from 0 to 7, with 0 being the highest priority. Each of the 8 fields in the register has an upper (fourth) bit reserved. This allows for a possible future expansion to 16 priority levels. Currently, the reserved bits always read 0. For future software compatibility, these bits should always be written as 0.

Table 6-12 Arbiter Master Priority (00E8)—PRI

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Rsvd	M7P			Rsvd	M6P			Rsvd	M5P			Rsvd	M4P		
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	Rsvd	M3P			Rsvd	M2P			Rsvd	M1P			Rsvd	M0P		
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	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

6.3.12 Base Address Register (00EC)—BA

This reserved register is used by the slave interface to determine the Arbiter valid address range. The valid address range is Base Address to (Base Address + 0x7F).

The Arbiter uses 7 significant bits for register addressing. Significant and high order address bits are shown in Table 6-14.

Table 6-13 Base Address (00EC)—BA

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

Bit	Name	Description
0:24	BAR	Upper 25 bits of the Base Address Value. Default is defined in a Verilog parameter, ARBITER_BASE_ADDRESS. The lower 7 bits are always 0. This register has no effect on MGT5100 implementation.
25:31	—	Reserved

Table 6-14 High Order and Significant Bits

Address Bit	Description
31:7	High Order bits
6	Selects Reserved or Working range
5:3	Selects double word in address range
2	Selects individual register within a double word
1:0	Selects byte within 4 byte register.

6.3.13 Reserved Registers (0080–00BF, 00EC, 00F0–00FC)

Reserved read-only registers.

Table 6-15 Reserved Registers (0080–00BF, 00EC, 00F0–00FC)

	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

SECTION 7
SYSTEM INTERFACE UNIT (SIU)

7.1 Overview

The following sections are contained in this document:

- Interrupt Controller, includes:
  - Interrupt Controller Registers—MBAR + 0x0500
- 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 Harpo core pins:

- cint
- smi
- int

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 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.

Table 7-1 Interrupt Sources (continued)

Source	No.	Description
General Timers	8	Generates interrupt in Input Capture mode or Internal Timer mode. Timers 6 and 7 can interrupt from NAP/DOZE power-down.
SmartComm and Peripherals	17	Various peripherals are priority programmed and encoded into HI or LO interrupt to the Interrupt Controller. SmartComm Controller interrupt is connected to HI interrupt.
RTC	2	Stopwatch and periodic
WakeUp	1(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	1(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 $\overline{\mathrm{SRESET}}$ output indication.
Total	35(49)	

Table 7-1 does not include machine-check bus errors or transaction handshaking. Core interrupt pins given in Section 7.2.1.1 through Section 7.2.1.3 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.

Table 7-2 System Management Interrupt Pin Interrupts (continued)

Interrupt	Description
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 2 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 WakeUp 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	SmartComm HI, IRQ0, Slice Timer1, CCS WakeUp	MSR[24]	Persistent (remains until cleared)
core_smi	System Management Interrupt	Slice timer2, Programmable interrupts	MSR[ee]	Persistent
core_int	Standard Interrupt	Programmable interrupts	MSR[ee]	Persistent

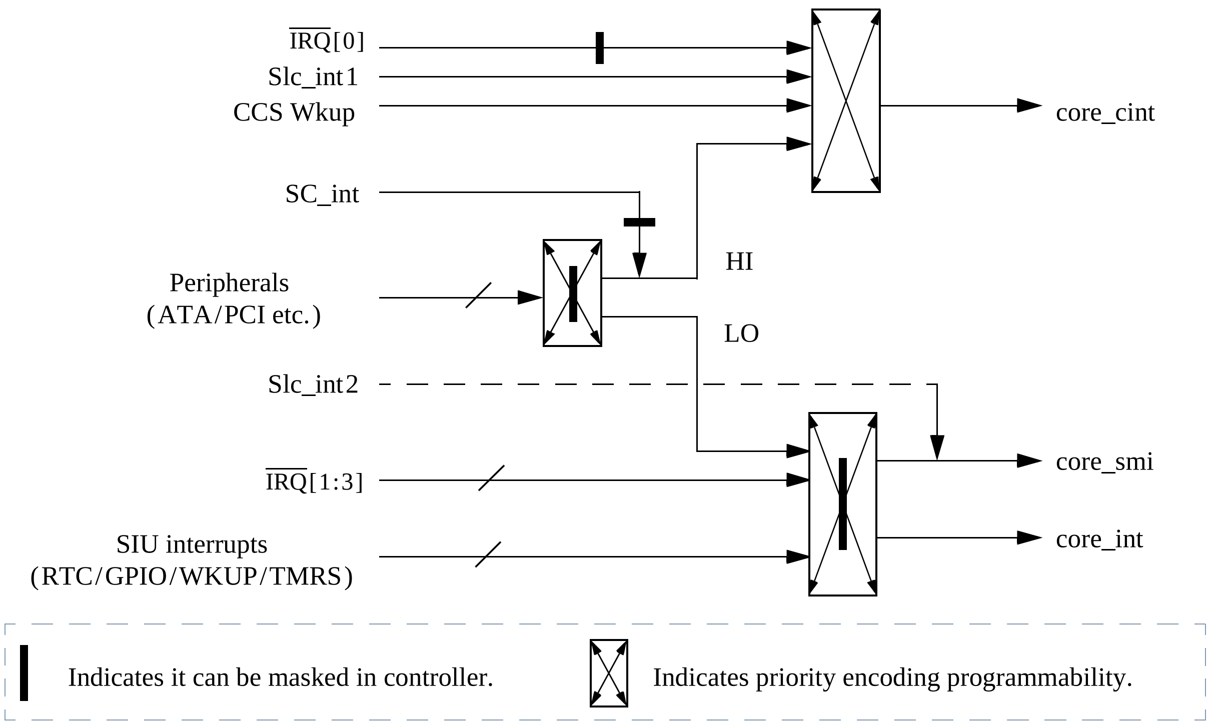

$\overline{\text{IRQ}}$[0]
Slc_int1
CCS Wkup
core_cint
SC_int
HI
Peripherals (ATA/PCI etc.)
LO
Slc_int2
core_smi
$\overline{\text{IRQ}}$[1:3]
SIU interrupts (RTC/GPIO/WKUP/TMRS)
core_int
Indicates it can be masked in controller.
Indicates priority encoding programmability.

Figure 7-1 Interrupt Sources and Core Interrupt Pins

External Interrupts can be programmed as level or edge sensitive. All internal interrupt sources are generated as level sensitive and are not programmable.

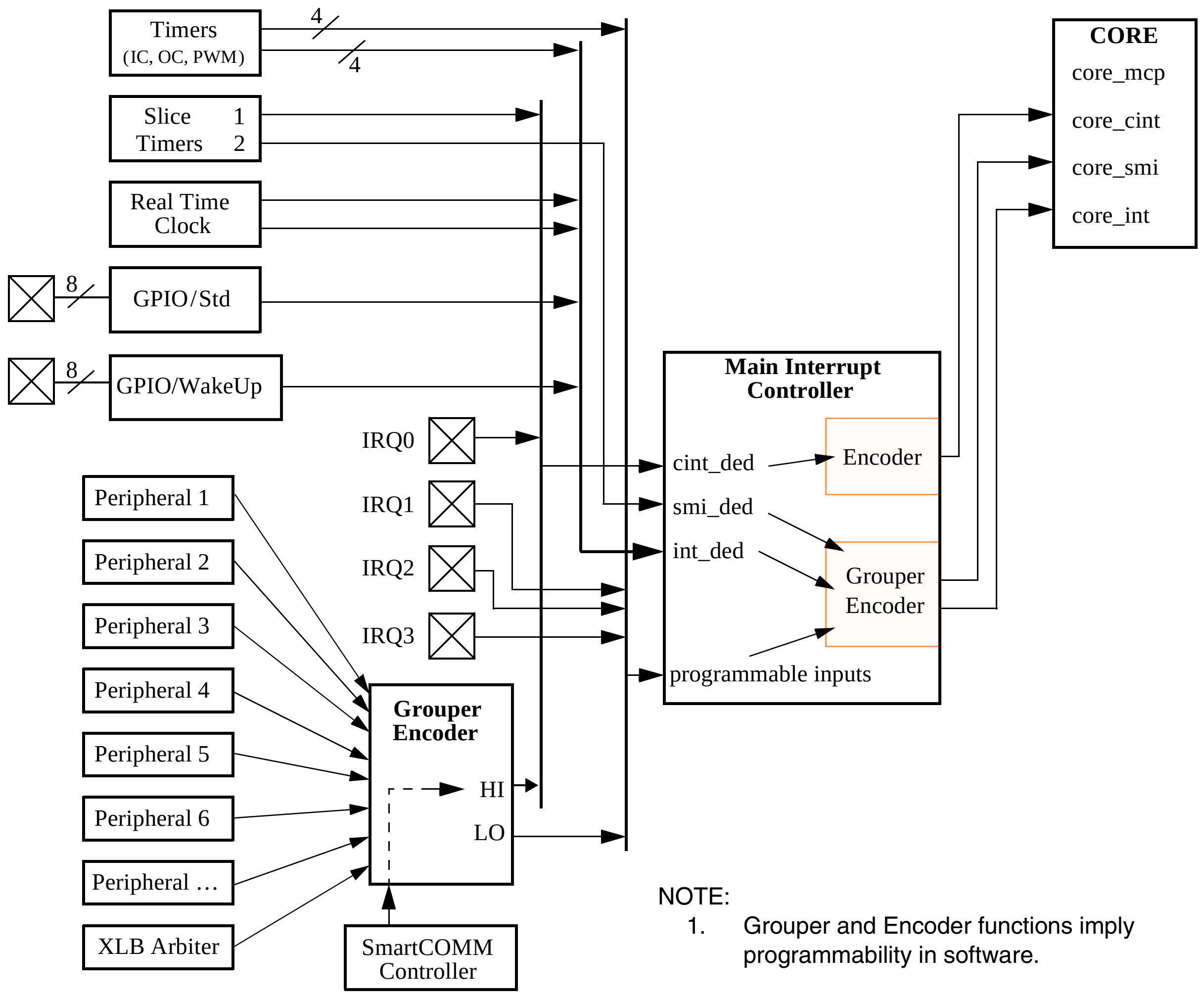
7.2.2 $\overline{\text{IRQ}}$[0:3] Interrupt Requests

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

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

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

7.2.3 Interface Description



NOTE:

1. Grouper and Encoder functions imply programmability in software.

Figure 7-2 Interrupt Controller Routing Scheme

7.2.4 Interrupt Controller Registers—MBAR + 0x0500

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:

- Peripheral Interrupt Mask (0500)
- Peripheral Priority and HI/LO Select 1 (0504)
- Peripheral Priority and HI/LO Select 2 (0508)
- Peripheral Priority and HI/LO Select 3 (050C)
- External Enable and External Types (0510)
- Critical Priority and Main Interrupt Mask (0514)
- Main Interrupt Priority and INT/SMI Select 1 (0518)
- Main Interrupt Priority and INT/SMI Select 2 (051C)
- PerStat, MainStat, CritStat Encoded (0524)
- Critical Interrupt Status All (0528)
- Main Interrupt Status All (052C)
- Peripheral Interrupt Status All (0530)
- Peripheral Interrupt Status All (0538)

7.2.4.1 Peripheral Interrupt Mask (0500)

Table 7-4 Peripheral Interrupt Mask (0500)

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

Bits	Name	Description
—	Per_mask	Bits 0:21—To mask/accept individual peripheral 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. Important—See Note [1].
0	Per_mask	SmartComm 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 (IRDA)
5	Per_mask	Peripheral 5 (Ethernet)

Bits	Name	Description
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 (Reserved)
12	Per_mask	Peripheral 12 (Reserved)
13	Per_mask	Peripheral 13 (SPI modf)
14	Per_mask	Peripheral 14 (SPI spif)
15	Per_mask	Peripheral 15, which is I2C1
16	Per_mask	Peripheral 16, which is I2C2
17	Per_mask	Peripheral 17, which is CAN1
18	Per_mask	Peripheral 18, which is CAN2
19	Per_mask	Peripheral 19, which is IR_RX
20	Per_mask	Peripheral 20, which is IR_TX
21	Per_mask	Peripheral 21, which is XLB Arbiter
22:31	—	Reserved
NOTE: 1. Setting these bits prevents an interrupt being presented to the core pins for the masked sources. Encoded status indications (PSe in Reg9) are suppressed, but the binary "all" status bits (PSa in RegC) are active as long as the source module is presenting an active input to the Interrupt Controller.		

7.2.4.2 Peripheral Priority and HI/LO Select 1 (0504)

Table 7-5 Peripheral Priority and HI/LO Select 1 (0504)

	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 4bits 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 = SmartComm 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 = IRDA
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 SmartComm 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 Peripheral Priority and HI/LO Select 2 (0508)

Table 7-6 Peripheral Priority and HI/LO Select 2 (0508)

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

Bits	Name	Description
12:15	Per11_pri	Peripheral 11 = Reserved
16:19	Per12_pri	Peripheral 12 = Reserved
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 Peripheral Priority and HI/LO Select 3 (050C)

Table 7-7 Peripheral Priority and HI/LO Select 3 (050C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Per16_pri				Per17_pri				Per18_pri				Per19_pri			
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	Per20_pri				Per21_pri				Reserved							
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
8:11	Per18_pri	Peripheral 18 = CAN2
12:15	Per19_pri	Peripheral 19 = IR_RX
16:19	Per20_pri	Peripheral 20 = IR_TX
20:23	Per21_pri	Peripheral 21 = XLB Arbiter
24:31	—	Reserved

7.2.4.5 External Enable and External Types (0510)

Table 7-8 External Enable and External Types (0510)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved				ECLR(4)				Etype0		Etype1		Etype2		Etype2	
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). It is expected to be a debug bit only.
—	EENA[x]	Individual enable bits for each $\overline{\text{IRQ}}$ input pin. Setting the associated bit lets the related $\overline{\text{IRQ}}$ pin generate interrupts. In either case, status indications in PSa and CSa (RegC) are active.
20	EENA0	$\overline{\text{IRQ}}$[0]
21	EENA1	$\overline{\text{IRQ}}$[1]
22	EENA2	$\overline{\text{IRQ}}$[2]
23	EENA3	$\overline{\text{IRQ}}$[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 Critical Priority and Main Interrupt Mask (0514)

Table 7-9 Critical Priority and Main Interrupt Mask (0514)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Crit0_Pri		Crit1_Pri		Crit2_Pri		Crit3_Pri		Reserved							Main_ Mask
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	Main_Mask															
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, $\overline{\text{IRQ}}$[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 $\overline{\text{IRQ}}$[0] the highest default priority (being the lowest numbered source).
2:3	Crit1_Pri	Priority encoding value for Slice_Timer1 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
—	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	SliceTimer2, which is hardwired to SMI interrupt output. See Note 2.
—	—	Interrupt sources below are bank/priority programmable (in Reg6 and Reg7).
16	Main_Mask1	$\overline{\text{IRQ}}$[1] ($\overline{\text{IRQ}}$[1] input pin interrupt)
17	Main_Mask2	$\overline{\text{IRQ}}$[2] ($\overline{\text{IRQ}}$[2] input pin interrupt)
18	Main_Mask3	$\overline{\text{IRQ}}$[3] ($\overline{\text{IRQ}}$[3] input pin interrupt)
19	Main_Mask4	LO_int (source programmable from Peripheral ints)
20	Main_Mask5	RTC_pint (Real time clock, periodic interrupt)

Bits	Name	Description
21	Main_Mask6	RTC_sint (Real time clock, stopwatch 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 $\overline{\text{IRQ}}$[1:3], is redundant with External ENA bits in Reg4, but both masks are applied.
2. SliceTimer2 is hard-coded and neither bank nor priority adjustable.

7.2.4.7 Main Interrupt Priority and INT/SMI Select 1 (0518)

Table 7-10 Main Interrupt Priority and INT/SMI Select 1 (0518)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Main1_Pri				Main2_Pri				Main3_Pri				Main4_Pri			
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	Main5_Pri				Main6_Pri				Main7_Pri				Main8_Pri			
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 ($\overline{\text{IRQ}}$[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 ($\overline{\text{IRQ}}$[2] input pin) priority encoding value.
8:11	Main3_pri	Main interrupt source 3 ($\overline{\text{IRQ}}$[3] input pin) priority encoding value.

Bits	Name	Description
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) 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.
20:23	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_timer2) 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 Main Interrupt Priority and INT/SMI Select 2 (051C)

Table 7-11 Main Interrupt Priority and INT/SMI Select 2 (051C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Main9_Pri				Main10_Pri				Main11_Pri				Main12_Pri			
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	Main13_Pri				Main14_Pri				Main15_Pri				Main16_Pri			
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 4bits 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 MGT5100 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.

Bits	Name	Description
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.
20:23	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 PerStat, MainStat, CritStat Encoded (0524)

Table 7-12 PerStat, MainStat, CritStat Encoded (0524)

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

Bits	Name	Description
0:1	—	Reserved
4:7	PSe	Peripheral Status Encoded—makes a singular indication of the current peripheral interrupt (6bits indicating 1 of 22 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 Reg0 (peripheral mask) and re-stated in RegC (peripheral status all). 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

Bits	Name	Description
11:15	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 RegB (main status all). 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
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 = $\overline{\text{IRQ}}$ input pin is the source. See Note 2. 01 = Slice Timer 1 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 Reg4 (bit 31). 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 $\overline{\text{IRQ}}$[1:3] respectively, if the $\overline{\text{IRQ}}$ 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 Reg4 be set to clear this interrupt indication. Only one method should be used, not both (this limit is only true for multiple edge-sensitive $\overline{\text{IRQ}}$ inputs).
2. For $\overline{\text{IRQ}}$[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 Reg4 be set to clear this interrupt indication. You can do both if desired, and you can do it regardless of the $\overline{\text{IRQ}}$[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 MGT5100 operate normally.

7.2.4.10 Critical Interrupt Status All (0528)

Table 7-13 Critical Interrupt Status All (0528)

	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, Reg9.
4	CSa0	indicates $\overline{\text{IRQ}}$[0] interrupt
5	CSa1	Slice Timer 1 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 Reg4 is cleared, then CSa status occurs, but controller does not assert a core interrupt.

7.2.4.11 Main Interrupt Status All (052C)

Table 7-14 Main Interrupt Status All (052C)

	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 2 (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
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, Reg5. 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 Peripheral Interrupt Status All (0530)

Table 7-15 Peripheral Interrupt Status All (0530)

	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															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:9	—	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, Reg9.
10	PSa0	SmartComm interrupt source
11	PSa1	PSC1
12	PSa2	PSC2
13	PSa3	PSC3
14	PSa4	IRDA
15	PSa5	Ethernet
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	Reserved
22	PSa12	Reserved
23	PSa13	SPI modf
24	PSa14	SPI spif
25	PSa15	I^{2}C1
26	PSa16	I^{2}C2
27	PSa17	CAN1
28	PSa18	CAN2
29	PSa19	IR_Rx
30	PSa20	IR_Tx
31	PSa21	XLB Arbiter
NOTE: 1. These interrupts are directly maskable by Reg0 Per_Mask. 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 Peripheral Interrupt Status All (0538)

Table 7-16 Peripheral Interrupt Status All (0538)

	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

7.3 General Purpose I/O (GPIO)

There are a total of 56 possible GPIO pins on the MGT5100. 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 level or 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 Port Configuration Register (MBAR+0B00)—GPIOPCR. 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 7-17 lists all 56 GPIO pins.

Table 7-17 GPIO Pin List

GPIO	Alternate Functionality	Interrupt	WakeUp
GPIO_TMR_0	Timer_GPIO/ATA/CAN2	Only as Timer IC	No
GPIO_TMR_1	Timer_GPIO/ATA/CAN2	Only as Timer IC	No
GPIO_TMR_2	Timer_GPIO/SPI	Only as Timer IC	No
GPIO_TMR_3	Timer_GPIO/SPI	Only as Timer IC	No
GPIO_TMR_4	Timer_GPIO/SPI	Only as Timer IC	No
GPIO_TMR_5	Timer_GPIO/SPI	Only as Timer IC	No
GPIO_TMR_6	Timer_GPIO	Only as Timer IC	No
GPIO_TMR_7	Timer_GPIO	Only as Timer IC	No
GPIO_PSC1_0	UART1/AC971/CODEC1	No	No
GPIO_PSC1_1	UART1/AC971/CODEC1	No	No
GPIO_PSC1_2	UART1/AC971	No	No
GPIO_PSC1_3	UART1/AC971/CODEC1	No	No
GPIO_WKUP_0(PSC1)	UART1/AC971/CODEC1	Yes	Yes
GPIO_PSC2_0	UART2/AC972/CODEC2/CAN1	No	No
GPIO_PSC2_1	UART2/AC972/CODEC2/CAN1	No	No
GPIO_PSC2_2	UART2/AC972/CAN2	No	No
GPIO_PSC2_3	UART2/AC972/CODEC2/CAN2	No	No

Table 7-17 GPIO Pin List (continued)

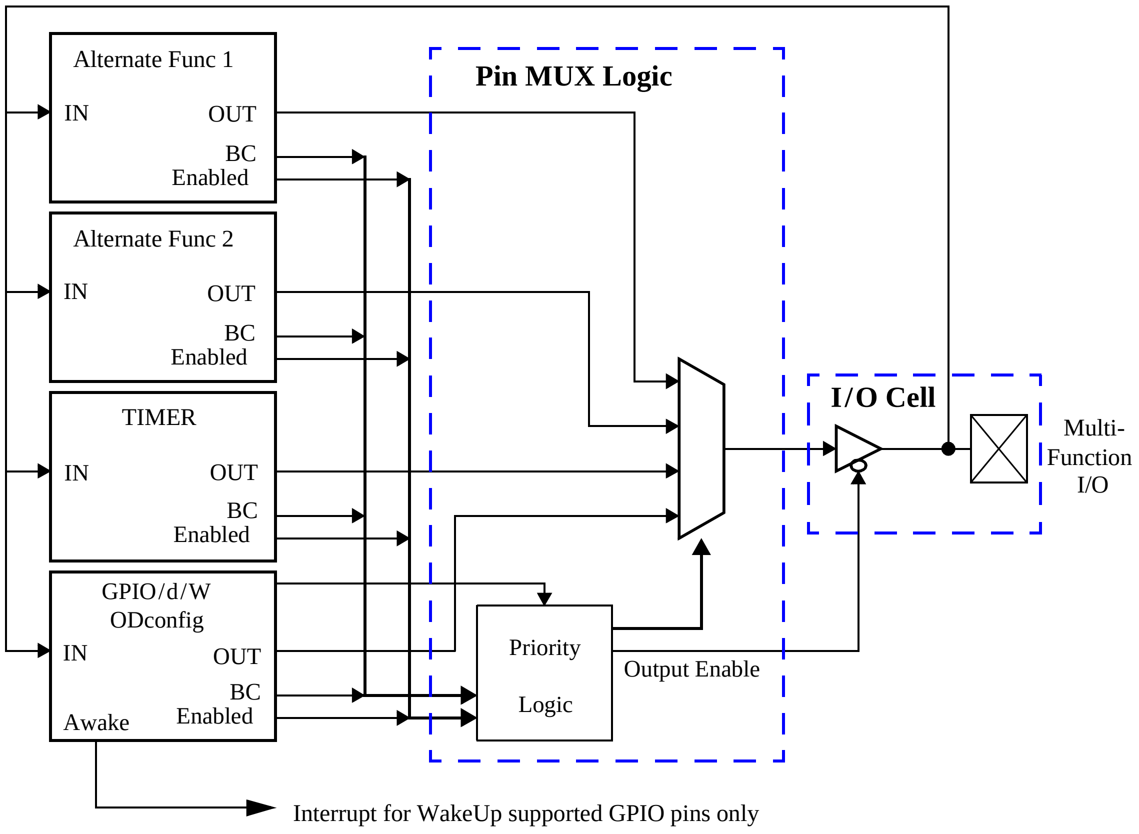
GPIO	Alternate Functionality	Interrupt	WakeUp
GPIO_WKUP_1(PSC2)	UART2/AC972/CODEC2	Yes	Yes
GPIO_PSC3_0	USB2/CODEC3/UART3	No	No
GPIO_PSC3_1	USB2/CODEC3/UART3	No	No
GPIO_PSC3_2	USB2/CODEC3/UART3	No	No
GPIO_PSC3_3	USB2/CODEC3/UART3	No	No
GPIO_SINT_0(PSC3)	USB2/UART3	Yes	No
GPIO_SINT_1(PSC3)	USB2	Yes	No
GPIO_PSC3_4	USB2/SPI	No	No
GPIO_PSC3_5	USB2/SPI	No	No
GPIO_SINT_2(PSC3)	USB2/SPI	Yes	No
GPIO_WKUP_2(PSC3)	USB2/SPI	Yes	Yes
GPIO_USB_0	USB1 (OE)	No	No
GPIO_USB_1	USB1 (PORTPWR)	No	No
GPIO_USB_2	USB1 (SPEED)	No	No
GPIO_USB_3	USB1 (SUSPEND)	No	No
GPIO_SINT_3(USB)	USB1 (CLOCK)	Yes	No
GPIO_ETHO_0(out only)	Ethernet	No	No
GPIO_ETHO_1(out only)	Ethernet	No	No
GPIO_ETHO_2(out only)	Ethernet/USB2	No	No
GPIO_ETHO_3(out only)	Ethernet/USB2	No	No
GPIO_ETHO_4(out only)	Ethernet/USB2	No	No
GPIO_ETHO_5(out only)	Ethernet/USB2	No	No
GPIO_ETHO_6(out only)	Ethernet/USB2	No	No
GPIO_ETHO_7(out only)	Ethernet/USB2	No	No
GPIO_ETHI_0	Ethernet	No	No
GPIO_ETHI_1	Ethernet	No	No
GPIO_ETHI_2	Ethernet	No	No
GPIO_ETHI_3	Ethernet	No	No
GPIO_SINT_4(ETH)	Ethernet/USB2	Yes	No
GPIO_SINT_5(ETH)	Ethernet/USB2	Yes	No
GPIO_SINT_6(ETH)	Ethernet/USB2	Yes	No
GPIO_SINT_7(ETH)	Ethernet/USB2	Yes	No
GPIO_WKUP_3(ETH)	Ethernet	Yes	Yes
GPIO_IRDA_0	IRDA TX	No	No
GPIO_IRDA_1	IRDA CLK (and/or USB CLK)	No	No
GPIO_WKUP_4(IRDA)	IR Remote Receiver	Yes	Yes

Table 7-17 GPIO Pin List (continued)

GPIO	Alternate Functionality	Interrupt	WakeUp
GPIO_WKUP_5(IRDA)	IR Keyboard Receiver (IRDA RX)	Yes	Yes
GPIO_WKUP_6	Dedicated GPIO Pin/SDRAM CS1	Yes	Yes
GPIO_WKUP_7	Dedicated GPIO Pin	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 (Register 0 in GPIO standard module) 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
- enhanced UART (with carrier detect input)
- 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.

7.3.1.2 PSC2 (UART2/AC97/CODEC2)

The PSC2 port has 5 pins with hardware support for:

- CODEC
- expanded UART (with carrier detect input)
- 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.

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.

7.3.1.4 USB1

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/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.

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

Total GPIO available on this port is:

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

7.3.1.6 IRDA

The IRD 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
- Consumer IR Receiver (1 pin)—non-operational in current version.
- Consumer IR Blaster (1pin)

The Consumer IR Receiver and IRDA configuration can exist simultaneously. The IRDA clock pin can be used as a Input USB clock and is separately programmable for this use.

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

7.3.1.7 I^{2}C

There are 2 I^{2}C ports consisting of 2 pins each. Although no GPIO is available on these pins, they can be alternately programmed as CAN1 pins (on I^{2}C1) and/or as the ATA Chip Selects (on I^{2}C2). If the alternate function is specified, the associated I^{2}C 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 2.5V 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 MGT5100 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. Table 7-18 shows the register organization.

Table 7-18 GPIO Standard Register Types—MBAR+0x0B00

Register	Description
Register 0	Port Configuration Register
Register 1–5	Simple GPIO configuration registers: Register 1—Enables Register 2—Open-Drain type Register 3—Data Direction (DD) Register 4—Data Output value (DO) Register 5—Data Input value (DI)
Register 6–8	Output-Only GPIO configuration registers: Register 6—Enables Register 7—Open Drain type Register 8—DO
Register 9–14	Simple Interrupt GPIO configuration registers: Register 9—Enables Register 10—Open-Drain type Register 11—DD Register 12—DO Register 13—Interrupt enables Register 14—Interrupt types
Register 15	Master Interrupt Enable and Bus Error Enable control register
Register 16	Interrupt Status, Input Value, and Global Bus Error Status register

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:

- Port Configuration Register (0B00)—GPIOPCR
- Simple GPIO Enables (0B04)—GPIOSEN,
- Simple GPIO Open Drain Type (0B08)—GPIOSOD
- Simple GPIO Data Direction (0B0C)—GPIOSDD
- Simple GPIO Data Output Values (0B10)—GPIOS-DO
- Simple GPIO Data Input Values (0B14)—GPIOSDI,
- GPIO Output-Only Enables (0B18)—GPIOOE
- GPIO Output-Only Data Value Out (0B1C)—GPIOODO
- GPIO Simple Interrupt Enables (0B20)—GPIOSIE
- GPIO Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD
- GPIO Simple Interrupt Data Direction (0B28)—GPIOSIDD
- GPIO Simple Interrupt Data Value Out (0B2C)—GPIOSIDO
- GPIO Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE
- GPIO Simple Interrupt Interrupt Types (0B34)—GPIOSIIT
- GPIO Simple Interrupt Master Enable (0B38)—GPIOSIME
- GPIO Simple Interrupt Status (0B3C)—GPIO-SIST

7.3.2.1.1 Port Configuration Register (MBAR+0B00)—GPIOPCR

Table 7-19 Port Configuration Register (0B00)—GPIOPCR

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

Bit	Name	Description
0:x	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
4:5	—	Reserved
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
9:10	IRDA	Infrared Data Association 000=All IrDA/IR pins are GPIOs 001=Consumer IR on ir_tx (Blaster), rest are GPIOs 010=Consumer remote on ir_rx, IrDA pins are GPIOs 011=Consumer IR on ir_tx (Blaster), consumer remote on ir_rx, rest are GPIOs 10X=IrDA on 3 pins, GPIO on ir_rx 11X=IrDA on 3 pins, consumer remote on ir_rx
12	—	Reserved

Bit	Name	Description
13:15	Ether	Ethernet 000=All 18 Ethernet pins are GPIOs 001=USB2 on Ethernet, see Note 3 010=Ethernet 10Mbit (7-wire) mode 011=Ethernet 7-wire and USB2, see Note 3 100=Ethernet 100Mbit without MD 101=Ethernet 100Mbit with MD 11X=Ethernet 100Mbit with MD
16:18	—	Reserved
19	USB	0=All 10 USB pins are GPIOs 1=USB
20:23	PSC3	Programmable Serial Controller 3 0000=All PSC3 pins are GPIOs 0001=USB2 on PSC3, no GPIOs available, see Note 3 0010=Reserved 0011=Reserved 0100=UART functionality without CD 0101=UARTe functionality with CD 011X=CODEC3 functionality 100X=Reserved 1010=Reserved 1011=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 11X=CODEC functionality
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 11X=CODEC functionality

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 or Table 2-1 or Table 2-2 to determine GPIO availability for the various PCR field settings.

7.3.2.1.2 Simple GPIO Enables (0B04)—GPIOSEN

Table 7-20 Simple GPIO Enables (0B04)—GPIOSEN

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Reserved		IRDA		ETHR				Reserved				USB			
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		PSC3						PSC2				PSC1			
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 (IR_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 Simple GPIO Open Drain Type (0B08)—GPIOSOD

Table 7-21 Simple GPIO Open Drain Type (0B08)—GPIOSOD

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Reserved		IRDA		ETHR				Reserved				USB			
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		PSC3						PSC2				PSC1			
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 (IR_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 Simple GPIO Data Direction (0B0C)—GPIOSDD

Table 7-22 Simple GPIO Data Direction (0B0C)—GPIOSDD

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

Bit	Name	Description
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 Simple GPIO Data Output Values (0B10)—GPIOSDO

Table 7-23 Simple GPIO Data Output Values (0B10)—GPIOSDO

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

Bit	Name	Description
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
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 Simple GPIO Data Input Values (0B14)—GPIOSDI

Table 7-24 Simple GPIO Data Input Values (0B14)—GPIOSDI

	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 (IR_TX pin)

Bit	Name	Description
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
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 Output-Only Enables (0B18)—GPIOOE

Table 7-25 GPIO Output-Only Enables (0B18)—GPIOOE

	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 Output-Only Data Value Out (0B1C)—GPIOODO

Table 7-26 GPIO Output-Only Data Value Out (0B1C)—GPIOODO

	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 Simple Interrupt Enables (0B20)—GPIOSIE

Table 7-27 GPIO Simple Interrupt Enables (0B20)—GPIOSIE

	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 Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD

Table 7-28 GPIO Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD

	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 Simple Interrupt Data Direction (0B28)—GPIOSIDD

Table 7-29 GPIO Simple Interrupt Data Direction (0B28)—GPIOSIDD

	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 Simple Interrupt Data Value Out (0B2C)—GPIOSIDO

Table 7-30 GPIO Simple Interrupt Data Value Out (0B2C)—GPIOSIDO

	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 Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE

Table 7-31 GPIO Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE

	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 Simple Interrupt Interrupt Types (0B34)—GPIOSIIT Register. Also, the Master Interrupt Enable bit must be set in the Simple Interrupt Master Enable (0B38)—GPIOSIME Register, before any Simple Interrupt pin can generate an Interrupt.		

7.3.2.1.14 Simple Interrupt Interrupt Types (0B34)—GPIOSIIT

Table 7-32 GPIO Simple Interrupt Interrupt Types (0B34)—GPIOSIIT

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

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 = 01 = 10 = 11 =
16:31	—	Reserved

7.3.2.1.15 Simple Interrupt Master Enable (0B38)—GPIOSIME

Table 7-33 GPIO Simple Interrupt Master Enable (0B38)—GPIOSIME

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			BE	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	BE	GPIO Simple Interrupt Bus Error Enable pin—If this bit is set to 1 and an improper access is made to this module, TEA on the XL Bus is generated. An improper access is defined as a read or write to any unimplemented register, or a write access to a read only register. Generally, this bit should be kept 0, since improper access on the module does not corrupt any register contents.
4:6	—	Reserved
7	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.
8:31	—	Reserved

7.3.2.1.16 Simple Interrupt Status (0B3C)—GPIOSIST

Table 7-34 GPIO Simple Interrupt Status (0B3C)—GPIOSIST

	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					BE1	BE2	BE3	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.
16:20	—	Reserved
21	BE1	Bus Error (type 1)—If set, it indicates a read of an unimplemented register was attempted. This is a read of any register from 10 to 1F. Clear bit with a Sticky bit write to 1. BE1 is **not** dependant on Register BE bit in GPIOSIME register. Bit can be used to detect software access errors. An improper access does not corrupt a register or its contents.
22	BE2	Bus Error (type 2)—If set, it indicates a write to an unimplemented register was attempted. This is a write to any register from 10 to 1F. Clear bit with a Sticky bit write to 1. BE2 is **not** dependant on Register BE bit in GPIOSIME register.
23	BE3	Bus Error (type 3)—If set, it indicates a write to a read-only register was attempted. This is only possible if Register 5 was accessed with a write. Clear bit with a Sticky bit write to 1. BE3 is **not** dependant on Register BE bit in GPIOSIME register.
24:31	—	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 distinicion between these two types of interrupts is made according to the powered state of MGT5100.

- 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:

- WakeUp GPIO Enables (0C00)—GPIOWE
- WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD
- WakeUp GPIO Data Direction (0C08)—GPIO-WDD
- WakeUp GPIO Data Value Out (0C0C)—GPIO-WDO
- WakeUp GPIO Interrupt Enable (0C10)—GPI-OWUE
- WakeUp GPIO Individual Interrupt Enable (0C14)—GPIOWSIE
- WakeUp GPIO Interrupt Types (0C18)—GPIOWT
- WakeUp GPIO Master Enables (0C1C)—GPIO-WME
- WakeUp GPIO Data Input Values (0C20)—GPIOWI
- WakeUp GPIO Status Register (0C24)—GPIOWS

7.3.2.2.1 WakeUp GPIO Enables (0C00)—GPIOWE

Table 7-35 WakeUp GPIO Enables (0C00)—GPIOWE

	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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD

Table 7-36 WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD

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

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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Data Direction (0C08)—GPIOWDD

Table 7-37 WakeUp GPIO Data Direction (0C08)—GPIOWDD

	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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Data Value Out (0C0C)—GPIOWDO

Table 7-38 WakeUp GPIO Data Value Out (0C0C)—GPIOWDO

	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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Interrupt Enable (0C10)—GPIOWUE

Table 7-39 WakeUp GPIO Interrupt Enable (0C10)—GPIOWUE

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

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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 MGT5100 is in Deep Sleep mode. **NOTE:** These enable bits apply ONLY when MGT5100 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 WakeUp GPIO Simple Interrupt Enable (0C14)—GPIOWSIE

Table 7-40 WakeUp GPIO Individual Interrupt Enable (0C14)—GPIOWSIE

	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 (IRDA_RX pin) Bit 3 controls GPIO_WKUP_4 (IR_RX 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 MGT5100 is **not** in Deep Sleep mode. **NOTE:** These enable bits apply **only** when MGT5100 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 WakeUp GPIO Interrupt Types (0C18)—GPIOWT

Table 7-41 WakeUp GPIO Interrupt Types (0C18)—GPIOWT

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

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 MGT5100 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 (IRDA_RX pin) ITYP4 controls GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Master Enables (0C1C)—GPIOWME

Table 7-42 WakeUp GPIO Master Enables (0C1C)—GPIOWME

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

Bit	Name	Description
0:2	—	Reserved
3	BE	WakeUp GPIO Bus Error Enable pin. If this bit is set to a 1 and an improper access is made to this module, a TEA on the XL Bus will be generated. An improper access is defined as a read or write to any unimplemented register, or a write access to a read only register. Generally, this bit should be kept 0, since improper accesses on the module do not corrupt any register contents.
4:6	—	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 WakeUp GPIO Data Input Values (0C20)—GPIOWI

Table 7-43 WakeUp GPIO Data Input Values (0C20)—GPIOWI

	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 (IRDA_RX pin) Bit 3 reflects GPIO_WKUP_4 (IR_RX 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 WakeUp GPIO Status Register (0C24)—GPIOWS

Table 7-44 WakeUp GPIO Status Register (0C24)—GPIOWS

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Istat								Reserved					BE1	BE2	BE3
W	rwc	rwc	rwc	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 (IRDA_RX pin) Bit 3 reflects interrupt on GPIO_WKUP_4 (IR_RX 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)
8:12	—	Reserved

Bit	Name	Description
13	BE1	Bus Error Enable 1—If set, indicates a read of an unimplemented register was attempted. This would be a read of any register from 10–1F. This bit can be cleared by a sticky-bit write to 1. This status bit is not dependant on the Bus Error Enable bit in GPIOWI. BE bits can be used to detect software access errors, but in no case is a register or its contents corrupted by an improper access.
14	BE2	Bus Error Enable 2—If set, indicates a write to an unimplemented register was attempted. This would be a write of any register from 10–1F. This bit can be cleared by a sticky-bit write to 1. This status bit is not dependant on the Bus Error Enable bit in GPIOWI.
15	BE3	Bus Error Enable 3—If set, indicates a write to a read-only register was attempted. This is only possible if GPIOWIE was accessed with a write. This would be a write of any register from 10–1F. This bit can be cleared by a sticky-bit write to 1. This status bit is not dependant on the Bus Error Enable bit in GPIOWI.
16:31	—	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.
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. 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 MGT5100 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[0–7] Enable and Mode Select (0600–0670)
- GPT[0–7] Counter Input (0604–0674)
- GPT[0–7] PWM Configuration (0608–0678)
- GPT[0–7] Status (060C–067C), read-only

7.4.4.1 GPT[*0–7*] Enable and Mode Select (0600–0670)

GPT0	0600	GPT4	0640
GPT1	0610	GPT5	0650
GPT2	0620	GPT6	0660
GPT3	0630	GPT7	0670

Table 7-45 GPT[*0–7*] Enable and Mode Select (0600–0670)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	OCPW								Reserved		OCT		Reserved		ICT	
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	WDen	Reserved		CE	Rsvd	Stop_Cont	Open_Drn	IntEn	Reserved		GPIO		Rsvd	Timer_MS		
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. 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[0–7] Counter Input (0604–0674).
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. See Section C, Addendum, for Pulse Input Capture.
16	WDen	Watchdog enable—bit enables watchdog operation. A timer expiration causes an internal MGT5100 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

Bit	Name	Description
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
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. 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 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

Bit	Name	Description
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.
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[*0–7*] Counter Input (0604–0674)

GPT0	0604	GPT4	0644
GPT1	0614	GPT5	0654
GPT2	0624	GPT6	0664
GPT3	0634	GPT7	0674

Table 7-46 GPT[*0–7*] Counter Input (0604–0674)

	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[*0–7*] PWM Configuration (0608–0678)

GPT0	0608	GPT4	0648
GPT1	0618	GPT5	0658
GPT2	0628	GPT6	0668
GPT3	0638	GPT7	0678

Table 7-47 GPT[*0–7*] PWM Configuration (0608–0678)

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

Bit	Name	Description
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[*0–7*] Status (060C–067C)

GPT0	060C	GPT4	064C
GPT1	061C	GPT5	065C
GPT2	062C	GPT6	066C
GPT3	063C	GPT7	067C

This is a read-only register.

Table 7-48 GPT[*0–7*] Status (060C–067C)

	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
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).

Bit	Name	Description
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 7.75uS to 508mS in 30nS steps (based on 33MHz 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:

- Terminal Count Register — TCNT0, 1 (0700,0710)
- Control Register — CTL0, 1 (0704, 0714)
- Count Value Register — TMRCNT0, 1 (0708, 0718) Read Only.
- Timer Status Register — TMRSTA0, 1 (070C, 071C) Read Only.

7.5.1.1 Terminal Count Register — TCNT0, 1 (0700, 0710)

TCNT0 0700 TCNT1 0710

Table 7-49 Terminal Count Register — TCNT0, 1 (0700,0710)

	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	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	Terminal Count															
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	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.

7.5.1.2 Control Register — CTL0, 1 (0704, 0714)

CTL0	0704	CTL1	0714

Table 7-50 Control Register — CTL0, 1 (0704, 0714)

	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 Count Value Register — TMRCNT0, 1 (0708, 0718)

TMRCNT0	0708	TMRCNT1	0718

Table 7-51 Count Value Register — TMRCNT0, 1 (0708, 0718)

	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 change while a read is in progress, but the actual Timer counter continues unaffected.

7.5.1.4 Timer Status Register — TMRSTA0, 1 (070C, 071C)

TMRSTA0	070C	TMRSTA1	071C

Table 7-52 Timer Status Register — TMRSTA0, 1 (070C, 071C)

	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

The clock runs as long as power is maintained and the crystal is running, regardless of MGT5100 power-down state.

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 MGT5100 power-down modes. At a soft reset or greater, 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 MGT5100 from any power down mode.

7.6.1 Real-Time Clock Signals

Table 7-53 Real-Time Clock Signals

Signal	I/O	Definition
RTC_XTAL[1]/EXT	I	Real-time Clock External Crystal/External Clock Input
RTC_XTAL[0]	I	Real-time Clock External Crystal

Figure 7-4 shows a suggested circuit using an Epson® MC-405 32.768KHz quartz crystal oscillator.

NOTE: External component values are highly dependent on the crystal. These values will be different for different brands of crystals.

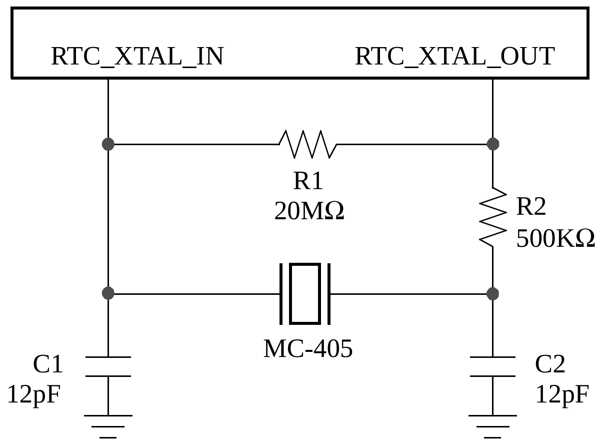


Figure 7-4 Diagram—Suggested Crystal Oscillator Circuit

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 32KHz 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).

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 (0800)
- RTC Date Set (0804)
- RTC New Year and Stopwatch (0808)
- RTC Alarm and Interrupt Enable (080C)
- RTC Current Time (0810), read-only
- RTC Current Date (0814), read-only
- RTC Alarm and Stopwatch Interrupt (0818), read-only
- RTC Periodic Interrupt and Bus Error (081C), read-only

7.6.3.1 Time Set (0800)

Table 7-54 RTC Time Set (0800)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved						set_time	pause_time	Reserved		SlctHour	C24Hour_set				
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		Minute_set						Reserved		Second_set					
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 (see Reg 4).
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
18:23	Minute_set	Minute written in RTC after successful state machine transition by set_time and pause_time bits.
24:25	—	Reserved

Bits	Name	Description
26:31	Second_set	Second written in RTC after successful state machine transition by set_time and pause_time bits.

7.6.3.2 Date Set (0804)

Table 7-55 RTC Date Set (0804)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved						set_date	pause_date	Reserved			Month_set				
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		Weekday_set						Reserved		Day_set					
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_date	Operation of pause_date and set_date is similar to pause_time and set_time described in Reg0.
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.
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.
24:25	—	Reserved
26:31	Date_set	New date written in RTC after state machine transition by set_date and pause_date bits. **NOTE:** Year_set in the following register is also part of the date set function.

7.6.3.3 New Year and Stopwatch (0808)

Table 7-56 RTC New Year and Stopwatch (0808)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Reserved							write_SW	SW_set							
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				Year_set											
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 Alarm and Interrupt Enable (080C)

Table 7-57 RTC Alarm and Interrupt Enable (080C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Reserved							Alm_enable	Reserved			Alm_24H_set				
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		Alm_Min_set						Reserved				MPEb	IntEn_day	IntEn_min	IntEn_sec
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bits	Name	Description
0:6	—	Reserved

Bits	Name	Description
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.
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_et	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.
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 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 Current Time (0810)

This is a read-only register.

Table 7-58 RTC Current Time (0810)

	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

Bits	Name	Description
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), whole 5-bit hour designates current time in 24-hour format. If 12-hour format is selected (SlctHour high in Reg 0), msb of hour field indicates: • AM(Hour[0]=0), or • PM(Hour[0]=1), and • Hour[1:4] designates current time in 12-hour format.
16:17	—	Reserved
18:23	Minute	Shows minutes in current time.
24:25	—	Reserved
26:31	Second	Shows seconds in current time.

7.6.3.6 Current Date (0814)

This is a read-only register.

Table 7-59 RTC Current Date (0814)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				Month				Weekday			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				Year											
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	Day	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 Alarm and Stopwatch Interrupt (0818)

This is a read-only register.

Table 7-60 RTC Alarm and Stopwatch Interrupt (0818)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved							Int_alm	Reserved							Int_SW
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_status	SW_min							
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	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.
20:31	SW_min	Minutes remaining in stopwatch.

7.6.3.8 Periodic Interrupt and Bus Error (081C)

This is a read-only register.

Table 7-61 RTC Periodic Interrupt and Bus Error (081C)

	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.
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 Test Register/Divides

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 7-62. Test Register/Divides (0820)

	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.		

SECTION 8
SDRAM MEMORY CONTROLLER

8.1 Overview

The following sections are contained in this document:

- Features
- Functional Description
  - SDRAM Configurations
  - Timing Considerations
  - External Signals (SDRAM Side)
- Operation
  - Block Diagram
  - Address Multiplexing and Address Pipeline Blocks
  - External Chip Select
  - Power-Up Initialization
  - Page Management
  - Transfer Size
  - Commands
- Programming the SDRAM Memory Controller
- SDRAM Memory Controller Registers (MBAR + 0x0100)
- Examples

The MGT5100 SDRAM Memory Controller (MC) supports access to external SDRAM memory from all masters on the XL bus. Single Data Rate (SDR) SDRAM and Double Data Rate (DDR) SDRAM is supported. The SDRAM MC minimizes loading and maximizes speed.

The SDRAM MC interfaces to the microprocessor bus internally, but leaves address decoding to the on-chip central address decoder. The SDRAM MC externally supports various sizes of SDRAM or DDR SDRAM memory systems.

Register descriptions in this document adhere to G2 microprocessor convention. What this means is:

- the left bit—the most significant bit (msb) of a register—is bit 0
- the right bit—the least significant bit (lsb) of a register—is bit 31

8.2 Features

The MGT5100 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_MEMCLK and $\overline{\text{MEM\_MEMCLK}}$.
- 32-bit data bus
- Maximum size—supports 256MB of memory:
  - 13 lines of row address
  - 11 lines of column address
  - 2 lines of internal bank address
  - a maximum of 2 pinned-out Chip Selects (CS)

  This comprises 26 address lines per physical CS, supporting up to 64MB of address space. With a 32-bit data bus this provides a total of 256MB of memory.
- Minimum size—supports 2MB of memory:
  - 11 lines of row address
  - 8 lines of column address
  - 2 lines of internal_bank address
  - 1 chip select
- 32Byte G2 critical word, first burst transfer
- Supports G2 bus, 2-stage address/data pipeline
- Supports sleep mode and SDRAM self-refresh mode
- Supports page mode and bursting to maximize the data rate

NOTE: SDRAM MC does not support error detect and parity check

8.3 Functional Description

8.3.1 SDRAM Configurations

The SDRAM MC interfaces to the G2 bus internally. However, address decoding is done by the on-chip central address decoder. Supported SDRAM devices are:

- 64Mbit
- 128Mbit
- 256Mbit
- 512Mbit

Figure 8-1 shows an example of a memory configuration. This example uses a 4Mbyte word x 8bit x 4 internal_bank DDR SDRAM memory chip to form a 2-bank 32-bit wide, 1-Gbit memory system.

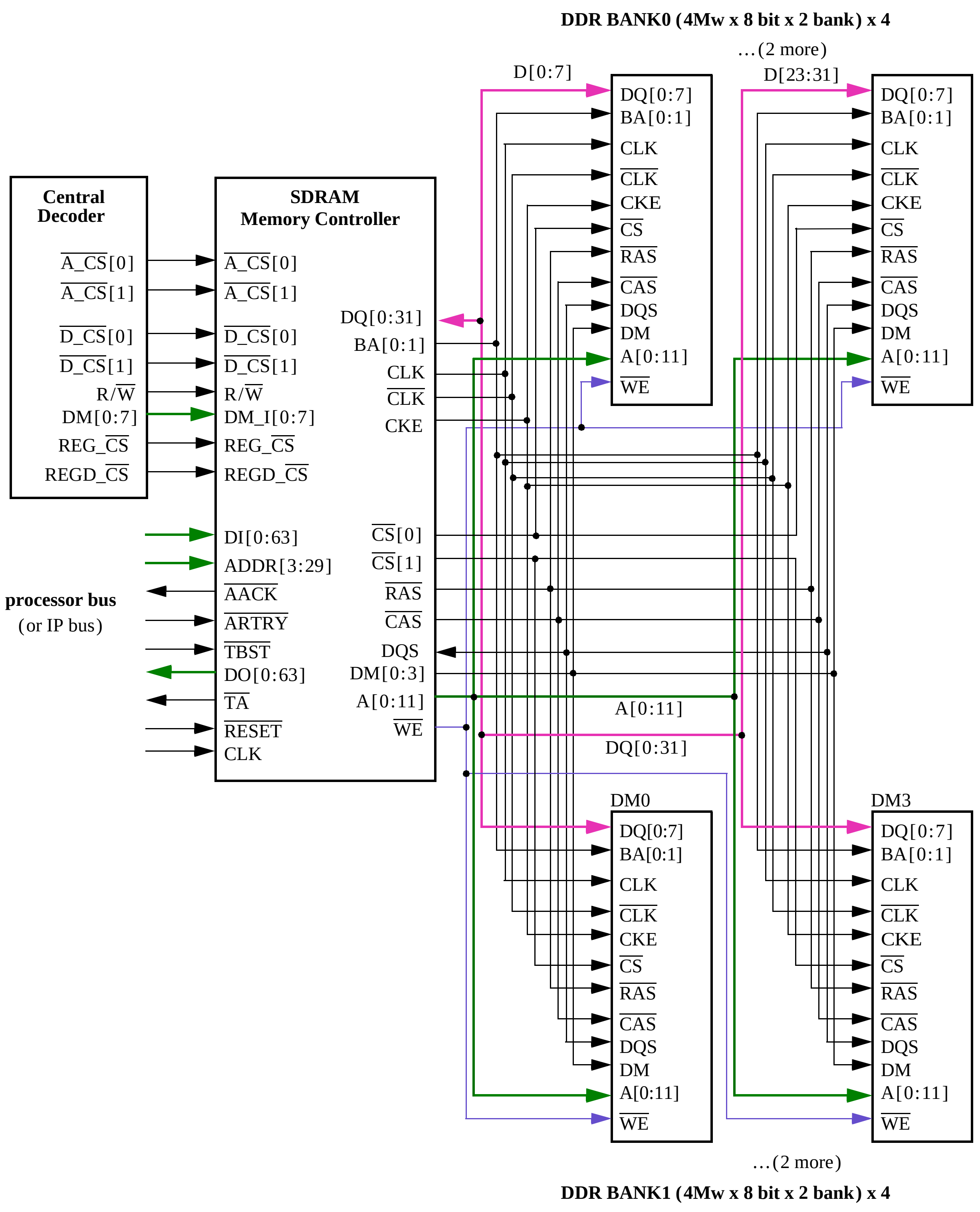


Figure 8-1 Block Diagram—SDRAM Subsystem Example

The SDRAM MC has:

- 1 13-bit multiplexed address bus
- 2 internal bank select lines
- 2 chip selects

The above components support up to either:

- 64MByte address x 32-bit data, or
- 256MB of storage when using 2 chip selects

The Internal SDRAM MC supports 2 chip selects.

- One chip select is pinned out all the time, (i.e., a dedicated pin).
- A second chip select 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.

Both chip selects contribute together to access the whole memory. If 2 chip selects are active, the memory is divided into 2 equal parts:

- $\overline{\text{MEM\_CS}}$[0] accesses the lower part
- $\overline{\text{MEM\_CS}}$[1] accesses the upper part

The SDRAM MC supports both:

- 4MB x 16bit (uses column address 7–0)
- 8MB x 16bit SDRAM configurations

The SDRAM MC does not support:

- a 4MB x 32bit SDRAM (uses column address 6–0)
- (general) memory architectures with less than 8 column lines

The SDRAM MC supports a 32-bit data bus.

NOTE: It is not possible to connect only a 16-bit device to one half of the data bus.

MGT5100 supports an SSO friendly I/O structure where known data is driven on all I/O lines except during a read, where SDRAM is driving. For example, no external bus-keepers are needed. During the switch from write to read, when I/O enables are *negating*, no data change can occur at output buffer input. Such a change is delayed at least one clock edge from the enabled negation.

If a DIMM memory module is used, the SDRAM MC needs a serial I/O interface to read the DIMM module EEROM. This interface is only needed during power-up initialization. One of the 2 I^2C chip interfaces can be used for this purpose by multiplexing its I/O pins.

The MGT5100 SDRAM MC implementation supports only a 32-bit memory data bus. The internal memory address bus is 26bits wide.

- The greatest possible row address width is 14.
- The greatest possible column address width is 12.

NOTE: Maximum row and maximum column can not be used at the same time. Row bits plus column bits must be less than or equal to 24.

In addition to row/column address lines, there are always two bank select bits. Therefore, the greatest possible address space is (2^26) x 32bit, or 256MB of total memory (per chip select).

The minimum number of row address bits is determined by the width of the device mode register(s), not by memory size. The SDRAM MC does not impose minimums on row, column, or bank width. However, address space memory will be fragmented if:

- the address bus is connected one-to-one to the address port, and
- the memory device pinout is not listed in Table 8-1

Table 8-1 Memory Configurations

Row Bits	Bank Bits	Column Bits	Address Range	Bytes (32-bit Bus)
—	0–2	1–8	1KB–8KB	8rB—32KB[1]
11	2	8	2MB	8MB[2,3]
12	2	8	4MB	16MB[2]
12	2	9	8MB	32MB[2]
12	2	10	16MB	64MB[2]
12	2	11	32MB	128MB[2]
12	2	12	64MB	256MB[2]
13	2	8	8MB	32MB[2]
13	2	9	16MB	64MB[2]
13	2	10	32MB	128MB[2]
13	2	11	64MB	256MB[2]
14	2	8	16MB	64MB[2]
14	2	9	32MB	128MB
14	2	10	64MB	256MB

NOTE:
1. These are trivial examples, not useful, but can be programmed into the controller. The only contiguous memory with 1 bank bit is a trivial example. The SDRAMC is really designed for memory with 2 bank bits.
2. The SDRAMC is not designed for memory with less than 8 column bits.
3. The SDRAMC is not designed for memory with greater than 8 column bits, but less than 12 row bits.

8.3.2 Timing Considerations

If maximum performance is desired, MGT5100 limits external memory to a maximum of 2 memory chips placed within 3–5cm of the MGT5100 processor. Maximum performance is not guaranteed if a second chip select is used and more than 2 memory devices are connected.

These restrictions allow a simple fixed delay to the read data clock to meet timing constraints. This method lets the controller avoid using a DLL or delay mirror.

Flight delay on the board can be up to 4ns. The MGT5100 SDRAM board trace delay must be under 2ns. Usually, this means SCRAM lines should be less than 20–25cm long and the memory bank should be placed within 3–5cm from the core

It is possible to drive 4 8-bit wide devices on a single chip select, but maximum performance in not guaranteed. The main problem is controller loading. The maximum load is 25pF/pin.

The MGT5100 implements a simple software adjustable, tap delay circuit, which consists of a buffer string with 8 selectable tap points. This circuit delays the clock used to latch read data from the SDRAM memories into the SDRAM MC. After reset, this delay string defaults to the mid-tap point. If minor read clock delay adjustments are required, the processor can program this delay circuit to other tap points.

For DDR memories, DQS signals from the memories are used as enable signals to determine which of the 4 data bytes are valid. The data itself is latched with the delayed read clock described above, not with the actual DQS signals from the DDR SDRAM memories. DQS signals are not used with SDR memories.

8.3.3 External Signals (SDRAM Side)

Table 8-2 SDRAM External Signals

Signal Name	Description
Bidirectional Signals	
MEM_MDQ[31:0]	Data
MEM_MDQS[3:0]	Data Strobe, DDR only
Outputs	
$\overline{\text{MEM\_RAS}}$	Row Address Select.
$\overline{\text{MEM\_CAS}}$	Column Address Select
$\overline{\text{MEM\_WE}}$	Write Enable
$\overline{\text{MEM\_CS}}$[0], $\overline{\text{MEM\_CS}}$[1]	Command Select. Each bank has a command select to enable commands
MEM_MEMCLK	Memory Clock (frequency is the same as the internal XL bus clock)

Table 8-2 SDRAM External Signals (continued)

Signal Name	Description
MEM_MBA[1:0]	Bank Address. Each SDRAM module has four internal banks. These two bits are used to select the internal bank. They are also used to select SDRAM the internal mode register during power up initialization.
MEM_DQM[3:0]	Data Mask. SDRAM MC drives each of these bits to mask corresponding data bytes which are invalid in the read/write operation.
MEM_MA[12:0]	Address. These are used as either row addresses or column addresses depending on the command issued. When they are used as column addresses, A10 is used as a control signal instead of an address line to control the precharge operation.
$\overline{\text{MEM\_MEMCLK}}$	Memory Clock Bar, DDR SDRAM only.
MEM_CLK_EN	Clock enable. When it is low, the SDRAM clock is disabled. Used in self refresh cycle, and can be used to delay the data in normal SDRAM system.

NOTE: Signals $\overline{\text{MEM\_RAS}}$, $\overline{\text{MEM\_CAS}}$, $\overline{\text{MEM\_WE}}$, $\overline{\text{MEM\_CS}}$[0], $\overline{\text{MEM\_CS}}$[1], plus MEM_MBA, MEM_MA[10], and MEM_CLK_EN are encoded to form a set of SDRAM operation commands to control the different SDRAM operations.

8.4 Operation

8.4.1 Block Diagram

Figure 8-2 shows the SDRAM MC block diagram. It is important to notice:

- the internal XL bus is 64bits wide
- the external interface to the SDRAM is only 32bits wide

The internal XL bus generates the SDRAM address using internal address ADDR[4:29] (32bit wide access).

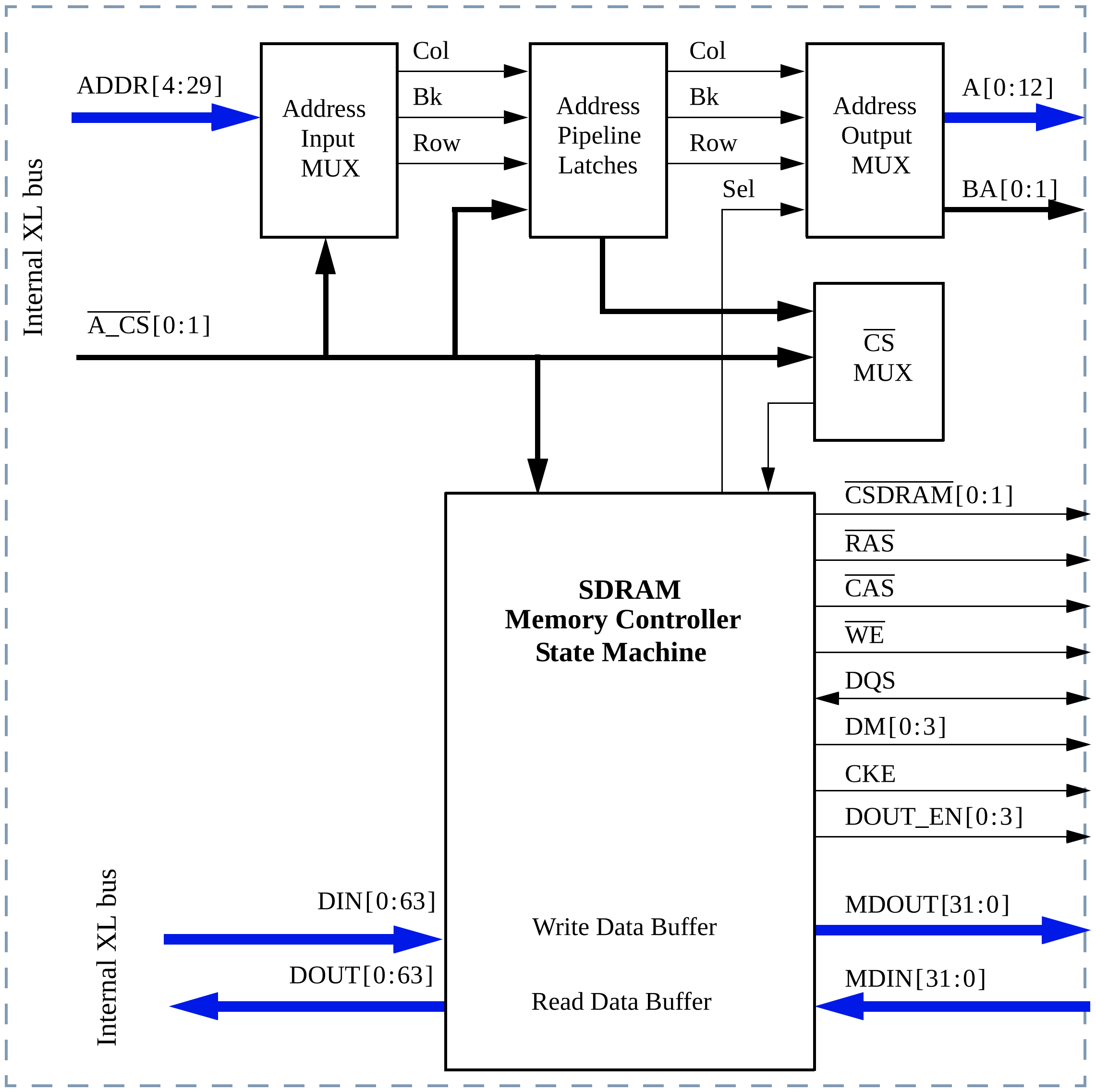


Figure 8-2 Block Diagram—SDRAM Memory Controller

8.4.2 Address Multiplexing and Address Pipeline Blocks

When the SDRAM MC receives the internal signal $\overline{\text{A\_CS}}$[0:1] (see Figure 8-2) as active from a central decoder, it does the following:

- latches the core internal address lines A[4:29]
- multiplexes the core internal address lines into:
  - row address lines
  - column address lines
  - internal bank addresses

In commonly used 64Mbit SDRAM modules, the minimum number of column address lines is 8.

In both 64 Mbit and 128 Mbit SDRAM modules, the number of row address lines is 12 and the internal bank number is 4 (2 address lines used).

To simplify input address multiplexing, the SDRAM MC routes the lower 22 bits (from low to high) to:

- 8 column addresses
- 2 internal bank addresses
- 12 row addresses

The SDRAM MC multiplexes the last 4 higher address lines as higher column or row addresses. This configuration limits the page size to either:

- 1 KB words (8 column addresses and 2 internal bank addresses), or
- 32 Kbits

The SDRAM MC supports only 4 internal bank configurations. Table 8-1 lists commonly used 64 Mbit and 128 Mbit SDRAM module configurations.

The SDRAM MC multiplexes the higher 4 bits (4:7) of the processor address for future 256 Mbit or 512 Mbit module use.

8.4.3 External Chip Select

The 2 pinned-out chip selects are derived from the 4 internal most significant bits, A[4:7]. One address line can be selected to control the memory breakpoint (see Table 8-12).

The maximum address range accessible by a single chip select is 128 MB. This is obtained by reflecting A4 to the output chip selects.

- If A4 is 0, $\overline{\text{MEM\_CS}}$[0] is active.
- If A4 is 1, $\overline{\text{MEM\_CS}}$[1] is active.

If A5 controls the chip select, maximum size for each single bank of memory is 64 MB.

Table 8-3 SDRAM Address Multiplexing

SDRAM	Device	Row bit x	Physical Address Multiplexing						
Device	Configure	Column bits x	—	—	—	—	—	—	—
—	—	Internal Bank	4	5	6	7	8:19	20,21	22,29
—	16Mx4bit	Internal Bank	4	5	6	7	8:19	20,21	22,29
—	—	12x10x4	x	x	Col9	Col8	8:19	20,21	22,29
—	8MBx8bits	—	—	—	—	—	Row	Bk	Col
64 Mbits	—	12x9x4	x	x	x	Col8	Row	Bk	Col
—	—	—	—	—	—	—	11–0	1–0	7–0
—	4MBx16bit	—	—	—	—	—	11–0	1–0	7–0

Table 8-3 SDRAM Address Multiplexing (continued)

SDRAM	Device	Row bit x	Physical Address Multiplexing						
—	—	12x8x4	x	x	x	x	11–0	1–0	7–0
—	32MBx4bit	12x8x4	x	x	x	x	11–0	1–0	7–0
—	—	12x11x4	x	Col10	Col9	Col8	11–0	1–0	7–0
—	16MBx8bit	—	—	—	—	—	Row	Bk	Col

8.4.4 Power-Up Initialization

The SDRAM and SDRAM MC must be initialized after power-up.

The SDRAM MC uses a I^{2}C serial interface block, if the system uses the DIMM module. This serial interface is used to read the memory parameters which are stored in the DIMMs EEPROM. If there is no DIMM module used, the parameter information may be obtained from the SDRAM specification. This information is required to initialize the memory system. See Section 18, Inter-Integrated Circuit (I^{2}C) for more information.

The steps below should be followed to initialize the memory system.

NOTE: The sequence might change slightly from device to device. Please refer to the device data sheet for the most relevant information.

STEP 1: After reset is deactivated, pause for the amount of time indicated in the SDRAM specification. Usually 100µs or 200µs.

STEP 2: Write to the SDRAM MC configuration and control registers to do the following:

- select the clock
- enable SDRAM operation
- set command delay values

If two chip selects are required, it is necessary to enable the second chip select before trying to access the memory bank.

STEP 3: Write to the SDRAM MC control register to set the MODE_EN bit.

When MODE_EN bit is set, a special MODE_SET register in the SDRAM MC can be written. A write to the MODE_SET register causes the SDRAM MC to generate the SDRAM mode (or extended mode) register write command. Thus, the data write to the SET_MODE register also writes to the SDRAM register. SDRAM commands are described in detail in Section 8.4.7, Commands.

STEP 4: Write to the SDRAM MC control register to set the clock enable (CKE) bit, and keep the MODE_EN set.

STEP 5: Write to set_mode register to program SDRAM extended mode register to enable DLL.

STEP 6: Write to SET_MODE register to program the SDRAM mode register to reset DLL mode.

STEP 7: Issue precharge for all commands to precharge all banks.

STEP 8: Issue 2 or more CBR refresh commands.

STEP 9: Write to the SET_MODE register to program the SDRAM mode register to set the SDRAM parameters.

STEP 10: Write to the SDRAM MC control register to clear MODE_EN bit. This bit is cleared by writing 0.

8.4.5 Page Management

To maximize memory access throughput, the SDRAM MC supports paging mode.

During memory access, the SDRAM MC maintains an active page for each memory bank. An active page is composed of the active rows in the internal banks. Page size is a variable of the number of active rows in a bank and can be from 1 row to as many rows as the internal bank number.

For example, if there are 10 column address lines used and 4 internal banks, the page size is 1KB to 4KB words. In a 32bit wide data bus implementation, the page size is 4KB to 16KB. Due to the address decoding implementation in this design, only the lower 256 words in a row are in a continuous physical address. Therefore, the effective active page size is from 256 words to 1KB words.

The SDRAM MC keeps track of memory accesses. If a read or write falls in the same page (row address does not change), the read or write is issued without issuing a precharge and bank active command. The active page remains open until one of the following conditions occurs:

- row address changes
- maximum page active duration is reached
- a refresh cycle starts

To close the page, the SDRAM MC must issue a precharge command.

To open a new page, the SDRAM MC issues a bank active command with page (row) and bank address.

8.4.6 Transfer Size

In the MGT5100 the internal data bus (XL bus) is 64bits wide, while the SDRAM external interface bus is 32bits wide. Therefore, each internal master read/write to the SDRAM MC generates 2 read/write commands to the SDRAM module. The SDRAM MC manages the word size translation (packing/unpacking) between 64- and 32-bit buses.

The SDRAM MC supports both:

- one-beat (single beat)
- burst transfer modes

The burst size is the G2 603e standard 4 double words (32Bytes). Using a 32-bit data bus, the SDRAM system does 8 accesses in each burst transfer.

The SDRAM MC follows the 603e critical double-word first, sequential burst transfer format. Based on the start address issued by the internal master, the order of the 4 double-words in a burst transfer is one of the following:

- 0, 1, 2, 3
- 1, 2, 3, 0
- 2, 3, 0, 1
- 3, 0, 1, 2

This format is used in most SDRAM modules.

Most SDRAM modules support only one transfer size, which is programmed into the Mode Register during initialization. The MGT5100 has a burst length equal to 8.

The SDRAM MC supports single-beat transfers (i.e., single read), to terminate a burst transfer, by issuing either:

- a burst stop command
- precharge command

To implement proper single-beat transfers, the SDRAM MC uses DM[0:3] to mask unwanted bytes or words. The SDRAM MC supports the following single-beat transfer sizes:

- 1Byte
- 2Byte
- 3Byte
- 4Byte
- 8Byte

8.4.7 Commands

When an internal bus master accesses the SDRAM, the SDRAM MC generates the corresponding SDRAM command. Table 8-4 lists SDRAM commands and truth tables supported by the SDRAM MC.

Table 8-4 SDRAM Commands

Function	Symbol	RAS	CAS	WE	BA[0:1]	A10	Other A
Burst stop	BST	H	H	L	X	X	X
Read	READ	H	L	H	V	L	V
Read with auto precharge	READA	H	L	H	V	H	V
Write	WRIT	H	L	L	V	L	V
Write with auto precharge	WRITA	H	L	L	V	H	V
Bank active	ACT	L	H	H	V	V	V
Precharge select bank	PRE	L	H	L	V	L	X
Precharge all banks	PALL	L	H	L	X	H	X
Mode register set	MRS	L	L	L	L	L	V
Extended mode register set	EMRS	L	L	L	H, L	L	V
CBR auto refresh	REF	L	L	H	X	X	X
Self refresh[5]	SREF	L	L	H	X	X	X

NOTE:
1. H = High
2. L = Low
3. V = Valid
4. X = Don’t care
5. Self refresh only can be issued when CKE is inactive (Low).

8.4.7.1 Mode and Extended Mode Register Set Command

Mode Register Set (MRS) and Extended Mode Register Set (EMRS) commands are used during SDRAM initialization.

In this phase, a bus master writes to a special SDRAM MC Mode Register. The SDRAM MC then generates the MRS or EMRS commands. In these two operations, data written to the SDRAM module is put on the SDRAM address bus. The SDRAM MC transfers the data in the internal data bus to the SDRAM address bus.

The BA[0:1] value, which decides if the command is MRS or EMRS, is contained in the data instead of an address as in normal write operations.

The last value written to the Mode Set Register contains SDRAM parameters used by the SDRAM MC for command generation. The parameters are:

- burst length
- wrap type
- latency mode

The SDRAM MC Mode Set Register must be enabled before writing, and disabled after writing. This is done by setting or clearing the Control Register MODE_SET_EN bit.

8.4.7.2 Precharge Select Bank and Precharge all Banks Commands

The precharge command puts SDRAM into an idle state. In this state, the following commands can be issued:

- refresh
- active
- mode register set

Other read/write commands can be issued after SDRAM is activated. A delay period is required after the precharge command and before any command can be issued. During initialization, the delay value is saved in the SDRAM MC Configuration Registers (CFG1, CFG2).

- The Precharge Select Bank Command precharges 1 internal_bank selected by BA[0:1].
- The Precharge All Banks Command precharges all banks in the SDRAM module.

After initialization, the SDRAM MC issues a Bank Active Command to allow access to the memory. For page mode support, the SDRAM MC keeps the active bank open as long as possible. The SDRAM MC does not issue the next precharge commands until one of the following conditions occurs:

- access to a new bank (row address changed)
- end of refresh period is reached
- end of maximum bank open duration is reached

The SDRAM MC issues the following:

- Precharge All Bank Command for initialize and refresh operations
- Precharge Select Bank Command when:
  - the open bank limit is reached
  - the row address of next access changes

If the following 2 conditions exist:

- an access is waiting in the address pipeline, and
- the SDRAM MC finds the row address is to change

If the operation has not started, the SDRAM MC issues a read/write with auto precharge command for the previous stage operation.

8.4.7.3 Bank Active Command

In an SDRAM module each internal bank can have one active row. The Bank Active Command activates one row.

In the MGT5100 SDRAM MC there are 2 bank address lines. They are located between row addresses and column addresses. Therefore, active rows in an SDRAM module can be in continuous memory locations. These continuous active rows form an active page.

After a precharge command, a Bank Active Command is required to access the bank. The SDRAM MC issues the Bank Active Command (with row and bank addresses) when it receives a read/write command from the master to access an inactive page after precharging the bank.

8.4.7.4 Read Command

When the SDRAM MC receives a master read command, it first checks whether the read is to an inactive row. If to an inactive row, the SDRAM MC issues the following commands:

1. a precharge command
2. a bank active command (with row and bank address)
3. a read command

Each command issued must meet certain delay requirements. These requirements are controlled by the saved value in the SDRAM MC Configuration Registers.

NOTE: Each read is a burst read operation.

Burst length is controlled by a value stored in the SDRAM MC Configuration Register. To support a G2 32Byte burst transfer, the value should be 8. Precisely, a 0x07 is written in the CFG2 Burst Length field.

The master read is a 64-bit operation. The MGT5100 SDRAM interface bus width is 32bits. Therefore, each 64-bit master data read requires 2 SDRAM interface operations:

- The first data-read reads the higher 4Bytes.
- The second data-read reads the lower 4Bytes.

The SDRAM MC packs the 2 reads together, then sends them to the G2 processor bus.

To terminate the burst read if the master read is a single-beat read, the SDRAM MC may issue either:

- a precharge command, or
- a burst stop command

If another command to access the same page is in the pipeline, the SDRAM MC can issue the next command without waiting for the previous burst transfer to finish. This is done only:

- after the single-beat command, and
- after meeting the required delay

In the DDR system, the burst stop command must be issued if the next command is a write command.

The SDRAM MC also drives DM[0:3] to mask unwanted data bytes.

During a read operation, the SDRAM MC issues precharge and active commands (if necessary) with row addresses as early as the G2 internal address tenure starts. The SDRAM MC issues a read command during the G2 data tenure with the column address saved in a latch.

8.4.7.5 Write Command

When the SDRAM MC receives a master write command, it first checks whether the write is to an inactive row. If to an inactive row, the SDRAM MC issues the following commands:

- a precharge command
- a bank active commands (with row and bank addresses)
- a write command

Each command must meet certain delay requirements. These requirements are controlled by the saved value in the SDRAM MC Configuration Register.

NOTE: Each write is a burst write operation.

Burst length is controlled by a value stored in the SDRAM MC Configuration Register. To support the G2 32 Byte burst transfer, this value should be 8.

The master write is a 64-bit operation. the MGT5100 SDRAM interface bus width is 32 bits. Therefore, each 64-bit master data write requires 2 SDRAM interface operations.

- The first data-write writes the higher 4 Bytes.
- The second data-write writes the lower 4 Bytes.

The SDRAM MC unpacks the G2 data, then sends them out in 2 consecutive writings.

To terminate the burst write if the master write is a single-beat write, the SDRAM MC may issue either:

- a precharge command, or
- a burst stop command (not valid for DDR devices)

If another command to access the same page is in the pipeline, the SDRAM MC can issue the next command without waiting for the previous burst transfer to finish. This is done only:

- after the single-beat command, and
- after meeting the required delay

The SDRAM MC also drives DM[0:3] to mask unwanted data bytes.

During a write operation, the SDRAM MC issues precharge and active commands (if necessary) with row addresses as early as the G2 internal address tenure starts. The SDRAM MC issues a write command during the G2 data tenure with the column address saved in a latch.

In case a DDR memory is used, the SDRAM MC drives the DQS to SDRAM properly.

8.4.7.6 Burst Stop Command

The Burst Stop Command stops the following operations:

- current read burst operations (both SDR and DDR SDRAM systems)
- write operations (SDR SDRAM system)

Both read/write and burst stop commands must meet certain delay requirements. These requirements are controlled by the saved value in the SDRAM MC Configuration Register.

In the single-beat read/write operation, if precharge is not necessary, the SDRAM MC may issue the burst stop command to terminate the burst operation.

Latency issues exist after the command is issue and before the transfer is stopped. the SDRAM MC Configuration Register stores the latency value. The SDRAM MC uses this value to control DM[0:4] to mask extra data bytes.

8.4.7.7 Auto-Refresh and Self-Refresh Commands

The SDRAM MC supports:

- auto-refresh in operation mode
- self-refresh in sleep mode

The SDRAM MC issues an auto-refresh command according to the interval value specified in the SDRAM MC Control Registers. Before executing auto-refresh, the SDRAM MC issues the Precharge All Bank Commands.

After refresh, the SDRAM is in an idle state and waits for an active command.

Delay requirements exist between the commands. The delay values are programmed in the SDRAM MC Configuration Registers. The refresh interval value in the Configuration Register is the number of the scaled clock that clocks the refresh counter.

If a memory access is in progress at the time of the refresh interval being reached, the SDRAM MC extends the refresh interval to wait for the transfer to finish.

For example, the typical SDRAM module requires 4096 refresh cycles in 64ms (which is often written in a data sheet as the minimum refresh time = 15.625µs). As an example, assume the SDRAM MC clock frequency is 100MHz. Then, the refresh interval without considering the transfer conflict should be:

$$15.625\mu s \times 100MHz = 1562 \text{ (clock cycles)}$$

Then, having an internal pre scaler (equal to 64) we get:

$$1562 \div 64 = 24$$

The number 23 should be the value stored in the SDRAM MC Configuration Register (as the pre scaler count starts at 0).

When the MGT5100 system enters sleep mode, a control bit in the Control Register is set. Software then does the following:

- issues Precharge All Banks Command(s)
- de-asserts the CKE output to stop the SDRAM clock
- issues a self-refresh command

Software can use control bits stored in the SDRAM MC Control Register to issue the SDRAM commands.

8.5 Programming the SDRAM Memory Controller

The SDRAM MC registers consist of:

- 1 16-bit Special Mode Register
- 1 32-bit Control Register
- 2 32-bit Configuration Registers
- 1 2-bit Address Select Register

All registers are word-aligned in memory. Register descriptions in the sections below adhere to G2 microprocessor convention, as the HARPO core is a true G2 microprocessor. What this means is:

- the left bit—the most significant bit (msb) of a register—is bit 0
- the right bit—the least significant bit (lsb) of a register—is bit 31

In addition:

- the Most Significant Byte (MSB) is Byte 0, Big Endian convention—comprised of bits 0:7
- the Last Significant Byte (LSB) is Byte 3—comprised of bits 24:31

8.6 SDRAM Memory Controller Registers (MBAR + 0x0100)

SDRAM MC registers are located at an offset from MBAR of 0x0100. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0100 + register address**

Hyperlinks to the SDRAM MC registers are provided below:

- Mode Register 0 (0100)—MOD, write only
- Extended Mode Register 0, DDR Only (0100)—MOD
- Control Register 1 (0104)—CTR
- Configuration Register 1 (0108)—CFG1
- Configuration Register 2 (010C)—CFG2
- Address Select Register (0110)—ADRSEL

8.6.1 Mode Register (MOD)

The 32-bit write-only Mode Register (MOD) initializes the external SDRAM Mode Register and the Extended Mode Register. This register is reset only by a power-up reset signal.

NOTE: The Extended Mode Register is used only in DDR to control DLL and drive capabilities.

During SDRAM MC initialization (boot-up), system software sets the Control Register bit MODE_SET_EN. This bit allows access to the MOD register. The MOD register can only be written when the Control Register MODE_SET_EN bit is set. For more information on the Control Register, see Section 8.6.2.

Also during boot-up, the SDRAM MC generates the mode (or extended mode) set command. Data written to the MOD register is automatically transferred to the SDRAM device. After being written, the MODE_SET_EN bit should be cleared by software.

Table 8-5 Mode Register 0 (0100)—MOD

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	BA[1:0]		Operating Mode					Read CAS Latency			BT	Burst Length			Rsvd	WS
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

Bit	Name	Description
0:1	BA[1:0]	Internal memory device bank addressing.
2:6	Operating Mode	Normal operation/Reset DLL (see vendor's manual).
7:9	Read CAS Latency	Refer to memory device data sheet. • SDRAM supported values are 2,3 • DDR values are 2,2.5 Define delay value from read to data available. Actual value should be found in SDRAM module specification.
10	BT	Burst Type—MGT5100 supports only sequential burst (no interleaved). Therefore, 0 should be written to this bit.
11:13	Burst Length	Burst Length selection—supported values can be found in the SDRAM module specification. SDRAM MC supports only burst length 8 (usually 011).
14	—	Reserved
15	WS	Write Strobe—writing 1 to this bit generates a write strobe pulse, which writes data to the SDRAM register. This bit is automatically cleared.
16:31	—	Reserved
NOTE: Bit allocation detail varies from device to device.		

Table 8-6 Extended Mode Register 0, DDR Only (0100)—MOD

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	BA[1:0]		Operating Mode									QFC	DS	DLL	Rsvd	WS
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

Bit	Name	Description
0:1	BA[1:0]	Internal memory device bank addressing.
2:10	Operating Mode	Normal operation/Reset DLL (see vendor's manual).
11	QFC	QFC function—refer to memory device datasheet.
12	DS	Drive strength—refer to memory device datasheet. Usually set to 1 to avoid ringing.
13	DLL	Enable/Disable DLL—refer to memory device datasheet.
14	—	Reserved
15	WS	Write Strobe—writing 1 to this bit generates a write strobe pulse, which writes data to the SDRAM register. This bit is automatically cleared.
16:31	—	Reserved
NOTE: Bit allocation detail varies from device to device.		

8.6.2 Control Register (CTR)

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 8-7 Control Register 1 (0104)—CTR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MODE Set En	Ck En	DDR Mode	Ref Cnt En	Fast XLB	Rsvd	Address Sel		DDR 32bit	Mem Data Drv	Refresh Counter Value					
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				DDR DQS En				Reserved			Buff DIMM	Rsvd	Soft Ref	Soft Pre	Rsvd
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	MODE Set En	Mode set enable—During SDRAM software (SW) initialization, writing 1 to this bit enables access to SDRAM internal mode_set register and extended_mode set_register by writing through the SDRAM MC mode register. After mode_set_register programming is finished, this bit must be cleared by SW.
1	Ck En	Clock enable—Writing 1 to this bit enables the SDRAM clock. Software can use the bit to disable the SDRAM clock in the sleep mode after the precharge and self_refresh commands issued.
2	DDR Mode	Double Data Rate SDRAM (DDR) mode—If bit is set, SDRAM is DDR type. If bit is cleared, normal SDRAM is in use. This bit must be written properly during SDRAM initialization.
3	Ref Cnt En	Auto refresh enable—If bit is set, the auto refresh counter is enabled and the SDRAM MC issues an auto refresh command when the counter reaches TC (terminal count). If bit is cleared, the refresh counter is disabled. In sleep mode, if self refresh command is issued, this bit should be set. On exiting sleep mode this bit should be cleared.
4	Fast XLB	Fast XL bus clock—If core clock to XL bus clock ratio is 1:1 or 1.5:1, write 1 to this bit. Otherwise, write 0.
5	—	Reserved
6:7	Address Sel	Row/Column Address selection—set value for these 2 bits according to Table 8-8.
8	DDR 32 bit	To be set only for those 'special' (not all) 32 bit DDR memory which use address line A8 to control the PRECHARGE-ALL operation. In all other case (when address line A10 is used) this bit must be cleared (set to 0).
9	Mem data drv	Memory data bus driving control. When set, the SDRAM MC drives the memory data bus always except read period. When cleared, the SDRAM MC only drive the bus during the write. It is suggested to set it always to 1 to reduce power consumption due to floating lines during no write operations.

Bit	Name	Description
10:15	Refresh Counter Value	This is the minimum time interval used by the AUTO REFRESH memory operation. It is measured in MCLK/64. To compute the value: 1) Read the data sheet “Average Periodic Refresh Interval” (i.e., 15.625uS), multiply by MCLK (i.e., 99MHz). Thus, 15.625uS x 99MHz = 1546 In the example above, round to the nearest lower integer **not** to exceed the maximum timing. 2) Divide the obtained value by the pre-scaler (= 64), round to lower integer: 1546/64 = 24 3) To be safe subtract 1 from the previously obtained value and write the value in register bits 10:15 in hexadecimal form, 0x17 (23).
16:19	—	Reserved
20:23	DDR DQS En	Each bit individually control one DDR DQS output. If set, it enables all four DQS pins for DDR modules, which have a DQS signal per byte.
24:26	—	Reserved
27	Buff DIMM	Set this bit to 1 in case buffered DIMM is in use. Clear it otherwise.
28	—	Reserved
29	Soft Ref	Auto refresh software generated command. When this bit sets, the SDRAM MC generates an auto refresh command to both banks of the SDRAM module. This bit is cleared automatically after the command generated.
30	Soft Pre	precharge all banks software generated command. When this bit sets, the SDRAM MC generates a precharge all bank command to both banks of the SDRAM module. This bit is cleared automatically after the command generated.
31	—	Reserved

The Table 8-8 indicates how address 4–7 are multiplexed internally to support higher column address or row address. Check the specific memory data sheet to determine the column/row to which the memory is mapped.

Table 8-8 Address Select

ADDR_SEL	G2 Address Line (Mapped to Column or Row Address)			
	4	5	6	7
00	Col 11	Col 10	Col9	Col 8
01	Col 10	Col 9	Col 8	Row 12
10	x	x	x	x
11	x	x	x	x

8.6.3 Configuration Register 1 (CFG1)

The 32-bit read/write Configuration Register 1 stores the delay values used and specific SDRAM commands. During initialization, software loads values to the register according to the SDRAM information obtained from the data sheet.

The SDRAM MC uses the stored values to generate the proper delay when issuing commands. The value stored is expressed in “number of clocks”. This register is reset only by a power-up reset signal.

All delay values specified between commands is a minimum value (except latency value, which is fixed). A longer delay also works, but with decreased overall performances.

Normally, but not always, SDRAM memory and DDR memory have different values. This occurs even if the data sheet timing is the same, because:

- SDRAM expresses delay in MCLK
- SDRAM DDR expresses delay in MCLK2

By definition:

MCLK—SDRAM MC clock—has the speed of the SDRAM interface and is equal to the internal XL bus clock. MCLK is fixed once at boot time via the HW RESET WORD setting. It is an integer multiple of the external reference clock (e.g., 66 MHz, 99 MHz or 132 MHz if a 33 MHz reference is used).

MCLK2—double frequency of MCLK—used mainly for DDR, as DDR uses both edges of a normal clock (MCLK) to read/write data.

Micron Technology, Inc® provides additional explanations and examples of using the DDR device MT46V16M8-75Z.

Table 8-9 Configuration Register 1 (0108)—CFG1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Single Read to Read/Write Delay				Rsvd	Single Write to Read/ Precharge Delay			Read CAS Latency				Rsvd	Active to Read Delay		
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	Precharge to Active Delay			Exit Refresh to no Read Delay				Rsvd	Write 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	Single Read to Read/Write Delay	For DDR—this is counted by MCLK2 starting from 0 (as a counter or a "C" routine would do). The equation to compute the value is: Clock Latency (CL) + Data Valid + 1 MCLK for turnaround (lets memory prepare for next operation, to switch from Rx to Tx if needed). For example, referring to DDR memory detailed above: Clock Latency (= 4 MCLK2) + Last Valid Data (= 2 MCLK) + 1 MCLK (= 2 MCLK2) => 8 MCLK2. Counting from 0 this implies writing 7 (or better 0x07). For SDR—this is counted by MCLK. The value suggested here is 12 or 0x0C.
4	—	Reserved
5:7	Single Write to Read/ Precharge Delay	This is the last data valid computed in MCLK2 count starting from 0. Thus: For DDR—write 0x03 (equivalent to 4 MCLK2) For SDRAM—there is 1 Write Latency. Thus, value should be 2 (equivalent to 3 MCLK)
8:11	Read CAS Latency	For DDR—this is expressed in MCLK2 If CL=2, write 6 If CL=2.5, write 7 For SDR—this is expressed in MCLK If CL=2, write 2 If CL=3, write 3 NOTE: CL=2.5 is not supported for SDR. For both kind of memory, in case a buffered DIMM is used, add one more MLCK. For example, add 2 MCLK2 for DDR.
12	—	Reserved
13:15	Active to Read Delay	Active Command to Read/Write delay—read into the memory data sheet the t_{RCD} value and express it in MCLK rounding up to nearest integer. EXAMPLE: For DDR: t_{RCD} = 20nsec If MCLK = 99MHz, then period is 10.1nsec and value to write is 2 (0x02). Usually set to 0x02 for SDR as well.
16	—	Reserved
17:19	Precharge to Active Delay	Precharge to Active Command Delay—read in memory data sheet the t_{RP} value and express it in MCLK cycles. EXAMPLE: For DDR: t_{RP} = 20nsec If MCLK = 99MHz, value to write is 0x02. Usually set to 0x02 for SDR as well.
20:23	Refresh to No-Read Delay	Refresh to Active Command Delay—sometime referred to as Refresh to No-Read Command Delay. This must be expressed in MCLK cycles. EXAMPLE: For DDR: t_{XSBNR} = 75nsec IF MCLK = 99MHz, value to write is 0x08. Usually set to 0x02 for SDR as well.
24	—	Reserved
25:27	Write Latency	For DDR—expressed in MCLK2. Correct value should be 3 For SDRAM—value should be 0, as there is no write latency for SDRAM.
28:31	—	Reserved

8.6.4 Configuration Register 2

The 32-bit read/write Configuration Register 2 stores the delay values used with (mostly) BURAT related SDRAM commands.

The SDRAM MC uses the stored values to generate the proper delay between the commands. The value stored is expressed in “number of clocks”. This register is reset only by a power-up reset signal.

All delay values specified between commands is a minimum value (except burst_length, which is fixed). A longer delay also works, but with decreased overall performances.

This register differs from CFG1, in that delays are expressed mainly in MCLK.

Table 8-10 Configuration Register 2 (010C)—CFG2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	brdtopre delay				bwt delay				brd delay				burst length			
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													Read Tap		
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	brdtopre delay	Burst to Read/Precharge delay—is strictly the number of clocks (counted starting from 0) needed to do an entire Burst plus 1 clock for turnaround. It is expressed in MCLK. For DDR—Write 0x04—MGT5100 supports only Burst of length equal to 8-bit word. For DDR memory, this means 4 MCLK, plus 1 for turnaround (5). Because the count starts at 0, the value is written as 4. For SDR—Write 0x08—same reason for value as for DDR, but memory accesses one word per MCLK cycle.
4:7	bwt delay	Burst Write delay—expressed in MCLK. Similar to Burst to Read/Precharge. However, DDR needs an extra MCLK to accommodate Write Latency, which SDR does not have. For DDR—Write 0x05—equivalent to 6 MCLK = 4+1 for Write Latency +1 Turnaround For SDR—Write 0x08—same as above.
8:11	brd delay	Burst Read to Write Delay—Borst + read latency + write latency + turn around For DDR—0x07 (8 MCLK)—4 MCLK + 1 MCLK + 2 MCLK, minus rounding write latency of 1.5 MCLKs to 2 MCLKs. For SDR—0x0B (12 MCLK)—8 MCLK + 2 MCLK + 0 MCLK + 1 MCLK
12:15	burst length	Burst Length: MGT5100 supports only a Burst Length equal to 8 consecutive words. Thus for both SDR and DDR write 0x07.

Bit	Name	Description
16:28	—	Reserved
28:31	Read Tap	It is used to ‘calibrate’ read delay. Value suggested is 0x04

8.6.5 Address Select Register (ADRSEL)

The 32-bit read/write Address Select Register indicates which XL bus bit the SDRAM MC uses for chip select. This register also defines the maximum addressable memory space to be used.

Table 8-12 shows the relationship between addressable space and selecting a chip select.

Table 8-11 Address Select Register (0110)—ADRSEL

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					XLB_SEL		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
5:6	XLB_SEL	XL Bus Select—used to multiplex which XL bus address bit controls chip selects for the memory modules. The total memory accessible by the SDRAM MC can be divided into 2 equal sized part. The breakpoint is then fixed by selecting the XLB_SEL bits as described in the Table 8-12.
7:31	—	Reserved

Table 8-12 XL Bus Select

XLB_SEL	Selected XL Bus Bit	Maximum Addressable Memory
00	7	32MB
01	6	64MB
10	5	128MB
11	4	256MB

Available chip selects are:

- CSDRAM_0 on ball B18
- CSDRAM_1 on ball C15 (MUXed with GPIO_WAKEUP6)

To use both chip selects:

1. CSDRAM_1 must first be enabled. Do this by setting the most significant bit (left most bit) of the GPIO Configuration Register, located at MBAR+0xB00, to 1. Take care not to change the other GPIO Configuration Register bits.

2. After the chip select pin is activated, the split between the 2 pins is selected. This is done by choosing the XLB_SEL bits value.

   Example—If XLB_SEL=b'00:

   – the first 32 MB of address are controlled by $\overline{\text{MEM\_CS}[0]}$
   – the second 32 MB of address are controlled by $\overline{\text{MEM\_CS}[1]}$.

8.7 Examples

8.7.1 Example—Physical Address Multiplexing

Examples of XL bus bit ordering are shown in Figure 8-3.

XL bus defaults are:

- Row address is bits 8:19
- Column address is bits 22:29
- Bank selects are bits 20 and 21

Use the DDR SDRAM memory module from Micron (MT46V32M16) as the example part. The Micron part is 8 MByte x 16 bit x 4 banks, which is 512 MB per part. Configuring this part to use a 32-bit bus, requires 2 modules. That equals 128 MB of data.

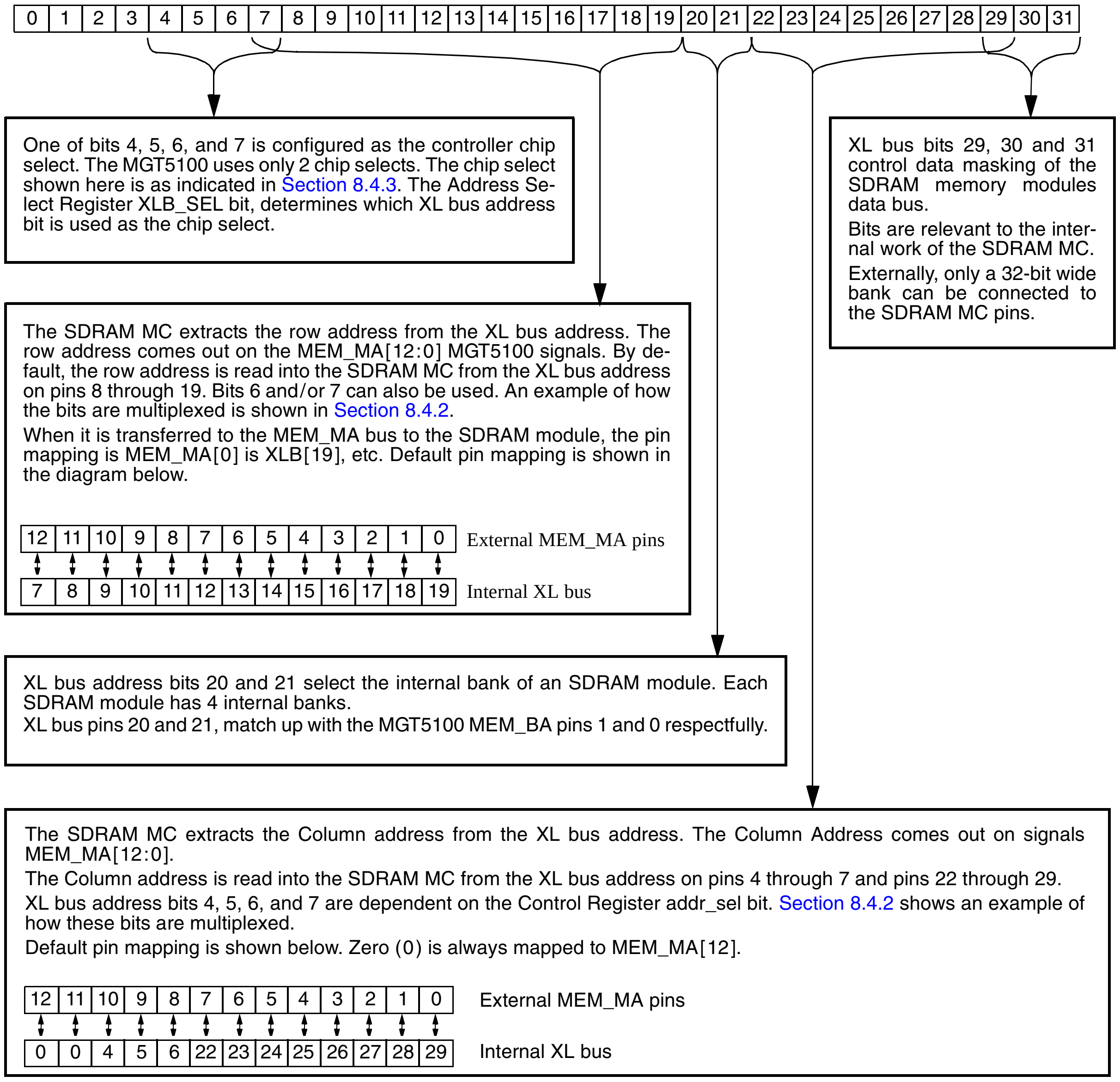
The Micron data sheet shows the following requirements:

- 13 row addresses
- 10 column addresses
- 2 bank selects

Table 8-8 shows the ADDR_SEL bit can be set to b'01. This setting allows access to 13 row addresses and 11 column addresses. This setting fits our needs and has one extra column bit.

Table 8-12 shows XLB_SEL bit assignments. Configure memories for a maximum accessible address space of 128 MB. Set XLB_SEL to b'10 to match 128 MB memory size value.

8.7.2 Example—Address Bus Mapping



This is an example of how the XL bus address enters the SDRAM MC and is broken down into its individual functions. Shown below is the 32-bit XL bus. The SDRAM MC uses bits 4 though 31.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

One of bits 4, 5, 6, and 7 is configured as the controller chip select. The MGT5100 uses only 2 chip selects. The chip select shown here is as indicated in Section 8.4.3. The Address Select Register XLB_SEL bit, determines which XL bus address bit is used as the chip select.

XL bus bits 29, 30 and 31 control data masking of the SDRAM memory modules data bus.

Bits are relevant to the internal work of the SDRAM MC.

Externally, only a 32-bit wide bank can be connected to the SDRAM MC pins.

The SDRAM MC extracts the row address from the XL bus address. The row address comes out on the MEM_MA[12:0] MGT5100 signals. By default, the row address is read into the SDRAM MC from the XL bus address on pins 8 through 19. Bits 6 and/or 7 can also be used. An example of how the bits are multiplexed is shown in Section 8.4.2.

When it is transferred to the MEM_MA bus to the SDRAM module, the pin mapping is MEM_MA[0] is XLB[19], etc. Default pin mapping is shown in the diagram below.

12	11	10	9	8	7	6	5	4	3	2	1	0	External MEM_MA pins
7	8	9	10	11	12	13	14	15	16	17	18	19	Internal XL bus

XL bus address bits 20 and 21 select the internal bank of an SDRAM module. Each SDRAM module has 4 internal banks.
XL bus pins 20 and 21, match up with the MGT5100 MEM_BA pins 1 and 0 respectfully.

The SDRAM MC extracts the Column address from the XL bus address. The Column Address comes out on signals MEM_MA[12:0].

The Column address is read into the SDRAM MC from the XL bus address on pins 4 through 7 and pins 22 through 29.

XL bus address bits 4, 5, 6, and 7 are dependent on the Control Register addr_sel bit. Section 8.4.2 shows an example of how these bits are multiplexed.

Default pin mapping is shown below. Zero (0) is always mapped to MEM_MA[12].

12	11	10	9	8	7	6	5	4	3	2	1	0	External MEM_MA pins
0	0	4	5	6	22	23	24	25	26	27	28	29	Internal XL bus

Figure 8-3 Example—Address Bus Mapping

8.7.3 Example—Setting Registers for Standard SDRAM

This example shows how to configure the SDRAM MC for using the Micron SDRAM module MT48LC8M16A2TG-75.

The memory is configured for 2 16-bit chips, totaling 64MB of addressable memory.

First, define what the last valid data value is. All values in the equations are relative to clock cycles. The last valid data value is defined as 2 clocks for each 32bits read from memory for a single beat transaction.

To calculate the value to be read into the Configuration Register 1 for this memory module, the equations below apply.

Assume the use of a 100MHz memory clock in the calculations. Calculate this example using a latency of 2.

Latency (read latency delay) = 2
(can be either 2 or 3 for current SDR SDRAM models)

Last valid data = 2ck
(2ck for each 32-bit read from memory for a single-beat transaction)

Burst read = The SDRAM MC always does a burst read which takes 8 clocks.

Table 8-13 Configuration Register 1 = 0xC222 2700

Bit	Description
SRD_DL	0x0C—Single read to write.
SWT_DL	0x02—Single write to read/precharge.
LATENCY_DL	0x02—Read Latency delay.
ACTORW	0x02—Active to read/write delay. Refer to Micron MT48LC8M16A2TG data sheet, page 34. From t_{RCD} symbol (Active to read or write delay).
PRETOAC	0x02—Precharge to active delay. Refer to Micron MT48LC8M16A2TG data sheet, page 34. From t_{RP} Symbol (Precharge command period).
REFTOAC	0x07—Refresh to active delay. Refer to Micron MT48LC8M16A2TG data sheet, page 34). From t_{XSR} Symbol (Exit Self Refresh to active command).
WT_LAT	0—Write latency to memory. Refer to Micron MT48LC8M16A2TG data sheet, page 50. The data sheet shows the Write command is on the same clock period as the valid data.

Table 8-14 Configuration Register 2 = 0x88B7 0004

Bit	Description
brdtopre = 8	Burst read to read/prech delay = 0x08 for SDR
bwt_dl = 8	Burst write to R/W delay = 0x08 for SDR
brd_dl = Hb	Burst read to write delay = 0x0C for SDR
burst_lth = 7	Burst length = 0x07 (only this value is supported)
rdly_tap = 4	Read delay tap—Set to center of tap delay = 4

Table 8-15 Control Register = 0xD858 000x[1]

Bit	Description
mode_set_en = 1	Mode set enable = 1, to access the mode_set_register in the SDRAM module.
clk_en = 1	Clock Enable = 1, to enable the SDRAM clock.
ddr_mode = 0	Double data rate SDRAM mode = 0, because this is **not** a DDR SDRAM.
ref_cnt_en = 1	Refresh counter enable = 1, because the auto-refresh option is desired.
fast_xlb = 1	Fast XL bus clock = 1, if a 1:1 ratio exists.
addr_sel = b00	Address selection = 00, to multiplex the address bits according to Table 8-8.
ddr_32bit = 0	DDR module support = 0, because an SDR module is being used.
m_data_drive = 1	Memory data bus driving control = 1, because it is desirable to always drive the memory data bus, so that the pins do not float and consume power.
ref_cnt = 011000	H18, Refresh counter enable—An example of refresh interval count being calculated is found in Section 8.4.7.7, Auto-Refresh and Self-Refresh Commands.
ddr_dqs_en = H0	DDR dqs enable i = H0, because this is a SDR module.
buff_dim = 0	Buffer DIMM = 0, because DIMM is not being used.
soft_ref = 0	Auto refresh = 0, on the first write to the Control register. The second time the Control register is written, this bit is set to do a software refresh.
soft_pre = 0	Precharge all = 0, on the first write to the Control register. The third time the Control register is written, this bit is set to do a software precharge.
NOTE: Last nibble can be varied according to SW operation: Auto Refresh or Precharge All.	

Table 8-16 Mode Register = H008D 0000

Bit	Description
mode_code = H008d	Operating mode—value comes from the memory modules specification, listed under operating mode. In most instances, under normal operation mode, these bits are set to 0.

Table 8-17 Load Sequence for Registers

Bit	Description
Write Configuration 1 Register	0xC222 2270—Load Configuration 1 Register.
Write Configuration 2 Register	0x88B7 0004—Load Configuration 2 Register.
Write Control Register	0xd858 0000—Load Control Register.
Write Control Register	0xd858 0002—Loading Control Register again, does a software refresh
Write Control Register	0xd858 0004—Loading Control Register again, does a software precharge.
Write ADRSEL Register	0x01
Write Mode Register	0x008D 0000—Load Mode Register to set burst length and latency.
Write Control Register	0x5858 0000—Load Control Register to remove SDRAM mode set enable. Most significant bit is now 0.

8.7.4 Example—Setting Registers for DDR SDRAM

This example shows how to configure the SDRAM MC for using the Micron DDR SDRAM Module MT46V16M8TG-75Z.

Assuming the use of a 99 MHz SDRAM MC clock in our calculations. MCLK, corresponds to having an external clock reference of 33 MHz. Calculate this example using a latency of 2. Configured to 64 MB using 4 chips to form a 32-bit bus.

Table 8-18 Configuration Register 1 = 0x7362 2830

Bit	Description
srd_dl = 7	Single read to write.
swt_dl = 3	Single write to read/precharge.
latency_dl = 6	Read Latency delay.
actorw = 2	Active to read/write delay = 2. Refer to Micron MT46V32M16TG data sheet, page 48. From t_{RCD} symbol (Active to read or write delay).
pretoac = 2	Precharge to active delay = 2. Refer to Micron MT46V32M16TG data sheet, page 48. From t_{RP} Symbol (Precharge command period).
reftoac = 7	Refresh to active delay = 8. Refer to Micron MT46V32M16TG data sheet, page 48). From t_{XSR} Symbol (Exit Self Refresh to active command).
wt_lat = 3	0—Write latency to memory.

Table 8-19 Configuration Register 2 = 0x4577 0004

Bit	Description
brdtopre = 4	Burst read to read/prech delay
bwt_dl = 5	Burst write to R/W delay
brd_dl = 7	Burst read to write delay
burst_lth = 7	Burst length
rdly_tap = 4	Read delay tap

Table 8-20 Control Register = 0xF051 0F00

Bit	Description
mode_set_en = 1	Mode set enable = 1, to access mode_set_register in DDR SDRAM module.
clk_en = 1	Clock Enable = 1, to enable DDR SDRAM clock.
ddr_mode = 1	Double Data Rate SDRAM mode = 1, because this is DDR SDRAM.
ref_cnt_en = 1	Refresh counter enable = 1, because the auto-refresh option is desired.
fast_xlb = 0	Fast XL bus clock = 0, because a 1:1 ratio does not exist. Assume the HARPO core runs more than 99 MHz.
addr_sel = ‘b00	Address selection = 00 to multiplex the address bits according to Table 8-8.
ddr_32bit = 0	DDR module support = 0, because a single 32-bit DDR module is not used.
m_data_drive = 1	Memory data bus driving control = 1, because it is desirable to always drive the memory data bus, so that the pins do not float and consume power.

Table 8-20   Control Register = 0xF051 0F00 (continued)

Bit	Description
ref_cnt = 011001	‘H18, Refresh counter enable—use 0x17, delay from data sheet is 15.625us
ddr_dqs_en = 0x0F	DDR dqs enable = 0x0F, because this is a DDR module.
buff_dim = 0	Buffer DIMM = 0, because DIMM is not being used.
soft_ref = 0	Auto refresh = 0 on the first write to the Control register. The second time the Control register is written, this bit is set to do a software refresh.
soft_pre = 0	Precharge all = 0 on the first write to the Control register. The third time the Control register is written, this bit is set to do a software precharge.

Table 8-21   Mode Register = 0x008D 0000 and 0x4009 0000

Bit	Description
mode_code = 0x048D 0000	Operating mode—value comes from the memory modules specification, listed under operating mode. This value expresses explicitly the RESET DLL condition. Under normal operation mode, a value of 0x008D 0000 shall be used.
extended mode = 0x4009 0000	Extended Mode—MGT5100 DDR does not implement the QFC algorithm and the driving strength is set to “reduced”. DLL are enabled.

Table 8-22   Load Sequence for Registers

Bit	Description
Write Configuration 1 register	0x7362 2830—Load Configuration 1 Register.
Write Configuration 2 register	0x4577 0004—Load Configuration 2 Register.
Write Control register	0xF051 0F00—Load Control Register.
Write Control register	0xF051 0F02—Load Control Register again, does a software precharge all.
Write Extended Mode register	0x4009 0000—Enable DLL and set output strength to “reduced”.
Write Mode register	0x048d 0000—Load Mode Register to set burst length and latency. Reset DLL, must be done each time DLL is enabled.
Write Control register	0xF051 0F02—Load Control Register again, does a software precharge all.
Write Control register	0xF051 0F04—Load Control Register again, does a software refresh.
Write ADRSEL register	0x01—sets addressable memory to 64 MB.
Write Mode register	0x008d 0000—Load Mode Register to set burst length and latency. Set SDRAM memory device into Normal Operating condition.
Write Control register	0x7051 0F00—Load Control Register to remove SDRAM mode set enable.

SECTION 9
CS/LP BOOT ROM/SRAM CONTROLLER

9.1 Local Bus Overview

This section focuses on the memory mapped device interface. This interface is referred to as LocalPlus. The following sections are contained herein:

- LocalPlus Bus (LP bus)
  - LocalPlus Features
  - LocalPlus Operation
  - Chip Select Configuration Address Spaces
  - Reset Configuration
- LocalPlus Controller (LPC) Design
  - Standard non-MUXed Mode
  - ALE Driven MUXed Mode
  - Signals
  - Interface Description
  - Physical Peripheral Connections
- LPC Programmer's Model
  - Interrupt and Bus Errors
  - Chip Select/LocalPlus Bus Registers—MBAR + 0x0300

The MGT5100 has a multi-function external local bus. This bus supports connections to PCI and ATA compliant devices, as well as memory mapped devices such as:

- Flash
- ROM
- SRAM

The PCI compliant interface is described in Section 10.3.

The ATA compliant interface is described in Section 11.6.

The local bus has been designed to accommodate several different peripheral interfaces with a minimum number of pins. The 32-bit address/data bus is shared by the different interfaces. It's definition changes, depending on where a given access falls in the memory map and which peripheral protocol must be supported. Separate control signals are used for each interface to insure proper operation without conflict or contention.

The default local bus interface is PCI. In this case the 32-bit bus is a MUXed address/data bus consistent with a 32-bit PCI protocol. Separate and dedicated PCI control signals (FRAME, DEVSEL, IRDY, TRDY, etc.) support the PCI protocol.

Other internal local bus masters (ATA or LocalPlus) gain access to the bus by making a request to the PCI control logic. An on-chip PCI arbiter determines which master controls the local bus during a given access and subsequently the resulting pin definition.

ATA accesses share the 32-bit local data bus. Separate and dedicated ATA control signals (chip selects, write, read, etc.) are provided.

- When an internal access is decoded to fall in the ATA controller address space, a request is made to the PCI arbiter.
- When ATA access is granted, the PCI address/data bus function is transformed into 16 bits of ATA data and 3 bits of ATA address.

External accesses to LocalPlus devices (Flash, ROM, SRAM etc.) are supported in a similar fashion. These devices occupy a separate location in the memory map and have independent control signals (chip selects, RWB, $\overline{TS}$ etc.).

- When an internal access is decoded to fall in the LocalPlus memory space, a request is made to the PCI arbiter.
- When granted, the 32-bit PCI address/data bus is transformed into different combinations of address/data bits to support access to these devices. The address/data bits depend on the mode assigned to a given Chip Select.

The LocalPlus bus provides the interface to boot code devices such as ROM or Flash. Several options are available in terms of device size, speed and configuration. These options are described in the sections below.

9.2 LocalPlus Bus (LP bus)

9.2.1 LocalPlus Features

LocalPlus has the following features:

- Interface to memory mapped or chip selected devices
- Several modes of operation (MUXed or non-MUXed), which allows interfaces to a variety of device configurations
- 6 Chip Select (CS) signals
- Programmable wait states per CS
- Configurable Boot code interface supporting PowerPC architecture code execution
- Dynamic bus sizing on some interfaces
- Byte swapping supported per CS

9.2.2 LocalPlus Operation

The LocalPlus interface consists of:

- an address bus
- a data bus
- Chip Select signals (CS0-5)
- 4 control signals:
  - RWB
  - ALE
  - ACK
  - TS

There are 2 primary modes of operation:

- MUXed
- non-MUXed

Each CS can be programmed to a different mode of operation (MUXed, non-MUXed, number of wait states, byte swapping etc.).

NOTE: CS0 is special and provides the initial G2 processor boot code access.

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 Section 11, ATA Controller.

In non-MUXed mode Address and Data are presented on the 32-bit local bus simultaneously, in a linear fashion. Different widths of address/data are available.

MUXed mode lets larger devices 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 $\overline{\text{ALE}}$ during the address phase and a $\overline{\text{TS}}$ during a separate data phase. This mode requires external logic to latch the address during the address tenure. An $\overline{\text{ACK}}$ 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).

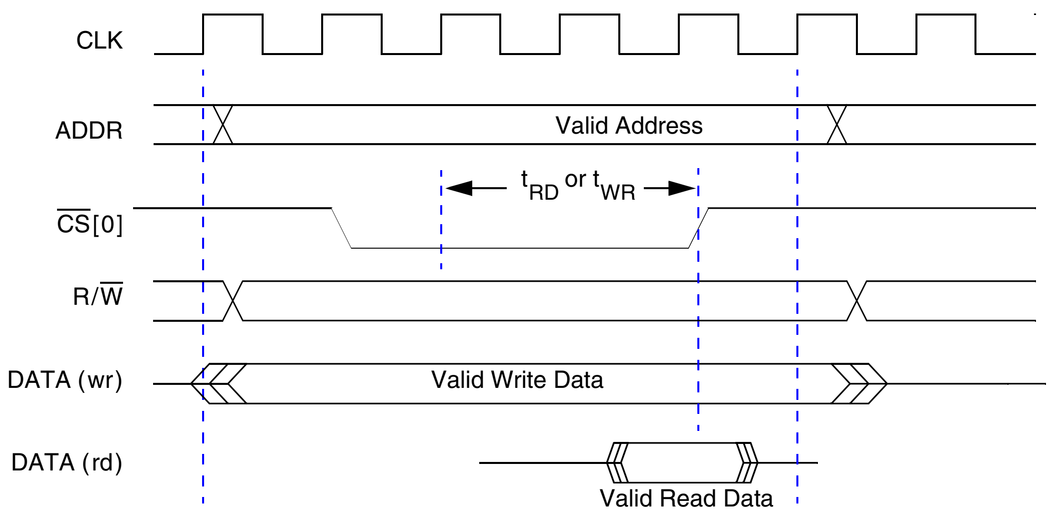
MUXed and non MUXed modes support a variety of device configurations and are configurable on a per CS basis.

9.2.2.1 Non-MUXed Mode

The non-MUXed mode allows the following configurations:

- 16-bit data and 16-bit address—128 KBytes accessible
- 8-bit data and 24-bit address—16 MBytes accessible

The above 2 options are selectable via the reset configuration word. Other configurations are possible via software configuration (e.g., 8-bit data and 16-bit address). Figure 9-1 shows the operation of Non-MUXed accesses.



NOTE:
1. t_{RD}/t_{WR} is wait states as programmed for corresponding access and chip select.
2. Read data has nominal setup/hold requirements around the $\overline{CS}$ negation.
3. Signals are driven with one-clock setup and hold outside of $\overline{CS}$ active.

Figure 9-1 Timing Diagram—Non-MUXed Access

9.2.2.2 MUXed Mode

The MUXed mode allows the following 3 configurations:

- 32-bit data, 25 bits of address, 2 bits of bank select—up to 512 MBytes accessible
- 16-bit data, 25 bits of address, 2 bits of bank select—up to 128 MBytes accessible
- 8-bit data, 25 bits of address, 2 bits of bank select—up to 64 MBytes accessible

Bank select bits are written in a register by the G2 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.

MUXed mode requires external logic to latch the address during the address tenure and to decode bank selects if they are encoded. Additional information and timing diagrams are provided in later sections.

9.2.2.3 Boot Configuration

After power-on reset (POR) the G2 processor accesses the local bus to fetch initial code sequences. Chip Select Boot ROM (CS0) is dedicated for this purpose. 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

- MUXed or non-MUXed mode.
- Data bus can be 16- or 32-bits wide in MUXed mode.
- Data bus can be 8- or 16-bits wide in non-MUXed mode.
- The number of wait states incured during boot code accesses can be 4 or 48 IP bus clock cycles.
- The boot address/exception table can be located at 0000 0100 (hex) or fff0 0100 (hex).

The PowerPC architecture compatible processor core requires 64-bit instruction fetches. During boot code accesses from CS0 space on-chip logic is provided to perform enough LocalPlus accesses to accumulate 64-bit instructions to be given to the G2 processor. For example, before passing the resulting 64-bit instruction to the G2 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:

– write transactions during boot (only instruction fetches are supported).
– an 8-bit wide MUXed mode configuration during boot, but only because there is no reset configuration option for this case.

After boot, CS0 space can be programmed to act as other MGT5100 Chip Select spaces (CS1-5). This capability is described in the sections below.

9.2.3 Chip Select Configuration Address Spaces

CS1-CS5 in MUXed mode:

- Supports 8-, 16- and 32-bit data reads and writes.
- Dynamic bus sizing is supported. 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 bits, which indicate the current transaction size.

- 64-bit access and burst access, are not supported. Internal logic is limited to 32-bits during non-boot accesses.

  Instruction accesses (i.e., read only) are supported on CS0 during boot, where this logic exists. This implies the G2 processor cannot execute code from the LP bus except during boot on CS0. If code exists in CSx spaces (other than CS0 at boot) the processor must transfer the code to SDRAM before execution can begin.

CS1-CS5 non-MUXed mode:

- In non-MUXed mode the data port size can be 8 or 16 bits.
- 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 bits are active only in MUXed mode).

9.2.4 Reset Configuration

The number of address/data bits and the ROM/SRAM controller timing at boot are controlled by bits in the RST_CONFIG word sampled during POR. The following 4 RST_CONFIG bits control boot device operation from reset:

- BootType
- BootSize
- BootWait
- BootSwap

Table 9-1 describes possible boot settings.

Table 9-1 BOOT_CONFIG (RST_CONFIG) Settings

Parameter	If Pulled Down (0)	If Pulled Up (1)	Notes
BootType	non-MUXed boot ROM accesses, referred to as single-tenure	MUXed boot ROM accesses, referred to as dual-tenure	BootSize is interpreted differently depending on BootType setting.
BootSize	For non-MUXed type: 8-bit boot ROM data 24-bit boot ROM address For MUXed type: 16-bit boot ROM data (25 bit address)	For non-MUXed type: 16-bit boot ROM data 16-bit boot ROM address For MUXed type: 32-bit ROM data (25 bit address)	For MUXed types address tenure always has capacity for 25 bits of address plus 2 programmable bank bits.
BootWait	Minimum Wait states 4 ipb_clk cycles	Maximum Wait states: 48 ipb_clk cycles	For BootType of 1 (MUXed), 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 = 2 Byte swap 32-bit access = 4 Byte swap

9.3 LocalPlus Controller (LPC) Design

The LocalPlus Controller has a very straight forward operation, but some special cases must be described. Internally, the LPC is driven by the IP Bus clock and looks like a slave module to the IPbus. The external Bus Clock (the one seen by peripherals) may be running at either:

- 1X the IP Bus clock
- 1/2X the IP Bus clock

Wait States are applied in IP Bus clocks regardless of this ratio. For the 1/2X case, there is no phase relationship guarantee between the internal clock and the external clock. A 180-degree phase shift may exist from one transaction to another. This generally means any programmed wait states must be increased by 1 to account for the possible half-period external clock slip.

Start/Stop registers to identify the CS address range for each CS output are contained in the MGT5100 MMAP register group. Registers in the LPC are accessed through the address range specified in the MGT5100 Internal Register Map. For more information, see Section 9.4, LPC Programmer's Model. These registers control the operation and description of a particular CS and peripheral, but only when a "hit" occurs in the MMAP module for a particular CS space.

Two major modes of operation are supported:

- Standard non-MUXed mode
- ALE driven MUXed mode

Within each mode, there is considerable flexibility to control the operation.

9.3.1 Standard non-MUXed Mode

In this mode, the peripheral address and data lines are limited to a total of 32. 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. No additional signals are available. If external signals are needed, external glue logic must be added.

Wait states are programmable and simply select how many IP Bus 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. Static data widths from 1 to 2 bytes are possible.

NOTE: 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.

9.3.2 ALE Driven MUXed Mode

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_b signal is asserted (active lo) during this address tenure. ALE width is programmable to be one or two IP Bus clocks (i.e., short or long ALE). The dedicated RWbar 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.

NOTE: One "dead cycle" exists between the address tenure and data tenure. This allows hold time for transparent latches, if used, in the external glue logic.

In ALE driven MUXed mode, Local Plus can access up to 512 MBytes of data:

- 25 address bits give 32 MBytes of address range
- plus 2 bank selects gives a total of 128 MBytes of address range.

  The data word is 32-bits wide giving a total of 512 MBytes of data.

9.3.3 Signals

The LPC consists of 3 major connection groups:

1. An internal IP Bus interface
2. Internal extensions to support external access (i.e., full address and "hit" signals)
3. The external I/O for connection to peripherals

The external I/O bus is shared with the PCI AD bus and requires arbitration for access to the external bus.

Table 9-2   LocalPlus External Signals

Signal	I/O	Definition
$\overline{\text{CS}}$ [5:0]	O	Chip Selects (active low), $\overline{\text{CS}}$[4] and $\overline{\text{CS}}$[5] used by ATA, if configured as such.
R/$\overline{\text{W}}$	O	Read/Write dedicated output.
AD_out [31:0]	I/O	AD address/data (bi-directional when used as data)
$\overline{\text{ACK}}$	I	External Acknowledge input
$\overline{\text{TS}}$	O	Dedicated Transfer Start output (multiplexed transactions only)

9.3.4 Interface Description

Figure 9-2 shows the LPC concept.

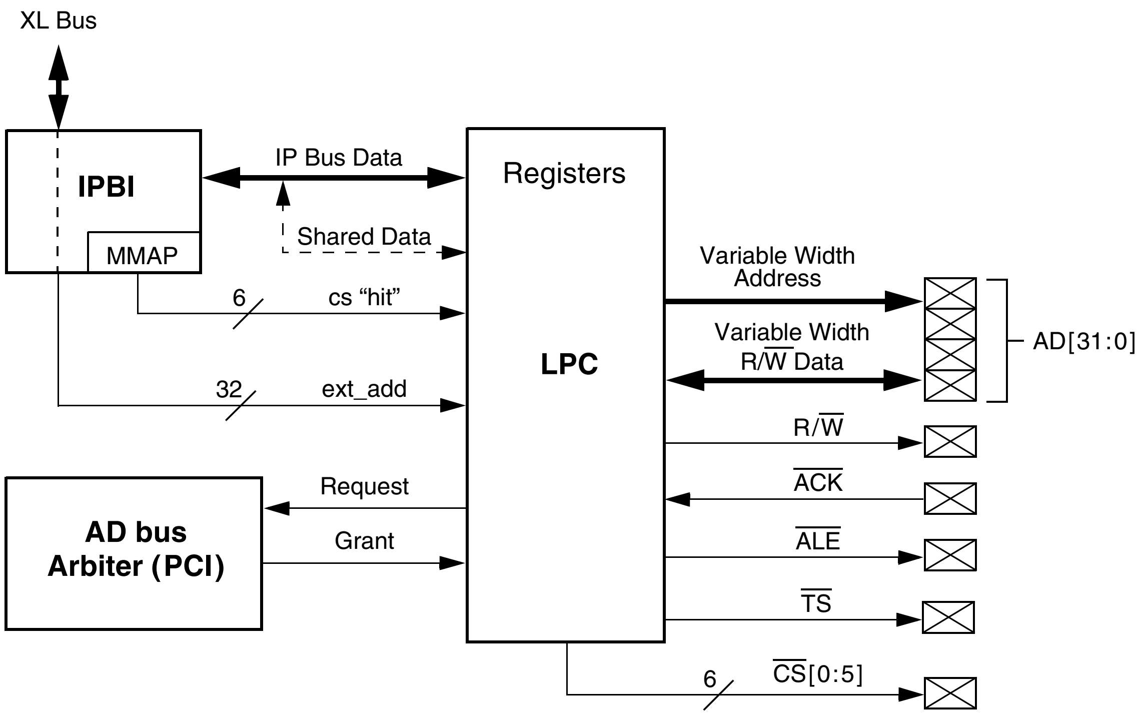


Figure 9-2   LPC Concept Diagram

Figure 9-3 shows ALE transactions.

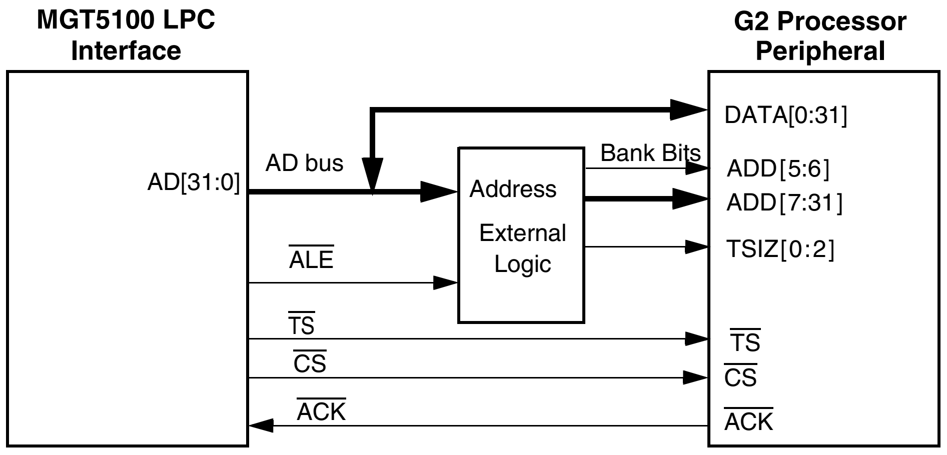


Figure 9-3 Using ALE Transactions

9.3.5 Physical Peripheral Connections

For non-MUXed peripheral connections (i.e., single tenure, address and data presented simultaneously on the AD[31:0] bus), the address inputs of a connected peripheral should connect to the AD bus bits lsb to lsb (AD[0] is the lsb).

For Data connections the peripheral data msb must connect to AD[31], which is the AD msb. Simply put, this means:

- the Address is oriented to the bottom of the AD bus
- the Data is oriented to the top of the AD bus

For MUXed transaction types the MGT5100 presents the address oriented to AD[0] (just as in non-MUXed) during the address tenure. During data tenure, data is presented in orientation to AD[0]. In this transaction, both Data and Address are orientated to the bottom of the AD bus.

The following tenure descriptions relate to MUXed mode transactions only.

9.3.5.1 During 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 TSIZE 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 TSIZEs of 1, 2, or 4 are supported.

TSIZE[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 occur.

The ALE signal is active low and remains asserted for either 1 or 2 system clocks. This depends on the programmed ALE width bit in the LPC control register.

AD[31] & AD[27] are unused and are driven low by the LPC during the address tenure.

9.3.5.2 During Data Tenure

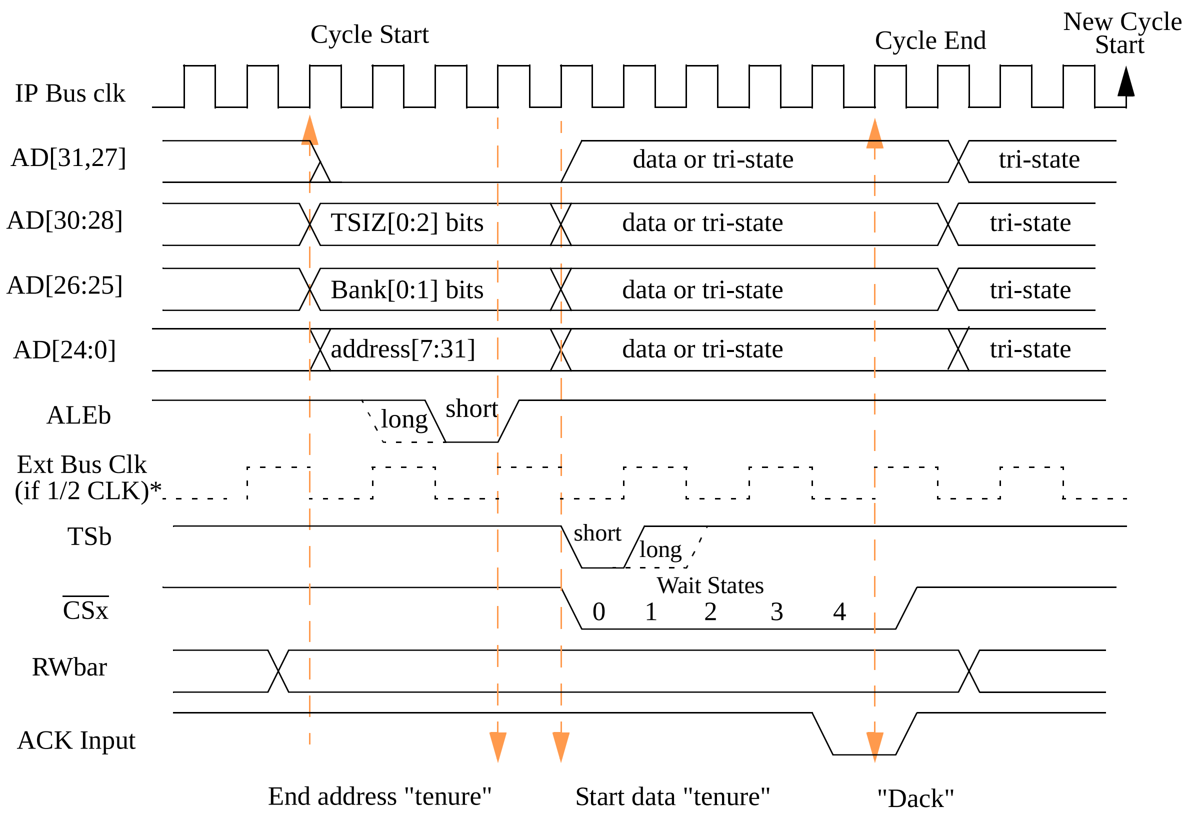
During Data Tenure, the following occurs:

- 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.

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 clock edge where the ACK input is detected as asserted, the LPC terminates the transaction and releases the bus on the **next** IP 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 9-4 shows a MUXed transaction type timing diagram.



NOTE:

1. During data tenure, the decision between data being driven or the AD bit being tri-stated is dependent on whether the transaction is a Read or a Write and what the programmed Data Size is. Unused bits in the data tenure (i.e., those in excess of the programmed data size) are driven to 0 by the LPC as a precaution to avoid floating bus condition.
2. The cycle terminates without an ACK, if the internal wait state condition expires. In either case, the data and control signals are maintained one clock cycle beyond CSx negation to assure hold time.
3. Use of ACK for termination is software programmable.
4. Ext Bus Clk (If 1/2 internal CLK) may occur 180 degrees phase shifted for any given transaction.

Figure 9-4 Timing Diagram—MUXed Transaction Type

9.4 LPC Programmer’s Model

Table 9-3 through Table 9-6 describe in detail the registers and bit meanings for configuring CS operation. There are six 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. Registers 8-31 are unimplemented.

NOTE: The address range registers for each CS reside in the MMAP register set rather than in the LPC register set.

9.4.1 Interrupt and Bus Errors

Although LPC has an implemented Interrupt Enable bit, there is **no** actual core Interrupt signal connection. An interrupt occurs when a specific peripheral in the associated configuration register is marked as read-only or write-only and is accessed in violation of this setting. It is important to distinguish “internal” versus “external” access when describing Status bits and error enables.

- Internal Access—always refers to the IP bus and/or XL buses in MGT5100. The LPC register set (and access to it) are considered internal transactions. Bus errors can occur during internal access (i.e., attempting to read or write an unimplemented register).
- External Access—always refers to transactions to external peripherals and these transactions always occur on the external AD bus.

Separate enables are provided for external and internal bus errors. Any enabled bus error is reflected as the assertion of ips_xfr_err on the IP bus. Then, by assertion of TEA on the XL bus. In general, enabled bus errors create a machine check exception at the core. The only way to alert the core to an LPC bus error is to enable the error to create a machine check and put the exception code in the machine check service routine.

To provide for software polling, status bits are always active, regardless of the enable bits.

9.4.2 Chip Select/LocalPlus Bus Registers—MBAR + 0x0300

There are 8 32-bit Chip Select/LocalPlus (CS/LP) bus 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:

- CS Boot ROM, (0300)
- CS Configuration (0304–0314)
- CS Control (0318)
- CS Status (031C)

9.4.2.1 CS Boot ROM, (0300)

Table 9-3 CS Boot ROM, (0300)

	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	AL	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 WaitX or used by itself, as specified by WTyp bits below. Wait states control how many IP Bus 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/2 equals number of external bus clocks from CS assertion to when data must be valid from boot device (this is because the default ratio of IP Bus to PCI Bus clock is 2 to 1 from reset).
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 **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	AL	ALE length—multiplexed transactions only 0=$\overline{\text{ALE}}$ width is 1 internal clock 1=$\overline{\text{ALE}}$ width is 2 internal clocks At boot time, internal clock is twice the frequency of external bus clock. Therefore, AL defaults to 1 (2 clocks) for boot device.
18	AA	ACK Active—multiplexed transactions only. This bit defines whether $\overline{\text{ALE}}$ 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. **cfg operation**—If rstcfg[14] on pad_eth_06 is high, indicating multiplexed mode boot device, then AA is assumed high as well. This lets the boot device shorten the Wait State period by asserting the ACK input.
19	CE	An individual Enable bit—allows CS operation for the corresponding CS pin. CE must be high to allow operation. Register 6 master enable bit must also be high, except when $\overline{\text{CS}}$[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=1Byte 01 =2Bytes 10 =3Bytes (**Not Supported**) 11=4Bytes (Not valid in non-MUXed case) **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[26:265]) during Address tenure of a multiplexed transaction. 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.

Bits	Name	Description
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.
30	WO	Write Only bit—If bit is high, the peripheral is treated as a write-only device. An attempted read access results in a bus error (as dictated by Chip Select Contro Register EBEE bit) and/or an interrupt (as dictated by Chip Select Control Register IE bit). In any case, no transaction is presented to the peripheral. A bus error means the internal cycle is terminated with a transfer error acknowledge (ips_xfr_err assertion to IP bus, TEA assertion to XL bus).
31	RO	Read Only bit—If bit is high, the peripheral is treated as a read-only device. An attempted write access results in a bus error (as specified by Chip Select Control Register EBEE bit) and/or an interrupt (as specified by Chip Select Control Register IE bit). In any case, no transaction is presented to the peripheral. **NOTE:** This bit is high from Reset, indicating Boot Device is Read-Only.

9.4.2.2 CS Configuration (0304–0314)

Table 9-4 CS Configuration (0304–0314)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WaitP								WaitX							
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	MX	AL	AA	CE	AS		DS		Bank		WTyp		WS	RS	WO	RO
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 System 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.
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

Bits	Name	Description
17	AL	$\overline{\text{ALE}}$ Length (multiplexed transactions only) 0=$\overline{\text{ALE}}$ width is 1 internal clock 1=$\overline{\text{ALE}}$ width is 2 internal clocks At boot time, internal clock is twice the frequency of external bus clock. Therefore, AL defaults to 1 (2 clocks) for boot device.
18	AA	ACK Active—multiplexed transactions only. This bit defines whether $\overline{\text{ALE}}$ 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.
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, except when $\overline{\text{CS}}$[0] is used for boot ROM. 1=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 (**Not Supported**) 11 = 25 bits (Not valid for non-MUXed, don't care in MUXed case) **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=1Byte 01 =2Bytes 10 =3Bytes (**Not Supported**) 11=4Bytes (Not valid in non-MUXed case)
24:25	Bank	Bank bits—are reflected on external AD lines (AD[26:25]) during address tenure of a multiplexed transaction. 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 results in a bus error (as specified by Chip Select Control Register EBEE bit) and/or an interrupt (as dictated by Chip Select Control Register IE bit). In any case, no transaction is presented to the peripheral. A bus error means the internal cycle is terminated with a transfer error acknowledge (ips_xfr_err assertion to IP bus, TEA assertion to XL bus).
31	RO	Read Only bit—If high, peripheral is treated as a read-only device. An attempted Write access results in a bus error (as specified by Chip Select Control Register EBEE bit) and/or an interrupt (as dictated by Chip Select Control Register IE bit). In any case, no transaction is presented to the peripheral.

9.4.2.3 CS Control (0318)

Table 9-5 CS Control (0318)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	EBEE			IBEE	IE			ME								
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																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	EBEE	External Bus Error Enable bit—Setting bit to 1 causes an external transaction that was denied because of an illegal access (i.e., Read-only or Write-only bits are set), to terminate with a Bus Error termination. This causes the IP bus to receive ips_xfr_err and the XL bus to receive transfer error acknowledge (TEA). If bit is 0, transaction is still denied, but results in a positive acknowledge on the internal busses. Regardless of this bit setting, a status bit (ROerr or WOerr) is set in LPC Status Register.
1:2	—	Reserved
3	IBEE	Internal Bus Error Enable bit—Setting this bit causes an illegal access to the LPC register set to generate an internal bus error condition. Otherwise, all access to the LPC register set is positively acknowledged. In either case, no register or bit is corrupted. Bus error status bits are always set in the status register, regardless of the IBEE setting.
4	IE	Interrupt Enable bit—This bit is Reserved and does not affect actual operation. BE AWARE—The Interrupt to the CPU is NOT connected so no actual Interrupt can be generated to the core. Software can detect bus errors by setting the appropriate BEE bit above and putting service code in the machine check ISR to query the LPC Status Register.
5:6	—	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

9.4.2.4 CS Status (031C)

Table 9-6 CS Status (031C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved		WOerr	ROerr	Rsvd	CSxerr			Reserved						B2	B3
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

Bits	Name	Description
0:1	—	Reserved

Bits	Name	Description
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. This status bit is always active regardless of bus error enable bit. 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. This status bit is always active regardless of bus error enable bit. 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.
8:13	—	Reserved
14	B2	Bus Error (type 2)—indicates an unimplemented LPC register was accessed by an internal write cycle. B2 is a sticky bit and must be written with 1 to be cleared. **NOTE:** No register or bit is corrupted, but it may indicate damaged software.
15	B3	Bus Error (type 3)—indicates an unimplemented LPC register was accessed by a internal read cycle. B3 is a sticky bit and must be written with 1 to be cleared. **NOTE:** No register or bit is corrupted, but it may indicate damaged software.
16:31	—	Reserved

SECTION 10 PCI CONTROLLER

10.1 Overview

The following sections are contained in this document:

- PCI External Signals
- PCI Interface, includes:
  - PCI XLB Configuration Registers—MBAR + 0x0D00
  - MGT5100 Application Interface Registers—MBAR + 0x0D00
  - PCI Transmit (Tx) Registers—MBAR + 0x3800
  - PCI Receive (Rx) Registers—MBAR + 0x3880

The Peripheral Component Interface (PCI) Controller is a high-performance bus, especially suitable for high data-rate applications like digital audio and video. The MGT5100 supports a PCI initiator interface and a limited target interface. See Figure 10-1.

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 PCI bus clock is always sourced from the MGT5100. An external output is available to drive the PCI bus clock. However, no input is available to let the MGT5100 be driven by an external source. Even in target mode, MGT5100 continues to drive the PCI clock for the system. This clock is selectable as:

- Same as the IP bus clock, or
- One-half the IP bus clock

Buffering and operation are designed to allow direct transactions to and from PCI-compliant and ATA-compliant disk drives. Full-duplex operation at both destinations allow data movement from the disk even while PCI-to-disk operations are in progress. In fact, data movement can be simultaneously occurring in both directions with data rates up to 10x MBytes/second. Transactions are time-separated by the internal PCI bus arbiter. If software time-separates block moves of data, the PCI data rate can be increased to approximately 40 MBytes/second (at a 27 MHz PCI clock).

The DWPCI Controller provides both target and initiator operation. MGT5100 is expected to operate mainly as an initiator. The DWPCI interface is optimized for initiator operation.

- As a target, limited but usable access to the internal bus (XLB) is supported.
- As an initiator, the PCI Controller is coupled directly to XLB (as a slave) and is also available on CommBus as a SmartDMA peripheral.

The DWPCI Controller provides for automatic retry of target disconnected transactions as well as latency time-outs. Hardware modules are placed in front of the DWPCI Controller to ease the software burden on MGT5100 application writers. These modules also do the necessary bus translation to connect the PCI to XLB and CommBus.

10.2 PCI External Signals

Table 10-1 PCI External Signals

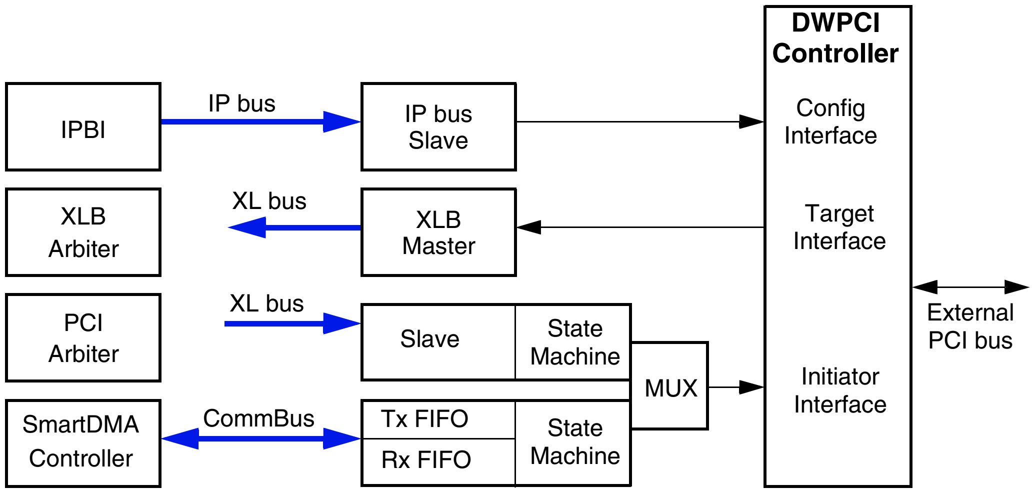
Signal	I/O	Definition
AD	I/O	Multiplexed Address and Data Bus (Shared with ATA and LPC)
PAR	I/O	Parity
C/BE	I	Command/Byte Enable
IRDY	I	Initiator Ready
RST	I	Reset
CLOCK	I	Clock
FRAME	I	Frame Start
PERR	I	Parity Error
GNT	I	Bus Grant
LOCK	I	Bus Lock

10.3 PCI Interface

The following sub-sections are provided:

- PCI XLB Initiator Interface
- PCI XLB Target Interface
- PCI XLB Configuration Interface (includes registers)
- PCI SmartDMA Initiator Interface (includes registers)

The internal interface for PCI transactions is dictated by the DWPCI application interfaces. Figure 10-1 provides an operational view of these interfaces. In each case, a hardware module (or "gasket") converts the DWPCI interface into a standard bus interface with various limitations and capabilities.



NOTE:
1. Configuration transactions are single-beat 32 bits on the IP bus.
2. Target transactions are single data-beat 64 bits on the XLB master.
3. Initiator transactions support XLB burst transactions.
4. Full DMA support is provided by the CommBus SmartDMA Subsystem.

Figure 10-1 Block Diagram—PCI Interface

Blocks labelled as *state machine*, indicate custom functions that were added to enhance DWPCI functionality (i.e., autonomous packet transactions). Status and control registers are included to augment the standard PCI status bits in DWPCI. The PCI arbiter is custom designed and internal to MGT5100. One Request/Grant pair is provided for MGT5100 arbitration of a single external PCI Master.

NOTE: The MGT5100 PCI Arbiter does not implement bus parking or any specific algorithm scheme. Each PCI transaction requires re-arbitration.

As shown in Figure 10-1, the DWPCI has 3 separate interfaces for PCI transactions. The configuration interface provides for the read and write of status and control registers associated with PCI configuration. If the MGT5100 is the configuring master (as expected), this interface is used to configure the DWPCI.

An external master would configure the DWPCI through the external PCI bus. The DWPCI, and thus MGT5100, can operate either as a PCI initiator or a PCI target.

- As a target, external requests are translated into single-beat 64-bit transactions on the XLB.
- As an initiator, the expected operation, 2 connection modes are supported:
  - XLB mode
  - CommBus mode

XLB Mode—Direct G2 core transactions (or any XLB master) are available on the XLB. Single-beat 64-bit data transactions are immediately transferred to the PCI bus. Each XLB data beat results in 2 32-bit data beats on the PCI bus (one beat may appear as a Null

Beat). No buffering is done and the XLB transaction stalls until the PCI transaction completes. Standard burst and critical double-word burst transactions are supported. However, each data beat may be stalled while the PCI transaction completes. Stall-time includes the PCI bus arbitration time.

NOTE:

1. XLB transactions are limited to 1-, 2-, 4-, or 8-Byte transactions that must be properly aligned to the indicated address boundary.
2. CDWF burst transactions rely on the cache-line features in DWPCI and may be affected by an errata involving the least significant bits of the PCI address. Section 10.3.1 gives more information.

CommBus Mode—The second connection mode is through the CommBus and SmartDMA Controller. This interface provides for high-speed, autonomous DMA transactions with the DWPCI operating as a standard SmartDMA peripheral. Full-duplex operation is supported and direct XLB transactions can also be interleaved while CommBus transactions are in progress. PCI arbitration occurs continuously to support transaction interleaving.

NOTE: DMA operation operates independent of the XLB. Non-PCI XLB transactions have 100% bandwidth available during PCI DMA activities, except when SmartDMA is bursting data on XLB (i.e., to SDRAM).

Four (4) major interfaces are described in the following sections:

1. PCI XLB Initiator Interface (Section 10.3.1)—provides direct processor access to the PCI bus (i.e., XLB mode). This interface includes configurable address translation suitable to implement PREP and CHRP protocols.
2. PCI XLB Target Interface (Section 10.3.2)—supports MGT5100-as-Target PCI operations.
3. PCI XLB Configuration Interface (Section 10.3.3)—includes the usual PCI Configuration registers (Type 0 Header), followed by registers to control the PCI XLB Initiator Interface and PCI XLB Target Interface.
4. PCI SmartDMA Initiator Interface (Section 10.3.4)—has 2 sections:
   - PCI SmartDMA Transmit (Tx) Initiator Interface, Section 10.3.4.1 (PCI write)
   - PCI SmartDMA Receive (Rx) Initiator Interface, Section 10.3.4.3 (PCI read). For example, CommBus mode.

   The PCI SmartDMA Initiator Interface provides for autonomous block transfers across the PCI bus. Sequential data organization from some starting addresses is assumed for each data "packet". Up to 65KBytes of data can be transferred without CPU intervention; SmartDMA does the internal data transfer. This interface is controlled by its own register set, with some assistance from the PCI XLB Configuration interface.

10.3.1 PCI XLB Initiator Interface

Certain control and status information (e.g., PCI command word) is written to or read from an IP bus module associated with this interface (see Section 10.3.3). Once this module is configured, XLB transactions are automatically converted to PCI transactions.

This interface provides for configurable "Windows" into the PCI space. Two Main Windows are configurable and two subwindows can be defined within the Main Windows.

Map Decoder logic is provided and configurable to support PREP and CHRP protocols. This address translation logic is generic and requires significant software knowledge to actually implement the named protocols. See Section 10.3.3 for related register descriptions.

The PCI XLB Initiator interface supports all aligned XLB transactions, including 64-bit and normal bursts (i.e., 32-Byte aligned bursts). CDWF burst operation (i.e., cache line burst) is not directly supported, except the PCI protocol has a configurable cache line operation.

10.3.2 PCI XLB Target Interface

This interface supports MGT5100-as-Target operation. The DWPCI configuration space is accessible by an external PCI master (as in any PCI Target). Two BAR registers are provided to identify two Memory spaces in MGT5100.

- For SDRAM, there is a fixed 128MByte BAR.
- For MGT5100 internal registers, there is a 32KByte BAR.

Be Aware—True SDRAM space may be smaller, or larger, than 128MByte. The system programmer must be aware of this, as the PCI configuration space always shows a full 128MBytes.

An externally initiated transaction is mastered onto the XLB with no distinction from any other possible XLB master. The transaction on XLB shows as a 32-bit single-beat transaction.

Latencies on the XLB may result in a 16/8 clock rule violation. If this occurs, the DWPCI component issues a Target Disconnect and handles the expected retry. There is no accommodation made for these possible latencies. Certain situations could result in infinite retries. If this condition occurs, software can override the 16/8 clock rule by programming the "latrule disable" bit high. In this case, DWPCI does not disconnect and proceeds as if no rule violation occurred.

10.3.3 PCI XLB Configuration Interface

This interface provides an IP bus interface to and from DWPCI configuration registers. In addition to containing the standard PCI configuration header space, more register space is provided in this module. This space contains the control and status registers for the PCI XLB Initiator interface, and the PCI XLB Target interface. Additionally, there are some global controls that may affect the PCI SmartDMA interfaces (e.g., Max Retries).

10.3.3.1 PCI XLB Configuration Registers—MBAR + 0x0D00

PCI XLB Configuration is controlled by 17 32-bit registers. These registers are located at an offset from MBAR of 0x0D00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0D00 + register address**

Registers in this section show 2 types of labels:

- **PCI Header**—These registers are physically contained in the DWCPI component and represent a Standard Type 0 Header space as defined by the PCI specification.
- **PCI**—These are MGT5100 registers associated with PCI status/control and are custom to the MGT5100.

A register overview is provided in Section 3.4.1, PCI XLB Configuration Registers.

Hyperlinks to the PCI XLB configuration registers are provided below:

- PCI Header: Device ID/Vendor ID (0D00)
- PCI Header: Status/Command (0D04)
- PCI Header: Class Code/Revision (0D08)
- PCI Header: BIST/Type/Latency/Cache (0D0C)
- PCI Header: BAR0 (0D10)
- PCI Header: BAR1 (0D14)
- PCI Header: Reserved (0D18–0D27)
- PCI Header: CardBus CIS Pointer (0D28)
- PCI Header: Subsystem Vendor ID (0D2C)
- PCI Header: Unused (0D30)
- PCI Header: Unused (0D34–0D3B)
- PCI Header: Max Lat/Min Grant/Interrupt (0D3C)
- PCI Interrupt Enable (0D60)
- PCI Status (0D64)
- PCI Control (0D68)
- PCI Mask/Value Read (0D6C)
- PCI Mask/Value Write (0D70)
- PCI Subwindow 1 (0D74)
- PCI Subwindow 2 (0D78)
- PCI Window Command/Control (0D7C)

10.3.3.1.1 PCI Header: Device ID/Vendor ID (0D00)

Table 10-2 PCI Header: Device ID/Vendor ID (0D00)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Device ID															
W																
RESET:	0x5801															

	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 MGT5100 Value is: 0x5801.
16:31	Vendor ID	This field is read-only and represents the PCI Vendor Id assigned to MGT5100 Value is: 0x1057.

10.3.3.1.2 PCI Header: Status/Command (0D04)

Several bits in this and other registers are read-write-clear (**rwc**). If a status bit is set by hardware, software can clear the bit by writing 1 to the corresponding bit position. This is similar to the Port Power Control (PPC) sticky-bit method. Such bits are marked in the write section of the register description as "**rwc**".

Table 10-3 PCI Header: Status/Command (0D04)

	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	01		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	Name	Description
0	PE	Parity Error Detected—This bit is set when a parity error is detected, even the Parity Error Response bit in the Command Register (bit 6) is disabled. This register is **rwc** via PCI configuration cycles.
1	SE	System Error Signalled—This bit is set whenever MGT5100 generates a PCI System Error on the $\overline{\text{SERR}}$ line. This register is **rwc** via PCI configuration cycles.
2	MA	Master Abort Received—is set when MGT5100 is the PCI master and terminates a transaction (except for Special Cycle) with a Master-Abort. This register is **rwc** via PCI configuration cycles.
3	TR	Target Abort Received—is set when MGT5100 is the PCI master and a transaction is terminated by a Target Abort from the currently-addressed target. This register is **rwc** via PCI configuration cycles.
4	TS	Target Abort Signalled—is set when MGT5100 is the PCI target and it terminates a transaction with a Target Abort. This register is **rwc** via PCI configuration cycles.

Bits	Name	Description
5:6	DT	DEVSEL# Timing—Hardwired 01. These bits encode a medium $\overline{\text{DEVSEL}}$ timing. This defines the slowest $\overline{\text{DEVSEL}}$ timing when MGT5100 is the PCI target (except configuration accesses).
7	DP	Master Data Parity Error—applies only when MGT5100 is the PCI master and is set, and only if the following conditions are met: MGT5100-as-master sets $\overline{\text{PERR}}$ during a read, or detected it as asserted by the target during a write. The Parity Error Response bit in the Command Register, is set to 1. This register is **rwc** via PCI configuration cycles.
8	FC	Fast Back-to-Back Capable—Hardwired 0. This read-only bit indicates MGT5100, as target, is **not** capable of accepting fast back-to-back transactions with other targets.
9	R	Reserved—Hardwired 0. Prior to the 2.2 PCI Specification, this was the User Defined Features (UDF) Supported bit. 1 = Supported User Defined Features 0 = Does not support UDF
10	66M	66MHz Capable—Hardwired 1. Indicates the PCI Controller is 66MHz capable. **NOTE:** Other system implementation concerns may currently prevent full 66MHz operation. This item is TBD.
11	C	Capabilities List—Hardwired 0. Indicates the PCI Controller does NOT implement the New Capabilities List Pointer Configuration Register in DWORD 13 of the Configuration Space.
22	F	Fast Back-to-Back Transfer Enable—controls whether MGT5100 as master can do fast back-to-back transactions to different devices. If all targets are fast back-to-back capable, Initialization software should set this bit. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles. Value1 indicates master can generate fast back-to-back transactions to different devices. Value 0 indicates fast back-to-back transactions are allowed only to same device.
23	S	SERR# enable—is an enable bit for the $\overline{\text{SERR}}$ driver. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles. Value 0 disables the SERR# driver. Value 1 enables the $\overline{\text{SERR}}$ driver. Address parity errors are reported only if this bit and bit 6 are 1. **NOTE:** MGT5100 is unable to assert SERR as an Initiator.
24	St	Address and Data Stepping—Hardwired 0. Indicates PCI Controller never uses address/data stepping. Initialization software should write 0 to this bit location.
25	PER	Parity Error Response—controls device response to parity errors. PER is programmable; read/write from both the IP bus and PCI bus Configuration cycles. When set, and a parity error is detected, PCI Controller asserts $\overline{\text{PERR}}$. When bit is 0, if a parity error occurs, device sets its Detected Parity Error status bit (bit 24), but does not assert $\overline{\text{PERR}}$.
26	V	VGA Palette Snoop Enable—Hardwired 0. Indicates PCI Controller is not VGA compatible. Initialization software should write 0 to this bit location.

Bits	Name	Description
27	MW	Memory Write and Invalidate Enable—bit enables using the Memory Write and Invalidate command. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles. When bit is 1, MGT5100-as-master may generate the command. When bit is 0, Memory Write must be used.
28	Sp	Special Cycle Monitor or Ignore—determines whether to ignore PCI Special Cycles. Since MGT5100-as-target does not recognize messages delivered via the Special Cycle operation, value 1 should never be programmed to this register. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles.
29	B	Bus Master Enable—bit indicates whether MGT5100 has the ability to serve as a PCI bus master. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles. 1 indicates ability is enabled. 0 indicates bit does not disable mastered transactions. It is meant to be read by configuration software If MGT5100 is used as a PCI bus master (via XLB or CommBus), 1 should be written to this bit during initialization. If register value is 0, this bit does not disable mastered transactions. It is meant to be read by configuration software.
30	M	Memory Access Control—bit controls PCI Controller response to Memory Space accesses. This bit is programmable; read/write from both the IP bus and PCI bus Configuration cycles. 0 disables the response. 1 lets controller recognize a Memory access.
31	IO	IO access Control—Hardwired 0. Bit is not implemented because no MGT5100 IO type space is accessible from the PCI bus. PCI base address registers are Memory address ranges only. Initialization software should write 0 to this bit location.

10.3.3.1.3 PCI Header: Class Code/Revision (0D08)

Table 10-4 PCI Header: Class Code/Revision (0D08)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Class Code															
W	▨															
RESET:	0x0B20															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Class Code								Revision ID							
W	▨								▨							
RESET:	0x00								0x00							

Bits	Name	Description
0:23	Class Code	Field is read-only and represents the PCI Class Code assigned to MGT5100. Value is: 0x0B2000.
24:31	Revision ID	Field is read-only and represents the PCI Revision ID for this MGT5100 version. Value is: 0x00.

10.3.3.1.4 PCI Header: BIST/Type/Latency/Cache (0D0D)

Table 10-5 PCI Header: BIST/Type/Latency/Cache (0D0C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BIST								Header Type							
W																
RESET:	0x00								0x00							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Lat Timer[0:4]					Lat Timer[5:7]			Reserved				Cache Line Size			
W																
RESET:	0x00					0	0	0	0x00							

Bits	Name	Description
0:7	BIST	Built-In Self Test—Hardwired 0x00. PCI Controller does not implement the BIST register. Initialization software should write 0x00 to this register location.
8:15	Header Type	Hardwired 0x00. PCI Controller implements a Type 0 PCI Configuration Space Header. Initialization software should write 0x00 to this register location.
16:23	Lat Timer [0:4] Lat Timer [5:7]	Latency Timer—This register contains the latency timer value, in PCI clocks, used when MGT5100 is the PCI master. The lower 3 bits of the register are hard-wired low. The upper 5 bits are programmable (read/write from both the IP bus and PCI bus Configuration cycles). This timer must be programmed to a non-zero value before DWPCI will operate.
24:31	Cache Line Size	These 4 bits are programmable (read/write from both the IP bus and PCI bus Configuration cycles). The value programmed specifies the cache-line size in units of PCI DWORDs.

10.3.3.1.5 PCI Header: BAR0 (0D10)

Table 10-6 PCI Header: BAR0 (0D10)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BAR0															
W																
RESET:	0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	BAR0	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
0:16	BAR0	This field represents the BAR register programmable portion, which points to MGT5100 Internal Registers. BAR0 is 17 bits wide, representing a BAR space of 32KBytes.
28	pref	prefetchable access—Hardwired 0. Indicates memory space defined by BAR0 is not prefetchable. This space is considered Memory-Mapped I/O.
29:30	range	Hardwired 00. Indicates 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/M#	IO or Memory Space—Hardwired 0. Indicates BAR0 is used for memory space. Configuration software should write 0 to this bit location. 0 = Memory 1 = I/O

10.3.3.1.6 PCI Header: BAR1 (0D14)

Table 10-7 PCI Header: BAR1 (0D14)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BAR1					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	00		0

Bits	Name	Description
0:4	BAR1	Field represents the BAR register programmable portion that points to MGT5100 external SDRAM. BAR1 is 5 bits wide, representing a BAR space of 128MBytes. **NOTE:** Smaller (or larger) SDRAM may be present, but this register always reflects 128MBytes of space.

Bits	Name	Description
5:27	—	Reserved
28	pref	Prefetchable access—Hardwired 0. Indicates memory space defined by BAR1 is not prefetchable.
29:30	range	Hardwired 00. Indicates 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/M#	IO or Memory Space—Hardwired 0. Indicates BAR1 is for memory space. Configuration software should write 0 to this bit location. 0 = Memory 1 = I/O

10.3.3.1.7 PCI Header: Reserved (0D18–0D27)

These registers (0D18–0D27) are reserved and always read 0.

10.3.3.1.8 PCI Header: CardBus CIS Pointer (0D28)

This register (0D28) contains the Card Information Structure (CIS) pointer for the CardBus card. All 32bits of register are programmable by the IP bus. From the PCI bus, it is read-only. Reset value is 0x00000000.

10.3.3.1.9 PCI Header: Subsystem Vendor ID (0D2C)

This register (0D2C) 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 0 value in these registers indicates no Subsystem Vendor and Subsystem ID is associated with the device. If used, software must write to these registers before a PCI bus master reads them.

All 32bits of the register are programmable by the IP bus. From the PCI bus, they are read-only. Reset value is 0x00000000.

10.3.3.1.10 PCI Header: Unused (0D30)

This register (0D30) is unused and always reads 0. MGT5100 does not use Expansion ROM.

10.3.3.1.11 PCI Header: Unused (0D34–0D3B)

These registers (0D34–0D3B) are unused and always read 0.

10.3.3.1.12 PCI Header: Max Lat, Min Grant, Interrupt (0D3C)

Table 10-8 PCI Header: Max Lat/Min Grant/Interrupt (0D3C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Max_Lat								Min_Grant							
W																
RESET:	0x00								0x00							

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31 lsb
R	Interrupt Pin								Interrupt Line							
W	▨	▨	▨	▨	▨	▨	▨	▨								
RESET:	0x01								0x00							

Bits	Name	Description
0:7	Max_Lat	Maximum Latency—Specifies how often, in units of 1/4 microseconds, the PCI Controller would like to have access to the PCI bus as master. Value 0 indicates device has no stringent requirement in this area. The register is read/write from the IP bus, but read-only from the PCI bus. **NOTE:** This register value determines the priority level the bus arbiter assigns. The PCI Controller contains the PCI bus arbiter and this register is not used; no external arbitration is supported. Software need not write or read this register.
8:15	Min_Gnt	Minimum Grant—Value programmed to this register indicates how long the PCI Controller (as master) wants to retain PCI bus ownership when it initiates a transaction. As with Max_Lat, this register is not used by the arbiter. Software need not write or read this register. The register is programmable from the IP bus, but read-only from the PCI bus.
16:23	Interrupt Pin	Hardwired 0x01. Indicates device uses Int A as an interrupt request pin. This should be 0x00 because MGT5100 does **not** use an interrupt request pin on the PCI bus.
24:31	Interrupt Line	Hardwired 0x00. Stores a value that identifies which PCI Interrupt Controller input is 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.

10.3.3.1.13 PCI Header: Unused (0D40–0D5F)

These registers (0D40–0D5F) are unused and always read 0.

10.3.3.2 MGT5100 Application Interface Registers—MBAR + 0x0D00

The following registers are not part of standard PCI Header space. These registers are customized to MGT5100 application interfaces.

Primarily, these registers pertain to the PCI XLB interface. SmartDMA Tx and Rx interfaces have their own registers and represent separate interrupt sources.

10.3.3.2.1 PCI Interrupt Enable (0D60)

Table 10-9 PCI Interrupt Enable (0D60)

	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							IE	Max Retries							
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	IE	Interrupt Enable—bit enables generation of a CPU interrupt when PCI errors occur in conjunction with the PCI XLB Initiator interface. Possible error conditions are listed in the PCI Status Register. **NOTE:** This interrupt is limited to direct XLB to PCI transactions only and is recorded in the Interrupt Controller as a Peripheral number 8 interrupt.
24:31	Max Retries	Maximum Retry—Indicates the maximum number of attempts that can occur before a XLB PCI Initiator interface aborts. Retries are per transaction, rather than being cumulative. A setting of 0 or 0xFF results in no retry abort (i.e., infinite retries are possible).

10.3.3.2.2 PCI Status (0D64)

This status register is provided in addition to status recording that may occur in the DWPCI (i.e., the standard PCI configuration status register). Since DWPCI does not generate an interrupt, this module traps error signals from DWPCI.

If enabled, this register generates a CPU interrupt.

Table 10-10 PCI Status (0D64)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					RE	TA	IA	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
5	RE	Retry Error—When set, bit indicates Maximum Retries occurred in an XLB PCI Initiated transaction. Write 1 to this bit to clear. The PCI Header Status Register may also need to be queried (and serviced) to fully clear this interrupt source.
6	TA	Target Abort—When set, bit indicates Target Abort occurred in an XLB PCI Initiated transaction. Write 1 to this bit to clear. This bit may be set in conjunction with a Retry attempt and does not necessarily indicate a fatal error condition. The PCI Header Status Register also needs to be queried (and serviced) to determine the severity of the error.
7	IA	Initiator Abort—When set, bit indicates Initiator Abort occurred in an XLB PCI Initiated transaction. Write 1 to this bit to clear. The PCI Header Status Register may also need to be queried (and serviced) to fully clear this interrupt source.
8:31	—	Reserved

10.3.3.2.3 PCI Control (0D68)

This register controls global DPWCI settings. Therefore, this register impacts PCI transactions from any interface.

Table 10-11 PCI Control (0D68)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved				PR	CM	DP	LD	Reserved							
RESET:	0	0	0	0	1	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

Bits	Name	Description
0:3	—	Reserved
4	PR	PCI Reset—Default is 1. PR is asserted externally until software clears this bit. When written as 1, external PCI Reset pin is asserted. When written as 0, external PCI Reset pin is negated. **NOTE:** Internal DWPCI module is NOT reset when this bit is set.

Bits	Name	Description
5	CM	config_setmsb—This bit controls how address translation occurs in the PCI to XLB Initiator interface. When set to 1, this bit is forced onto the msb of the address sent to an external PCI device during any configuration access, regardless of the window hit. This applies only to configuration PCI command types where 1 is applied after any mask operation. This relates to “direct” versus “indirect” configuration access. When set to 0, this bit relinquishes control of the address msb to the Mask/Value and determines whether the Apply Translation bit is set or not (which is a per-Window control bit).
6	DP	Disable Posting—a control bit that applies only when MGT5100 is a Target. When set to 1, posted write operations are disabled. When set to 0, posted writes are enabled. This register must be written to before the second clock of $\overline{\text{FRAME}}$ assertion by an external PCI master.
7	LD	Latrule Disable—a control bit that applies only when MGT5100 is a Target. When set, this bit prevents the PCI Controller from automatically issuing a retry of the transaction due to the PCI16/8 clock rule. The bit must be set before the 15th PCI clock for the first transfer and before the 7th clock for other transfers.

10.3.3.2.4 PCI Mask/Value Read (0D6C)

In the current design, this register uses different read and write locations. Appendix C, Addendum, gives more information.

Table 10-12 PCI Mask/Value Read (0D6C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Window1 Mask								Window1 Value							
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	Window2 Mask								Window2 Value							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Window1 Mask	Used for PCI to XLB Initiator accesses on Window1 space, as defined by PCI1_START/STOP registers in MMAP module. This byte determines which upper 8 bits of the XLB address should be translated for use as a PCI address. 1 indicates corresponding bit should be translated according to Value byte below. This provides a method for overlaying a PCI page address onto the XLB address. 0 in the Mask Byte indicates the XLB address bit should be passed to PCI unaltered. **NOTE:** A Window Translation bit is in the Window Command and Control Register. This bit can turn the operation ON and OFF.
8:15	Window1 Value	Used for any Masked bit (as described above). The corresponding Value bit contained here is inserted into XLB address for presentation as a PCI address. **NOTE:** The Mask/yValue operation is limited to the Most Significant Byte only.
16:23	Window2 Mask	Same as Window1 Mask, except this byte applies to accesses on Window2 space, as defined by PCI2_START/STOP registers in MMAP module.
24:31	Window2 Value	Same as Window1 Value, but for Window2.

10.3.3.2.5 PCI Mask/Value Write (0D70)

In the current design, this register use different read and write locations. Appendix C, Addendum, gives more information.

Table 10-13 PCI Mask/Value Write (0D70)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	Window1 Mask								Window1 Value							
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	Window2 Mask								Window2 Value							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	Window1 Mask	Used for PCI to XLB Initiator accesses on Window1 space (as defined by PCI1_START/STOP registers in MMAP module). This byte determines which upper 8 bits of the XLB address should be translated for use as a PCI address. 1 indicates the corresponding bit should be translated according to Value Byte below. This gives a method for overlaying a PCI page address onto the XLB address. 0 in Mask Byte indicates XLB address bit should be passed to PCI unaltered. **NOTE:** There is a Window Translation bit in the Window Command and Control Register that can turn this operation ON and OFF.

Bits	Name	Description
8:15	Window1 Value	For any Masked bit (as described above), the corresponding Value bit contained here is inserted into the XLB address for presentation as a PCI address. **NOTE:** Mask/Value operation is limited to the Most Significant Byte only.
16:23	Window2 Mask	Same as Window1 Mask, except this byte applies to access on Window2 space, as defined by PCI2_START/STOP registers in MMAP module.
24:31	Window2 Value	Same as Window1 Value but for Window2.

10.3.3.2.6 PCI Subwindow 1 (0D74)

Table 10-14 PCI Subwindow 1 (0D74)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Subwindow1 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	Subwindow1 Stop Address															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
31:16	Subwindow1 Start Address	Register defines a Subwindow within either Main Window. Start Address is applied as an equal-to-or-greater-than test to any Main Window access in progress. If a Subwindow is detected, the Window command and control settings for the Subwindow are applied, rather than the Main Window settings.
15:0	Subwindow1 Stop Address	Stop Address is applied as a less-than test to any XLB address already accepted as a hit within a Main Window address. No correlation exists between Subwindows and Main Window1 and 2. Once a Main Window address range is detected, both Subwindow checks occur. Therefore, Subwindows1 and 2 should not overlap.

10.3.3.2.7 PCI Subwindow 2 (0D78)

Table 10-15 PCI Subwindow 2 (0D78)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Subwindow2 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	Subwindow2 Stop Address															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
31:16	Subwindow2 Start Address	Register defines a Subwindow within either Main Window. Start Address is applied as an equal-to-or-greater-than test to any Main Window access in progress. If a Subwindow is detected, the Window command and control settings for the Subwindow are applied, rather than the Main Window settings.
15:0	Subwindow2 Stop Address	Stop Address is applied as a less-than test to any XLB address already accepted as a hit within a Main Window address. There is no correlation between Subwindows and Main Window1 and 2. Once a Main Window address range is detected, both Subwindow checks occur. Therefore, Subwindows1 and 2 should not overlap.

10.3.3.2.8 PCI Window Command/Control (0D7C)

Table 10-16 PCI Window Command/Control (0D7C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Win1 Command			▨	▨	Win1 Control			Win2 Command			▨	▨	Win2 Control		
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	Sub1 Command			▨	▨	Sub1 Control			Sub2 Command			▨	▨	Sub2 Control		
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:2	Win1Command (0:2)	Main Window1 Command—When the XLB address range falls within the Main PCI Window1 and not within an active Subwindow, these 3 bits are the upper 3 bits driven on the PCI command lines. The lsb is determined by the type of transaction, either: 0 for read. 1 for write. The 3 bits are left justified to let software write the PCI command value in its natural 4-bit position. However, the right-most bit always reads 0.
5:7	Win 1 Control (0:2)	Main Window1 Control—When the XLB address range falls within the Main PCI Window1 (not within an active Subwindow) and this bit[0] is set, the following non-contiguous address translation is done: PCI_address[31:0] = {0x0000,XLB_address[9:19],XLB_address[27:31]} If used, the algorithm is applied prior to Main PCI Window address translation. (see bit[2] of this field). For normal addressing, this bit should be clear. Bit[1] is reserved and must be written 0. If written as 1, it causes the "disable posting" signal to be asserted to DWPCI during Initiator transactions for the given window. At present, there is no known reason to for this type of operation; posting only affects Target transactions. Bit[2], if set during a decoded Main Window1 access (and not a Subwindow), then Address translation indicated by the corresponding Mask/Value registers is applied. Otherwise, no Mask/Value translation is applied. This bit essentially serves as a enable bit for the Mask/Value operation.

Bits	Name	Description
8:10	Win 2 Command (0:2)	Same as Win 1 Command, except this applies to Window2
13:15	Win 2 Control (0:2)	Same as Win 1 Control, except this applies to Window2
16:18	Sub 1 Command (0:2)	Same as Win 1 Command, except this applies to Subwindow1.
21:23	Sub 1 Control (0:2)	Same as Win 1 Control, except this applies to Subwindow1.
24:26	Sub 2 Command (0:2)	Same as Win 1 Command., except this applies to Subwindow2.
29:31	Sub 2 Control (0:2)	Same as Win 1 Control, except this applies to Subwindow2.
3:4 11:12 19:20 27:28	—	Reserved

10.3.4 PCI SmartDMA Initiator Interface

10.3.4.1 PCI SmartDMA Transmit (Tx) Initiator Interface

This interface is used for DMA write transfers to the PCI bus. It consists of a Tx FIFO integrated as a SmartDMA peripheral. As such, it is generally controlled by the SmartDMA Controller through a pre-described program loop.

As with all SmartDMA peripherals, this interface can be accessed and controlled directly through the IP bus interface. However, this path does not generally lend itself to high throughput. With one exception, under SmartDMA control the XLB remains available for other activity. The exception is occasional XLB burst traffic, if the source data is on an XLB resource (e.g., SDRAM).

The Tx FIFO consists of 32 long-words and supports PCI bursts up to 8 long-words. This PCI burst size is programmable, up to the maximum of 8.

The usual method writes a PCI command word and address to the control register, along with the number of bytes to be transmitted (Packet_Size). The module waits for the Tx FIFO to fill, then begins transmitting data to the PCI bus.

Transmission continues until the specified number of bytes are sent. Software must handle filling the Tx FIFO to support the specified number of bytes. At this point, software must restart the procedure by (at least) rewriting the Packet_Size register.

NOTE: "Software" in this context usually means an autonomous SmartDMA task loop rather than the processor.

If desired, a CPU interrupt can be generated at the end of each Packet or when various error conditions occur.

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 to a new starting address.

- The largest burst size is 8.
- The largest packet size is 65,535.

A packet typically consists of many PCI data bursts. The Tx Controller stalls until sufficient bytes are in the Tx FIFO to support a full burst. It continues in this mode until the entire packet is transmitted.

NOTE: Even though some signals are referred to in bytes, this DMA module only does long-word access to and from PCI. For example, the 2 least significant bits of the Packet_Size value are ignored.

10.3.4.2 PCI Transmit (Tx) Registers—MBAR + 0x3800

PCI Tx is controlled by 14 32-bit registers. These registers are located at an offset from MBAR of 0x3800. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3800 + register address**

Hyperlinks to the PCI Tx registers are provided below:

- PCI Tx Packet Size (3800)
- PCI Tx Start Address (3804)
- PCI Tx Transaction Control (3808)
- PCI Tx Enables (380C)
- PCI Tx Next Address (3810)
- PCI Tx Last Word (3814)
- PCI Tx Done Counts (3818)
- PCI Tx Status Bits (381C)
- PCI Tx FIFO Data (3840)
- PCI Tx FIFO Status (3844)
- PCI Tx FIFO Control (3848)
- PCI Tx Alarm (384E)
- PCI Tx Read Pointer (3852)
- PCI Tx Write Pointer (3856)

NOTE: Hyperlinks to the PCI Receive (Rx) Registers—MBAR + 0x3880 are on page 10-32.

10.3.4.2.1 PCI Tx Packet Size (3800)

Table 10-17 PCI Tx Packet Size (3800)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Packet_Size															
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 Tx Controller to send over PCI. If Master_Enable bit is high and Reset_Controller bit is low, writes to this register completes a restart sequence.
16:31	—	Reserved

10.3.4.2.2 PCI Tx Start Address (3804)

Table 10-18 PCI Tx Start Address (3804)

	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 presented for the first PCI Packet DWORD. The PCI Tx Controller tracks and calculates the necessary address for subsequent transactions (addressing is assumed to be sequential from the start address).

10.3.4.2.3 PCI Tx Transaction Control (3808)

Table 10-19 PCI Tx Transaction Control (3808)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				PCI_cmnd				Max_Retrys							
W																
RESET:	0	0	0	0	0111				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																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	—	Reserved
4:7	PCI_cmd	This field contains the PCI command to present during the address phase of each PCI transaction. Default is write memory. This field is NOT checked for consistency. If an illegal value is written, unpredictable results occur. If default value is not used, the user should write this register only once prior to any packet restart.
8:15	Max_Retrys	This field contains the maximum number of retrys to allow per transaction. A slow or malfunctioning target might issue infinite disconnects, which could permanently tie up the PCI bus. A non-zero Max_Retrys value detects this condition and generates a CPU interrupt (if Abort Enable bit is high). Setting Max_Retrys to 0xFF allows infinite retry cycles to occur. A value of 0x00 should be avoided.
16:23	Max_Beats	This field contains the desired number of PCI data beats to attempt on each PCI transaction. A default setting of 0 represents a maximum of 8 beats per transaction. The Tx Controller waits until sufficient bytes are in the Tx FIFO to support the indicated number of beats. Each beat is 4 Bytes. When a packet is nearly complete, and less than the Max_Beats number of bytes remain to complete the packet, the Tx Controller automatically issues single-beat transactions until the packet is finished.
24:31	—	Reserved

10.3.4.2.4 PCI Tx Enables (380C)

Enables (Byte 0 Only Significant, All Bit Fields).

Table 10-20 PCI Tx Enables (380C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	RC	RF	Rsvd	CM	BE	AE	NE	ME	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

Bits	Name	Description
0	RC	Reset Controller—user writes bit high to put Tx Controller in a reset state. Other register bits are NOT affected. This reset is intended for recovery from an error condition or to re-load the start address when Continuous mode is selected. This bit does not prohibit register access. However, it must be negated to initiate a restart sequence (i.e., writing the Packet_Size register). If this bit is used to reload a start address, the Start_Add register must be written prior to asserting this bit.
1	RF	Reset FIFO—this pin state bit directly feeds FIFO reset pin and is active high. FIFO is reset and existing data is flushed. Alarm and granularity settings are not affected. The RC bit and reset FIFO bit operate independently. However, both must be low for normal operation.
2	—	Reserved
3	CM	Continuous Mode—user writes bit high to activate CM. In CM, Start_Add value is ignored at each packet restart, PCI address is auto-incremented from one packet to the next. The Packets_Done status byte becomes active, indicating how many packets were transmitted since the last Reset Controller condition. If bit is low, software is responsible for updating Start_Add value at each packet restart.
4	BE	Bus Error—user writes bit high to enable bus error indications. This means illegal IP bus transactions are terminated with a transfer error acknowledge. See PCI Tx Status Bits (381C) for BE descriptions. Normally this bit is low (negated), since illegal IP bus access is not destructive to register contents. Although, it may indicate damaged software. This bit does not affect interrupt generation from the module. However, a TEA_b occurs on XLB, which creates a Machine Check Exception.
5	AE	Abort Error—user writes bit high to enable CPU Interrupt generation if an abnormal packet transmission termination occurs. See PCI Tx Status Bits (381C) for possible error conditions. The SmartDMA Transmit module interrupt is Peripheral Interrupt #10. It may be desirable to mask CPU interrupts when SmartDMA is controlling operation. Status bits should be polled to prevent a possible lock-up condition.
6	NE	Normal termination Enable—user writes 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 type of SmartDMA and/or core program control in effect. A typical software model has CPU intervention between each Packet, with a SmartDMA task loop controlling intra-packet operation (i.e., filling the Tx FIFO).
7	ME	Master Enable—the Tx Controller Master Enable signal. User writes bit high to enable operation. This bit can be toggled low to allow out-of-order register updates prior to generating a Restart sequence. In which case, transmission begins when ME is written high. This bit should NOT be used as such in a continuous mode, because it has the side effect of resetting the Packets_Sent status counter.
8:31	—	Reserved

10.3.4.2.5 PCI Tx Next Address (3810)

Table 10-21 PCI Tx Next Address (3810)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Next_Address															
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	Next_Address															
W	Reserved															
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. It is updated at the successful completion of each PCI data beat. This register represents a byte address and is updated with the user-written Start_Add value when the Start_Add is reloaded. It is intended to be accurate even if an abnormal PCI bus termination occurs.

10.3.4.2.6 PCI Tx Last Word (3814)

Table 10-22 PCI Tx Last Word (3814)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Last_Word															
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	Last_Word															
W	Reserved															
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 long word fetched from FIFO. It is intended for use when an abnormal PCI termination has corrupted FIFO data integrity (for that word).

10.3.4.2.7 PCI Tx Done Counts (3818)

Table 10-23 PCI Tx Done Counts (3818)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Bytes_Done															
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	Packets_Done															
W	Reserved															
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 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 transmitted and is active only if continuous mode is in effect. If the following occurs, the counter resets: • Reset Controller bit is asserted (normal way to restart continuous mode). • Master Enable bit is negated. In this way, 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 This assumes Packet_Size is the same for all restart sequences.

10.3.4.2.8 PCI Tx Status Bits (381C)

Table 10-24 PCI Tx Status Bits (381C)

	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	TE	IE
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
7	NT	Normal termination—flag sets when packet terminates normally. It is not set for abnormally terminated packets. See Note.
8	BE3	Bus Error type 3—flag sets when an IP bus transaction attempts to write to a read-only register. Flag bit sets regardless of bus error (BE) enable bit. See Note. If software is polling this byte and wishes to disregard this error, this bit must be masked. No register bit corruption occurs for this (or any other) bus error case.
9	BE2	Bus Error type 2—flag sets when an IP bus transaction attempts to write to a reserved register (an entire 32-bit register, not just a reserved bit or byte). Flag bit sets regardless of BE. See Note. If software is polling this byte and wishes to disregard this error, this bit must be masked.
10	BE1	Bus Error type 1—flag sets when an IP bus transaction attempts to read a reserved register (an entire 32-bit register, not just a reserved bit or byte). Flag bit sets regardless of BE. See Note. If software is polling this byte and wishes to disregard this error, this bit must be masked.
11	FE	FIFO error—flag sets when Tx FIFO asserts its FIFO error output. See Note. If abort error enable (AE) bit is set, a CPU interrupt generates. Error source is determined by reading FIFO error status register. The error condition must be cleared at the FIFO prior to clearing this Sticky bit or the flag continues to assert.
12	SE	System error—flag sets in response to Tx Controller entering an illegal state. See Note. If AE bit is set, a CPU interrupt generates. In normal operation this should never occur. To recover, assert Reset Controller (RC) bit and clear flag.
13	RE	Retry error—flag sets if Max_Retrys setting is non-zero and PCI transaction has performed retries in excess of the setting. See Note. If AE bit is set, a CPU interrupt generates. Retry counter is reset at beginning of each transaction (i.e., it is NOT cumulative throughout a packet) and generally indicates a damaged or improperly accessed target.
14	TE	Target abort error—flag sets if DWPCI Controller issued a target abort, which means the addressed PCI target has signalled an abort. See Note. If AE bit is set, a CPU interrupt is generated. Application software must query target's status register to determine error source. Coherency of Tx FIFO data and Tx Controller status registers (Next_Address, Bytes_Done, etc.) should remain valid.
15	IE	Initiator abort error—flag sets if PCI Controller issues an initiator abort flag. This generally means no target responded, but further status information can be read from PCI configuration interface. See Note. If AE bit is set, a CPU interrupt is generated. Coherency of Tx FIFO data and Tx Controller status registers (Next_Address, Bytes_Done, etc.) should remain valid.
16:31	—	Reserved
NOTE: Flag does not **require** clearing, but does not clear until 1 is written, in which case 0 is read (i.e., negated).		

10.3.4.2.9 PCI Tx FIFO Data (3840)

Table 10-25 PCI Tx FIFO Data (3840)

	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	FIFO data port—reading from this location pops data from FIFO, writing data pushes data into FIFO. During typical operation, the SmartDMA Controller pushes data here. The Tx Controller pops data. Therefore, user programs should not read here. See Note.
NOTE: Only full long word access is allowed. If all byte enables are not asserted when accessing this location, FIFO data is corrupted.		

10.3.4.2.10 PCI Tx FIFO Status (3844)

Table 10-26 PCI Tx FIFO Status (3844)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									Err	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	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 logic-OR of other flag bits and can be polled for detection of FIFO errors. After clearing the offending condition, writing 1 to this bit clears flag.
10	UF	UnderFlow—flag indicates read pointer surpassed write pointer. FIFO was read beyond empty. Resetting FIFO clears this condition. Writing 1 to this bit clears flag.

Bits	Name	Description
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	FR	Frame Ready—This bit is not used and may read as 1 or 0.
13	Full	FIFO is full. Cleared by reading or resetting FIFO
14	Alarm	This bit reflects a physical Requestor signal, which connects to the SmartDMA module. PCI Tx Requestor number is 8. If high, it indicates Requestor is active and means FIFO is nearing empty. The near-empty threshold is described as less than X number of bytes remaining, where X is defined by the Alarm setting in PCI Tx FIFO Control Register. If low, it indicates Requestor is negated and means FIFO is nearing full. The near-full threshold is described as less than X number of free bytes (space) remaining, where X is 4 times the granularity setting in PCI Tx FIFO Control Register.
15	Empty	FIFO is empty. Cleared by writing data to FIFO.
16:31	—	Reserved

10.3.4.2.11 PCI Tx FIFO Control (3848)

Table 10-27 PCI Tx FIFO Control (3848)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		WFR	Reserved		GR			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

Bits	Name	Description
0:1	—	Reserved
2	WFR	When bit is set, FIFO Controller assumes next data transmitted is End of Frame (EOF). See Note 1.
3:4	—	Reserved
5:7	GR	Granularity—bits control a high “watermark” point at which FIFO negates an Alarm condition (i.e., request for data). GR represents the number of free bytes times 4. See Notes 2 and 3.
8:31	—	Reserved
NOTE: 1. This module does not support Framing. Bit should remain low. 2. A granularity setting of 0 should be avoided. Such a setting means the Alarm bit (and Requestor signal) will not negate until FIFO is completely full. The SmartDMA module may do up to 2 more data writes after a Requestor negation, due to its internal pipelining. 3. Granularity setting is in bytes for this FIFO (due to an implementation error). A setting of 1–4 is equivalent to granularity of 1 (5–7 equals granularity of 2). Appendix C, Addendum, gives more information.		

10.3.4.2.12 PCI Tx Alarm (384E)

Table 10-28 PCI Tx Alarm (384E)

	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 a low level “watermark”, which is the point where FIFO asserts a request for SmartDMA Controller data filling. Value is in bytes. For example, with Alarm = 32, alarm condition occurs when FIFO contains less than 32Bytes. Once asserted, alarm does not negate until high level mark is reached, as specified by PCI Tx FIFO Control Register granularity bits.

10.3.4.2.13 PCI Tx Read Pointer (3852)

Table 10-29 PCI Tx Read Pointer (3852)

	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 FIFO RAM.

10.3.4.2.14 PCI Tx Write Pointer (3856)

Table 10-30 PCI Tx Write Pointer (3856)

	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 Write address presented to the FIFO RAM.

10.3.4.3 PCI SmartDMA Receive (Rx) Initiator Interface

This interface is used for DMA reads from the PCI bus. It consists of an Rx FIFO integrated as a SmartDMA peripheral. As such, it is generally controlled by the SmartDMA Controller through a pre-described program loop.

As with all SmartDMA peripherals, this interface can be accessed and controlled directly through the IP bus interface, if desired. However, this path does not generally lend itself to high throughput. Under SmartDMA control, the XLB remains available for other activity.

The Rx FIFO consists of 32 long-words and supports PCI bursts up to 8 long-words. This burst size is programmable (up to the maximum of 8).

The general approach is to write a PCI command word and address to the control register, including the number of bytes to be transmitted (Packet_Size). The module verifies enough space is available in the Rx FIFO and immediately begins the PCI read transactions.

Transmission continues until the specified number of bytes are received. Software must handle emptying the Rx FIFO to support the specified number of bytes. 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.
- The largest Packet_Size is 65,535.

A packet typically consist of many PCI data bursts. The Rx Controller stalls until enough space is available in the Rx FIFO to support a full burst. It continues in this mode until the entire packet is transmitted.

NOTE: Even though some signals are referred to in bytes, this DMA module only does long word access to and from PCI. For example, the 2 least significant bits of the Packet_Size value are ignored.

10.3.4.4 PCI Receive (Rx) Registers—MBAR + 0x3880

PCI Rx is controlled by 13 32-bit registers. These registers are located at an offset from MBAR of 0x3880. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3880 + register address**

Hyperlinks to the PCI Rx registers are provided below:

- PCI Rx Packet Size (3880)
- PCI Rx Start Address (3884)
- PCI Rx Transaction Command (3888)
- PCI Rx Enables (388C)
- PCI Rx Next Address (3890)
- PCI Rx Done Counts (3898)
- PCI Rx Status Bits (389C)
- PCI Rx FIFO Data (38C0)
- PCI Rx FIFO Status (38C4)
- PCI Rx FIFO Control (38C8)
- PCI Rx Alarm (38CC)
- PCI Rx Read Pointer (38D0)
- PCI Rx Write Pointer (38D4)

10.3.4.4.1 PCI Rx Packet Size (3880)

Table 10-31 PCI Rx Packet Size (3880)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Packet_Size															
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 this register with the number of bytes for Rx Controller to fetch over PCI. If Master_Enable bit is high and Reset_Controller bit is low, writing to this register completes a restart sequence.
16:31	—	Reserved

10.3.4.4.2 PCI Rx Start Address (3884)

Table 10-32 PCI Rx Start Address (3884)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Start_Add															
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	Start_Add															
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 desired current packet starting address to this register. This is the address first presented on the external PCI bus, then auto-incremented as needed. The register itself does NOT increment as the PCI packet proceeds.

10.3.4.4.3 PCI Rx Transaction Command (3888)

Table 10-33 PCI Rx Transaction Command (3888)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved				PCI_cmnd				Max_Retrys							
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 / W	Reserved			FB		Max_Beats			Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
4:7	PCI_cmd	Field contains the PCI command to present during the address phase of each PCI transaction. Default is read memory multiple. This field is NOT checked for consistency. If an illegal value is written, unpredictable results occur. If default value is not used, the user should write this register only once prior to any packet restart.
	Max_Retrys	Field contains the maximum number of retrys allowed per transaction. A slow or malfunctioning target might issue infinite disconnects, which could permanently tie up the PCI bus. If Abort Enable bit is high, a non-zero Max_Retrys value detects this condition and generates a CPU interrupt. Setting Max_Retrys to 0xFF allows infinite retry cycles to occur. A value of 0x00 should be avoided.
19	FB	Full Burst—is a special test bit that lets unlimited burst and packet size transactions occur. Use of this bit is **not** recommended. This bit must be maintained low for normal operation.

Bits	Name	Description
21:23	Max_Beats	Field contains the desired number of PCI data beats to attempt on each PCI transaction. A default setting of 0 represents the maximum of 8 beats per transaction. Rx Controller waits until sufficient space is in the Rx FIFO to support the indicated number of beats. Each beat is 4 Bytes. When a packet is nearly complete, and less than the Max_Beats number of bytes remain to complete the packet, Rx Controller automatically issues single-beat transactions until packet is finished.
24:31	—	Reserved

10.3.4.4.4 PCI Rx Enables (388C)

Table 10-34 PCI Rx Enables (388C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	RC	RF	FE	CM	BE	AE	NE	ME	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

Bits	Name	Description
0	RC	Reset Controller—bit is written high to put Rx 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. RC bit does not prohibit register access. However, it must be negated to initiate a Restart sequence (i.e., writing the Packet_Size register). If used to reload a start address, the Start_Add register must be written prior to asserting this bit.
1	RF	Reset FIFO—this pin state directly feeds the FIFO reset pin and is active high. FIFO is reset and flushed of any existing data. RC bit and RF bit operate independently. However, both must be low for normal operation.
2	FE	Flush Enable—causes a flush signal to be generated to the Rx FIFO Controller. This flush is necessary to insure the SmartDMA requestor is asserted and all data left in the Rx FIFO is transferred out and the SmartDMA task loop can complete. FE is active high.
3	CM	Continuous Mode—bit is written high to activate CM. In CM Start_Add value is ignored at each packet restart, PCI address is auto-incremented from one packet to the next. The Packets_Done status word becomes active, indicating how many packets were received since the last reset Controller condition. If the continuous bit is low, software is responsible for updating Start_Add value at each packet restart.

Bits	Name	Description
4	BE	Bus Error—user writes this bit high to enable bus error indications. This means illegal IP bus transactions are terminated with a transfer error acknowledge. See PCI for BE descriptions. Normally this bit is low (negated), since illegal IP bus access is not destructive to register contents. Although, it may indicate damaged software. This bit does not affect interrupt generation from the module. However, a TEA_b occurs on XLB, which creates a Machine Check Exception.
5	AE	Abort Error—user writes bit high to enable CPU Interrupt generation if an abnormal packet transmission termination occurs. See PCI Rx Status Bits (389C) for possible error conditions. The SmartDMA Receive module interrupt is Peripheral Interrupt #9. It may be desirable to mask CPU interrupts when SmartDMA is controlling operation. Status bits should be polled to prevent a possible lock-up condition.
6	NE	Normal termination Enable—user writes bit high to enable CPU Interrupt generation at conclusion of a normally terminated packet transmission. This may or may not be desirable, depending on the type of SmartDMA and/or core program control in effect. A typical software model has CPU intervention between each Packet, with a SmartDMA task loop controlling intra-packet operation (i.e., filling the Tx FIFO).
7	ME	Master Enable—the Rx Controller Master Enable signal. This bit is written high to enable operation. It can be toggled low to allow out-of-order register updates prior to generating a restart sequence. In which case, transmission begins when ME is written back high. However, this signal should **not** be used as such in continuous mode. It has the side effect of resetting the Packet_Sent status counter.
16:31	—	Reserved

10.3.4.4.5 PCI Rx Next Address (3890)

Table 10-35 PCI Rx Next Address (3890)

	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. Register is updated at the successful completion of each PCI data beat. The data represents a byte address and is updated with a user-written Start_Add value when Start_Add is reloaded. The intent of this register is to be accurate, even if an abnormal PCI bus termination occurs.

10.3.4.4.6 PCI Rx Done Counts (3898)

Table 10-36 PCI Rx Done Counts (3898)

	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 is asserted (normal way to restart continuous mode). • Master Enable bit 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.

10.3.4.4.7 PCI Rx Status Bits (389C)

Table 10-37 PCI Rx Status Bits (389C)

	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	TE	IE
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
7	NT	Normal termination—flag sets when packet terminates normally. It is not set for abnormally terminated packets. See Note 1.
8	BE3	Bus Error type 3—flag sets when an IP bus transaction attempts to write to a read-only register. Flag bit sets regardless of bus error enable bit (BE). If software is polling this byte and wishes to disregard this error, it must mask this bit. See Notes 1 and 2.
9	BE2	Bus Error type 2—flag sets when an IP bus transaction attempts to write to a reserved register (an entire 32-bit register, not a reserved bit or byte). Flag sets regardless of BE. If software polls this byte and wishes to disregard this error, this bit must be masked. See Note 1.
10	BE1	Bus Error type 1—flag sets when an IP bus transaction attempts to read a reserved register (an entire 32-bit register, not a reserved bit or byte). Flag sets regardless of BE. If software polls this byte and wishes to disregard this error, this bit must be masked. See Note 1.
11	FE	FIFO error—flag sets when Rx FIFO asserts a FIFO error output. If abort error enable (AE) bit is set, a CPU interrupt generates. The error source is determined by reading the FIFO error status register. Error condition must be cleared at the FIFO prior to clearing this sticky bit. Otherwise, flag continues to assert. See Note 1.
12	SE	System error—flag sets in response to the Rx Controller entering an illegal state. If AE bit is set, a CPU interrupt generates. In normal operation this should never occur. To recovery, assert the Reset Controller (RC) bit and clear this flag. See Note 1.
13	RE	Retry error—flag sets if the Max_Retrys setting is non-zero and the PCI transaction does retries in excess of the setting. If AE bit is set, a CPU interrupt generates. The retry counter is reset at the beginning of each transaction (it is NOT cumulative throughout a packet) and would generally indicate a damaged or improperly accessed target. Non-zero Max_Retrys values are intended for debug or diagnostic use only. See Note 1.
14	TE	Target abort error—flag sets if PCI Controller issued a target abort, which means the addressed PCI target signalled an abort. If AE bit is set, a CPU interrupt generates. Application software must query the Target status register to determine the error source. Coherency of Rx FIFO data and Rx Controller status registers (Next_Address, Bytes_Done, etc.) should remain valid. See Note 1.
15	IE	Initiator abort error—flag sets if PCI Controller issues an initiator abort flag. This generally means no target responded, but further status information can be read from the PCI configuration interface. If AE bit is set, a CPU interrupt generates. Coherency of Rx FIFO data and Rx Controller status registers (Next_Address, Bytes_Done, etc.) should remain valid. See Note 1.
16:31	—	Reserved

NOTE:
1. Flag does not **require** clearing, but does not clear until 1 is written, in which case 0 is read (i.e., negated). All other flag bits operate similarly.
2. No register bit corruption occurs for this (or any other) bus error case.

10.3.4.4.8 PCI Rx FIFO Data (38C0)

Table 10-38 PCI Rx FIFO Data (38C0)

	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	FIFO data port—Reading from this location "pops" data from the FIFO; writing "pushes" data into the FIFO. During normal operation the SmartDMA Controller pops data here. The Rx Controller pushes data. Therefore, user programs should not write here. **NOTE:** Only full long word access is allowed. If all byte enables are not asserted when accessing this location, FIFO data is corrupted.

10.3.4.4.9 PCI Rx FIFO Status (38C4)

Table 10-39 PCI Rx FIFO Status (38C4)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									Err	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	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 logic-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 surpassed 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	FR	Frame Ready. This bit is not used and may read as 1 or 0.
13	Full	FIFO is full. Cleared by reading or resetting FIFO.

Bits	Name	Description
14	Alarm	This bit reflects a physical Requestor signal, which connects to the SmartDMA module. PCI Rx Requestor number is 7. If high, it indicates the Requestor is active and means the FIFO is nearing full. This near-full threshold is described as less than X number of free-bytes (space) remaining, where X is defined by the Alarm setting in PCI Rx Alarm Register. If low, it indicates the Requestor is negated and means the FIFO is nearing empty. This near-full threshold is described as less than X number of bytes remaining, where X is four times the granularity setting in PCI Rx Alarm Register.
15	Empty	FIFO is empty. Cleared by data being written to the FIFO.
16:31	—	Reserved

10.3.4.4.10 PCI Rx FIFO Control (38C8)

Table 10-40 PCI Rx FIFO Control (38C8)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved		WFR	Reserved		GR			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

Bits	Name	Description
0:1	—	Reserved
2	WFR	When this bit is set, the FIFO Controller assumes next data received is End of Frame (EOF). See Note 1.
3:4	—	Reserved
5:7	GR	Granularity—bits control low “watermark” point at which FIFO negates Alarm condition (i.e., request for data). It represents the number of bytes times 4. See Notes 2 and 3.
8:31	—	Reserved

NOTE:
1. This module does not support Framing. This bit should remain low.
2. A granularity setting of 0 should be avoided. Such a setting means the Alarm bit (and Requestor signal) will not negate until the FIFO is completely empty. due to its internal pipelining, the SmartDMA module may do up to 2 more data reads after negation of a Requestor.
3. Granularity setting is in bytes for this FIFO (due to an implementation error), a setting of 1–4 is equivalent to granularity of 1 (5–7 equals granularity of 2). Appendix C, Addendum, gives more information.

10.3.4.4.11 PCI Rx Alarm (38CC)

Table 10-41 PCI Rx Alarm (38CC)

	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 high level "watermark", which is the point where FIFO asserts request for SmartDMA Controller data emptying. Value is in free bytes (space). For example, with Alarm = 32, alarm condition occurs when FIFO contains less than 32 free bytes. Once asserted, alarm does not negate until low level mark is reached, as specified by FIFO control register 12 granularity bits.

10.3.4.4.12 PCI Rx Read Pointer (38D0)

Table 10-42 PCI Rx Read Pointer (38D0)

	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	Read Pointer—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 read address presented to FIFO RAM.

10.3.4.4.13 PCI Rx Write Pointer (38D4)

Table 10-43 PCI Rx Write Pointer (38D4)

	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	Write Pointer—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 FIFO RAM.

SECTION 11
ATA CONTROLLER

11.1 Overview

The following sections are contained in this document:

- SmartComm Key Features
- ATA Register Interface, includes:
  - ATA Host Registers—MBAR + 0x3A00
  - ATA FIFO Registers—MBAR + 0x3A00
  - ATA Drive Registers—MBAR + 0x3A00
- ATA Host Controller Operation
- Signals and Connections
- ATA Interface Description
- ATA Bus Background
- ATA RESET/Power-Up
- ATA I/O Cable Specifications
- ATA Electrical Characteristics

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 MGT5100 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 $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ on MGT5100 pins, they are asserted and internally held low for an appropriate period of time to satisfy ATA reset. An MGT5100 GPIO may be used to drive ATA reset independently if special software control is needed.

Figure 11-1 shows the ATA Controller Interface.

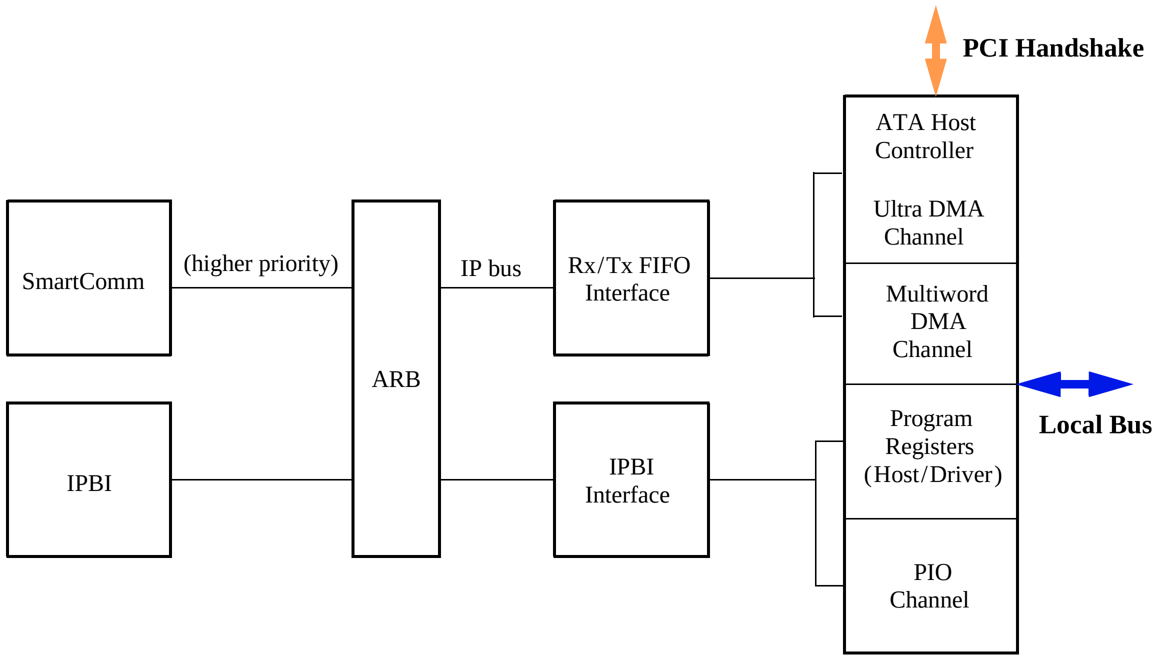


Figure 11-1 ATA Controller Interface

11.2 SmartComm Key Features

11.2.1 SmartComm Read

1. microprocessor sets up descriptors in SC RAM and initiates a transfer.
2. SC 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, SC is interrupted and moves data from FIFO to an internal destination.

11.2.2 SmartComm Write

1. microprocessor sets up descriptors in SC RAM and initiates a transfer.
2. SC hits on an ATA command FIFO space and writes a command (ATA drive register address, transfer size) into FIFO.
3. SC 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 SC SRAM buffering.

11.3 ATA Register Interface

The IP bus interface module contains all software-programmable ATA Controller registers and the IP bus glue logic needed to read and write these registers. The IP bus 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 timing registers are designed to accommodate clock speeds up to 108MHz. Enough counter bits are provided to count ATA timing, based on clock speeds up to 108MHz.

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 (3A00)—HCFG
- ATA Host Status (3A04)—HSR
- ATA PIO Timing 1 (3A08)—PIO1
- ATA PIO Timing 2 (3A0C)—PIO2
- ATA Multiword DMA Timing 1 (3A10)—DMA1
- ATA Multiword DMA Timing 2 (3A14)—DMA2
- ATA Ultra DMA Timing 1 (3A18)—UDMA1
- ATA Ultra DMA Timing 2 (3A1C)—UDMA2
- ATA Ultra DMA Timing 3 (3A20)—UDMA3
- ATA Ultra DMA Timing 4 (3A24)—UDMA4
- ATA Ultra DMA Timing 5 (3A28)—UDMA5

11.3.1.1 Host Configuration (3A00)—HCFG

Table 11-1 ATA Host Configuration (3A00)—HCFG

	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 11-28.)
2:5	—	Reserved
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 Host Status (3A04)—HSR

Table 11-2 ATA Host Status (3A04)—HSR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	TIP	UREP	Reserved				RERR	WERR	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

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 Command Register hut bit (see Table 11-28.).
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 PIO Timing 1 (3A08)—PIO1

Table 11-3 ATA PIO Timing 1 (3A08)—PIO1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	pio_t0								pio_t2_8							
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	pio_t2_16								Reserved							
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.

Bits	Name	Description
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 PIO Timing 2 (3A0C)—PIO2

Table 11-4 ATA PIO Timing 2 (3A0C)—PIO2

	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 Multiword DMA Timing 1 (3A10)—DMA1

Table 11-5 ATA Multiword DMA Timing 1 (3A10)—DMA1

	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 Multiword DMA Timing 2 (3A14)—DMA2

Table 11-6 ATA Multiword DMA Timing 2 (3A14)—DMA2

	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 Ultra DMA Timing 1 (3A18)—UDMA1

Table 11-7 ATA Ultra DMA Timing 1 (3A18)—UDMA1

	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 Ultra DMA Timing 2 (3A1C)—UDMA2

Table 11-8 ATA Ultra DMA Timing 2 (3A1C)—UDMA2

	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. (May not be needed).
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 Ultra DMA Timing 3 (3A20)—UDMA3

Table 11-9 ATA Ultra DMA Timing 3 (3A20)—UDMA3

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	udma_tmli								udma_taz							
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	udma_tenv								udma_tsri							
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 Ultra DMA Timing 4 (3A24)—UDMA4

Table 11-10 ATA Ultra DMA Timing 4 (3A24)—UDMA4

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	udma_tss								udma_trfs							
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	udma_trp								udma_tac							
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.

Bits	Name	Description
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 Ultra DMA Timing 5 (3A28)—UDMA5

Table 11-11 ATA Ultra DMA Timing 5 (3A28)—UDMA5

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	udma_tzah								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

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

11.3.2 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 (3A3C)—RTFDWR
- ATA Rx/Tx FIFO Status (3A40)—RTFSR
- ATA Rx/Tx FIFO Control (3A44)—RTFCR
- ATA Rx/Tx Alarm (3A48)—RTFAR
- ATA Rx/Tx FIFO Read Pointer (3A4C)—RTFRPR
- ATA Rx/Tx FIFO Write Pointer (3A50)—RTFWPR

11.3.2.1 Rx/Tx FIFO Data Word (3A3C)—RTFDWR

Table 11-12 ATA Rx/Tx FIFO Data Word (3A3C)—RTFDWR

	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 SmartDMA Controller pushes data here. **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 Rx/Tx FIFO Status (3A40)—RTFSR

Table 11-13 ATA Rx/Tx FIFO Status (3A40)—RTFSR

	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.
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.

Bits	Name	Description
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 Rx/Tx FIFO Control (3A44)—RTFCR

Table 11-14 ATA Rx/Tx FIFO Control (3A44)—RTFCR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		WFR	Reserved		GR			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

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

11.3.2.4 Rx/Tx Alarm (3A48)—RTFAR

Table 11-15 ATA Rx/Tx Alarm (3A48)—RTFAR

	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 SmartDMA Controller data filling. Value is in bytes. For example, with Alarm = 32, alarm condition occurs when FIFO contains 32Bytes 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 Rx/Tx FIFO Read Pointer (3A4C)—RTFRPR

Table 11-16 ATA Rx/Tx FIFO Read Pointer (3A4C)—RTFRPR

	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 Rx/Tx FIFO Write Pointer (3A50)—RTFWPR

Table 11-17 ATA Rx/Tx FIFO Write Pointer (3A50)—RTFWPR

	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 MGT5100 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 (3A5C)—DCTR, write-only
- ATA Drive Alternate Status (3A5C)—DASR, read-only
- ATA Drive Data (3A60)—DDR, R/W
- ATA Drive Features (3A64)—DFR, write-only
- ATA Drive Error (3A64)—DER, read-only
- ATA Drive Sector Count (3A68)—DSCR, R/W
- ATA Drive Sector Number (3A6C)—DSNR, R/W
- ATA Drive Cylinder Low (3A70)—DCLR, R/W
- ATA Drive Cylinder High (3A74)—DCHR, R/W
- ATA Drive Device/Head (3A78)—DDHR, R/W
- ATA Drive Command (3A7C)—DCR, write-only
- ATA Drive Device Status (3A80)—DSR, read-only

11.3.3.1 Device Control (3A5C)—DCTR

Table 11-18 ATA Drive Device Control (3A5C)—DCTR

	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:** For MGT5100 ATA Host Controller, enabling INTRQ is mandatory for DMA/UDMA data transfer modes.
7:31	—	Reserved

11.3.3.2 Drive Alternate Status (3A5C)—DASR

Table 11-19 ATA Drive Alternate Status (3A5C)—DASR

	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 Drive Data (3A60)—DDR

Table 11-20 ATA Drive Data (3A60)—DDR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data								Data							
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 Drive Features (3A64)—DFR

Table 11-21 ATA Drive Features (3A64)—DFR

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

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 Drive Error (3A64)—DER

Table 11-22 ATA Drive Error (3A64)—DER

	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 Drive Sector Count (3A68)—DSCR

Table 11-23 ATA Drive Sector Count (3A68)—DSCR

	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 Drive Sector Number (3A6C)—DSNR

Table 11-24 ATA Drive Sector Number (3A6C)—DSNR

	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 Drive Cylinder Low (3A70)—DCLR

Table 11-25 ATA Drive Cylinder Low (3A70)—DCLR

	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 Drive Cylinder High (3A74)—DCHR

Table 11-26 ATA Drive Cylinder High (3A74)—DCHR

	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 Drive Device/Head (3A78)—DDHR

Table 11-27 ATA Drive Device/Head (3A78)—DDHR

	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 Drive Command (3A7C)—DCR

Table 11-28 ATA Drive Command (3A7C)—DCR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	Data								Rsvd	HUT	FR	FE	IE	UDMA	READ	WRITE
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

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.

Bits	Name	Description
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 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 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 Device Status (3A80)—DSR

Table 11-29 ATA Drive Device Status (3A80)—DSR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	DRDY	Data		DRQ	Reserved		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	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 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.
8:31	—	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 (ipg_clk)
- the minimum data transfer mode cycle-time passed in the INDENTIFY DEVICE 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.
3. Count =

$$\frac{(\text{ATA\_mode\_timing\_spec} + \text{clock\_period} - 1)}{\text{clock\_period}}$$

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:

Count =

$$\frac{(\text{ATA\_mode\_timing\_spec} + 2 * \text{XCVR\_PROP\_DLY} + \text{clock\_period} - 1)}{\text{clock\_period}}$$

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:

Count =

$$\frac{(4 + \mathrm{t2CYC\_spec}[\mathrm{mode}] + \mathrm{clock\_period} - 1)}{\mathrm{clock\_period}}$$

4. Write the calculated count in the timing registers provided in the ATA host register memory map.
5. Write ATA drive registers per ATA-4 specification using Host Controller register memory map to the setup drive for desired operation.
6. 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.
7. Write ata_dma_mode register to indicate UDMA/DMA READ/WRITE operations for UDMA/DMA data transfer modes.
8. 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 MGT5100 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 IP bus interface module.

The ATA drive registers are also set up by the software through ATA IP bus interface 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 IP bus interface 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 11-30 PIO Timing Requirements

Counter	Start from	Activity at end	Dependencies
t0	t1	go to IDLE	t2=0, t2i=0, t4=0
t1[1]	N/A (Use t1 instead)	—	—
t2	t1	Latch Read_Data	IORDY_reg=1
t2i	t2	—	—
t3[2]	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)	—	—
t9[3]	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, and shall be well documented.
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 $\overline{\text{CS}}$[1]FX/$\overline{\text{CS}}$[3]FX/ADDR negate.

If ATA drive address space is hit by microprocessor, the ATA IP bus interface 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 IP bus interface 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 Smartcomm FIFO interface. When this interface is fully defined, the design specifics may be detailed. Table 11-31 shows the timing requirements specified in the ATA-4 spec for multiword DMA data transfers.

Table 11-31 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

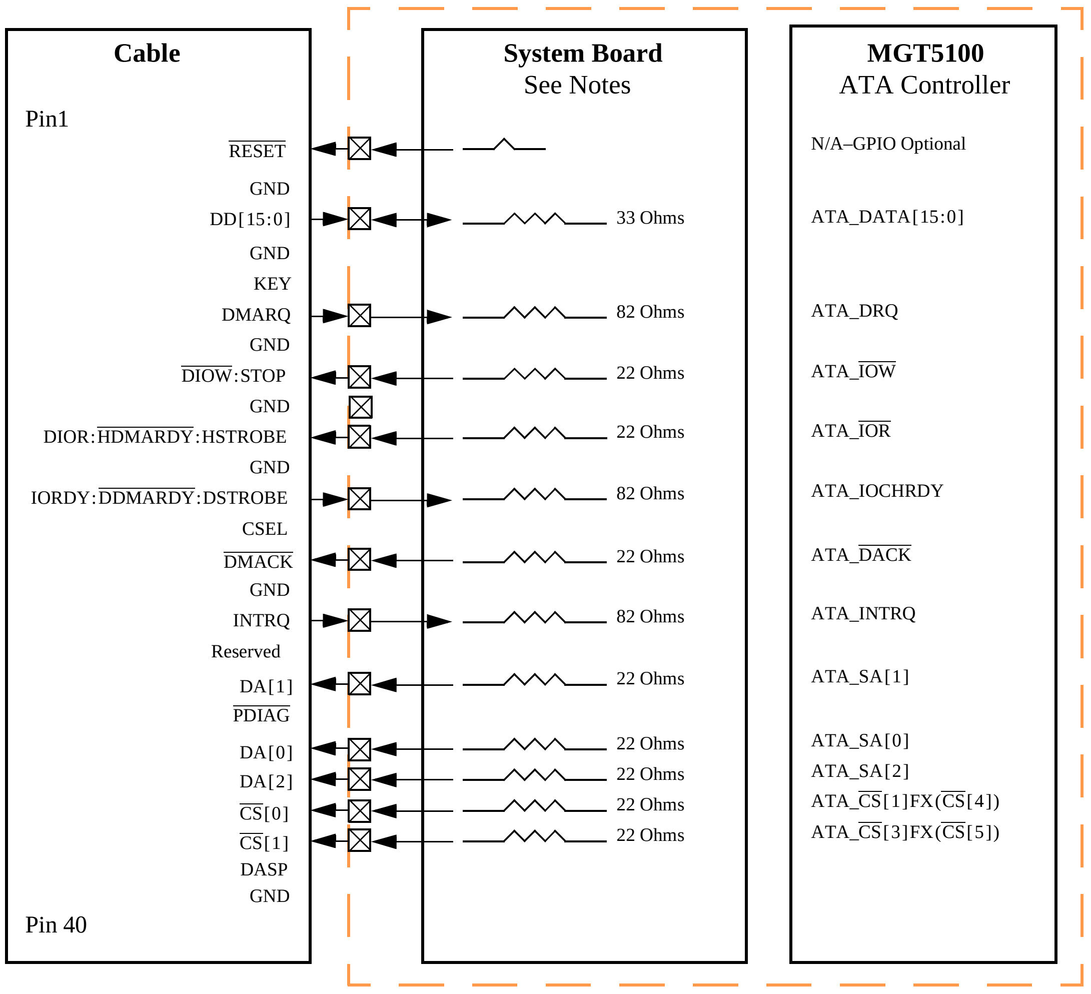
Table 11-32 MGT5100 External Signals

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).
$\overline{\text{CS}}$[1]FX	O	Chip select connected to $\overline{\text{CS}}$[0] on ATA cable.
$\overline{\text{CS}}$[3]FX	O	Chip select connected to $\overline{\text{CS}}$[1] on ATA cable.

Table 11-32 MGT5100 External Signals (continued)

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_$\overline{\text{WE}}$	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	NC[1]	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 MGT5100 ATA signals.

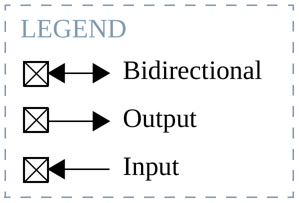


Figure 11-2 Connections—Controller Cable, System Board, MGT5100

11.6 ATA Interface Description

Table 11-33 ATA Controller External Connections

Pin#	Cable	I/O	System Board	I/O	MGT5100
1	RESET	O	$\overline{\text{RESET}}$: 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	—	—	—	—
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	—	—	—	—

HOST		DEVICE
CS[0], CS[1]	→	Chip Select to select Command Block registers.
DA[2:0]	→	Address to access drive registers or data ports.
DD[15:0]	→	8-, 16-bit data interface.
DIOR:HDMARDY:HSTROBE	→	DIOR→Asserted by host to read drive registers or data ports. HDMARDY→Host ready to receive UDMA data in bursts. Negated to pause. HSTROBE→Host signal for UDMA data out bursts. Data latched in drive registers from DD[15:0] on both edges of HSTROBE. Host stops generating HSTROBE edges to pause.
DIOW:STOP	→	DIOW→Asserted by host to write drive registers or data ports. Negated by host before initiation of UDMA. STOP→Negated by host before UDMA burst. Assertion by host signals termination of UDMA.
DMACK	→	Host response to DMARQ by drive to initiate DMA transfers.
DMARQ	←	Asserted by drive for DMA data transfers from/to host. For multiword DMA, data direction is controlled by DIOR and DIOW. MARQ is negated by drive when DMACK is received from host.drivere-asserts DMARQ for more DMA transfers.
INTRQ	←	INTRQ used by selected drive to interrupt host. If (nIEN bit == 0 && drive is selected), INTRQ must be enabled through tri-state and must be driven asserted or negated. If (nIEN == 1 || drive is not selected), INTRQ = 1'bz. When INTRQ asserted, drive must negate it within 400ns of negation of DIOR that reads STATUS register or within 400ns of negation of DIOW that writes the COMMAND register. When drive is selected by writing to Device/Head register and interrupt is pending, INTRQ must be asserted within 400ns of negation of DIOW that writes the Device/Head register. When drive is de-selected by writing to Device/Head register and interrupt is pending, INTRQ must be negated within 400ns of negation of DIOW that writes the Device/Head register.
IORDY:DDMARDY:DSTROBE	←	IORDY is negated by drive to extend host transfer cycle (read or write) for PIO modes 3 and above. DDMARDY→drive ready to receive UDMA data out bursts. Negated to pause. DSTROBE→drive signal from UDMA data in bursts. Data latched in host registers from DD[15:0] on both edges of DSTROBE. Drive stops generating DSTROBE edges to pause.
PDIAG:CBLID	—	$\overline{\text{PDIAG}}$→is asserted by drive 1 to indicate to drive 0 that it has completed diagnostics. CBLID→Host may sample CBLID after Power-ON or hardware reset is completed for all drives on the cable, to detect presence or absence of 80 conductor cable. If CBLID is detected as connected to ground then 80-conductor cable is present. If drive 1 is present, Host should issue IDENTIFY DEVICE or IDENTIFY PACKET DEVICE and use returned data to determine if drive is compliant with ATA-3 or subsequent standards. Drives complaint with ATA-3 or above, release PDIAG no later than after the first command following a Power-ON or hardware reset sequence.
RESET	←	RESET used by host to reset drive.
CSEL	—	CSEL negated, drive address is 0 CSEL asserted, drive address is 1

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 11-34 summarizes the different ATA standards.

MGT5100 is compliant with the latest officially published ANSI ATA-4 interface.

Table 11-34 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 11-35 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 ($\overline{\mathrm{CS}}$[1]$\overline{\mathrm{FX}}$, $\overline{\mathrm{CS}}$[3]$\overline{\mathrm{FX}}$, DA[2:0]) present on the physical ATA interface. Table 11-36 gives details.

Table 11-36 ATA Register Address/Chip Select Decoding

Address						Function	
System Address	CS[1]FX	CS[3]FX	DA[2]	DA[1]	DA[0]	READ ($\overline{\mathrm{DIOR}}$)	WRITE ($\overline{\mathrm{DIOW}}$)
						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

Table 11-36 ATA Register Address/Chip Select Decoding (continued)

Address						Function	
System Address	CS[1]FX	CS[3]FX	DA[2]	DA[1]	DA[0]	READ ($\overline{\text{DIOR}}$)	WRITE ($\overline{\text{DIOW}}$)
						Control Block Registers	
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.

NOTE:

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$\rightarrow${Cylinder High (0x01F5), Cylinder Low (0x01F4)}
- Head$\rightarrow$Drive/Head (0x01F6)
- Sector$\rightarrow$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 ($6136 \div 8 = 767$). The number of heads is then multiplied by the same number ($16 \times 8 = 128$). This fits well within the limits set by the BIOS and a larger disk is recognized for its true size ($767 \times 128 \times 63 \times 512$ = 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:

$$\text{LBA} \rightarrow (\text{Cylinder\#} \times \text{HeadCount} + \text{Head\#}) \times \text{SectorCount} + \text{Sector\#} - 1$$

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)}

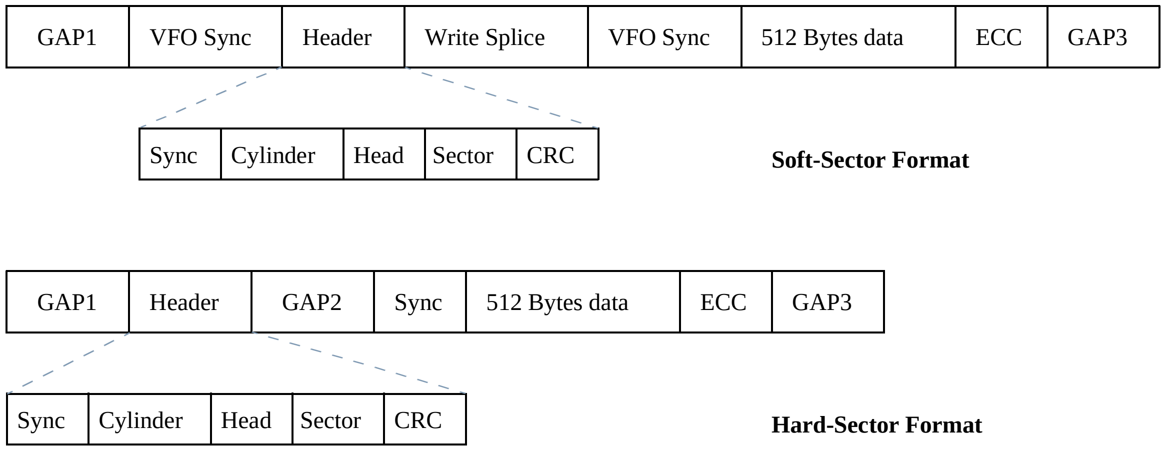


Figure 11-4 ATA Sector Format

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 11-35. Timing and sequence information are given in Appendix A.

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 512Bytes 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.

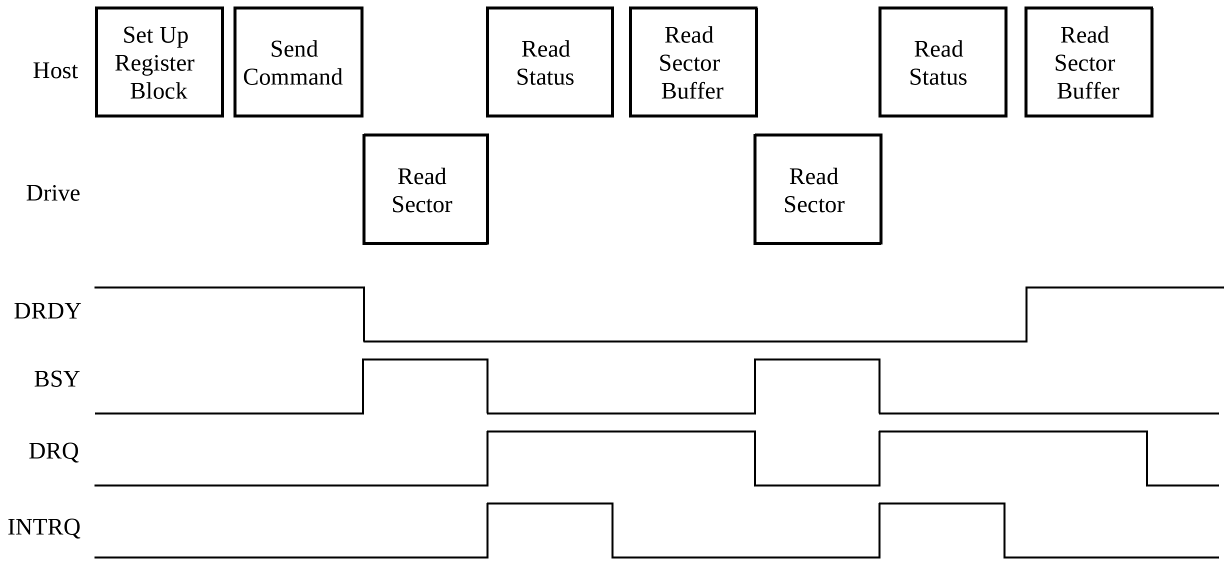


Figure 11-5 Timing Diagram—PIO Read Command (Class 1)

11.7.4.1.1 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 512Bytes 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.

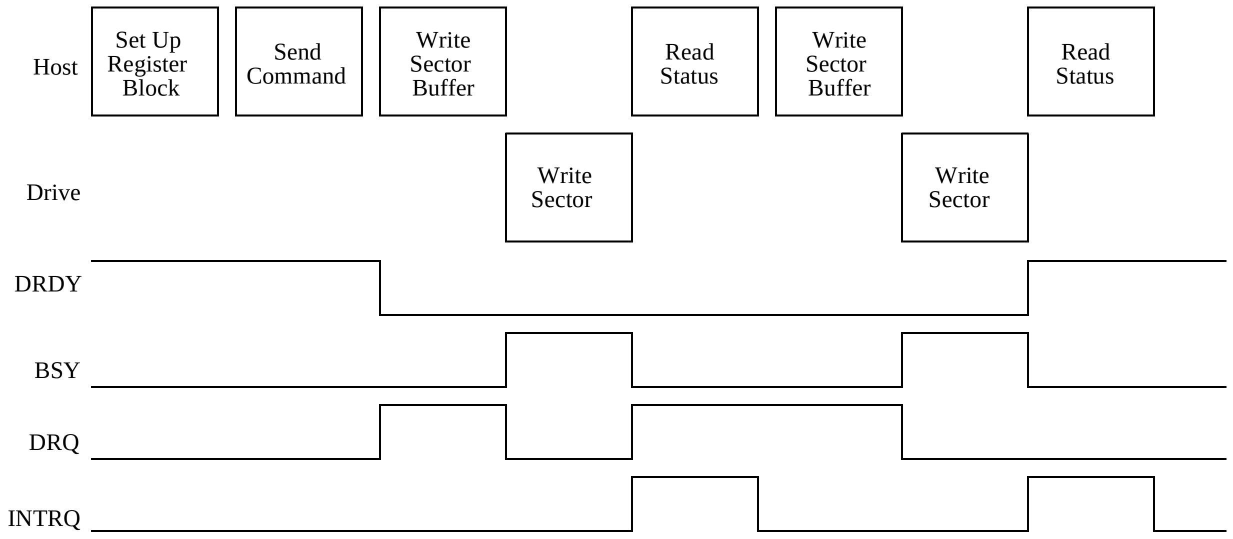


Figure 11-6 Timing Diagram—PIO Write Command (Class 2)

11.7.4.1.1 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.

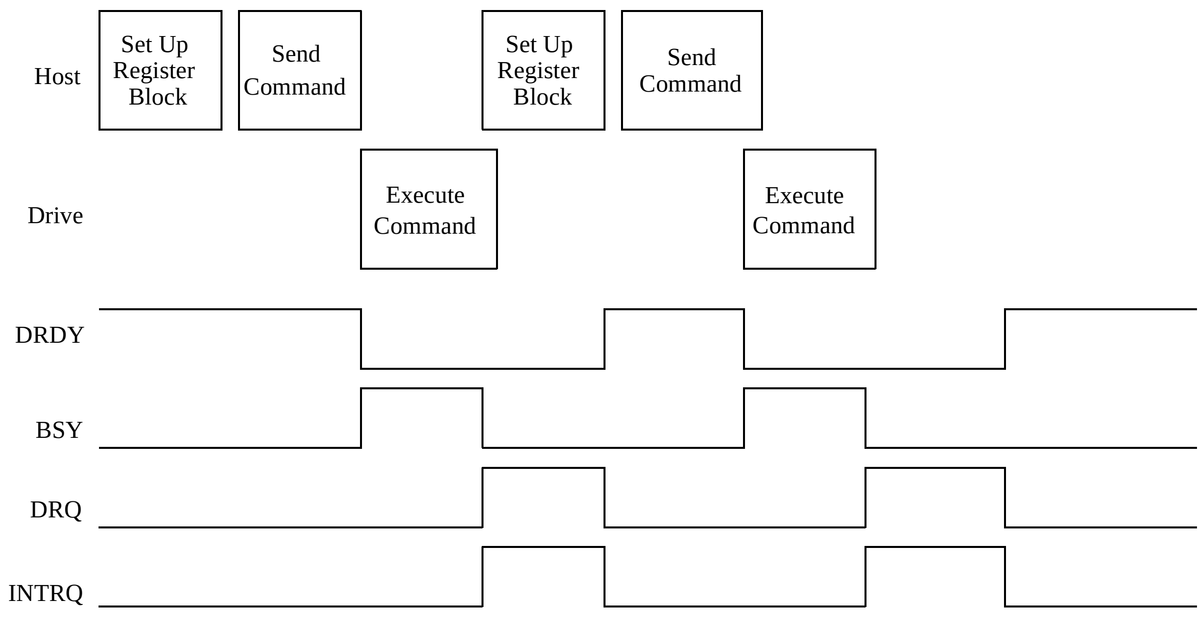


Figure 11-7 Timing Diagram—Non-Data Command (Class 3)

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 03h to features register to set transfer mode, based on value in sector count register.
3. Write command code EFh 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 11-37 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.

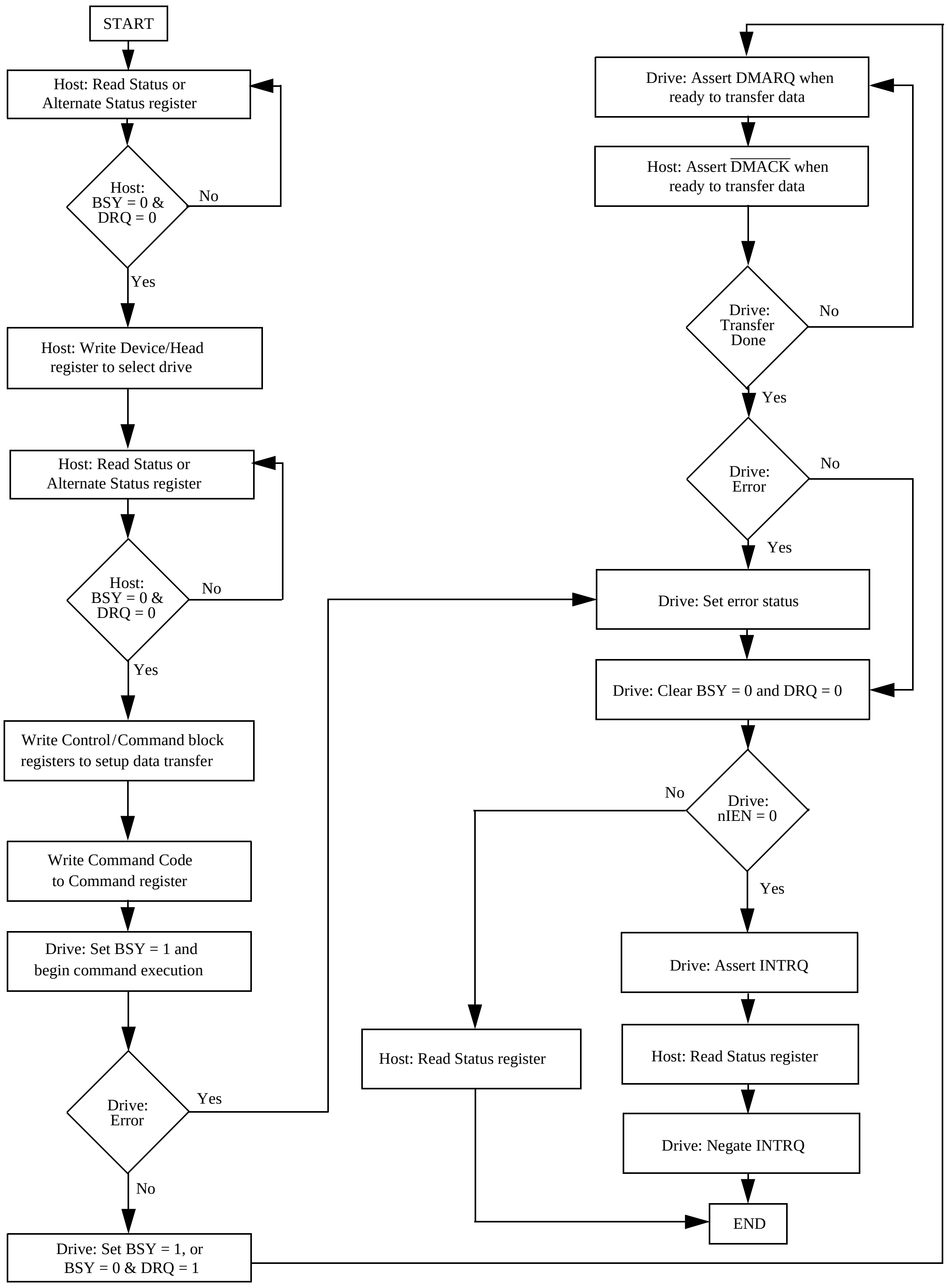

START
Host: Read Status or Alternate Status register
Host: BSY = 0 & DRQ = 0
No
Yes
Host: Write Device/Head register to select drive
Host: Read Status or Alternate Status register
Host: BSY = 0 & DRQ = 0
No
Yes
Write Control/Command block registers to setup data transfer
Write Command Code to Command register
Drive: Set BSY = 1 and begin command execution
Drive: Error
Yes
No
Drive: Set BSY = 1, or BSY = 0 & DRQ = 1
Drive: Assert DMARQ when ready to transfer data
Host: Assert $\overline{\text{DMACK}}$ when ready to transfer data
Drive: Transfer Done
No
Yes
Drive: Error
No
Yes
Drive: Set error status
Drive: Clear BSY = 0 and DRQ = 0
Drive: nIEN = 0
No
Yes
Host: Read Status register
Drive: Assert INTRQ
Host: Read Status register
Drive: Negate INTRQ
END

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.

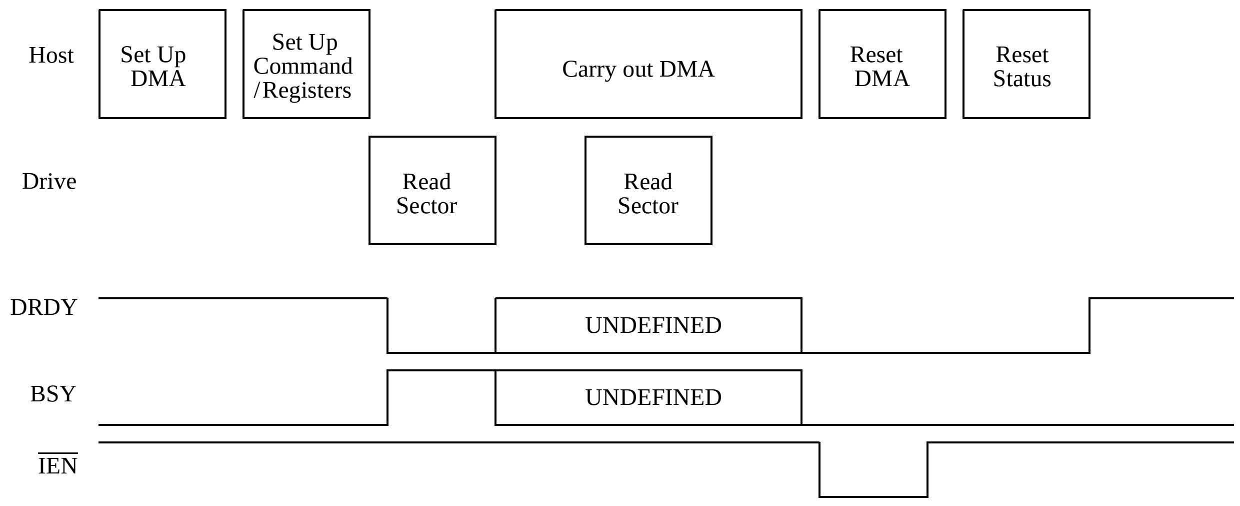


Figure 11-9 Timing Diagram—DMA Command (Class 4)

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 66MHz.

Table 11-38 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 11-38 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 25µs after power has stabilized within system specified tolerance. A signal assertion less than 20ns 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 MGT5100 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 5µs apart.

Figure 11-10 shows the Reset timing diagram. Table 11-39 gives timing characteristics.

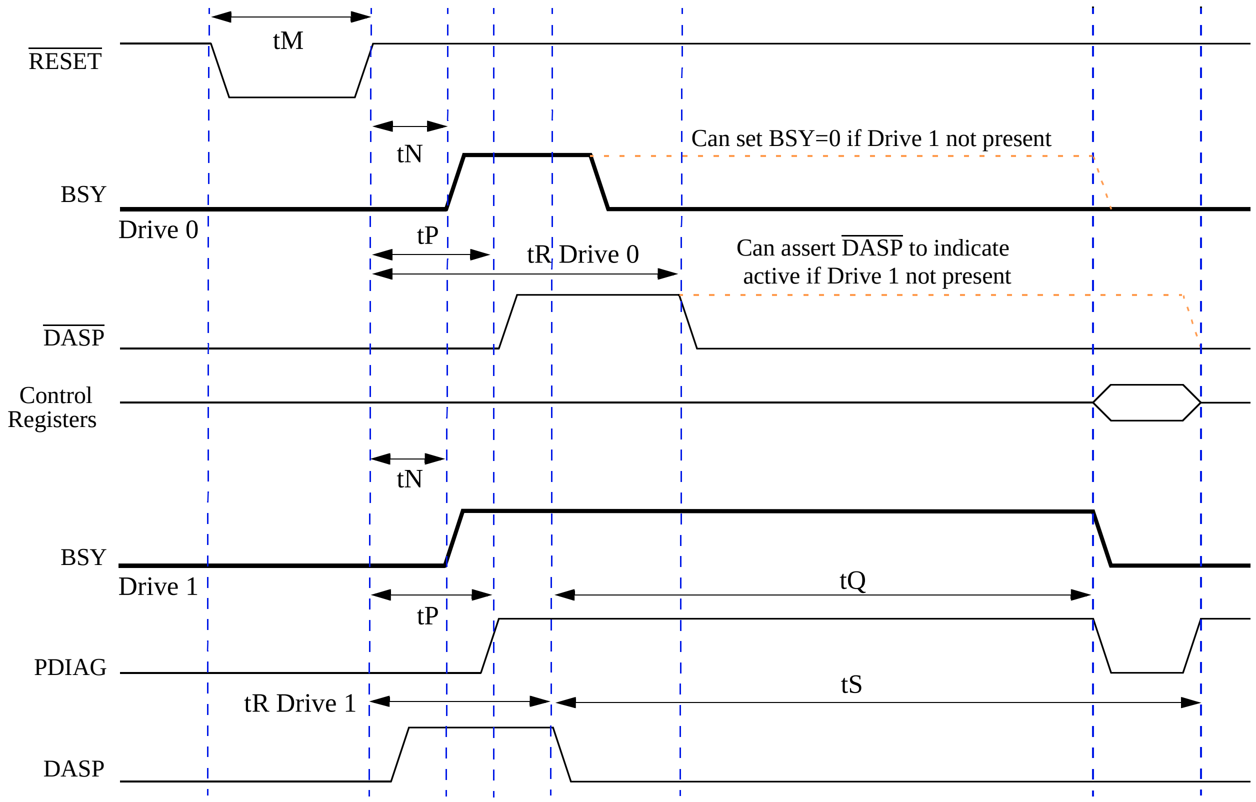

$\overline{\text{RESET}}$
tM
tN
Can set BSY=0 if Drive 1 not present
BSY
Drive 0
tP
tR Drive 0
Can assert $\overline{\text{DASP}}$ to indicate active if Drive 1 not present
$\overline{\text{DASP}}$
Control Registers
tN
BSY
Drive 1
tQ
tP
PDIAG
tR Drive 1
tS
DASP

Figure 11-10 Timing Diagram—Reset Timing

Table 11-39 Reset Timing Characteristics

Name	PIO Timing Parameter	Min/Max	Timing
tM	Reset pulse width	Min	25 µs
tN	Reset negated to BSY active setup	Max	400 ns
tP	Reset negated to $\overline{\text{DASP}}$ inactive setup	Max	1 ms
tQ	$\overline{\text{DASP}}$ active to $\overline{\text{PDIAG}}$ active setup	Max	30 s
tR	Drive 0—Reset negated to $\overline{\text{DASP}}$ active setup	Max	450 ms
	Drive 1—Reset negated to $\overline{\text{DASP}}$ active setup	Max	400 ms
tS	$\overline{\text{DASP}}$ active to $\overline{\text{PDIAG}}$ inactive setup	Max	30.5 s

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.

11.10 ATA Electrical Characteristics

The ATA interface requires external components to connect to an ATA-compliant drive due to electrical loading and noise considerations on the local bus.

The MGT5100 ATA characteristics are driven by PCI drivers. The ATA interface must be driven by level shifting drivers on the board. Proper termination series resistors are also needed on the board. The board must be designed for level shifters and proper termination resistors.

Table 11-40 gives the standard DC electrical characteristics. Table 11-41 gives AC electrical characteristics.

Table 11-40 DC Electrical Characteristics

Symbol	Description	Min	Max
I_{OL}[1]	Driver sink current	4 mA	—
I_{OH}[2]	Driver source current	400 µA	—
V_{IH}	Voltage input high	2.0 VDC	—
V_{IL}	Voltage input low	—	0.8 VDC
V_{OH}	[Voltage output high $(I)_{OH} = -400\,\mu A$]	2.4 VDC	—
V_{OL}	[Voltage output low $(I)_{OL} = 12\,mA$]	—	0.5 VDC

NOTE:
1. I_{OL} for DASP shall be 12 mA minimum to meet legacy timing and signal integrity. This is not driven by MGT5100.
2. I_{OH} value at 400 µA is insufficient in the case of DMARQ that is typically pulled low by a 5.6 KΩ resistor. This is not driven by MGT5100.

Table 11-41 AC Electrical Specifications

Symbol	Description	Min	Max
T_{RISE}	Rise time for any signal on AT interface. See Note 1.	5ns	—
T_{FALL}	Fall time for any signal on AT interface. See Note 1.	5ns	—
C_{IN}	Host input capacitance	—	25pF
C_{OUT}	Host output capacitance	—	25pF
C_{IN}	Device input capacitance	—	20pF
C_{OUT}	Device output capacitance	—	20pF
NOTE: 1. t_{RISE} and t_{FALL} are measured from 10–90% of full signal amplitude with a total capacitive load of 40pF.			

11.10.1 ATA Timing Diagrams

Figure 11-11 shows the PIO mode timing diagram. Table 11-42 gives the PIO mode timing specifications.

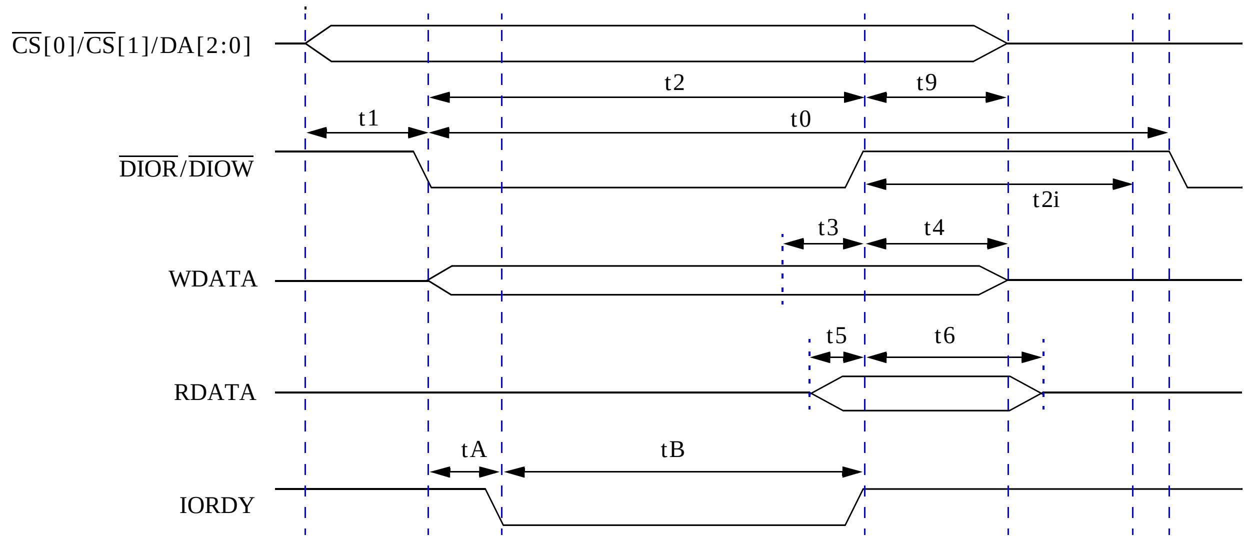

$\overline{CS}$[0]/$\overline{CS}$[1]/DA[2:0]
$\overline{DIOR}$/$\overline{DIOW}$
WDATA
RDATA
IORDY
t1
t2
t9
t0
t2i
t3
t4
t5
t6
tA
tB

Figure 11-11 Timing Diagram—PIO Mode

Table 11-42 PIO Mode Timing Specifications

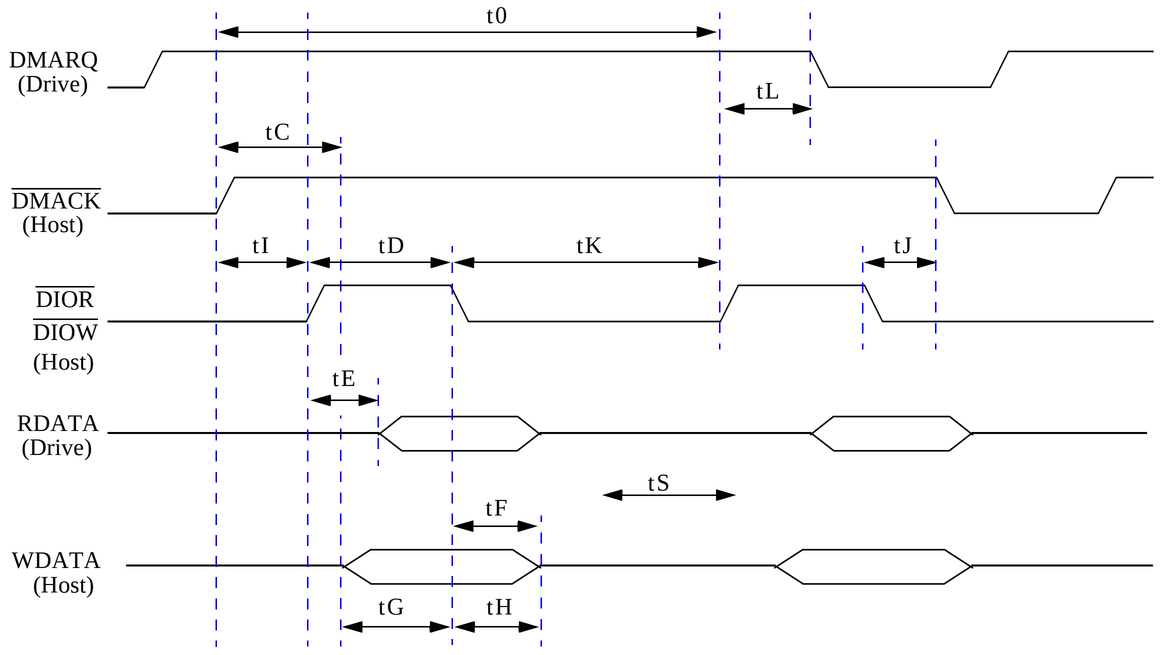
Name	PIO Timing Parameter	Min/ Max	Mode 0 (ns)	Mode 1 (ns)	Mode 2 (ns)	Mode 3 (ns)	Mode 4 (ns)	See Note 1
t0	Cycle Time	min	600	383	240	180	120	Yes
t1	Address valid to $\overline{DIOR}$/$\overline{DIOW}$ setup	min	70	50	30	30	25	Yes
t2	$\overline{DIOR}$/$\overline{DIOW}$ pulse width							Yes
	16-bit	min	165	125	100	80	70	
	8-bit	min	290	290	290	80	70	

Table 11-42 PIO Mode Timing Specifications (continued)

Name	PIO Timing Parameter	Min/ Max	Mode 0 (ns)	Mode 1 (ns)	Mode 2 (ns)	Mode 3 (ns)	Mode 4 (ns)	See Note 1
t2i	DIOR/DIOW recovery time	min	—	—	—	70	25	Yes
t3	DIOW data setup	min	60	45	30	30	20[2]	No
t4	DIOW data hold	min	30	20	15	10	10	Yes
t5	DIOR data setup	min	50	35	20	20	20	No
t6	DIOR data hold	min	5	5	5	5	5	No
t9	IOR/DIOW to address valid hold	min	20	15	10	10	10	No
tA	IORDY setup	max	35	35	35	35	35	No
tB	IORDY pulse width	max	1250	1250	1250	1250	1250	No

NOTE:
1. Implemented in MGT5100 (Yes, No).
2. t3 is not counted because data is guaranteed at the start of DIOW.

Figure 11-12 shows the Multiword DMA timing diagram. Table 11-43 gives the Multiword DMA timing specifications.



NOTE:
1. Regardless of the signal electrical properties, the direction of signal assertion is toward top of page; negation is toward bottom of page.

Figure 11-12 Timing Diagram—Multiword DMA

Table 11-43 Multiword DMA Timing Specifications

Name	Multiword DMA Timing Parameters	Min/Max	Mode 0 (ns)	Mode 1 (ns)	Mode 2 (ns)
t0	Cycle Time	min	480	150	120
tC	$\overline{\text{DMACK}}$ to DMARQ delay	max	—	—	—
tD	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ pulse width (16-bit)	min	215	80	70
tE	$\overline{\text{DIOR}}$ data access	max	150	60	50
tG	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ data setup	min	100	30	20
tF	$\overline{\text{DIOR}}$ data hold	min	5	5	5
tH	$\overline{\text{DIOW}}$ data hold	min	20	15	10
tI	DMACK to $\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ setup	min	0	0	0
tJ	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ to $\overline{\text{DMACK}}$ hold	min	20	5	5
tKr	$\overline{\text{DIOR}}$ negated pulse width	min	50	50	25
tKw	$\overline{\text{DIOW}}$ negated pulse width	min	215	50	25
tLr	$\overline{\text{DIOR}}$ to DMARQ delay	max	120	40	35
tLw	$\overline{\text{DIOW}}$ to DMARQ delay	max	40	40	35

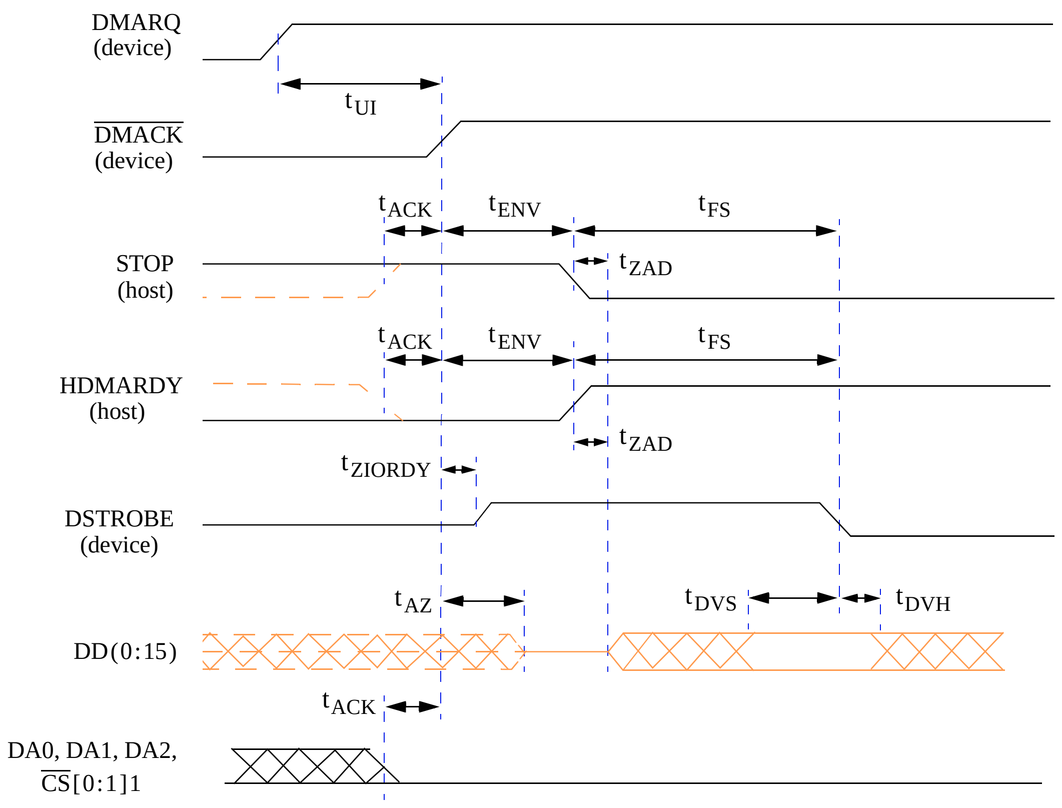


Figure 11-13 Timing Diagram—Initiating an Ultra DMA Data In Burst

Table 11-44 Ultra DMA Timing Specification

Name	MODE 0 (ns)		MODE 1 (ns)		MODE 2 (ns)		Comment
	Min	Max	Min	Max	Min	Max	
(t)$_{\text{2CYC}}$	240	—	160	—	120	—	Typical sustained average two cycle time.
(t)$_{\text{CYC}}$	114	—	75	—	55	—	Cycle time allowing for asymmetry and clock variations from STROBE edge to STROBE edge
(t)$_{\text{2CYC}}$	235	—	156	—	117	—	Two-cycle time allowing from clock variations, from rising edge to next rising edge or from falling edge to next falling edge of STROBE.
(t)$_{\text{DS}}$	15	—	10	—	7	—	Data setup time at recipient.
(t)$_{\text{DH}}$	5	—	5	—	5	—	Data hold time at recipient.
(t)$_{\text{DVS}}$	70	—	48	—	34	—	Data valid setup time at sender, to STROBE edge.
(t)$_{\text{DVH}}$	6	—	6	—	6	—	Data valid hold time at sender, from STROBE edge.
(t)$_{\text{FS}}$	0	230	0	200	0	170	First STROBE time for drive to first negate DSTROBE from STOP during a data in burst.
(t)$_{\text{LI}}$	0	150	0	150	0	150	Limited Interlock time. See Notes 1 and 2.
(t)$_{\text{MLI}}$	20	—	20	—	20	—	Interlock time with minimum. See Notes 1 and 2.
(t)$_{\text{UI}}$	0	—	0	—	0	—	Unlimited interlock time. See Notes 1 and 2.
(t)$_{\text{AZ}}$	—	10	—	10	—	10	Maximum time allowed for output drivers to release from being asserted or negated
(t)$_{\text{ZAH}}$	20	—	20	—	20	—	Minimum delay time required for output drivers to assert or negate from released state
(t)$_{\text{ZAD}}$	0	—	0	—	0	—	
(t)$_{\text{ENV}}$	20	70	20	70	20	70	Envelope time—from $\overline{\text{DMACK}}$ to STOP and $\overline{\text{HDMARDY}}$ during data out burst initiation.
(t)$_{\text{SR}}$	—	50	—	30	—	20	STROBE to $\overline{\text{DMARDY}}$ time, if $\overline{\text{DMARDY}}$ is negated before this long after STROBE edge, the recipient receives no more than one additional data word.
(t)$_{\text{RFS}}$	—	75	—	60	—	50	Ready-to-Final STROBE time—no $\overline{\text{STROBE}}$ edges are sent this long after negation of $\overline{\text{DMARDY}}$.
(t)$_{\text{RP}}$	160	—	125	—	100	—	Ready-to-Pause time—the time recipient waits to initiate pause after negating $\overline{\text{DMARDY}}$.
(t)$_{\text{IORDYZ}}$	—	20	—	20	—	20	Pull-up time before allowing IORDY to be released.
(t)$_{\text{ZIORDY}}$	0	—	0	—	0	—	Minimum time drive waits before driving IORDY
(t)$_{\text{ACK}}$	20	—	20	—	20	—	Setup and hold times for $\overline{\text{DMACK}}$, before assertion or negation.
(t)$_{\text{SS}}$	50	—	50	—	50	—	Time from STROBE edge to negation of DMARQ or assertion of STOP, when sender terminates a burst.

Table 11-44 Ultra DMA Timing Specification (continued)

Name	MODE 0 (ns)		MODE 1 (ns)		MODE 2 (ns)		Comment
	Min	Max	Min	Max	Min	Max	

NOTE:

1. t_{UI}, t_{MLI}, t_{LI} indicate sender-to-recipient or recipient-to-sender interlocks. That is, one agent (either sender or recipient) is waiting for the other agent to respond with a signal before proceeding.
   - t_{UI} is an unlimited interlock that has no maximum time value.
   - t_{MLI} is a limited time-out that has a defined minimum.
   - t_{LI} is a limited time-out that has a defined maximum.
2. All timing parameters are measured at the connector of the drive to which the parameter applies. For example, the sender shall stop generating STROBE edges t_{RFS} after negation of DMARDY. Both STROBE and DMARDY timing measurements are taken at the connector of the sender. Even though the sender stops generating STROBE edges. The receiver may receive additional STROBE edges due to propagation delay. All timing measurement switching points (low to high and high to low) are taken at 1.5V.

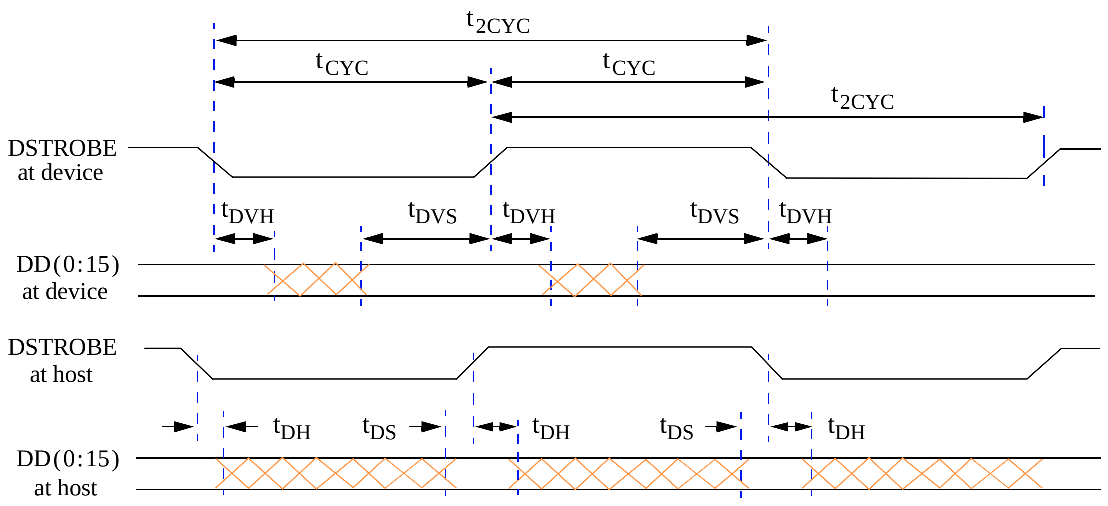


Figure 11-14 Timing Diagram—Sustained Ultra DMA Data In Burst

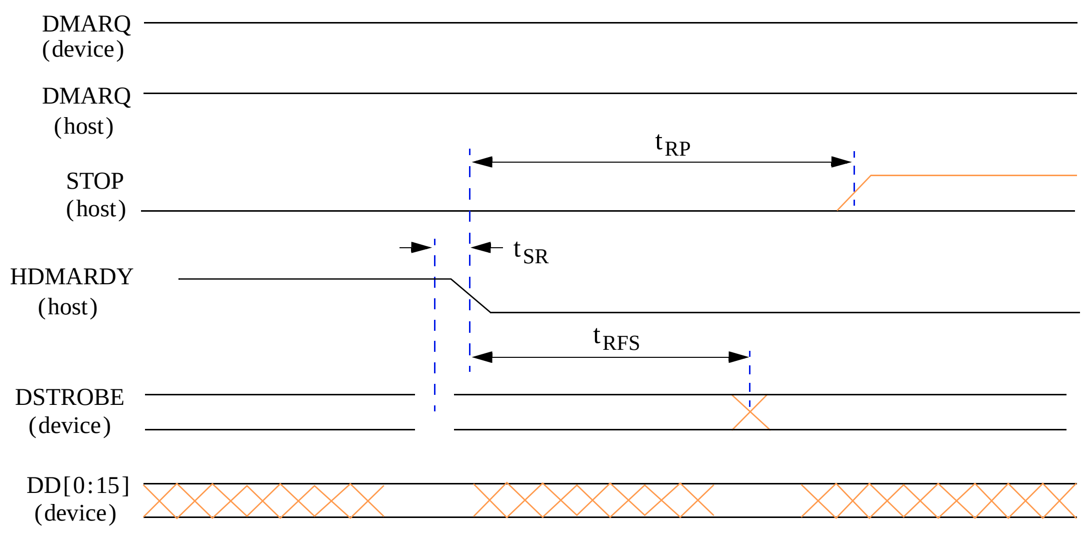


Figure 11-15 Timing Diagram—Host Pausing an Ultra DMA Data In Burst

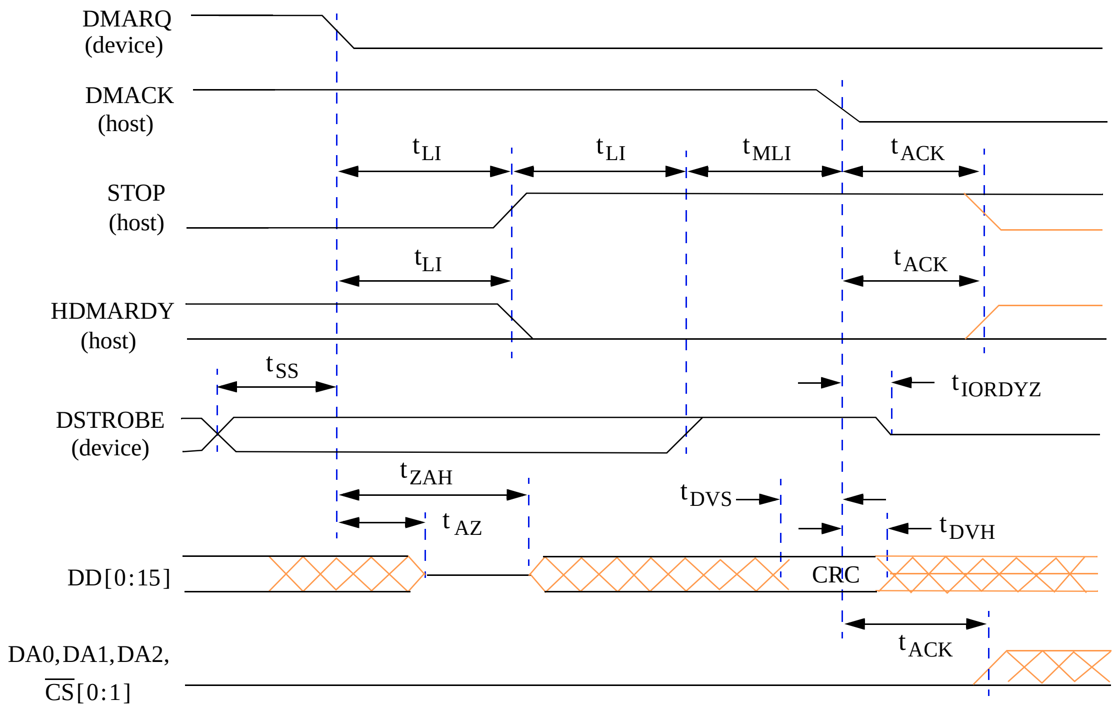


Figure 11-16 Timing Diagram—Drive Terminating Ultra DMA Data In Burst

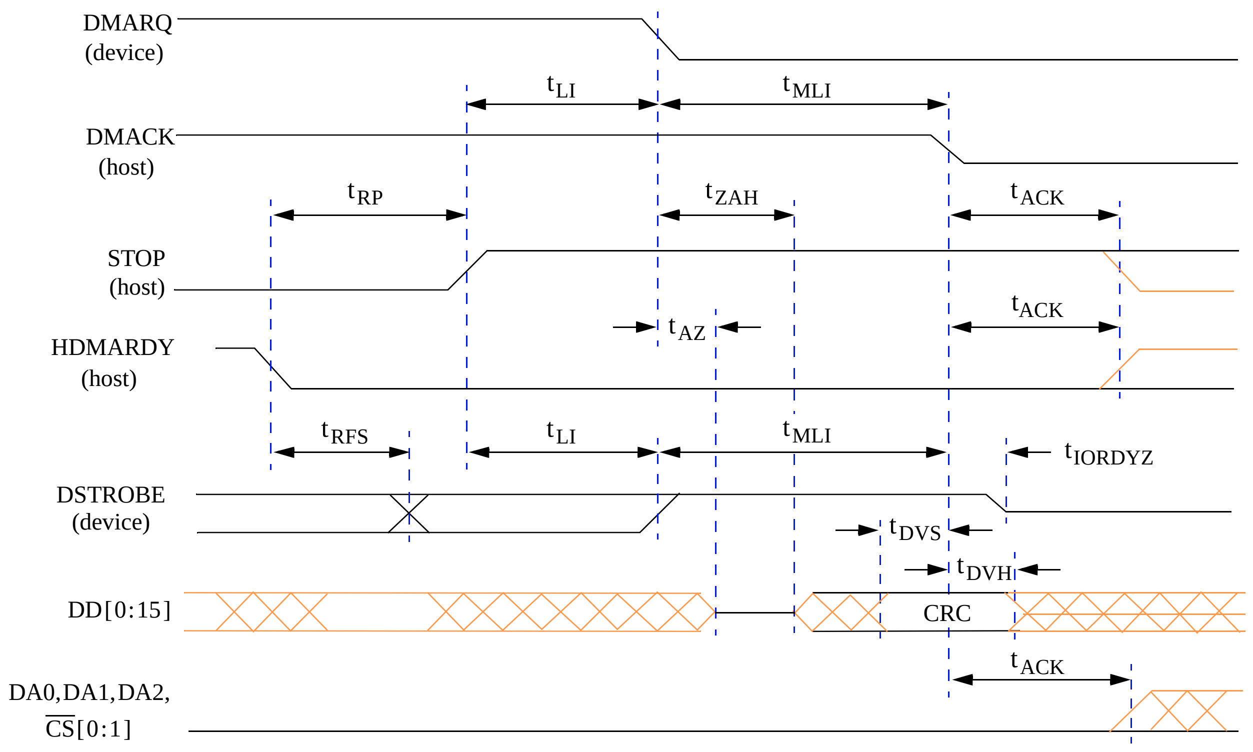


Figure 11-17 Timing Diagram—Host Terminating Ultra DMA Data In Burst

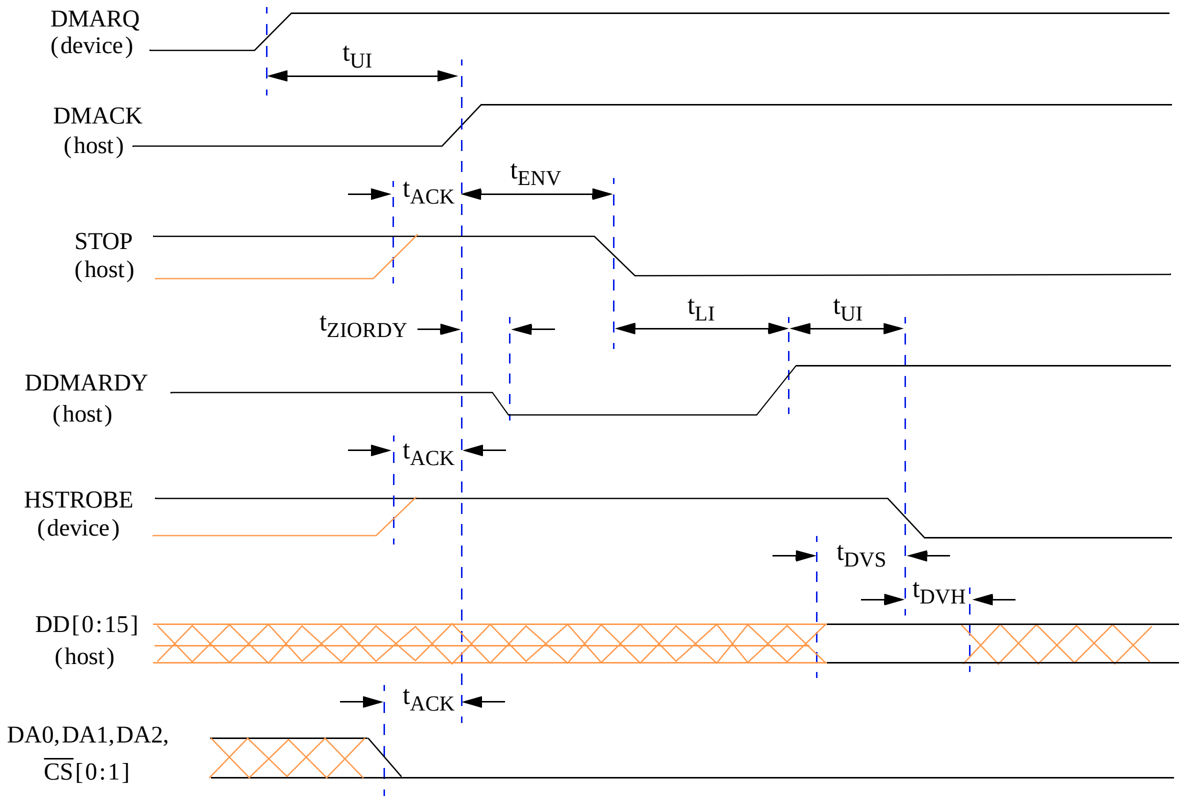


Figure 11-18 Timing Diagram—Initiating an Ultra DMA Data Out Burst

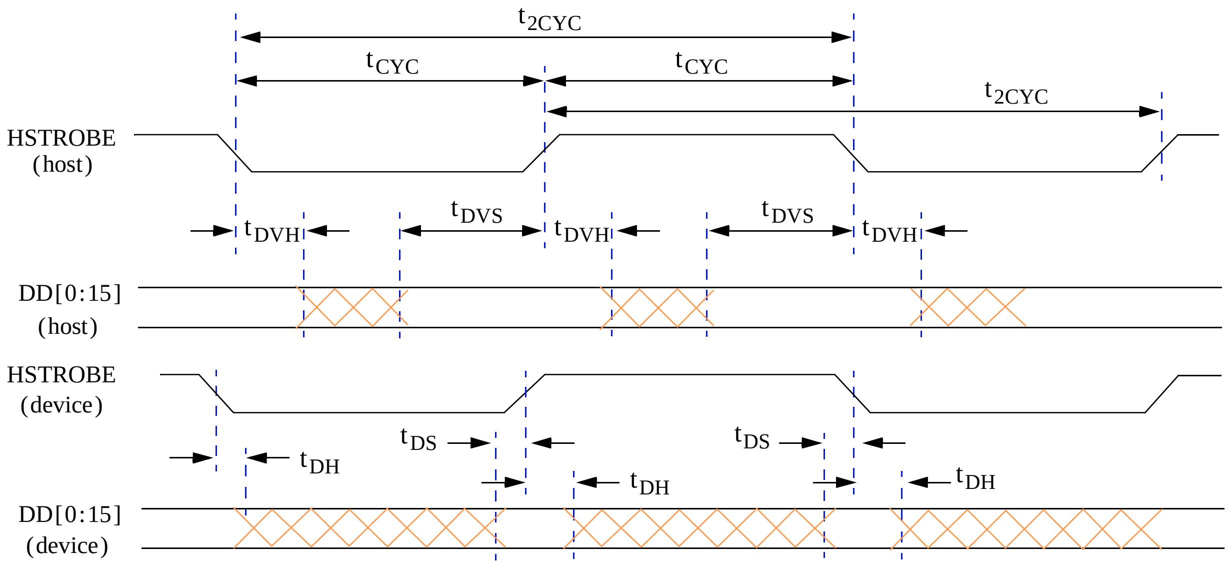


Figure 11-19 Timing Diagram—Sustained Ultra DMA Data Out Burst

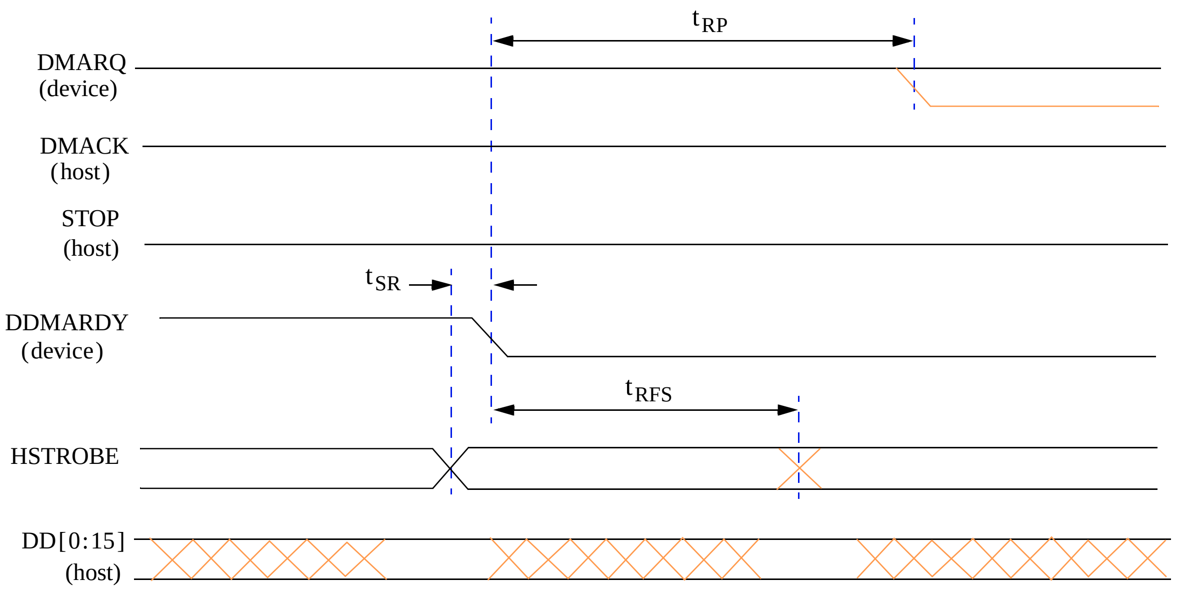


Figure 11-20 Timing Diagram—Drive Pausing an Ultra DMA Data Out Burst

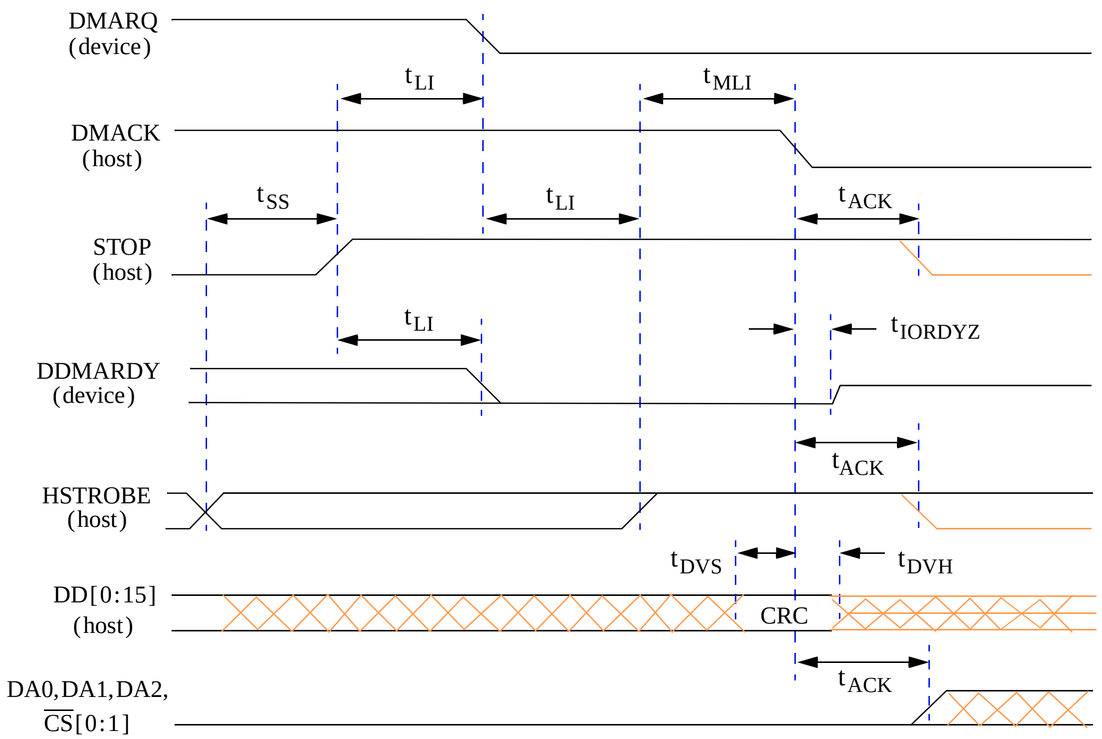


Figure 11-21 Timing Diagram—Host Terminating Ultra DMA Data Out Burst

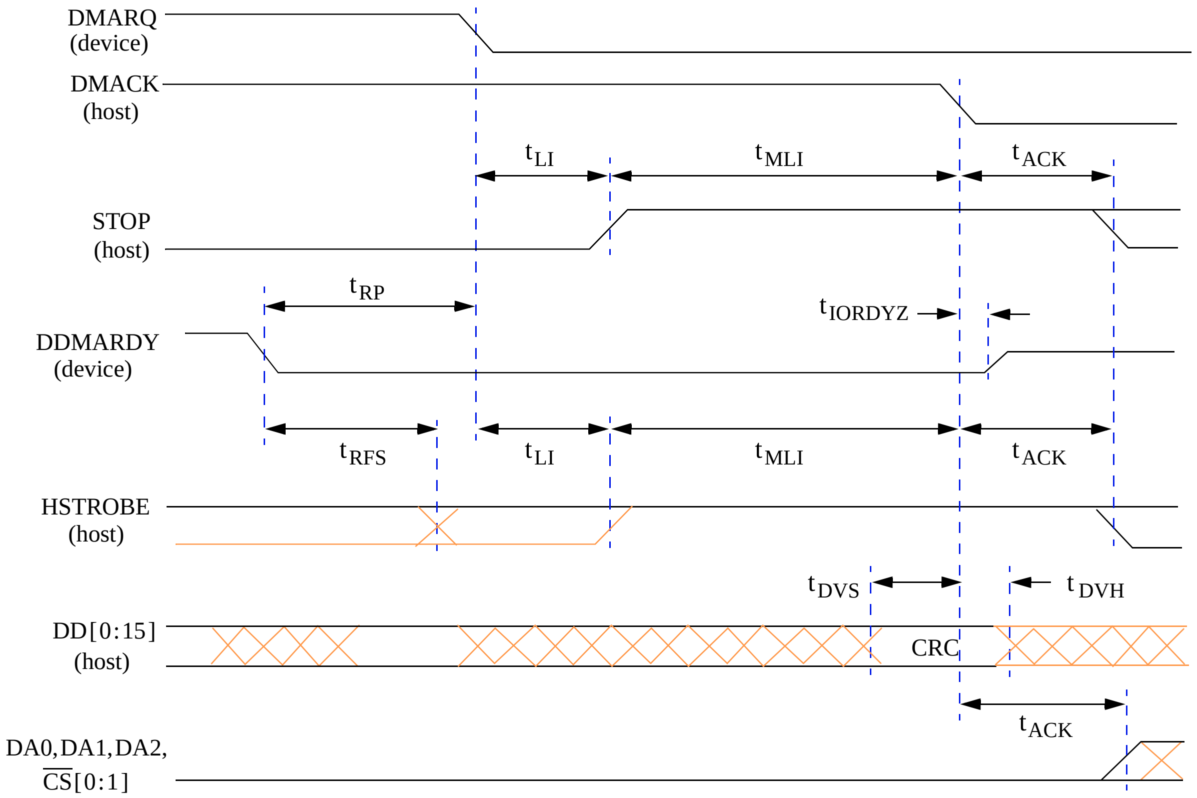


Figure 11-22 Timing Diagram—Drive Terminating Ultra DMA Data Out Burst

SECTION 12
UNIVERSAL SERIAL BUS (USB)

12.1 Overview

The following sections are contained in this document:

- Data Transfer Types
- Host Controller Interface
- Host Control (HC) Operational Registers, includes:
  - Control and Status Partition—MBAR + 0x1000
  - Memory Pointer Partition—MBAR + 0x1000
  - Frame Counter Partition—MBAR + 0x1000
  - Root Hub Partition—MBAR + 0x1000

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

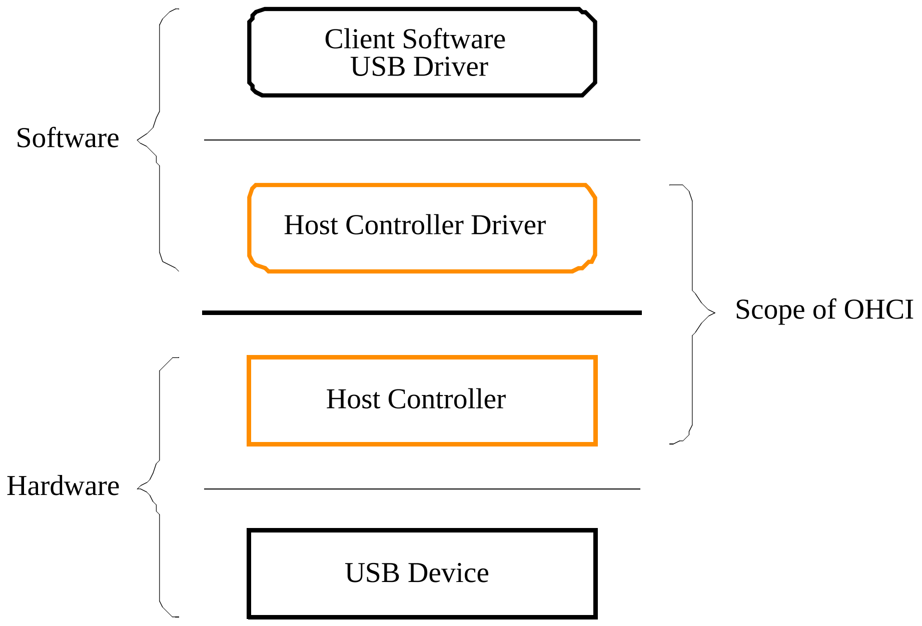


Figure 12-1 USB Focus Areas

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 non-periodic. 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.

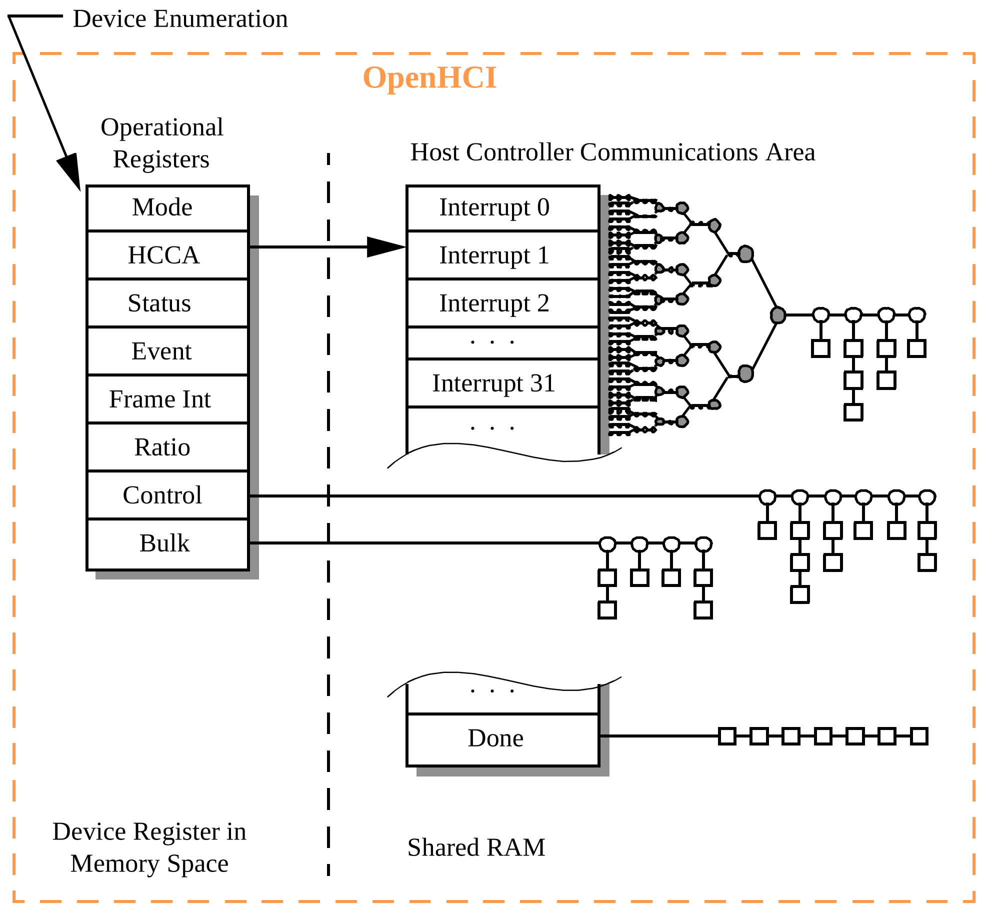


Figure 12-2 Communication Channels

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 8192Bytes 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.

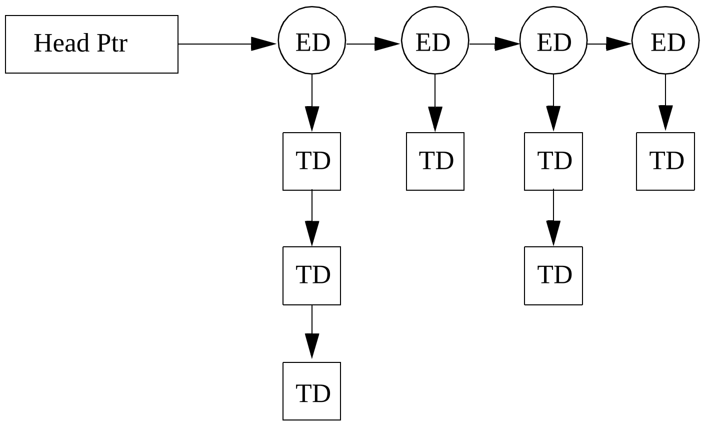


Figure 12-3 Typical List Structure

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.

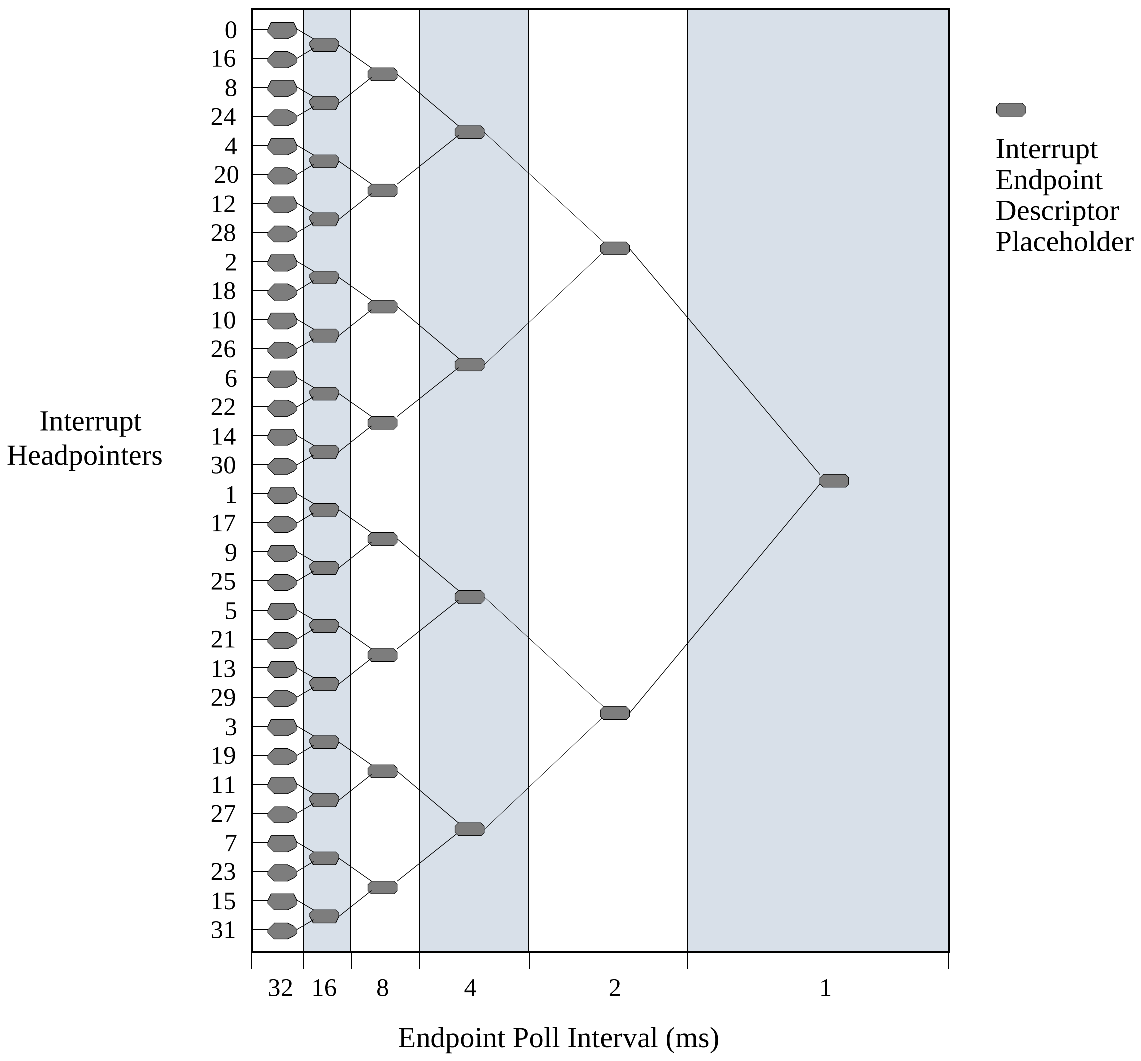


Figure 12-4 Interrupt ED Structure

Figure 12-5 shows a sample interrupt endpoint schedule. The schedule shows:

- two endpoint descriptors at a 1 ms poll interval
- two endpoint descriptors at a 2 ms poll interval
- one endpoint descriptor at a 4 ms poll interval
- two endpoint descriptors at an 8 ms poll interval
- two endpoint descriptors at a 16 ms poll interval
- two endpoint descriptors at a 32 ms poll interval.

NOTE: Unused interrupt endpoint placeholders are bypassed and the link is connected to the next available endpoint in the hierarchy.

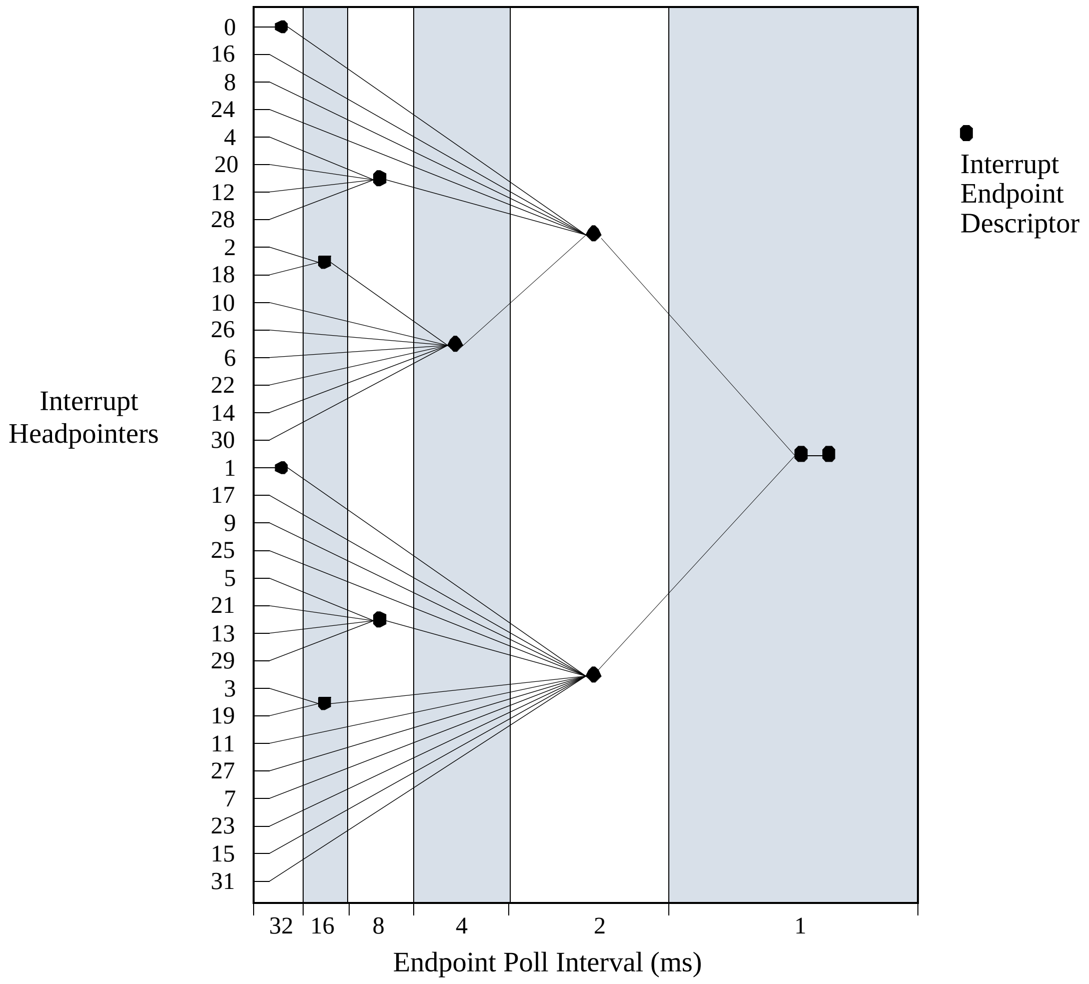


Figure 12-5 Sample Interrupt Endpoint Schedule

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/$\overline{\text{W}}$ 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 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:

- HC Revision Register (1000)—HcRevision
- HC Control Register (1004)—HcControl
- HC Command Status Register (1008)—HcCommandStatus
- HC Interrupt Status Register (100C)—HcInterruptStatus
- HC Interrupt Enable Register (1010)—HcInterruptEnable
- HC Interrupt Disable Register (1014)—HcInterruptDisable

12.4.1.1 HC Revision (1000)—HcRevision

Table 12-1 HC Revision Register (1000)—HcRevision

	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	0	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.

12.4.1.2 HC Control (1004)—HcControl

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 HC Control Register (1004)—HcControl

	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 1 ms 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.

Bits	Name	Description
26	BLE	Bulk List Enable—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	Control List Enable—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	Isochronous Enable—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.
29	PLE	Periodic List Enable—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	Control Bulk Service Ratio—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

12.4.1.3 HC Command Status (1008)—HcCommandStatus

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 Scheduling Overrun Count 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 Scheduling Overrun field in HcInterruptStatus register.

Table 12-3 HC Command Status Register (1008)—HcCommandStatus

	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.
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.

12.4.1.4 HC Interrupt Status (100C)—HcInterruptStatus

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 HC Interrupt Status Register (100C)—HcInterruptStatus

	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
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.

12.4.1.5 HC Interrupt Enable (1010)—HcInterruptEnable

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 HC Interrupt Enable Register (1010)—HcInterruptEnable

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	MIE	OC	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									RHSC	FNO	UE	RD	SF	WDH	SO
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	OwnershipChange 0=Ignore 1=Enable interrupt generation due to ownership change
2:24	—	Reserved
25	RHSC	RootHubStatusChange 0=Ignore 1=Enable interrupt generation due to root hub status change.
26	FNO	FrameNumberOverflow 0=Ignore 1=Enable interrupt generation due to frame number overflow.
27	UE	UnrecoverableError 0=Ignore 1=Enable interrupt generation due to unrecoverable error.

Bits	Name	Description
28	RD	Resume Detected 0 = Ignore 1 = Enable interrupt generation due to resume detect.
29	SF	Start of Frame 0 = Ignore 1 = Enable interrupt generation due to start of frame.
30	WDH	Writeback Done Head 0 = Ignore 1 = Enable interrupt generation due to HcDoneHead writeback.
31	SO	Scheduling Overrun 0 = Ignore 1 = Enable interrupt generation due to scheduling overrun.

12.4.1.6 HC Interrupt Disable (1014)—HcInterruptDisable

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 HC Interrupt Disable Register (1014)—HcInterruptDisable

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIE	OC	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									RHSC	FNO	UE	RD	SF	WDH	SO
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 0 = Ignore 1 = Disable interrupt generation due to Ownership Change
2:24	—	Reserved

Bits	Name	Description
25	RHSC	Root Hub Status Change 0=Ignore 1=Disable interrupt generation due to root hub status change.
26	FNO	Frame Number Overflow 0=Ignore 1=Disable interrupt generation due to frame number overflow.
27	UE	Unrecoverable Error 0=Ignore 1=Disable interrupt generation due to unrecoverable error.
28	RD	Resume Detected 0=Ignore 1=Disable interrupt generation due to resume detect.
29	SF	Startof Frame 0=Ignore 1=Disable interrupt generation due to start of frame.
30	WDH	Writeback Done Head 0=Ignore 1=Disable interrupt generation due to Hc Done Head writeback.
31	SO	Scheduling Overrun 0=Ignore 1=Disable interrupt generation due to scheduling overrun.

12.4.2 Memory Pointer Partition—MBAR + 0x1000

This HC partition uses 7 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:

- HC Communication Register (1018)—HcHCCA
- HC Period Current ED Register (101C)—HcPeriodCurrentED
- HC Control Head ED Register (1020)—HcControlHeadED
- HC Control Current ED Register (1024) HcControlCurrentED
- HC First Bulk Head List ED Register (1028)—HcBulkHeadED
- HC Bulk List Current ED Register (102C)—HcBulkCurrentED
- HC Last Completed Transfer Register (1030)—HcDoneHead

12.4.2.1 HC Communication (1018)—HcHCCA

The HC Communication register contains the HCCA physical address. 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 256Bytes. 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 HC Communication Register (1018)—HcHCCA

	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

12.4.2.2 HC Period Current ED (101C)—HcPeriodCurrentED

The HC Period Current Endpoint Descriptor (ED) register contains the physical address of the current isochronous or interrupt endpoint descriptor.

Table 12-8 HC Period Current ED Register (101C)—HcPeriodCurrentED

	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

12.4.2.3 HC Control Head ED (1020)—HcControlHeadED

The HC Control Head Endpoint Descriptor register contains the physical address of the first endpoint descriptor of the Control list.

Table 12-9 HC Control Head ED Register (1020)—HcControlHeadED

	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

12.4.2.4 HC Control Current ED (1024) HcControlCurrentED

The HC Control Current Endpoint Descriptor register contains the physical address of the current control list endpoint descriptor.

Table 12-10 HC Control Current ED Register (1024) HcControlCurrentED

	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

12.4.2.5 HC First Bulk Head List ED (1028)—HcBulkHeadED

The HC Bulk Head Endpoint Descriptor register contains the physical address of the first bulk list endpoint descriptor.

Table 12-11 HC First Bulk Head List ED Register (1028)—HcBulkHeadED

	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

12.4.2.6 HC Bulk List Current ED (102C)—HcBulkCurrentED

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 HC Bulk List Current ED Register (102C)—HcBulkCurrentED

	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 HcControl ControlListFilled. • 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

12.4.2.7 HC Last Completed Transfer (1030)—HcDoneHead

The HC Last Completed Transfer Descriptor 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 HC Last Completed Transfer Register (1030)—HcDoneHead

	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

12.4.3 Frame Counter Partition—MBAR + 0x1000

This HC partition uses 5 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:

- HC Frame Interval Register (1034)—HcFmInterval
- HC Frame Bit-time Remaining Register (1038)—HcFmRemaining
- HC Timing Reference Register (103C)—HcFmNumber
- HC Start Processing Periodic List Register (1040)—HcPeriodicStart
- HC Commit Transfer Register (1044)—HcLSThreshold

12.4.3.1 HC Frame Interval (1034)—HcFmInterval

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 HC Frame Interval Register (1034)—HcFmInterval

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

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

Bits	Name	Description
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.

12.4.3.2 HC Frame Bit-time Remaining (1038)—HcFmRemaining

This register is a 14-bit count-down counter containing the remaining current Frame bit-time.

Table 12-15 HC Frame Bit-time Remaining Register (1038)—HcFmRemaining

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	FRT	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		FR													
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.

12.4.3.3 HC Timing Reference (103C)—HcFmNumber

The HC Timing Reference 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 HC Timing Reference Register (103C)—HcFmNumber

	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

12.4.3.4 HC Start Processing Periodic List (1040)—HcPeriodicStart

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 HC Start Processing Periodic List Register (1040)—HcPeriodicStart

	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.

12.4.3.5 HC Commit Transfer (1044)—HcLSThreshold

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 HC Commit Transfer Register (1044)—HcLSThreshold

	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	0	0	0	0	0	0	0	0	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.

12.4.4 Root Hub Partition—MBAR + 0x1000

This HC partition uses 4 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:

- HC Root Hub Descriptor A Register (1048)—HcRhDescriptorA
- HC Root Hub Descriptor B Register (104C)—HcRhDescriptorB
- HC Root Hub Status Register (1050)—HcRhStatus
- HC RH Port Status Register (1054)—HcRhPortStatus[1:NDP]

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.

12.4.4.1 HC Root Hub Descriptor A (1048)—HcRhDescriptorA

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 HC Root Hub Descriptor A Register (1048)—HcRhDescriptorA

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	POTPGT								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			NOCP	OCPM	DT	NPS	PSM	NDP							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	POTPGT	PowerOnToPowerGoodTime—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	NoOverCurrentProtection—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.

Bits	Name	Description
20	OCPM	OverCurrentProtectionMode—describes how the Root Hub port overcurrent status is reported. At reset, OCPM should reflect the same mode as PowerSwitchingMode. OCPM is valid only if NoOverCurrentProtection is cleared. 0 = Overcurrent status is reported collectively for all downstream ports. 1 = Overcurrent status is reported on a per-port basis.
21	DT	DeviceType—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	NoPowerSwitching—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	PowerSwitchingMode—specifies how the root hub port power switching is controlled. PSM is implementation-specific and is only valid if the NoPowerSwitching 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 PortPowerControlMask bit is set, port responds only to port power commands (Set/ClearPortPower). ¤ If port mask is cleared, port is controlled only by the global power switch (Set/ClearGlobalPower).
24:31	NDP	NumberDownstreamPorts—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.

12.4.4.2 HC Root Hub Descriptor B (104C)—HcRhDescriptorB

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 HC Root Hub Descriptor B Register (104C)—HcRhDescriptorB

	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

12.4.4.3 HC Root Hub Status (1050)—HcRhStatus

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 HC Root Hub Status Register (1050)—HcRhStatus

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRWE	Reserved													OCIC	LPSC
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	DRWE	Reserved													OCI	LPS
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0	CRWE	ClearRemoteWakeUpEnable (write) • Writing 1 clears DRWE. • Writing 0 has no effect.
1:13	—	Reserved

Bits	Name	Description
14	OCIC	OverCurrentIndicatorChange—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	LocalPowerStatusChange (read)—Root Hub does not support the local power status feature. Thus, this bit is always read as 0. SetGlobalPower (write) • In global power mode (PSM=0), LPSC is written to 1 to turn on power to all ports (clear PortPowerStatus). • In per-port power mode, LPSC sets PortPowerStatus only on ports whose PPCM bit is not set. Writing 0 has no effect.
16	DRWE	DeviceRemoteWakeUpEnable (write)—enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote WakeUp event. 1 = ConnectStatusChange is a remote WakeUp event. SetRemoteWakeUpEnable (read). 1 = Sets DRWE. 0 = Has no effect.
17:29	—	Reserved
30	OCI	OverCurrentIndicator—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	LocalPowerStatus—Root Hub does not support the local power status feature. This bit is always read as 0 (write) ClearGlobalPower. In global power mode (PSM=0), bit is written to 1 to turn off power to all ports (clear PortPowerStatus). In per-port power mode, bit clears PortPowerStatus only on ports whose PPCM bit is not set. Writing 0 has no effect.

12.4.4.4 HC RH Port Status (1054)—HcRhPortStatus[1:NDP]

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.

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 HC RH Port Status Register (1054)—HcRhPortStatus[1:NDP]

	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

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

Bits	Name	Description
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 over-current 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]).

SECTION 13 SMARTCOMM/SMARTDMA

13.1 Overview

The following sections are contained in this document:

- SmartComm Functional Description
- SmartComm DMA Registers—MBAR+0x1200
- On-Chip SRAM
- SmartComm Timer Registers (SCTMR)—MBAR+0x0400

SmartComm provides an efficient, integrated approach to gathering and manipulating data sets from a broad range of communication interfaces. SmartComm consists of the:

- SmartDMA (SDMA) module, with interfaces to:
  - peripherals using the CommBus
  - the processor using an IP bus
  - other chip resources using an XL bus
- a defined set of communication-oriented peripherals
- local buffer memory
- standard bus interfaces

SDMA is a sophisticated, user-programmable DMA engine that interprets a series of C-language "for-loop" style descriptors to perform a user-configurable series of data movements and manipulation. To enhance performance and minimize impact on the external bus bandwidth, an 8KByte SmartComm dedicated RAM array is provided. Using the Multiply-and-Accumulate (MAC) unit and a general purpose logic unit, the user has complete control of operations on data occurring while data is moved from source(s) to destination.

The MGT5100 SmartComm I/O subsystem is based on the SmartComm device being developed by ISD for Vantage. For the MGT5100, SmartComm consists of SDMA and the following peripheral functions:

- 10/100 Fast Ethernet Controller (FEC)
- Three full-duplex Programmable Serial Controllers (PSCs) supporting:
  - RS-232 connection to a full function external modem (V.34 and V.90)
  - 1200-baud POTS modem
  - Standard UART interface to a terminal/computer for debug support
  - Interface to an external CODEC for soft modem support
  - Digital interface to an external audio CODEC ’97 (AC97) controller
- An IR Controller supporting IrDA data and control capabilities
- An I^{2}C interface

Many of the serial/parallel port pins serve multiple functions, which allows flexibility in optimizing the system to meet a specific set of integration requirements. For a description of the pin multiplexing scheme and which functions are supported, refer to Section 2, Signal Descriptions.

Other peripheral functions are included in MGT5100, but are not directly supported by the SmartDMA. These peripherals include:

- A Serial Peripheral Interface (SPI), which:
  - supports a 6.25MHz rate as a master
  - supports a 12.5MHz rate as a slave
- Basic IR Controller
- USB master/hub controller

13.2 SmartComm Functional Description

The SmartComm I/O subsystem consists of the following:

- a SmartComm DMA Controller
- an on-chip SRAM
- a set of peripheral interface modules with DMA controllable:
  - transmit (Tx)
  - receive (Rx)

The SmartComm DMA unit provides a front-line interrupt control and data movement interface. The Interface is on a separate peripheral bus to several on-chip peripheral functions or bus interfaces. This independant control of data movement leaves the G2 core free to concentrate on higher level activities, which increases overall system performance.

SmartComm DMA can control data movement on the following peripherals and interfaces:

- PCI bus
- ATA Controller
- Ethernet
- PSC
- I2C
- IrDA

SmartComm DMA can also do general purpose DMA. Most data transactions are between the peripheral/interface and SDRAM. However, data movement can be programmed to occur between any two memory mapped locations.

SmartComm can manage up to 16 simultaneously enabled DMA tasks, from up to 32 DMA requestors. Also included are the following, which can be used to do data operations as the data is transferred:

- A hardware logic unit
- A hardware CRC unit

SmartComm uses internal buffers to prefetch reads and post writes such that bursting is used whenever possible. This optimizes both internal and external bus activity.

The SmartComm DMA unit does data transfers by following pre-defined task loops. It stores and retrieves data to and from pre-defined data descriptors. The task loop and descriptor definitions are normally storred in the on-chip SRAM, but can also be stored in external SDRAM or other memory mapped storage locations.

FIFO interfaces are implemented between the DMA and each peripheral/interface. As FIFO's are filled or emptied, automatic requests are made to the DMA unit. Based on programmable water mark levels, the DMA unit moves data to and from the FIFO's. This method insures uninterrupted data movement at the given peripheral/interface rate.

A set of pre-defined task loops and descriptors are available from Motorola for each supported MGT5100 peripherals/interfaces. This includes a general purpose DMA function.

13.3 SmartComm DMA Registers—MBAR+0x1200

A register overview is provided in Section 3.4.2, SmartComm DMA Registers.

Hyperlinks to the SmartComm DMA registers are provided below:

- Task Bar (1200)—taskBar
- Current Pointer (1204)—currentPointer
- End Pointer (1208)—endPointer
- Variable Pointer (120C)—variablePointer
- Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl
- Interrupt Pending (1214)—IntPend
- Interrupt Mask (1218)—IntMask
- Task Control (121C)—TCR0, TCR1
- Task Control (1220)—TCR2, TCR3
- Task Control (1224)—TCR4, TCR5
- Task Control (1200)—TCR6, TCR7
- Task Control (122C)—TCR8, TCR9
- Task Control (1200)—TCRA, TCRB
- Task Control (1234)—TCRC, TCRD
- Task Control (1238)—TCRE, TCRF
- Initiator Priority (123C)—IPR0–3
- Initiator Priority (1240)—IPR4–7
- Initiator Priority (1244)—IPR8–11
- Initiator Priority (1248)—IPR12–15
- Initiator Priority (124C)—IPR16–19
- Initiator Priority (1250)—IPR20–23
- Initiator Priority (1254)—IPR24–27
- Initiator Priority (1258)—IPR28–31
- Reserved Register 1 (125C)—res1
- Reserved Register 2 (1260)—res2
- Reserved Register 3 (1264)—res3
- Reserved Register 4 (1268)—res4
- Reserved Register 5 (126C)—res5
- Debug Module Comparator 1 (1270)—Value1
- Debug Module Comparator 2 (1274)—Value2
- Debug Modulator Control (1278)—Control
- Debug Module Status (127C)—Status

13.3.1 Task Bar (1200)—taskBar

Table 13-1. Task Bar (1200)—taskBar

	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	Base address register for SmartComm tasks.

13.3.2 Current Pointer (1204)—currentPointer

Table 13-2. Current Pointer (1204)—currentPointer

	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	Pointer to current task instruction (LCD or DRD).

13.3.3 End Pointer (1208)—endPointer

Table 13-3. End Pointer (1208)—endPointer

	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	Pointer to last instruction of current task.

13.3.4 Variable Pointer (120C)—variablePointer

Table 13-4. Variable Pointer (120C)—variablePointer

	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	Pointer to current task's variable table.

13.3.5 Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl

Table 13-5. Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IntVect1 (not used)								IntVect2 (not used)							
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	PtdCntrl (not used)															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	Name	Description
0:7	IntVect1	Interrupt Vector 1 (not used in MGT5100)
8:15	IntVect2	Interrupt Vector 2 (not used in MGT5100)
16:31	PtdCntrl	PTD Control register (not used)

13.3.6 Interrupt Pending (1214)—IntPend

Table 13-6. Interrupt Pending (1214)—IntPend

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	Reserved														EU[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
1:14	—	Reserved
15	EU[0]	Execution Unit
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 correspnding 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.

13.3.7 Interrupt Mask (1218)—IntMask

Table 13-7. Interrupt Mask (1218)—IntMask

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	Reserved														EU[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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	Name	Description
0	DBG	Debug
1:14	—	Reserved
15	EU[0]	Execution Unit

Bit	Name	Description
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 correspnding 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.

13.3.8 Task Control 0, 1 (121C)—TCR0, TCR1

Table 13-8. Task Control (121C)—TCR0, TCR1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCR0															
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															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:15	TCR0	Task control register for task 0.
16:31	TCR1	Task control register for task 1.

13.3.9 Task Control 2, 3 (1220)—TCR2, TCR3

Table 13-9. Task Control (1220)—TCR2, TCR3

	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.
16:31	TCR3	Task control register for task 3.

13.3.10 Task Control 4, 5 (1224)—TCR4, TCR5

Table 13-10. Task Control (1224)—TCR4, TCR5

	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.
16:31	TCR5	Task control register for task 5.

13.3.11 Task Control 6, 7 (1228)—TCR6, TCR7

Table 13-11. Task Control (1200)—TCR6, TCR7

	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.
16:31	TCR7	Task control register for task 7.

13.3.12 Task Control 8, 9 (122C)—TCR8, TCR9

Table 13-12. Task Control (122C)—TCR8, TCR9

	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.
16:31	TCR9	Task control register for task 9.

13.3.13 Task Control A, B (1230)—TCRA, TCRB

Table 13-13. Task Control (1200)—TCRA, TCRB

	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.
16:31	TCRB	Task control register for task 11.

13.3.14 Task Control C, D (1234)—TCRC, TCRD

Table 13-14. Task Control (1234)—TCRC, TCRD

	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.
16:31	TCRD	Task control register for task 13.

13.3.15 Task Control E, F (1238)—TCRE, TCRF

Table 13-15. Task Control (1238)—TCRE, TCRF

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	TCRE															
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	TCRF															
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.
16:31	TCRF	Task control register for task 15.

13.3.16 Initiator Priority 0–3 (123C)—IPR0–3

Table 13-16. Initiator Priority (123C)—IPR0–3

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	IPR0								IPR1							
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	IPR2								IPR3							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	IPR0	Initiator Prioity register for initator 0.
8:15	IPR1	Initiator Prioity register for initator 1.
16:23	IPR2	Initiator Prioity register for initator 2.
24:31	IPR3	Initiator Prioity register for initator 3.

13.3.17 Initiator Priority 4–7 (1240)—IPR4–7

Table 13-17. Initiator Priority (1240)—IPR4–7

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	IPR4								IPR5							
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 Prioity register for initator 4.
8:15	IPR5	Initiator Prioity register for initator 5.
16:23	IPR6	Initiator Prioity register for initator 6.
24:31	IPR7	Initiator Prioity register for initator 7.

13.3.18 Initiator Priority 8–11 (1244)—IPR8–11

Table 13-18. Initiator Priority (1244)—IPR8–11

	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 Prioity register for initator 8.
8:15	IPR9	Initiator Prioity register for initator 9.
16:23	IPR10	Initiator Prioity register for initator 10.
24:31	IPR11	Initiator Prioity register for initator 11.

13.3.19 Initiator Priority 12–15 (1248)—IPR12–15

Table 13-19. Initiator Priority (1248)—IPR12–15

	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 Prioity register for initator 12.
8:15	IPR13	Initiator Prioity register for initator 13.
16:23	IPR14	Initiator Prioity register for initator 14.
24:31	IPR15	Initiator Prioity register for initator 15.

13.3.20 Initiator Priority 16–19 (124C)—IPR16–19

Table 13-20. Initiator Priority (124C)—IPR16–19

	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 Prioity register for initator 16.
8:15	IPR17	Initiator Prioity register for initator 17.
16:23	IPR18	Initiator Prioity register for initator 18.
24:31	IPR19	Initiator Prioity register for initator 19.

13.3.21 Initiator Priority 20–23 (1250)—IPR20–23

Table 13-21. Initiator Priority (1250)—IPR20–23

	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 Prioity register for initator 20.
8:15	IPR21	Initiator Prioity register for initator 21.
16:23	IPR22	Initiator Prioity register for initator 22.
24:31	IPR23	Initiator Prioity register for initator 23.

13.3.22 Initiator Priority 24–27 (1254)—IPR24–27

Table 13-22. Initiator Priority (1254)—IPR24–27

	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 Prioity register for initator 24.
8:15	IPR25	Initiator Prioity register for initator 25.
16:23	IPR26	Initiator Prioity register for initator 26.
24:31	IPR27	Initiator Prioity register for initator 27.

13.3.23 Initiator Priority 28–31 (1258)—IPR28–31

Table 13-23. Initiator Priority (1258)—IPR28–31

	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 Prioity register for initator 28.
8:15	IPR29	Initiator Prioity register for initator 29.
16:23	IPR30	Initiator Prioity register for initator 30.
24:31	IPR31	Initiator Prioity register for initator 31.

13.3.24 Reserved Register 1 (125C)—res1

Table 13-24. Reserved Register 1 (125C)—res1

	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

13.3.25 Reserved Register 2 (1260)—res2

Table 13-25. Reserved Register 2 (1260)—res2

	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

13.3.26 Reserved Register 3 (1264)—res3

Table 13-26. Reserved Register 3 (1264)—res3

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	res3															
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	res3															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	res3	Reserved

13.3.27 Reserved Register 4 (1268)—res4

Table 13-27. Reserved Register 4 (1268)—res4

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	res4															
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	res4															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	res4	Reserved

13.3.28 Reserved Register 5 (126C)—res5

Table 13-28. Reserved Register 5 (126C)—res5

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	res5															
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	res5															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:31	res5	Reserved

13.3.29 Debug Module Comparator 1 (1270)—Value1

Table 13-29. Debug Module Comparator 1 (1270)—Value1

	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.

13.3.30 Debug Module Comparator 2 (1274)—Value2

Table 13-30. Debug Module Comparator 2 (1274)—Value2

	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.

13.3.31 Debug Modulator Control (1278)—Control

Table 13-31. Debug Modulator Control (1278)—Control

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

Bit	Name	Description
0:31	Control	Debug Module Control.

13.3.32 Debug Module Status (127C)—Status

Table 13-32. Debug Module Status (127C)—Status

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

Bit	Name	Description
0:31	Status	Debug Module Status.

13.4 On-Chip SRAM

MGT5100 contains 8KBytes of on-chip SRAM. This memory is directly accessable by the SmartComm DMA unit. It is used primarily as storage for task table and buffer descriptors used by SmartComm 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 MGT5100 internal register space and is also accessable by the processor core. As such it can be used for other purposes, such as scratch pad storage. The 8Kbyte SRAM starts at location MBAR + 0x4000.

13.5 SmartComm Timer Registers (SCTMR)—MBAR+0x0400

Table 13-33. SCTMR Register/Address—MBAR+0x0400

Address	Register Name	Description	Reset Value
0x0000	STA	SmartComm Timer Status Register	
0x0000–0x000C	—	reserved	
0x0010	TMR1PTC	Timer 1 Prescale and Termination Count Register	
0x0014	TMR1CTR	Timer 1 Control and Enable Register	
0x0018	TMR1STA	Timer 1 Force and Status Register	
0x001C	TMR1INFO	Timer 1 Info Register	
NOTE: Timers 2–7 are similar on address offsets 0x0020–0x007C			

SECTION 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 SmartComm 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 Smartcomm bus 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 Smartcomm. Smartcomm 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 7Wire interface, which uses a subset of the MII pins.

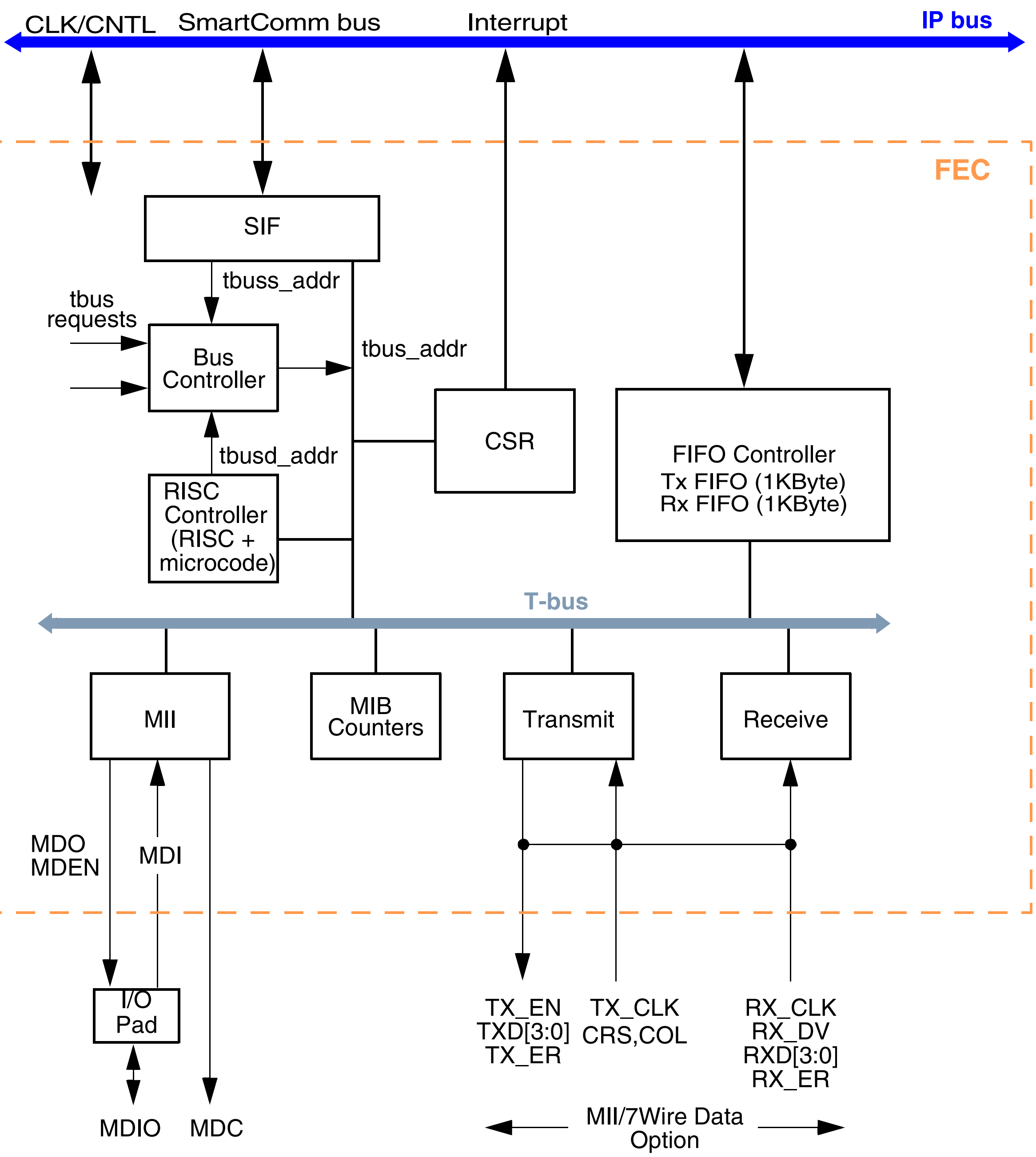


Figure 14-1 Block Diagram—FEC

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 EEE 802.3 MII
  - 10-Mbps EEE 802.3 MII
  - 10-Mbps 7Wire 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 10Mbps 7Wire Interface Operation

If the external transceiver supports 10 Mbps only and uses a 7Wire 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. In addition, the X_TEST register SLOT and COLL bits are intended for use with internal loopback.

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 7Wire interface and supports only 10-Mbps Ethernet data. The 7Wire 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 7Wire 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 7Wire—transmit data output	
tdata[1]	ETH2	MII—transmit data bit 1 output	

Table 14-1 Signal Properties (continued)

Signal Name	Chip Pin	Function	Reset State
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 7Wire—rena input	
rx_clk	ETH9	MII—Rx clock input 7Wire—Rx clock input	
col	ETH10	MII—collision input 10 Mbps 7Wire—collision input	
tx_clk	ETH11	MII—transmit clock input 7Wire—transmit clock input	
rdata[0]	ETH12	MII—Rx data bit 0 input 7Wire—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 deasserted. 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.

Table 14-4 MMI Format Definitions (continued)

Name	Description
<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 SmartComm task loops. Since the FEC software model is Smartcomm-based, there is no similarity with existing CPM-based products' coding.

The CSRs are used for mode control, interrupts and extraction of status information. SmartComm 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 SmartComm 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 2-KByte memory map space. This is divided into two sections of 512Bytes and an additional 1-KBytes of reserved space. The first 512Bytes 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 14-8
400–7FF	Reserved

14.4.2 Control and Status (CSR) Memory Map

Table 14-7 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
024	ECNTRL	Ethernet Control Register
028-03C		Reserved
040	MII_DATA	MII data register
044	MII_SPEED	MII Speed Register
04C-05C		Reserved
064	MIB_CONTROL	MIB Control/Status Register
068-07C		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

Address	Mnemonic	Name
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
120	GADDR1	Upper 32-bits of group hash table
124	GADDR2	Lower 32-bits of group hash table
140	FIFO_ID	Fifo Revision and Size
144	X_WMRK	Transmit fifo watermark
148-17C		Reserved
180	FM_CNTRL	FIFO Memory Control
184	RFIFO_DATA	Receive FIFO Data
188	RFIFO_STATUS	Receive FIFO Status
18C	RFIFO_CNTRL	Receive FIFO Control
190	RFIFO_LRF_PTR	Recv FIFO Last Read Frame Pointer
194	RFIFO_LWF_PTR	Recv FIFO Last Write Frame Pointer
198	RFIFO_ALARM	Receive FIFO Alarm
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_CNTRL	Transmit FIFO Control
1B0	TFIFO_LRF_PTR	Xmit FIFO Last Read Frame Pointer
1B4	TFIFO_LWF_PTR	Xmit FIFO Last Write Frame Pointer
1B8	TFIFO_ALARM	Transmit FIFO Alarm
1BC	TFIFO_RDPTR	Transmit FIFO Read Pointer
1C0	TFIFO_WRPTR	Transmit FIFO Write Pointer
1C4	RESET_CNTRL	FIFO Reset Control
1C8	XMIT_FSM	Transmit FSM Control
1CC-1FF		Reserved

14.4.3 MIB Block Counters Memory Map

Table 14-8 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 2047Bytes. 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 14-8 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 64Bytes, 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 64Bytes, bad crc
220	RMON_T_JAB	RMONTxPackets greater than MAX_FL bytes, badcrc
224	RMON_T_COL	RMON Tx collision count
228	RMON_T_P64	RMON Tx 64Byte packets
22C	RMON_T_P65TO127	RMON Tx 65 to 127Byte packets
230	RMON_T_P128TO255	RMON Tx 128 to 255Byte packets
234	RMON_T_P256TO511	RMON Tx 256 to 511Byte packets
238	RMON_T_P512TO1023	RMON Tx 512 to 1023Byte packets
23C	RMON_T_P1024TO2047	RMON Tx 1024 to 2047Byte packets
240	RMON_T_P_GTE2048	RMON Tx packets with greater than 2048Bytes
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

Table 14-8 MIB Counters (continued)

Address	Mnemonic	Description
28C	RMON_R_MC_PKT	RMON Rx Multicast Packets
290	RMON_R_CRC_ALIGN	RMON Rx Packets with CRC/Align error
294	RMON_R_UNDERSIZE	RMON Rx Packets less than 64Bytes, 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 64Bytes, bad crc
2A0	RMON_R_JAB	RMONRxPackets greater than MAX_FL bytes, badcrc
2A4	RMON_R_RESVD_0	Reserved
2A8	RMON_R_P64	RMON Rx 64Byte packets
2AC	RMON_R_P65TO127	RMON Rx 65 to 127Byte packets
2B0	RMON_R_P128TO255	RMON Rx 128 to 255Byte packets
2B4	RMON_R_P256TO511	RMON Rx 256 to 511Byte packets
2B8	RMON_R_P512TO1023	RMON Rx 512 to 1023Byte packets
2BC	RMON_R_P1024TO2047	RMON Rx 1024 to 2047Byte packets
2C0	RMON_R_P_GTE2048	RMON Rx packets with greater than 2048Bytes
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

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:

- FEC ID Register (3000)—FEC_ID, read-only
- Interrupt Event Register (3004)—IEVENT, R/W
- Interrupt Enable Register (3008)—IMASK
- Ethernet Control Register (3024)—ECNTRL, R/W
- Descriptor Individual Address 1 Register (3118)—IADDR1
- Descriptor Individual Address 2 Register (311C)—IADDR2
- Descriptor Group Address 1 Register (3120)—GADDR1
- Descriptor Group Address 2 Register (3124)—GADDR2

- MII Management Frame Register (3040)—MII_DATA, R/W
- MII Speed Control Register (3044)—MII_SPEED, R/W
- MIB Control Register (3064)—MIB_CONTROL, R/W
- Receive Control Register (3084)—R_CNTRL, R/W
- Hash Register (3088)—R_HASH, read-only
- Rx Destination Address Low Register (309C)—R_DA_LOW, read-only
- Rx Destination Address High Register (30A0)—R_DA_HIGH, read-only
- Tx Control Register (30C4)—X_CNTRL, R/W
- Tx Status Register (30D0)—XMIT.X_STATUS, read-only
- Physical Address Low Register (30E4)—PADDR1
- Physical Address High Register (30E8)—PADDR2
- Opcode/Pause Duration Register (30EC)—OP_PAUSE, R/W
- Tx FIFO Watermark Register (3144)—X_WMRK, R/W
- Tx Rx FIFO Status Register (3188,31A8)—TFIFO_STATUS, RFIFO_STATUS
- Tx Rx FIFO Control Register (318C,31AC)—TFIFO_CNTRL, RFIFO_CNTRL
- Tx Rx FIFO Last Read Frame Pointer Register (3190,31B0)—TFIFO_LRF_PTR, RFIFO_LRF_PTR
- Tx Rx FIFO Last Write Frame Pointer Register (3194,31B4)—TFIFO_LWF_PTR, RFIFO_LWF_PTR
- Tx Rx FIFO Alarm Pointer Register (3198,31B8)—TFIFO_ALARM, RFIFO_ALARM
- Tx Rx FIFO Read Pointer Register (319C,31BC)—TFIFO_RDPTR, RFIFO_RDPTR
- Tx Rx FIFO Write Pointer Register (31A0,31C0)—TFIFO_WRPTR, RFIFO_WRPTR
- Reset Control Register (31C4)—RESET_CNTRL
- Transmit FSM Register (31C8)—XMIT_FSM

14.5.1 FEC ID (3000)—FEC_ID

The read-only FEC ID register (FEC_ID) identifies the FEC block and revision.

Table 14-9 FEC ID Register (3000)—FEC_ID

	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

Bits	Name	Description
16:20	—	Reserved
21	DMA	DMA function is included in the FEC 0 = FEC does not include DMA (Smartcomm 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 Interrupt Event (3004)—IEVENT

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 14-10 Interrupt Event Register (3004)—IEVENT

	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 / W	Reserved															
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

Bits	Name	Description
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. Frame is truncated with a bad crc. Remainder of 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 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 SmartComm 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 SmartComm are soft reset.
15:31	—	Reserved

14.5.3 Interrupt Enable (3008)—IMASK

The IMASK register provides control over the interrupt events allowed to generate an interrupt. All implemented bits in this CSR are R/$\overline{\text{W}}$. 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 14-11 Interrupt Enable Register (3008)—IMASK

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBEEN	BREN	BTEN	GRAEN	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	TFINTEN	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 Ethernet Control (3024)—ECNTRL

The ECNTRL register is a read/write user register that can enable/disable the FEC. Some fields may be altered by hardware.

Table 14-12 Ethernet Control Register (3024)—ECNTRL

	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.

14.5.5 MII Management Frame (3040)—MII_DATA

The MII_DATA register is user accessable. This 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 14-13 MII Management Frame Register (3040)—MII_DATA

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	ST		OP		PA					RA					TA	
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	DATA															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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.

14.5.6 MII Speed Control (3044)—MII_SPEED

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 14-14 MII Speed Control Register (3044)—MII_SPEED

	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 this register to 0x0000_000A results in an MDC frequency of 25MHz * 1/10 = 2.5 MHz. Table 14-15 shows MII_SPEED optimum values as a function of the ipb_clk frequency.
31	—	Reserved

Table 14-15 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

14.5.7 MIB Control (3064)—MIB_CONTROL

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 14-16 MIB Control Register (3064)—MIB_CONTROL

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIB_DISABLE	MIB_IDLE	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	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

14.5.8 Receive Control (3084)—R_CNTRL

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 14-17 Receive Control Register (3084)—R_CNTRL

	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	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										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	0

Bits	Name	Description
0:4	—	Reserved
5:15	MAX_FL	Maximum Frame Length—User R/W field. Resets to decimal 1518. 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.

Bits	Name	Description
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).
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. System clock is substituted for TX_CLK when LOOP is asserted. DRT must be set to 0 when asserting LOOP.

14.5.9 Hash (3088)—R_HASH

The read-only R_HASH register provides address recognition information from the Rx block about the frame currently being received. This register is read by the Microcontroller. These bits provide the Microcontroller with information used in the address recognition subroutine.

Table 14-18 Hash Register (3088)—R_HASH

	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	FCEDC	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

14.5.10 Rx Destination Address Low (309C)—R_DA_LOW

The R_DA_LOW register is written by receive logic and is read-only from the T-bus. The R_DA_LOW register contains the lower 32 bits (first 4 Bytes) of the 48-bit destination address field of the current receive frame. Byte 0 is the first byte transmitted on the network at the start of the frame. This register is used by the internal address recognition logic. This register is not reset.

Table 14-19 Rx Destination Address Low Register (309C)—R_DA_LOW

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

Bits	Name	Description
0:31	R_DA_LOW	Bytes 0(bits 31:24), 1 (bits 23:16), 2 (bits 15:8) and 3 (bits 7:0) of the 6-Byte destination address. These are the first 4 Bytes of the receive frame.

14.5.11 Rx Destination Address High (30A0)—R_DA_HIGH

The R_DA_HIGH register is written by the receive logic and is read-only from the T-bus. The R_DA_HIGH register contains Bytes 4 and 5 of the 6-Byte destination address of the current receive frame. This register is used by the internal address recognition logic. Byte 0 is the first byte transmitted on the network at the start of the frame. This register is not reset.

Table 14-20 Rx Destination Address High Register (30A0)—R_DA_HIGH

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R_DA_HIGH															
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	R_DA_HIGH	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-Byte destination address.
16:31	—	Reserved

14.5.12 Tx Control (30C4)—X_CNTRL

This 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 14-21 Tx Control Register (30C4)—X_CNTRL

	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, 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, MAC transmits a MAC Control PAUSE frame. Next, MAC clears the TFC_PAUSE bit and resumes transmitting data frames. **Note:** If transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, MAC may still transmit a MAC Control PAUSE frame.

Bits	Name	Description
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 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, 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 Tx Status (30D0)—XMIT.X_STATUS

This is a read-only status register. Register fields are written by hardware and updated after the frame transmission (TF_INT signal asserts) completes.

This register is not initialized to a known value at reset.

Table 14-22 Tx Status Register (30D0)—XMIT.X_STATUS

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						DEF	HB	LC	RL	RC				UN	CSL
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	DEF	Defer—sets if the transmit state machine had to defer while trying to transmit this frame. This bit is not set if a collision occurred during transmission.
7	HB	Heartbeat Error—sets if collision input was not asserted during the heartbeat window following transmission completion. Bit is set only if the HBC bit is also set.
8	LC	Late Collision—sets if a collision occurred after the collision window (7 Bytes PA + 1 Byte SFD + 56 Bytes data) has passed.

Bits	Name	Description
9	RL	Retry Limit—sets if a collision occurred on all 16 attempts to transmit a frame. Retry Count (RC) indicates the number of retries required to transmit the frame. 0 if no collisions occurred. If RL = 1, this field has not meaning.
10:13	RC	Retry Count—indicates the number of retries required to transmit the frame. 0 if no collisions occurred. If RL = 1, this field has no meaning.
14	UN	Underrun—sets if data was not received from the Tx FIFO during transmission, resulting in the transmission being aborted.
15	CSL	Carrier Sense Lost—sets if carrier sense dropped out or was never asserted during frame transmission without a collision.
16:31	—	Reserved—bits read as 0.

14.5.14 Physical Address Low (30E4)—PADDR1

The PADDR1 register is written by the user. This register contains the lower 32bits (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 14-23 Physical Address Low Register (30E4)—PADDR1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR1															
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	PADDR1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:31	PADD R1	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.

14.5.15 Physical Address High (30E8)—PADDR2

The PADDR2 register is written by the user. This register contains the upper 16bits (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 14-24 Physical Address High Register (30E8)—PADDR2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PADDR2															
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	TYPE															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 16bits are a constant value, hex 8808.

14.5.16 Opcode/Pause Duration (30EC)—OP_PAUSE

The OP_PAUSE register is read/write accessible. This register contains the 16-bit op-code, and 16-bit pause duration fields used in transmission of a PAUSE frame. The op-code 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 must be initialized.

Table 14-25 Opcode/Pause Duration Register (30EC)—OP_PAUSE

	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	0

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

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.

14.5.17 Descriptor Individual Address 1 (3118)—IADDR1

The IADDR1 register is written by the user. This register contains the upper 32bits 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 14-26 Descriptor Individual Address 1 Register (3118)—IADDR1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IADDR1															
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	IADDR1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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.

14.5.18 Descriptor Individual Address 2 (311C)—IADDR2

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 14-27 Descriptor Individual Address 2 Register (311C)—IADDR2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IADDR2															
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	IADDR2															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:31	IADDR2	The lower 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 31. • Bit 0 contains hash index bit 0.

14.5.19 Descriptor Group Address 1 (3120)—GADDR1

The GADDR1 register is written by the user. This register contains the upper 32 bits 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 14-28 Descriptor Group Address 1 Register (3120)—GADDR1

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GADDR1															
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	GADDR1															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:31	GADDR1	The GADDR1 register contains the upper 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 63. • Bit 0 contains hash index bit 32.

14.5.20 Descriptor Group Address 2 (3124)—GADDR2

The GADDR2 register is written by the user. 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. This register must be initialized.

Table 14-29 Descriptor Group Address 2 Register (3124)—GADDR2

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GADDR2															
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	GADDR2															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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.

14.5.21 Tx FIFO Watermark (3144)—X_WMRK

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 14-30 Tx FIFO Watermark Register (3144)—X_WMRK

	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 xFIFO. 0001 = 128 Bytes written to xFIFO. 0010 = 192 Bytes written to xFIFO. 0011 = 256 Bytes written to xFIFO. 0100 = 320 Bytes written to xFIFO. 0101 = 384 Bytes written to xFIFO. 0110 = 448 Bytes written to xFIFO. 0111 = 512 Bytes written to xFIFO. 1000 = 576 Bytes written to xFIFO. 1001 = 640 Bytes written to xFIFO. 1010 = 704 Bytes written to xFIFO. 1011 = 768 Bytes written to xFIFO. 1100 = 832 Bytes written to xFIFO. 1101 = 896Bytes written to xFIFO. 1110 = 960 Bytes written to xFIFO. 1111 = 1024 Bytes written to xFIFO.

14.5.22 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. Smartcomm supports byte, word or longword addesses. The processor supports longword access only. All register name access is longword aligned

Table 14-31 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
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

14.5.23 FIFO Data Register (3184,31A4)—TFIFO_DATA, RFIFO_DATA

This is 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.

14.5.24 FIFO Status Register(3188,31A8)—TFIFO_STATUS, RFIFO_STATUS

The FIFO status register contains 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 14-32 Tx Rx FIFO Status Register (3188,31A8)—TFIFO_STATUS, RFIFO_STATUS

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				Fram e[0]	Fram e[1]	Fram e[2]	Fram e[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[3:0]	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 to a “one”.
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 to a “one”.
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 to a “one”.
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.

Bits	Name	Description
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

14.6 FIFO Control Register(318C,31AC)—TFIFO_CNTRL, RFIFO_CNTRL

The FIFO Control Register provides 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 Smartcomm.

Table 14-33 Tx Rx FIFO Control Register (318C,31AC)—TFIFO_CNTRL, RFIFO_CNTRL

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Re-served	WFR[1]	WFR[0]	COMP	FRAME	GR[2]	GR[1]	GR[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 SmartDMA 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 SmartDMA. 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.

Bits	Name	Description
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.

14.7 FIFO Last Read Frame Pointer Register(3190,31B0)—TFIFO_LRF_PTR, RFIFO_LRF_PTR

The last read frame pointer (LRFP) is 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 14-34 Tx Rx FIFO Last Read Frame Pointer Register (3190,31B0)—TFIFO_LRF_PTR, RFIFO_LRF_PTR

	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						LR-FP[9]	LR-FP[8]	LR-FP[7]	LR-FP[6]	LR-FP[5]	LR-FP[4]	LR-FP[3]	LR-FP[2]	LR-FP[1]	LR-FP[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 FIFO Last Write Frame Pointer Register(3194,31B4)—TFIFO_LWF_PTR, RFIFO_LWF_PTR

The last write frame pointer (LWFP) is 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 14-35 Tx Rx FIFO Last Write Frame Pointer Register (3194,31B4)— TFIFO_LWF_PTR, RFIFO_LWF_PTR

	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						LR-FP[9]	LR-FP[8]	LR-FP[7]	LR-FP[6]	LR-FP[5]	LR-FP[4]	LR-FP[3]	LR-FP[2]	LR-FP[1]	LR-FP[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.

14.9 FIFO Alarm Pointer Register(3198,31B8)—TFIFO_ALARM, RFIFO_ALARM

This pointer provides high/low level alarm information to the user integration logic and the SmartComm 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 14-36 Tx Rx FIFO Alarm Pointer Register (3198,31B8)—TFIFO_ALARM, RFIFO_ALARM

	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]	Alarm [8]	Alarm [7]	Alarm[6]	Alarm[5]	Alarm[4]	Alarm [3]	Alarm [2]	Alarm[1]	Alarm[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).

14.10 FIFO Read Pointer Register(319C,31BC)—TFIFO_RDPTR, RFIFO_RDPTR

The read pointer is a FIFO maintained pointer which points to the next FIFO location to be read. The read pointer can be both read and written.

Table 14-37 Tx Rx FIFO Read Pointer Register (319C,31BC)—TFIFO_RDPTR, RFIFO_RDPTR

	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						REA D[9]	REA D[8]	REA D[7]	READ[6]	READ[5]	READ[4]	REA D[3]	REA D[2]	READ[1]	READ[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.

14.11 FIFO Write Pointer Register(31A0,31C0)—TFIFO_WRPTR, RFIFO_WRPTR

The write pointer is a FIFO maintained pointer which points to the next FIFO location to be written. The write pointer can be both read and written.

Table 14-38 Tx Rx FIFO Write Pointer Register (31A0,31C0)—TFIFO_WRPTR, RFIFO_WRPTR

	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						WRITE[9]	WRITE[8]	WRITE[7]	WRITE[6]	WRITE[5]	WRITE[4]	WRITE[3]	WRITE[2]	WRITE[1]	WRITE[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	WRITE[9:0]	WRITE Pointer. This pointer indicates the next location to be written by the FIFO controller.

14.12 Reset Control Register(31C4)—RESET_CNTRL

The reset control register allows reset of the FIFO controllers.

Table 14-39 Reset Control Register (31C4)—RESET_CNTRL

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						RCTL[1]	RCTL[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	RCTL[1]	0 = No Reset FIFO controllers 1 = Reset FIFO controllers

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

14.13 Transmit FSM Register(31C8)—XMIT_FSM

The transmit finite state machine register controls operation of appending CRC. Typical use is enabled and CRC appended.

Table 14-40 Transmit FSM Register (31C8)—XMIT_FSM

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						XF-SM[1]	XF-SM[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

14.14 Initialization Sequence

This section describes which registers are hardware reset, which are reset by FEC RISC, and what locations the user must initialize prior to enabling the FEC.

14.14.1 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 14-41 shows the effect deasserting ETHER_EN has on Ethernet MAC operation and registers.

Table 14-41 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

14.14.2 User Initialization (Prior to Asserting ETHER_EN)

The user needs to initialize portions 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 14-42.

Table 14-42 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)

14.14.2.1 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 14-43 shows RISC initialization operations common to FEC.

Table 14-43 Microcontroller Initialization (FEC)

Description
Initialize BackOff random number seed
Activate Receiver
Activate Transmit

14.14.3 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.

14.14.3.1 Receive Frame Status Word

Table 14-2 below defines the format for the receive frame status word

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										

Figure 14-2 Receive Frame Status Word Format

Bits 31-28, 26-25, 19 and 15-11—Reserved

L—Last in Frame, written by the FEC
   0 = The buffer is not the last in a frame.
   1 = 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 Nonoctet 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—

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—

14.14.3.2 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 14-3below for the format of the transmit frame control word.

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

Figure 14-3 Transmit Frame Control Word Format

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).

14.14.4 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 threshold 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.

14.14.5 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.

14.14.6 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, then the microcontroller runs the address recognition subroutine, as shown in Figure 14-5.

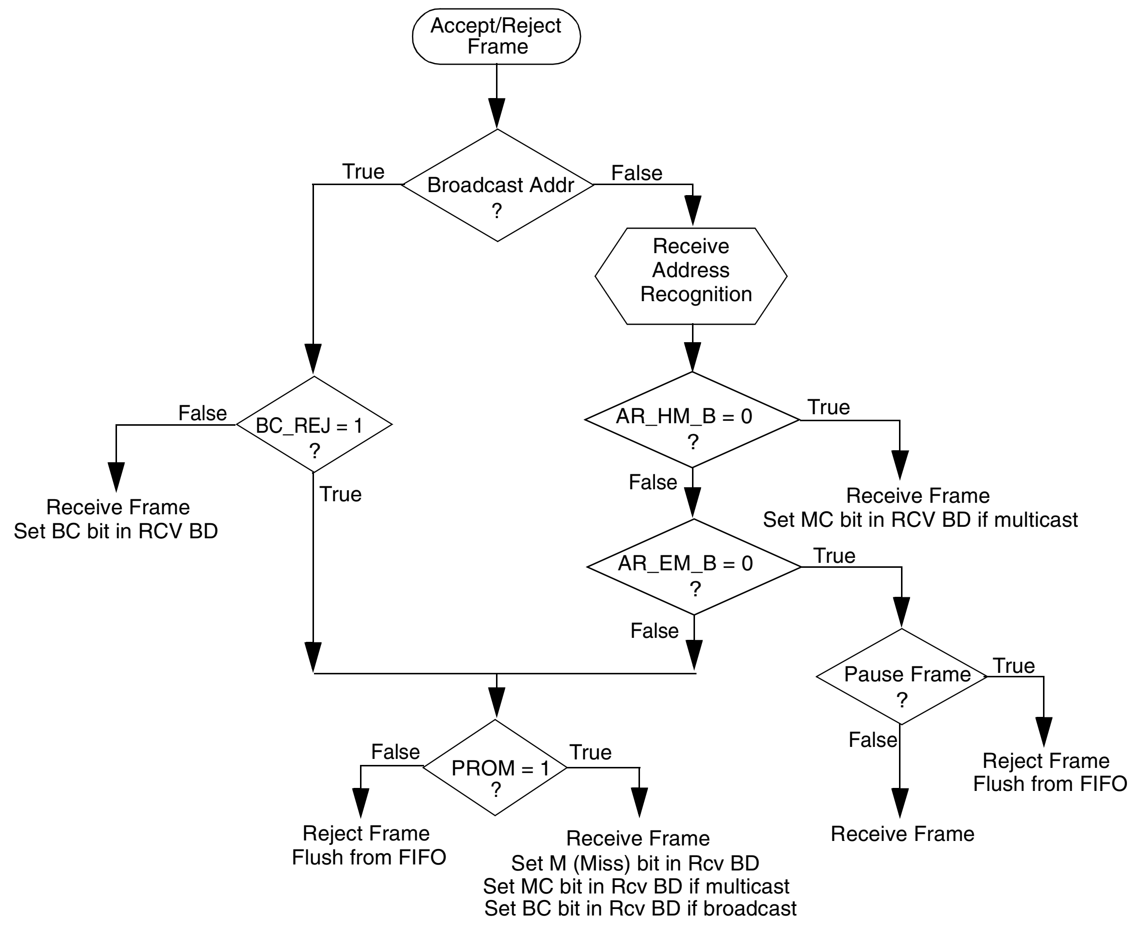
If the DA is a group (multicast) address and flow control is disabled, then the microcontroller will perform a group hash table lookup using the 64-entry hash table programmed in GADDR1 and GADDR2. If a hash match occurs, AR_HM_B (address recognition hash match bar) is set to 0 and the receiver accepts 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, then 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 then, 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, then 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.

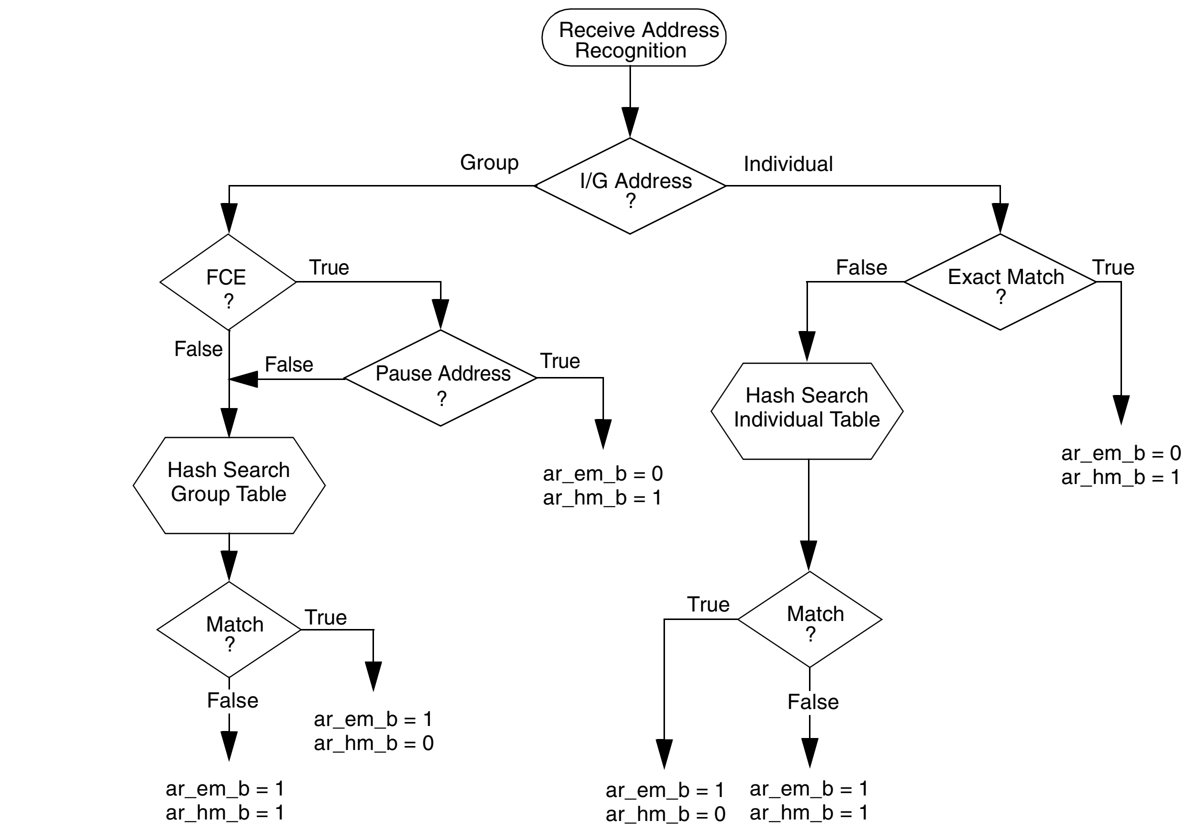
Similarly, if the DA is a broadcast address, broadcast reject (R_CNTRL.BC_REJ) is asserted, and promiscuous mode is enabled, then the frame will be accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame will be rejected and the MISS bit will be cleared.

In general, when a frame is rejected, it is flushed from the FIFO.


Accept/Reject Frame
Broadcast Addr ?
True
False
Receive Address Recognition
BC_REJ = 1 ?
False
True
Receive Frame Set BC bit in RCV BD
AR_HM_B = 0 ?
True
False
Receive Frame Set MC bit in RCV BD if multicast
AR_EM_B = 0 ?
True
False
Pause Frame ?
True
False
Reject Frame Flush from FIFO
Receive Frame
PROM = 1 ?
False
True
Reject Frame Flush from FIFO
Receive Frame Set M (Miss) bit in Rcv BD Set MC bit in Rcv BD if multicast Set BC bit in Rcv BD if broadcast

NOTES:
BC_REJ - field in R_CNTRL register (BroadCast REJect)
Check Address - microcode Address Recognition subroutine; returns AR_HM_B and AR_EM_B
AR_EM_B - bit in RECV.AR_DONE register (address recognition exact match bar)
AR_HM_B - bit in RECV.AR_DONE register (address recognition hash match bar)
PROM - field in R_CNTRL register (PROMiscous mode)
Pause Frame - valid PAUSE frame received

Figure 14-4 Ethernet Address Recognition - receive block decisions



NOTES:
FCE - field in R_CNTRL register (Flow Control Enable)
AR_EM_B - bit in RECV.AR_DONE register (address recognition exact match bar)
AR_HM_B - bit in RECV.AR_DONE register (address recognition hash match bar)
I/G - Individual/Group bit in Destination Address (least significant bit in first byte received in MAC frame)

Figure 14-5 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

Table 14-44 Destination Address to 6-Bit Hash

48-bit DA	6-bit hash (in hex)	hash decimal value
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
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

Table 14-44 Destination Address to 6-Bit Hash

48-bit DA	6-bit hash (in hex)	hash decimal value
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

14.14.7 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.

14.14.8 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 interpacket gap between receive frames is less than 28 bit times the following frame may be discarded by the receiver.

14.14.9 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 ones. 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, then no retransmission is performed and the end of frame buffer is closed with an LC error indication.

14.14.10 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 dma'd 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.

14.14.11 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.

14.14.11.1 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, then 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.

14.14.11.2 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 set. 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.

SECTION 15
PROGRAMMABLE SERIAL CONTROLLERS (PSC)

15.1 Overview

The following sections are contained in this document:

- PSC Registers—MBAR + 0x2000, 0x2400, 0x2800
- PSC Module Signal Definitions
- PSC Operation

The MGT5100 has 3 independent Programmable Serial Controllers (PSCs):

- PSC1—MBAR + 0x2000
- PSC2—MBAR + 0x2400
- PSC3—MBAR + 0x2800

Each PSC can be clocked by CLKIN, eliminating the need for an external crystal. 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

Unless otherwise specified, all references and descriptions in this chapter refer to "UART Mode", as opposed to "modem mode".

15.1.1 PSC1—MBAR + 0x2000

PSC1 is used to interface externally to a modem, a CODEC, an AC97 device, an IR Controller (UART type) or to a device that supports a UART protocol. When PSC1 interfaces to a CODEC, $\overline{\text{CTS}}$ is used as the bit-clock input and a Carrier Detect (CD) is used as the frame sync input. When PSC1 is used to support AC97, $\overline{\text{CTS}}$ is used as the bit-clock input and $\overline{\text{RTS}}$ is used as the frame sync output. Some external AC97 devices require a reset signal. This is provided using one of the MGT5100 GPIO signals.

15.1.2 PSC2—MBAR + 0x2400

PSC2 has the same functionality as PSC1.

15.1.3 PSC3—MBAR + 0x2800

PSC3 has the same functionality as PSC1 and PSC2 except that it does not support AC97.

Figure 15-1 shows a simplified block diagram.

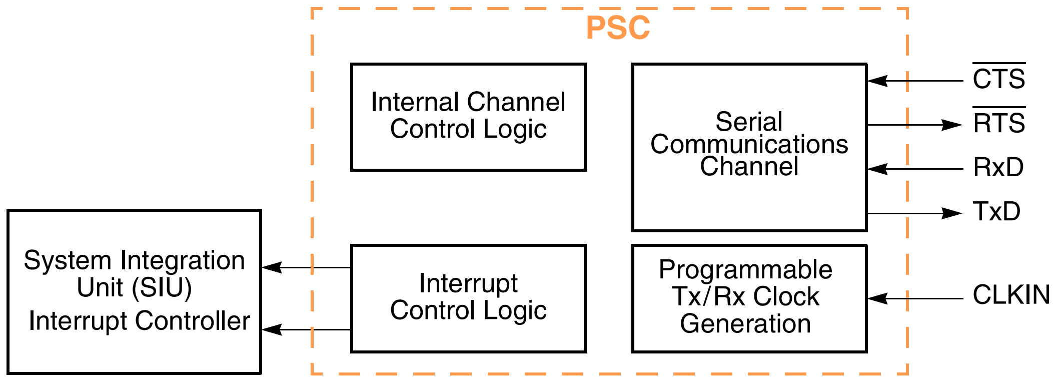

PSC
Internal Channel Control Logic
Serial Communications Channel
System Integration Unit (SIU) Interrupt Controller
Interrupt Control Logic
Programmable Tx/Rx Clock Generation
$\overline{\text{CTS}}$
$\overline{\text{RTS}}$
RxD
TxD
CLKIN

Figure 15-1 Simplified Block Diagram

PSC1 and PSC2 can provide synchronous operation and a CODEC interface for soft modem support. They can be programmed to function like a UART or in one of the following three modem modes:

- An 8-bit CODEC interface
- A 16-bit CODEC interface
- An Audio CODEC’97 (AC97) digital interface controller

PSC3 can be programmed to do all the above functions except AC97.

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.

Each PSC can interface to a CODEC through a serial port consisting of Tx and Rx serial data and serial bit-clock and frame inputs from the CODEC. Digital sample data is transferred to and from the CODEC through the serial port.

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 MGT5100 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.

When programmed as a UART the PSC serial communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter deriving an operating frequency from CLKIN or an external clock using the timer pin. 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.

15.1.4 Features

PSC features include:

- Each is clocked by CLKIN (IP Bus clock), eliminating the need for an external crystal
- Full-duplex asynchronous/synchronous receiver/transmitter channel
- 512-byte receiver (Rx) FIFO
- 256-byte transmitter (Tx) FIFO
- Independently programmable receiver and transmitter clock sources
- Programmable UART 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
- Each channel is programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic WakeUp mode for multidrop applications
- Four maskable interrupt conditions
- PSC Tx and Rx FIFOs can be programmed to interrupt either the SmartComm DMA or the CPU when they require filling or emptying, respectively.
- 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
- Programmable to interface to an 8- or 16-bit CODEC for soft modem support
- Programmable to function as a digital AC97 Controller (PSC1 and PSC2 only)
- No parity error, framing error, or line break detection in modem mode

15.2 PSC Registers—MBAR + 0x2000, 0x2400, 0x2800

Each PSC uses 37 32-bit registers. These registers are located at an offset as indicated below:

- PSC1 = MBAR + 0x20000 + register address
- PSC2 = MBAR + 0x24000 + register address
- PSC3 = MBAR + 0x28000 + register address

Hyperlinks to the PSC registers are provided below:

- UART Mode 1 (2000, 2400, 2800)—MR1_[1, 2, 3]
- Other Modes (2000, 2400, 2800)—MR1_[1, 2, 3]

- UART Mode (2004, 2404, 2804)—SR[1, 2, 3]
- Modem Status (2004, 2404, 2804)—SR[1, 2, 3]

- All Modes (2008, 2408, 2808)—CR[1, 2, 3]

- UART/Modem8&16 Rx Buffers (200C, 240C, 280C)—RB[1, 2, 3]
- AC97 Rx Buffers (200C, 240C)—RB[1, 2]

- Input Port UART Change (2010, 2810, 2410)—IPCR[1, 3, 2]
- Modem Mode (2010, 2810, 2410)—IPCR[1, 3, 2]
- All Modes (2010, 2410, 2810)—ACR[1, 2, 3]

- UART Mode (2014, 2414, 2814)—ISR[1, 2, 3]
- Modem Mode (2014, 2414, 2814)—ISR[1, 2, 3]

- Counter Timer UART (2018, 2418, 2818)—CTUR[1, 2, 3]
- Other Modes (2018, 2418, 2818)—CTUR[1, 2, 3]

- UART/Modem8&16 Set (2038, 2438, 2838)—OP1_[1, 2, 3]
- AC97 Mode (2038, 2438)—OP1_[1, 2]

- UART Mode 2 (2000, 2400, 2800)—MR2_[1, 2, 3]
- Other Modes (2000, 2400, 2800)—MR2_[1, 2, 3]

- UART Mode (2004, 2404, 2804)—CSR[1, 2, 3]
- Other Modes (2004, 2404, 2804)—CSR[1, 2, 3]

- Interrupt Vector (2030, 2430, 2830)—IVR[1, 2, 3]

- UART/Modem8&16 Tx Buffers (200C, 240C, 280C)—TB[1, 2, 3]
- AC97 Tx Buffers (200C, 240C)—TB[1, 2]

- Input UART (2034, 2434, 2834)—IP[1, 2, 3]
- Input Modem8&16 Mode (2034, 2434, 2834)—IP[1, 2, 3]
- Input AC97 Mode (2034, 2434)—IP[1, 2]

- Interrupt UART (2014, 2414, 2814)—IMR[1, 2, 3]
- Modem Mode (2014, 2414, 2814)—IMR[1, 2, 3]

- Counter Timer UART (201C, 241C, 281C)—CTLR[1, 2, 3]
- Other Modes (201C, 241C, 281C)—CTLR[1, 2, 3]

- UART/Modem8&16 Reset (203C, 243C, 284C)—OP0_[1, 2, 3]
- AC97Bit Reset (203C, 243C)—OP0_[1, 2]

- UART Mode (2040, 2440, 2840)—SICR[1, 2, 3]
- Modem8&16 Mode (2040, 2440, 2840)—SICR[1, 2, 3]
- AC97 Mode(2040, 2440, 2840)—SICR[1, 2]

- Rx FIFO Number of Data (2058, 2458, 2858)—RFNUM[1, 2, 3]
- Rx FIFO Data (2x60)—RFDATA[1, 2, 3]
- Rx FIFO Status (2064, 2464, 2864)—RFSTAT[1, 2, 3]
- Rx FIFO Control (2068, 2468, 2868)—RFCNTL[1, 2, 3]
- Rx FIFO Alarm (206E, 246E, 286E)—RFALARM[1, 2, 3]
- Rx FIFO Read Pointer (2072, 2472, 2872)—RFRPTR[1, 2, 3]
- Rx FIFO Write Pointer (2076, 2476, 2876)—RFWPTR[1, 2, 3]
- Rx FIFO Last Read Frame PTR (207A, 247A, 287A)—RFLRFPTR[1, 2, 3]
- Rx FIFO Last Write Frame PTR (207C, 247C, 287C)—RFLWFPTR[1, 2, 3]

- Tx FIFO Number of Data (205C, 245C, 285C)—TFNUM[1, 2, 3]
- Tx FIFO Data (2x80)—TFDATA[1, 2, 3]
- Tx FIFO Status (2084, 2484, 2884)—TFSTAT[1, 2, 3]
- Tx FIFO Control (2088, 2488, 2888)—TFCNTL[1, 2, 3]
- Tx FIFO Alarm (208E, 248E, 288E)—TFALARM[1, 2, 3]
- Tx FIFO Read Pointer (2092, 2492, 2892)—TFRPTR[1, 2, 3]
- Tx FIFO Write Pointer (2096, 2496, 2896)—TFWPTR[1, 2, 3]
- Tx FIFO Last Read Frame PTR (209A, 249A, 289A)—TFLRFPTR[1, 2, 3]
- Tx FIFO Last Write Frame PTR (209C, 249C, 289C)—TFLWFPTR[1, 2, 3]

Flowcharts in Section 15.4.6, describe basic PSC module programming. PSC module operation is controlled by writing control bytes into the appropriate registers.

15.2.1 Mode Register 1 (2x00)—MR1_[*1, 2, 3*]

The UART Mode registers control configuration. MR1_*n* 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*n*[MISC]. After MR1_*n* is read or written, the pointer points to MR2_*n*.

Table 15-1 UART Mode 1 (2000, 2400, 2800)—MR1_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	RxRTS	RxIRQ/ FFULL	ERR	PM		PMT	B/C	
RESET:	0	0	0	0	0	0	0	0

Table 15-2 Other Modes (2000, 2400, 2800)—MR1_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved	RxIRQ/ FFULL	Reserved					
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RxRTS	**UART**—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_*n*[TxRTS]. Not used in modem 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**—Reserved
1	RxIRQ/ FFULL	Receiver interrupt select. 0 = RxRDY is the source that generates IRQ 1 = FFULL is the source that generates IRQ.
2	ERR	**UART**—Error mode—Configures the FIFO status bits, SR*n*[RB,FE,PE]. This bit is not used in modem mode. It is hardwired to 1. 0 = Unused 1 = Block mode—SR*n* values are the logical OR of the status for all characters received after the last Reset Error Status command for the channel was issued. See Section 15.2.6. **Other**—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 15-3. PM is not used in modem mode. **Other**—Reserved

Bit	Name	Description
5	PMT	**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 modem mode. See Table 15-3. **Other**—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 modem mode. 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits **Other**—Reserved

Table 15-3 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 (2x00)—MR2_[*1, 2, 3*]

MR2_*n* can be read or written when the mode register pointer points to it, which occurs after any access to MR1_*n*. An MR2_*n* access does not update the pointer.

Table 15-4 UART Mode 2 (2000, 2400, 2800)—MR2_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	CM		TxRTS	TxCTS	SB			
RESET:	0	0	0	0	0	0	0	0

Table 15-5 Other Modes (2000, 2400, 2800)—MR2_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	CM		Reserved					
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	CM	Channel mode—Selects a channel mode. Table 15.4.3, describes individual modes. CM is used in both UART and modem modes. 00 = Normal 01 = Automatic echo 10 = Local loop-back 11 = Remote loop-back
2	TxRTS	**UART**—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 modem 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 UOP[RTS] 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**—Reserved
3	TxCTS	**UART**—Transmitter clear-to-send—If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. TxCTS is not used in modem 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**—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 modem mode. See Table 15-6. **Other**—Reserved

Table 15-6 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 (2x04)—SR[*1, 2, 3*]

The read-only SR*n* register shows status of the transmitter, the receiver, and the FIFO.

Table 15-7 UART Mode (2004, 2404, 2804)—SR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	RB	FE	PE	OE	TxEMP	TxRDY	FFULL	RxRDY	CDE	Reserved						
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RB	Received Break—detects breaks originating in middle of received character. Such a break must persist until the end of next detected character time. RB is not used (always 0) in modem mode. 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.
1	FE	Framing Error— is not used (always 0) in modem 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.
2	PE	Parity Error—valid only if RxRDY = 1. PE is not used (always 0) in modem mode. 0 = No parity error occurred. 1 = If MR1_*n*[PM]=0x (with parity or force parity), corresponding FIFO character was received with incorrect parity. If MR1*n*[PM]=11 (multidrop), PE stores received A/D bit.
3	OE	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. OE 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. OE is cleared by the RESET ERROR STATUS command in CR*n*.
4	TxEMP	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*n*[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.

Bit	Name	Description
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 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. TxRDY clears when the number of data bytes in the Tx FIFO becomes greater than the TFALARM value. In UART mode this bit only asserts if the Tx is enabled.
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.
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.
8	CDE	DCD Error 0 = The DCD input is negated while receiving data. 1 = No error
9:15	—	Reserved

15.2.4 Modem Mode (2x04)—SR[*1, 2, 3*]

This is a read-only register.

Table 15-8 Modem Status (2004, 2404, 2804)—SR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved			OE	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
0:2	—	Reserved
3	OE	Overrun Error—Indicates whether an overrun occurs. (For purposes of overrun, FIFO full means all space in the FIFO is occupied; the Rx FIFO threshold is irrelevant to overrun.) 0 = No overrun occurred. 1 = One or more characters in the received data stream have been lost. OE sets on receipt of a new sample when the FIFO is full and a sample is already in the shift register waiting for an empty FIFO position. When this occurs, the sample in the receiver shift register is lost. OE is cleared by the CR*n* RESET ERROR STATUS command.

Bit	Name	Description
4	URERR	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 Tx 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 (OE = 1), and as such it is now cleared the same way as OE, by a RESET ERROR STATUS command in the CR*n* and not by a reset Tx command in the CR*n*.
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 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. TxRDY clears when the number of data bytes in the Tx FIFO becomes greater than the TFALARM value. In modem mode this bit will become asserted even when the Tx is disabled, unlike in UART mode.
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.
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 receiver buffer FIFO.
8:15	—	Reserved

15.2.5 Clock-Select Register (2x04)—CSR[*1, 2, 3*]

Although the PSC module has an external clock input port, this port does not come out to a pin on the MGT5100. This is true for all 3 PSCs. Therefore, the user must avoid writing values of “1110” or “1111” to the RCS or TCS fields in any PSC CSR register. The only valid clock source is a prescaled version of the IP Bus clock.

Table 15-9 UART Mode (2004, 2404, 2804)—CSR[*1, 2, 3*]

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

Table 15-10 Other Modes (2004, 2404, 2804)—CSR[*1, 2, 3*]

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

Bit	Name	Description
0:7	—	Reserved
8:11	RCS	**UART**—Receiver Clock Select—Selects the clock source for Rx channel. 0000 -1101 = Prescaled CLKIN (IP bus clock) 1110 = INVALID 1111 = INVALID **Other**—Reserved
12:15	TCS	**UART**—Transmitter Clock Select—Selects the clock source for Tx channel. 0000-1101 = Prescaled CLKIN (IP bus clock) 1110 = INVALID 1111 = INVALID **Other**—Reserved

15.2.6 Command Register (2x08)—CR[*1, 2, 3*]

The write-only command registers (CR*n*), supply commands to the PSC in both UART and modem modes. Only multiple commands that do not conflict can be specified in a single write to a CR*n*. For example, reset Tx and enable Tx cannot be specified in one command.

Table 15-11 All Modes (2008, 2408, 2808)—CR[*1, 2, 3*]

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

Bit	Value	Command	Description
0:8	—		Reserved

Bit	Value	Command	Description
9:11 see note	000	no command	—
	001	reset mode register pointer	Causes mode register pointer to point to MR1*n*.
	010	reset receiver	Immediately disables receiver, clears SR*n*[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 reconfiguring the receiver.
	011	reset transmitter	In UART mode, immediately disables Tx and clears SR*n*[TxEMP, TxRDY]. No other registers are altered. Because it places Tx in a known state, use this command instead of TRANSMITTER DISABLE when reconfiguring transmitter. In modem 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*n*[RB, FE, PE, OE]. In block mode, clears all error bits after a data block is received. In modem mode, command clears OE and URERR.
	101	reset break change interrupt	Clears the delta break bit, ISR*n*[DB]. Command has no effect in modem 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 modem 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
12:13 see note	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*n*[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 modem 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 AC ‘97 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.
	10	transmitter disable	Terminates Tx operation and clears SR*n*[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*n*[TxEMP,TxRDY] are negated. In modem mode SR*n*[TxEMP] is negated. Tx does not clear SR*n*[TxRDY] unless PSC is in remote loop-back or auto-echo mode. In modem mode, unlike UART mode, the Tx FIFO may be loaded while Tx is disabled.
	11	—	Reserved, do not use.
14:15 see note	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*n*[PM] $\neq$ 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 modem 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.			

15.2.7 Rx Buffer Registers (2x0C)—RB[*1, 2, 3*]

Data is read from the Rx FIFO by reading from this read-only register. The Rx FIFO size is 512 bytes.

Table 15-12 UART/Modem8&16 Rx Buffers (200C, 240C, 280C)—RB[*1, 2, 3*]

	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 15-13 AC97 Rx Buffers (200C, 240C)—RB[*1, 2*]

	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

Bit	Name	Description
0:19 (AC97) and 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. Tmeslot #0 data is in bits 0:15. Bit 20 is 1 in the first sample of a new frame. **UART/Modem8** (0:31)—Received data—For modem8 mode, 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. **UART/Modem16** (0:31)—Received data—For modem16 mode, 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.
20 (AC97)	SOF	**AC97**—AC97 Start of Frame indicator. 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.
21:31 (AC97)	—	**AC97**—Reserved

15.2.8 Tx Buffer Registers (2x0C)—TB[*1, 2, 3*]

Data is written to the Tx FIFO by writing to this write-only register. The Tx FIFO size is 256 bytes.

Table 15-14 UART/Modem8&16 Tx Buffers (200C, 240C, 280C)—TB[*1, 2, 3*]

	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 15-15 AC97 Tx Buffers (200C, 240C)—TB[*1, 2*]

	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

Bit	Name	Description
0:19 (AC97) and 0:31 (other)	TB	**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. Since timeslot #0 is only 16 bits wide, it is stored in RB[0:15]. **UART/Modem8** (0:31)—Transmit data—For modem8 mode, 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 two bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data.Transmit data. **UART/Modem16** (0:31)—Transmit data—For modem16 mode, data can be written 2 or 4 bytes at a time. For two bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data
20 (AC97)	SOF	**AC97**—AC97 Start of Frame indicator. 0 = TB[0:19] is not the first sample in the frame. 1 = TB[0:15] is the first sample in a new frame. The number 0 slot is called the TAG slot.
21:31 (AC97)	—	**AC97**—Reserved

15.2.9 Input Port Change Register (2x10)—IPCR[*1, 2, 3*]

The read-only IPCR register shows the current state and change-of-state for the modem control input port.

Table 15-16 Input Port UART Change (2010, 2810, 2410)—IPCR[*1, 3, 2*]

	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	1	1	0	0

Table 15-17 Modem Mode (2010, 2810, 2410)—IPCR[*1, 3, 2*]

	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	1	1	0	0

Bit	Name	Description
0	SYNC	**Modem**—Sync detected. 0 = Has not detected sync. 1 = Detected sync (ext_clk=1 in modem8/modem16 or scc_rts_b=1 in AC97 mode) **UART**—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 1bit duration determined by the CSR, CTUR and CTLR) has occurred at scc_dcd_b 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 scc_cts_b 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 DCD port. This input is double latched and same as DCD of IP. 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 CTS port. This input is double latched and same as DCD of IP. 0 = The current state of the CTS input port is low. 1 = The current state of the CTS input port is high.

15.2.10 Auxiliary Control Register (2x10)—ACR[*1, 2, 3*]

The write-only ACR register controls Tx/Rx handshaking.

Table 15-18 All Modes (2010, 2410, 2810)—ACR[*1, 2, 3*]

	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).

15.2.11 Interrupt Status Register (2x14)—ISR[*1, 2, 3*]

The read-only ISR register provides status for all potential interrupt sources. Register contents are 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 15-19 UART Mode (2014, 2414, 2814)—ISR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved				DB	RxRDY FFULL	TxRDY	Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-20 Modem Mode (2014, 2414, 2814)—ISR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved					RxRDY FFULL	TxRDY	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 = IPC has no effect on the interrupt. 1 = Enable the interrupt for IPC in the ISR register.
1:4	—	Reserved
5	DB	**UART**—Delta Break **Modem**—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
—	DEOF	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.
8:15	—	Reserved

15.2.12 Interrupt Mask Register (2x14)—IMR[*1, 2, 3*]

The read-only IMR register selects corresponding bits in the ISR that cause an interrupt.

- If one ISR bit sets and the corresponding IMR bit also sets, 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 15-21 Interrupt UART (2014, 2414, 2814)—IMR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W	IPC	Reserved				DB	RxRDY FFULL	TxRDY	Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-22 Modem Mode (2014, 2414, 2814)—IMR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved															
W	IPC	Reserved					RxRDY FFULL	TxRDY	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.
1:4	—	Reserved
5	DB	**UART**—Delta Break 0 = DB has no effect on the interrupt. 1 = Enable the interrupt for DB **Modem**—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
—	DEOF	Detect End of Frame 0 = DEOF has no effect on the interrupt. 1 = Enable the interrupt for DEOF.
8:15	—	Reserved

15.2.13 Counter Timer Upper Register (2x18)—CTUR[*1, 2, 3*]

These registers hold the upper bytes of the preload value used by the timer to provide a given baud rate.

Table 15-23 Counter Timer UART (2018, 2418, 2818)—CTUR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R	CT[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-24 Other Modes (2018, 2418, 2818)—CTUR[*1, 2, 3*]

	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	CT	**UART**—Baud rate prescale value. The baud rate is calculated as: Baud rate = (system clock frequency) / (CT[15:0] x 16 x 2) The minimum CT value is 1; 0 denotes the counter stop. **Other**—Reserved

15.2.14 Counter Timer Lower Register (2x1C)—CTLR[*1, 2, 3*]

These registers hold the lower bytes of the preload value used by the timer to provide a given baud rate.

Table 15-25 Counter Timer UART (201C, 241C, 281C)—CTLR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R	CT[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Table 15-26 Other Modes (201C, 241C, 281C)—CTLR[*1, 2, 3*]

	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	CT	**UART**—Baud rate prescale value. The baud rate is calculated as: Baud rate = (system clock frequency) / (CT[15:0] x 16 x 2) The minimum CT value is 1; 0 denotes the counter stop. **Other**—Reserved

15.2.15 Interrupt Vector Register (2x30)—IVR[*1, 2, 3*]

This register is not used since the MGT5100 does not use interrupt vectors supplied by the peripherals.

Table 15-27 Interrupt Vector (2030, 2430, 2830)—IVR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R	IVR[0:7]							
W								
RESET:	1	1	1	1	0	0	0	0

Bit	Name	Description
0:7	IVR	Interrupt Vector— Not applicable for MGT5100.

15.2.16 Input Port (2x34)—IP[*1, 2, 3*]

This read-only IP register shows the current state of the input ports.

Table 15-28 Input UART (2034, 2434, 2834)—IP[*1, 2, 3*]

	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 15-29 Input Modem8&16 Mode (2034, 2434, 2834)—IP[*1, 2, 3*]

	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 15-30 Input AC97 Mode (2034, 2434)—IP[*1, 2*]

	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. **UART/Modem 8&16**—Reserved
1	TGL	**AC97/Modem 8&16**—Test usage. Toggle by frame sync. **UART**—Reserved
2:5	—	Reserved
6	DCD	Current state of the scc_dcd_b input. 0 = scc_dcd_b is low 1 = scc_dcd_b is high
7	CTS	Current state of the scc_cts_b input 0 = scc_cts_b is low 1 = scc_cts_b is high

15.2.17 Output Port 1 Bit Set (2x38)—OP1_[*1, 2, 3*]

This is a write-only register. Output ports are asserted by writing to this register.

Table 15-31 UART/Modem8&16 Set (2038, 2438, 2838)—OP1_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R								
W	Reserved						RES	RTS
RESET:	0	0	0	0	0	0	0	0

Table 15-32 AC97 Mode (2038, 2438)—OP1_[*1, 2*]

	msb 0	1	2	3	4	5	6	7 lsb
R								
W	Reserved						RES	Reserved
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 scc_res_b (scc_res_b becomes 0).
7	RTS	**UART/Modem 8&16**—Assert RTS output. 0 = No operation 1 = Asserts output port scc_rts_b (scc_rts_b becomes 0). **AC97**—Reserved

15.2.18 Output Port 0 Bit Reset (2x3C)—OP0_[*1, 2, 3*]

This is a write-only register. Output ports are negated by writing to this register.

Table 15-33 UART/Modem8&16 Reset (203C, 243C, 284C)—OP0_[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R								
W	Reserved						RES	RTS
RESET:	0	0	0	0	0	0	0	0

Table 15-34 AC97Bit Reset (203C, 243C)—OP0_[*1, 2*]

	msb 0	1	2	3	4	5	6	7 lsb
R								
W	Reserved						RES	Reserved
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 scc_res_b (scc_res_b becomes 1).
7	RTS	**UART/Modem 8&16**—Assert RTS output. 0 = No operation 1 = Negates output port scc_rts_b (scc_rts_b becomes 1). **AC97**—Reserved

15.2.19 PSC Control Register (2x40)—SICR[*1, 2, 3*]

This register sets the main operation mode.

Table 15-35 UART Mode (2040, 2440, 2840)—SICR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R/W	Reserved				RxDCD	SIM[2:0]		
RESET:	0	0	0	0	0	0	0	0

Table 15-36 Modem8&16 Mode (2040, 2440, 2840)—SICR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R/W	Reserved		DTS1	SHDIR	Reserved	SIM[2:0]		
RESET:	0	0	0	0	0	0	0	0

Table 15-37 AC97 Mode(2040, 2440, 2840)—SICR[*1, 2*]

	msb 0	1	2	3	4	5	6	7 lsb
R/W	ACRB	AWR	Reserved			SIM[2:0]		
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 port scc_res_b. 0 = The transceiver recovers from low power mode in AC97. 1 = The transceiver stays in the current state. **UART/Modem 8&16**—Reserved
1	AWR	**AC97**—AC97 Warm Reset (to the PSC and off-chip AC97 CODEC) 0 = AC97 warm reset is negated. RTS output functions normally as the AC97 frame sync. 1 = Force “1” on RTS output, which is used as the AC97 frame sync, and the PSC recovers from AC97 power down mode. **UART/Modem 8&16**—Reserved
2	DTS1	**Modem 8&16**—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. **UART/AC97**—Reserved
3	SHDIR	**Modem 8&16**—Shift Direction. 0 = msb first 1 = lsb first **UART/AC97**—Reserved

Bit	Name	Description
4	RxDCD	**UART**—Receiver DCD control. 0 = scc_dcd_b input is ignored 1 = scc_dcd_b input is effective **Modem 8&16/AC97**—Reserved
5:7	SIM[2:0]	PSC operation mode. **CAUTION**: When the operating mode change occurs, all Rx/Tx and error statuses are reset. Rx and Tx are disabled. 000 = UART 001 = 8-bit soft modem 010 = 16-bit soft modem 011 = AC97 100 = illegal 101 = illegal 110 = illegal 111 = Illegal

15.2.20 Rx FIFO Number of Data (2x58)—RFNUM[*1, 2, 3*]

Table 15-38 Rx FIFO Number of Data (2058, 2458, 2858)—RFNUM[*1, 2, 3*]

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

Bit	Name	Description
0:6	—	Reserved
7:15	COUNT	Number of data bytes in the Rx FIFO.

15.2.21 Tx FIFO Number of Data (2x5C)—TFNUM[*1, 2, 3*]

Table 15-39 Tx FIFO Number of Data (205C, 245C, 285C)—TFNUM[*1, 2, 3*]

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

Bit	Name	Description
0:6	—	Reserved
7:15	COUNT	Number of data bytes in the Tx FIFO.

15.2.22 Rx FIFO Data (2x60)—RFDATA[*1, 2, 3*]

The RFDATA register (2060, 2460, 2860) is for test use and not used in normal operation.

15.2.23 Rx FIFO Status (2x64)—RFSTAT[*1, 2, 3*]

Table 15-40 Rx FIFO Status (2064, 2464, 2864)—RFSTAT[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				Frame[3]	Frame[2]	Frame[1]	Frame[0]	Rsvd	Error	UF	OF	FR	FULL	ALARM	EMPTY
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 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 FIFOs, 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 SmartComm or CPU. See Section 15.2.25 for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

15.2.24 Rx FIFO Control (2x68)—RFCNTL[*1, 2, 3*]

Table 15-41 Rx FIFO Control (2068, 2468, 2868)—RFCNTL[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved		WFR	COMP	FRAME	GR[2:0]		
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 15.2.25 for details.

15.2.25 Rx FIFO Alarm (2x6E)—RFALARM[*1, 2, 3*]

Table 15-42 Rx FIFO Alarm (206E, 246E, 286E)—RFALARM[*1, 2, 3*]

	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	"Amost full" threshold level. Amount of empty space remaining in the Rx FIFO at which the ALARM bit in the status register goes high/active.

The standard FIFO Controller used in MGT5100 peripherals, such as the PSCs, was designed to control either:

- a transmit (Tx) FIFO
- a receive (Rx) FIFO

This is determined by one of the inputs being hard-wired either high or low.

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:
  – SmartComm
  – CPU (alternate)
- GRANULARITY specifies a threshold at which the interrupt goes away.

15.2.25.1 Rx FIFO

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.

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

where:

the Rx FIFO size is 512 bytes (Tx FIFO size is 256 bytes)

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.

Unlike the ALARM value, GRANULARITY value represents a number of data bytes, **not** empty space.

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 (almost empty).

When SmartComm is servicing the FIFOs, this process works well. However, if the CPU is servicing the FIFOs, the interrupt has no hysteresis. For Example, the ALARM level is used for both activating and deactivating the CPU interrupt.

When using SmartComm you must specify a non-zero GRANULARITY to get FIFO underrun errors. This is due to its internal pipelining. SmartComm does not immediately stop accessing the FIFO when the FIFO interrupt goes away.

15.2.25.2 Tx FIFO

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.

The FIFO interrupt stays active until SmartComm writes enough data into the Tx FIFO to reach the GRANULARITY level. Once the GRANULARITY level is reach, the interrupt goes away.

Tx FIFO GRANULARITY is specified in terms of empty bytes, **not** a number of data bytes as with the Rx FIFO. 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)

15.2.26 Rx FIFO Read Pointer (2x72)—RFRPTR[*1, 2, 3*]

Table 15-43 Rx FIFO Read Pointer (2072, 2472, 2872)—RFRPTR[*1, 2, 3*]

	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.

15.2.27 Rx FIFO Write Pointer (2x76)—RFWPTR[*1, 2, 3*]

Table 15-44 Rx FIFO Write Pointer (2076, 2476, 2876)—RFWPTR[*1, 2, 3*]

	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.

15.2.28 Rx FIFO Last Read Frame PTR (2x7A)—RFLRFPTR[*1, 2, 3*]

Table 15-45 Rx FIFO Last Read Frame PTR (207A, 247A, 287A)—RFLRFPTR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				LFP											
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.

15.2.29 Rx FIFO Last Write Frame PTR (2x7C)—RFLWFPTR[*1, 2, 3*]

Table 15-46 Rx FIFO Last Write Frame PTR (207C, 247C, 287C)—RFLWFPTR[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				LFP											
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.

15.2.30 Tx FIFO Data (2x80)—TFDATA[*1, 2, 3*]

The TFDATA register (2080, 2480, 2880) is for test use and not used in normal operation.

15.2.31 Tx FIFO Status (2x84)—TFSTAT[*1, 2, 3*]

Table 15-47 Tx FIFO Status (2084, 2484, 2884)—TFSTAT[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				Frame[3]	Frame[2]	Frame[1]	Frame[0]	Rsvd	Error	UF	OF	FR	FULL	ALARM	EMPTY
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved

Bit	Name	Description
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.
14	ALARM	The FIFO is requesting service from either SmartComm or CPU. See Section 15.2.25 for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

15.2.32 Tx FIFO Control (2x88)—TFCNTL[*1, 2, 3*]

Table 15-48 Tx FIFO Control (2088, 2488, 2888)—TFCNTL[*1, 2, 3*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved		WFR	COMP	FRAME	GR[2:0]		
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. 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 15.2.25 for details.

15.2.33 Tx FIFO Alarm (2x8E)—TFALARM[*1, 2, 3*]

Table 15-49 Tx FIFO Alarm (208E, 248E, 288E)—TFALARM[*1, 2, 3*]

	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 empty” threshold level. Amount of data remaining in the Tx FIFO at which the ALARM bit in the status register goes high/active.

15.2.34 Tx FIFO Read Pointer (2x92)—TFRPTR[*1, 2, 3*]

Table 15-50 Tx FIFO Read Pointer (2092, 2492, 2892)—TFRPTR[*1, 2, 3*]

	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

15.2.35 Tx FIFO Write Pointer (2x96)—TFWPTR[*1, 2, 3*]

Table 15-51 Tx FIFO Write Pointer (2096, 2496, 2896)—TFWPTR[*1, 2, 3*]

	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

15.2.36 Tx FIFO Last Read Frame PTR (2x9A)—TFLRFPTR[*1, 2, 3*]

Table 15-52 Tx FIFO Last Read Frame PTR (209A, 249A, 289A)—TFLRFPTR[*1, 2, 3*]

	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

15.2.37 Tx FIFO Last Write Frame PTR (2x9C)—TFLWFPTR[*1, 2, 3*]

Table 15-53 Tx FIFO Last Write Frame PTR (209C, 249C, 289C)—TFLWFPTR[*1, 2, 3*]

	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 Module Signal Definitions

Figure 15-2 shows both external and internal signal groups.

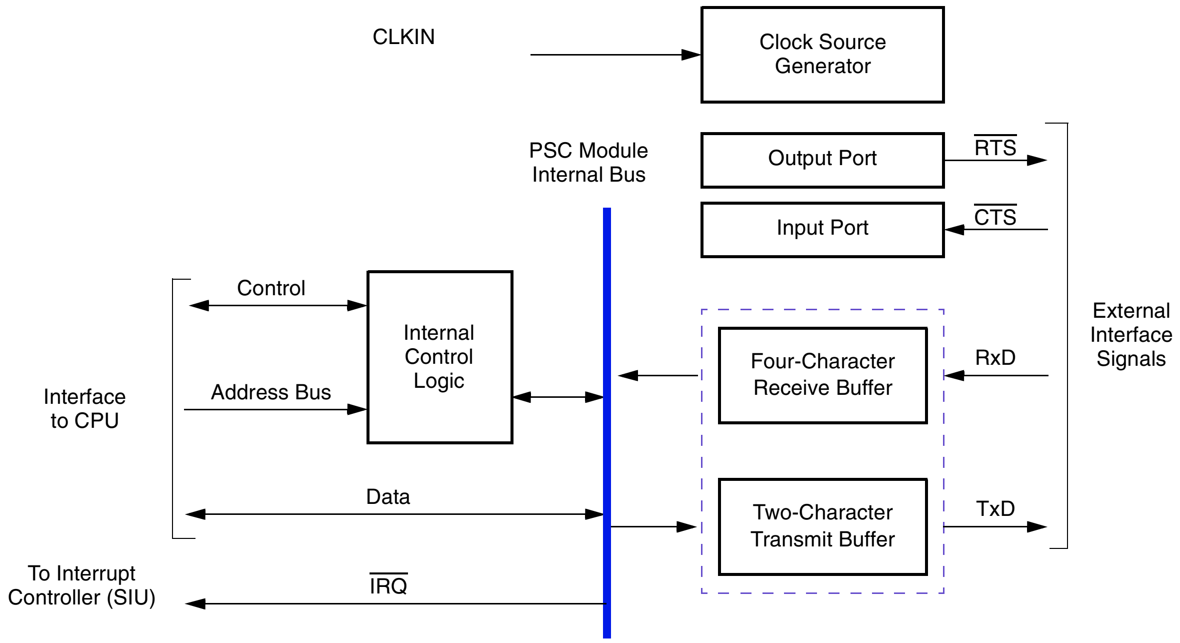


Figure 15-2 Block Diagram—PSC

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*n* bits. The interrupt level of a PSC module is programmed in the Interrupt Controller in the system integration module (SIU). The PSC uses the autovector for the programmed interrupt level.

The PSC can automatically transfer data using the DMA, rather than interrupting the core. When IMR[FFULL] is 1 and Rx FIFO is full, it can send an interrupt to a DMA channel so the FIFO data can be transferred to memory.

Table 15-54 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 15-54 PSC Module Signals

Signal	Description
TxD	Transmitter Serial Data Output—In UART mode, 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. For modem mode, TxD is held low when Tx is disabled or idle. Data is shifted out on TxD on the rising edge of the clock signal driving $\overline{\text{CTS}}$ input. Transfers can be specified as either lsb or msb 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. For modem mode, data received on RxD is sampled on the falling edge of the clock signal driving PSC1 and PSC $\overline{\text{CTS}}$ input. Transfers can be specified as either lsb or msb first.
$\overline{\text{CTS}}$	Clear-to-Send—UART mode. This input can generate an interrupt on a change of state.
$\overline{\text{RTS}}$	Request-to-Send—UART mode. 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.
CLK	In modem mode, CLK must be driven by the serial bit-clock from the external CODEC or AC97 Controller.
Frame	In AC97 mode FRAME is the frame sync, or start-of-frame (SOF), output to the external AC97 Controller. When this mode is used, the AC97 BIT_CLK, which is input on CLK, is divided by 256. In modem8 or modem16 mode FRAME is the frame sync input from the external Codec.

Figure 15-3 shows a signal configuration for a PSC/RS-232 interface.

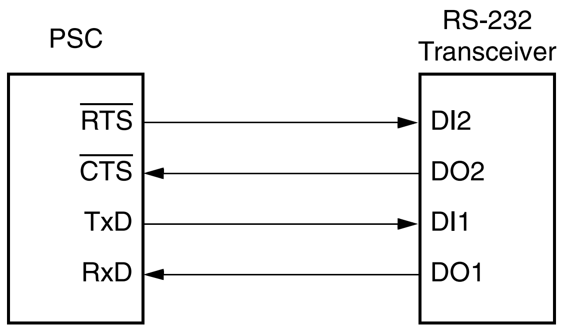


Figure 15-3 PSC/RS-232 Interface

Figure 15-3 shows a signal configuration for a PSC1/CODEC and PSC2/CODEC interface.

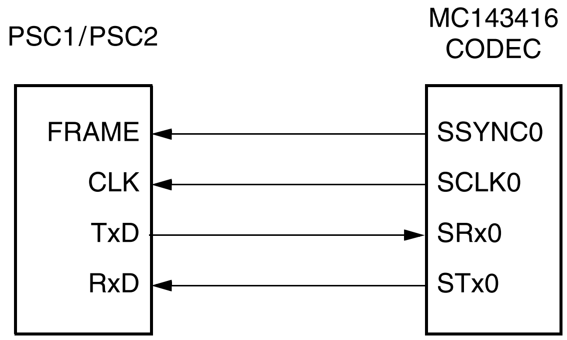


Figure 15-4 PSC1/CODEC and PSC2/CODEC Interface

Figure 15-5 shows a signal configuration for a PSC1/CODEC and PSC2/CODEC interface. An MGT5100 general-purpose I/O (GPIO) is used as a reset to the AC97 device.

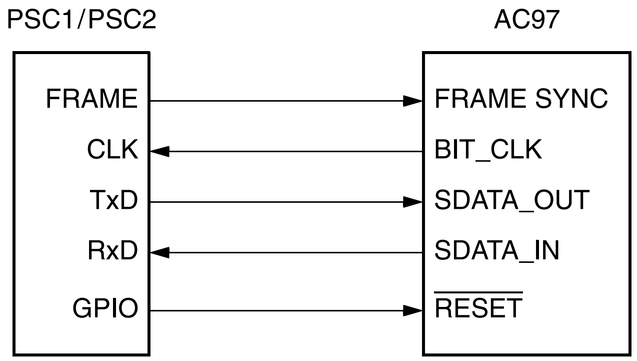


Figure 15-5 PSC1/AC97 and PSC2/AC97 Interface

15.4 PSC Operation

This section describes operation of the clock source generator, transmitter, and receiver.

15.4.1 Transmitter/Receiver Clock Source

CLKIN is the IP Bus clock and 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 Clock Generator cannot produce standard baud rates if CLKIN is used, so the 16-bit divider should be used.

15.4.1.1 Programmable Divider

Figure 15-6 shows the PSC Tx and Rx clock source. CLKIN supplies an asynchronous clock source that is divided by 32, then divided by the 16-bit value programmed in CTUR*n* and CTLR*n*. See Section 15.4.1.3.

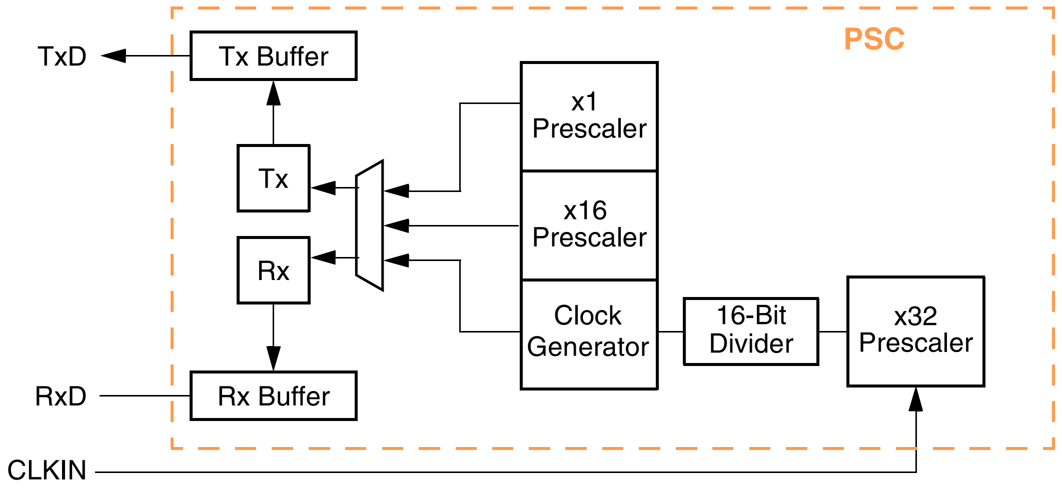


Figure 15-6 Clocking Source Diagram

15.4.1.2 Calculating Baud Rates

The following sections describe how to calculate baud rates.

15.4.1.3 CLKIN Baud Rates

When CLKIN is the PSC clocking source, it goes through a divide-by-32 prescaler. It then passes through the 16-bit divider of the concatenated CTUR*n* and CTLR*n* registers. Using a 54 MHz CLKIN, the baud-rate calculation is as follows:

$$\text{Baudrate} = \frac{54\text{ MHz}}{[32 \text{ x divider}]}$$

Let baud rate = 9600; the divider can be calculated as follows:

$$\text{Divider} = \frac{54\text{ MHz}}{[32 \text{ x } 9600]} = 176\ (\text{decimal}) = 0\text{x}00\text{B}0$$

Therefore CTUR*n* = 0x00 and CTLR*n* = 0xB0.

15.4.1.4 External Clock

Use of an external clock is not an option for the PSCs on the MGT5100.

15.4.2 Transmitter and Receiver Operating Modes

Figure 15-7 is a functional block diagram of Tx and Rx showing command and operating registers. The sections below give an overview; details are given in Section 15.2.

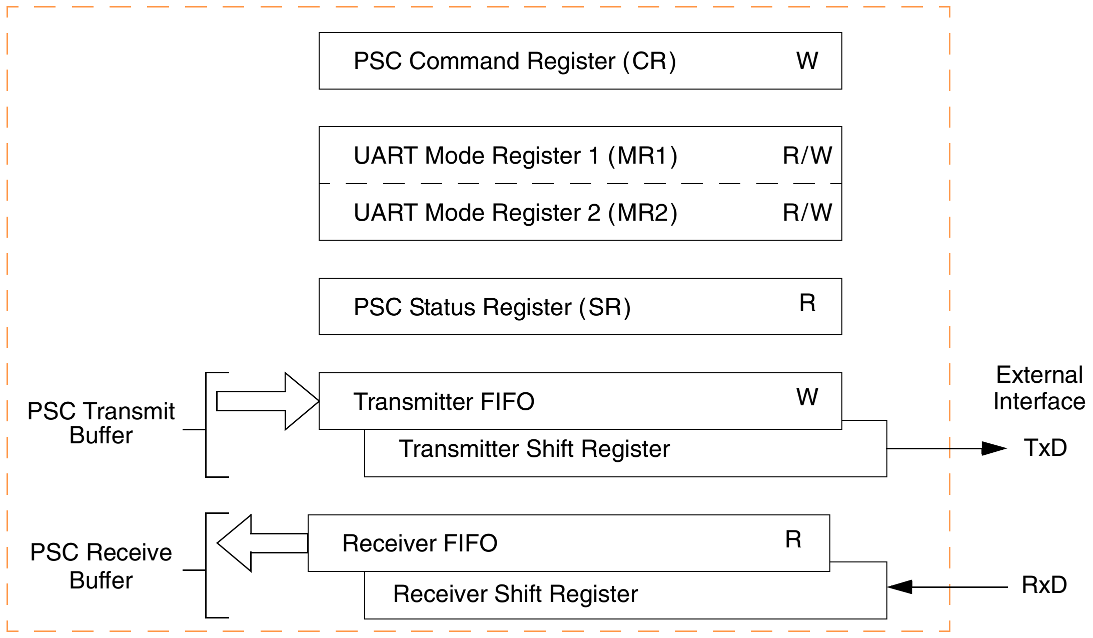

PSC Command Register (CR) W
UART Mode Register 1 (MR1) R/W
UART Mode Register 2 (MR2) R/W
PSC Status Register (SR) R
PSC Transmit Buffer
Transmitter FIFO W
Transmitter Shift Register
External Interface
TxD
PSC Receive Buffer
Receiver FIFO R
Receiver Shift Register
RxD

Figure 15-7 Functional Diagram—Tx and Rx

15.4.2.1 Transmitting in UART Mode

After a hardware reset, PSC1, 2, and 3 are in UART mode. The transmitter is enabled through the PSC command register (CR*n*). When it is ready to accept a character, the PSC sets SR*n*[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*n*[TxEMP], is set. Transmission resumes and TxEMP is cleared when the CPU loads a new character into the PSC Tx buffer (TB*n*).

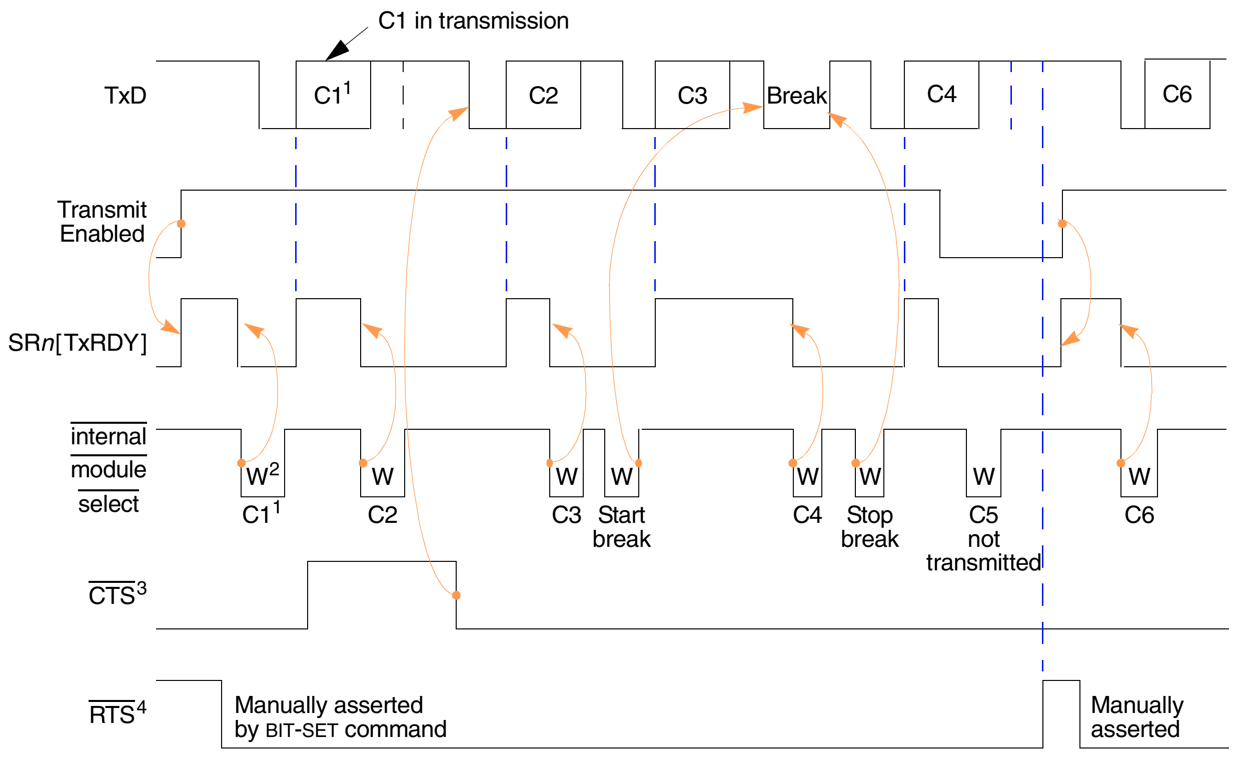
- If the transmitter receives a disable command, it continues until any character in the Tx shift register is completely sent.
- If the transmitter is reset through a software command, operation stops immediately (see Section 15.4.2.1).

NOTE: The transmitter is re-enabled through the CR*n* 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 15-8 shows the transmitter functional timing information.



NOTE:
1. Cn = transmit characters
2. W = write
3. MR2n[TxCTS] = 1
4. MR2n[TxRTS] = 1

Figure 15-8 Timing Diagram—Transmitter

15.4.2.2 Transmitter in Modem Mode

After a hardware reset, PSC1, 2, and 3 are in UART mode. PSC1, 2, or 3 can be put in one of the modem modes by writing the appropriate value for SICR[SIM]. The other SICR fields should be initialized at the same time. Set the Tx FIFO ALARM register level.

Figure 15-9 shows a CODEC interface (lsb first) timing diagram example.

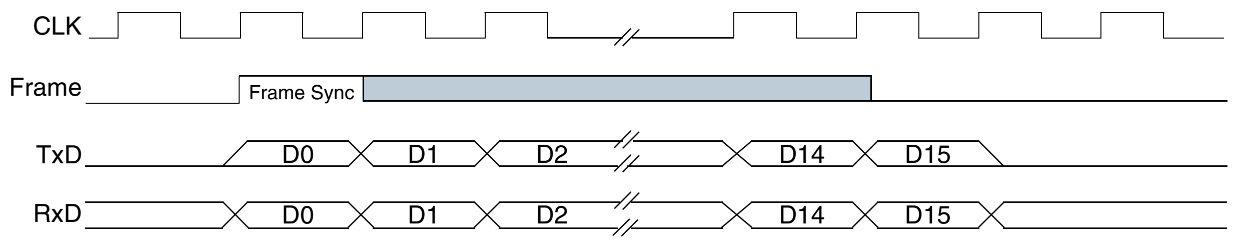


Figure 15-9 Timing Diagram—16-Bit CODEC Interface (lsb First)

Figure 15-10 shows a CODEC interface (msb first) timing diagram example.

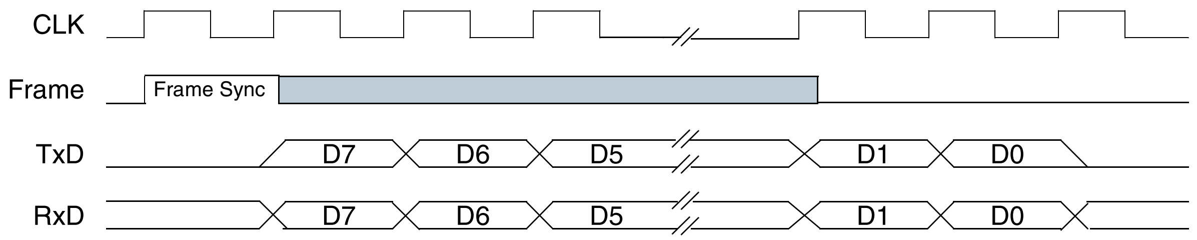


Figure 15-10 Timing Diagram—8-Bit CODEC Interface (msb First)

Figure 15-11 shows a AC97 interface timing diagram example.

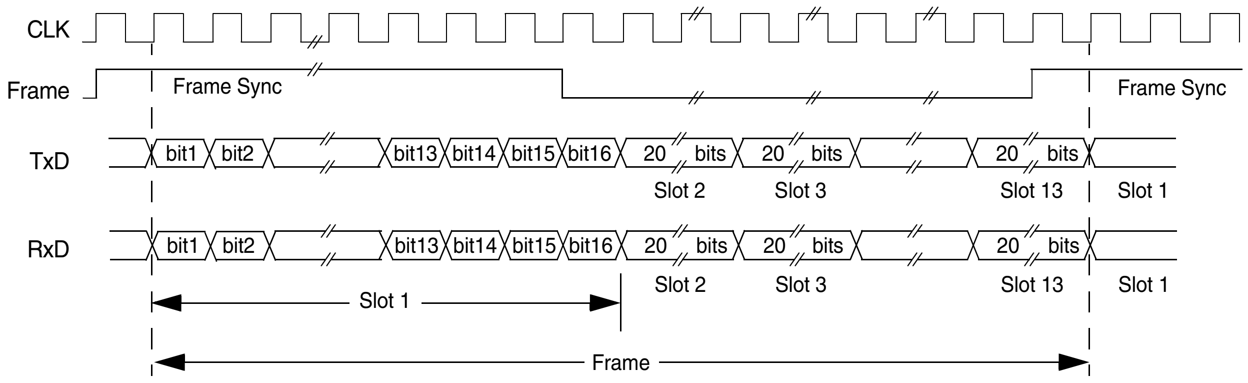


Figure 15-11 Timing Diagram—AC97 Interface

For more AC97 Controller interface information, refer to the *Audio CODEC'97 Component Specification.*

When interfaced to an 8- or 16-bit CODEC (SICR[MODE]=001 or 010), PSC1, 2, and 3 starts to send a sample during the 1-bit clock cycle after the rising edge of frame sync, according to the value of SICR[DTS1]. The frame sync pulse width makes no difference. SICR[SHDIR] controls whether bits are shifted out msb or lsb first. After the 8- or 16-bit sample is sent, 0s are sent until the next frame sync.

When PSC1 or PSC2 interfaces to an AC97 Controller (SICR[MODE]=011), the PSC starts transmitting time slot 1 data, one bit-clock cycle after the rising edge of frame sync, regardless of the value of SICR[DTS1]. 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).

Because Rx data is sampled on the falling edge of the bit-clock, 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.

15.4.2.3 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.

15.4.2.4 Receiver in UART Mode

After a hardware reset, PSC1, 2, and 3 are in UART mode. The receiver is enabled through its CR*n*, as described in Section 15.2.6. Figure 15-12 shows the receiver functional timing.

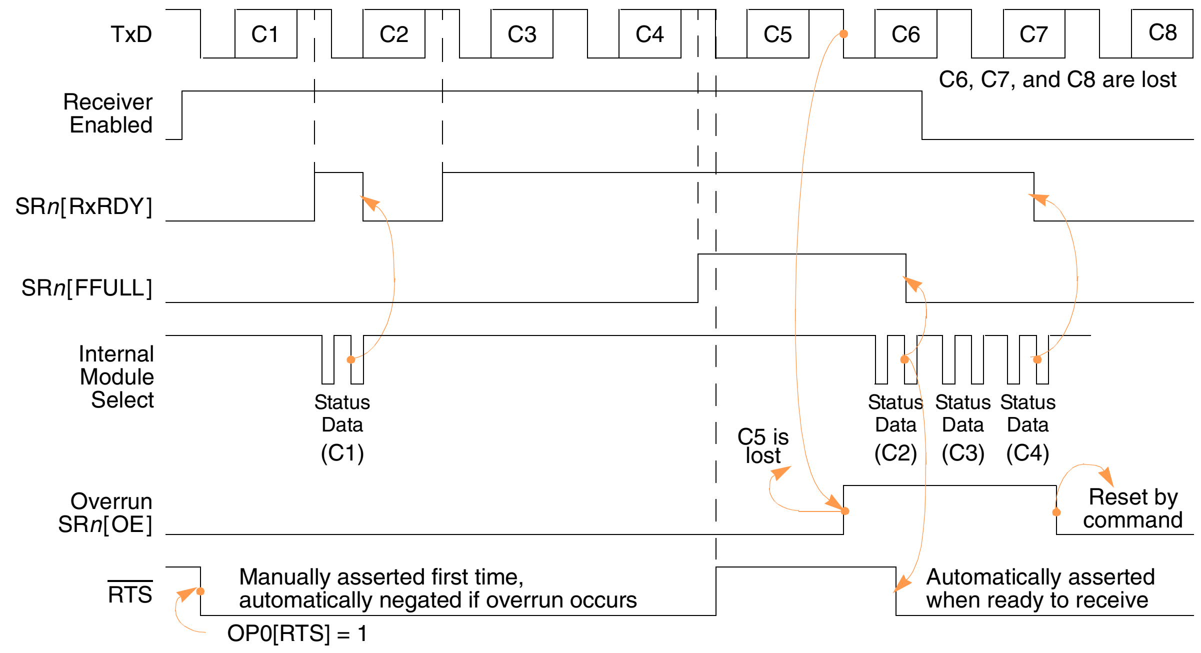


Figure 15-12 Timing Diagram—Receiver

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*n*[RxRDY] is set. If the character is less than 8bits, the most significant unused bits in the receiver holding register are cleared.

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 (OE), and received break (RB) conditions set respective error and break flags in SR*n* at the received character boundary and are valid only if SR*n*[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*n*[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*n* error bits and SR*n*[RxRDY].
- If the break lasts until the next character time, the receiver places an all-0 character into the Rx FIFO and sets SR*n*[RB, RxRDY].

15.4.2.5 Receiver in Modem Mode

After a hardware reset, PSC is in UART mode. Modem modes are chosen by setting SICR[MODE]. Other SICR fields should be initialized at the same time. Set the Rx FIFO ALARM level.

In modem mode, the serial bit-clock CLK is always an input to the PSC. When interfacing to an 8- or 16-bit CODEC, the frame sync FRAME is also an input to the PSC. When an AC97 Controller is used, the frame sync FRAME is an output.

Figure 15-9 and Figure 15-10 show PSC-CODEC interface timing diagrams. Figure 15-11 shows an example timing diagram for the PSC1 and PSC2-AC97 interface.

When an 8- or 16-bit CODEC is specified (SICR[MODE]=001 or 010), the PSC begins receiving a sample at either:

- the rising edge of frame sync, or
- 1 bit-clock cycle after the rising edge of frame sync, according to the SICR[DTS1] value.

The frame sync pulse width makes no difference. SICR[SHDIR] controls whether the sample is shifted in msb or lsb first. After the 8- or 16-bit sample is received, the receiver shift register shuts off until the next frame sync occurs.

When an AC97 Controller is specified (SICR[MODE]=011), 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.

- 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.

15.4.2.6 FIFO Stack

The receive FIFO stack consists of the FIFO and a receiver shift register connected to the RxD (see Figure 15-7). 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.

Section 15.2.25 describes how the Tx and Rx FIFO “almost empty” and “almost full” thresholds work.

Block error mode is always selected because MR1*n*[ERR] is hard-wired high. In block mode SR*n* 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*n* 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 FIFOs are unaffected, and SR*n*[OE] 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.

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.
- 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.

15.4.3 Looping Modes

The PSC can be configured to operate in various loopback modes as shown in Figure 15-12. These modes are useful for local and remote system diagnostic functions and can be used by in modem mode as well as UART mode. The modes are described below and in Section 15.2.

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.

15.4.3.1 Automatic Echo Mode

In automatic echo mode, shown in Figure 15-9, the PSC automatically resends 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.

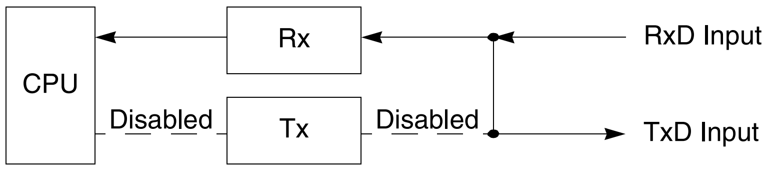


Figure 15-13 Automatic Echo

Because the transmitter is inactive, SR*n*[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.

15.4.3.2 Local Loop-Back Mode

Figure 15-14 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.

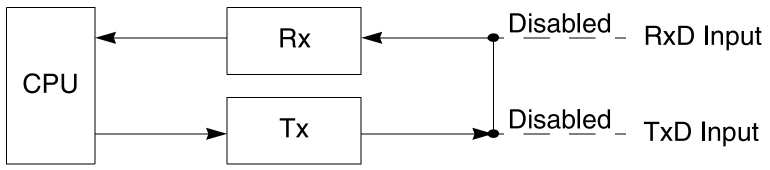


Figure 15-14 Local Loop-Back

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.

15.4.3.3 Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 15-15, 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.

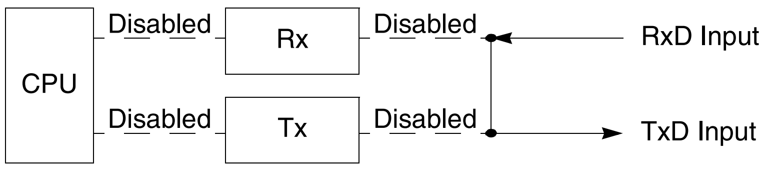


Figure 15-15 Remote Loop-Back

15.4.4 Multidrop Mode

Setting MR1*n*[PM] programs the PSC to operate in a WakeUp 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*n*[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 15-16 shows functional timing information for multidrop mode.

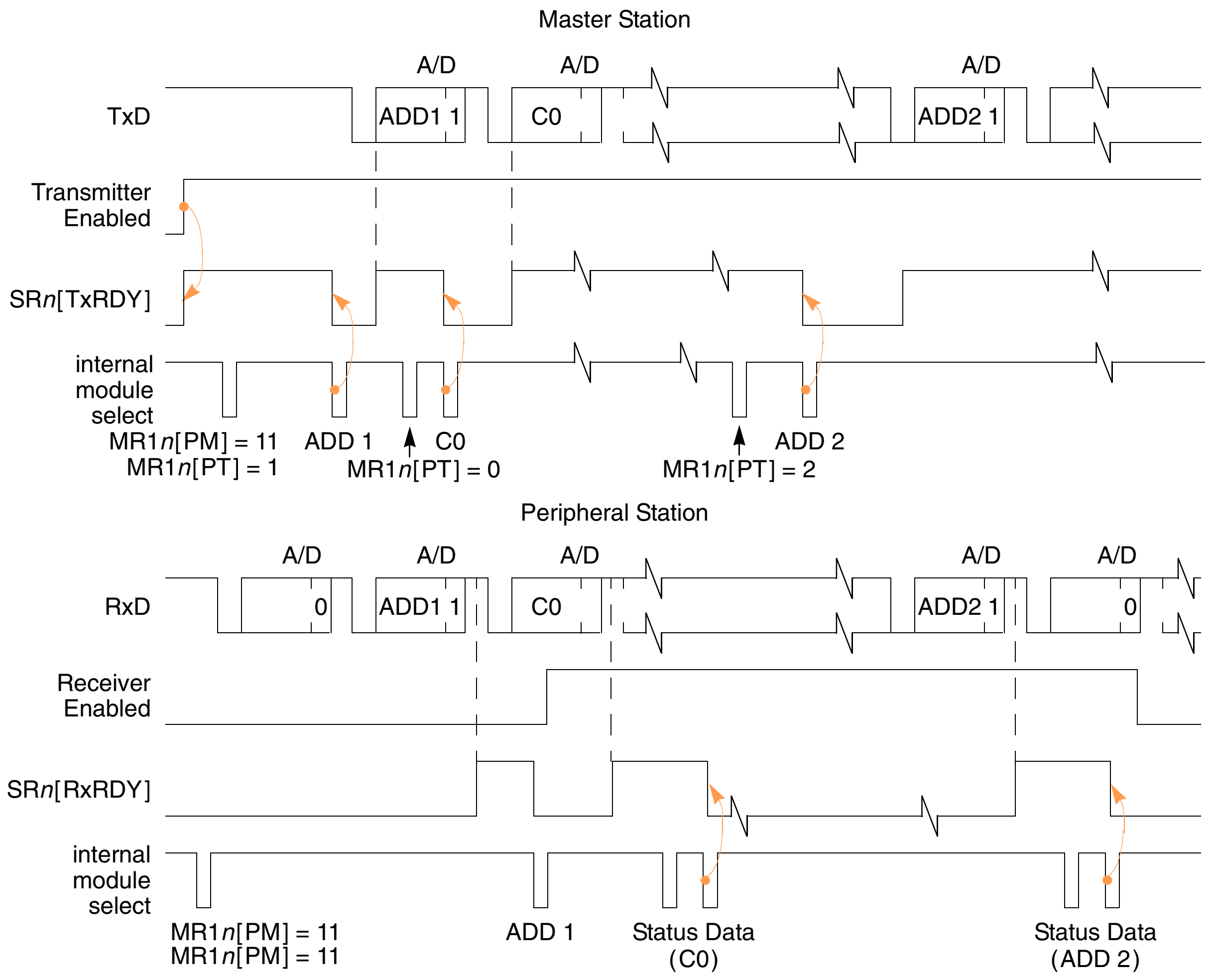


Figure 15-16 Timing Diagram—Multidrop Mode

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*n*[PT]. MR1*n* 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*n*[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.

15.4.5 Bus Operation

This section describes PSC module bus operation during read, write, and interrupt acknowledge cycles.

15.4.5.1 Read Cycles

The PSC module responds to reads with byte data. Reserved registers return 0s.

15.4.5.2 Write Cycles

The PSC module accepts write data as bytes. Write cycles to read-only or reserved registers complete normally without exception processing, but data is ignored.

NOTE: The PSC module is accessed by the CPU with zero wait states, as CLKIN is used for the PSC module.

15.4.6 Programming

Figure 15-17 through Figure 15-21 shows the software programming flowchart. The following items are shown:

- PSC module initialization—(sheets 1 and 2) consists of SINIT and CHCHK. Before SINIT is called at system initialization, the calling routine allocates 2 words on the system stack. On return to the calling routine, SINIT passes PSC status data on the stack. If SINIT finds no errors, Tx and Rx are enabled. SINIT calls CHCHK to perform the checks. When called, SINIT places the PSC in local loop-back mode and checks for the following errors:
  - Transmitter never ready
  - Receiver never ready
  - Parity error
  - Incorrect character received
- I/O driver routine—(sheets 4 and 5) consists of INCH, the terminal input character routine that gets a character from the receiver, and OUTCH, which sends a character to the transmitter.
- Interrupt handling—(sheet 4) consists of SIRQ, which is executed after the PSC module generates an interrupt caused by a change-in-break (beginning of break). SIRQ then clears the interrupt source, waits for the next change-in-break interrupt (end of break), clears the interrupt source again, then returns from exception processing to the system monitor.

15.4.6.1 PSC Module Initialization Sequence

NOTE: PSC module registers can be accessed by word or byte operations, but only data Byte D[7:0] is valid.

Table 15-55 shows the PSC module initialization sequence.

Table 15-55 PSC Module Initialization Sequence

Register	Setting
CR*n*	Reset the receiver and transmitter—UART or modem modes. Reset the mode pointer (MISC[2–0] = 0b001)—UART or modem modes.
IVR*n*	Program the vector number for a PSC module interrupt—UART or modem modes.
IMR*n*	Enable the preferred interrupt sources—UART or modem modes.
ACR*n*	Initialize the input enable control (IEC bit)—UART mode only.
CSR*n*	Select the receiver and transmitter clock. Use timer as source if required—UART mode only.
MR1*n*	If preferred, program operation of receiver ready-to-send (RxRTS bit)—UART mode only. Select receiver-ready or FIFO-full notification (RxRDY/FFULL bit)—UART or modem modes. Select parity mode and type (PM and PT bits)—UART mode only. Select number of bits per character (B/Cx bits)—UART mode only.
MR2*n*	Select the mode of operation (CMx bits)—UART or modem modes. If preferred, program operation of transmitter ready-to-send (TxRTS)—UART mode only. If preferred, program operation of clear-to-send (TxCTS bit)—UART mode only. Select stop bit length (SBx bits)—UART mode only.

Table 15-55 PSC Module Initialization Sequence (continued)

Register	Setting
RFALARM	UART or modem modes—Choose Rx FIFO “almost full” threshold level.
PCR	
TFALARM	UART or modem modes—Choose Tx FIFO “almost empty” threshold level.
SICR	Modem mode only Choose the desired modem mode. Choose whether msb or lsb is to be transferred first. Choose 0- or 1-bit delay between rising edge of frame sync and first bit of time slot 1. Activate/deactivate AC97 warm and cold resets.
CR*n*	Enable the receiver and transmitter—UART or modem modes.

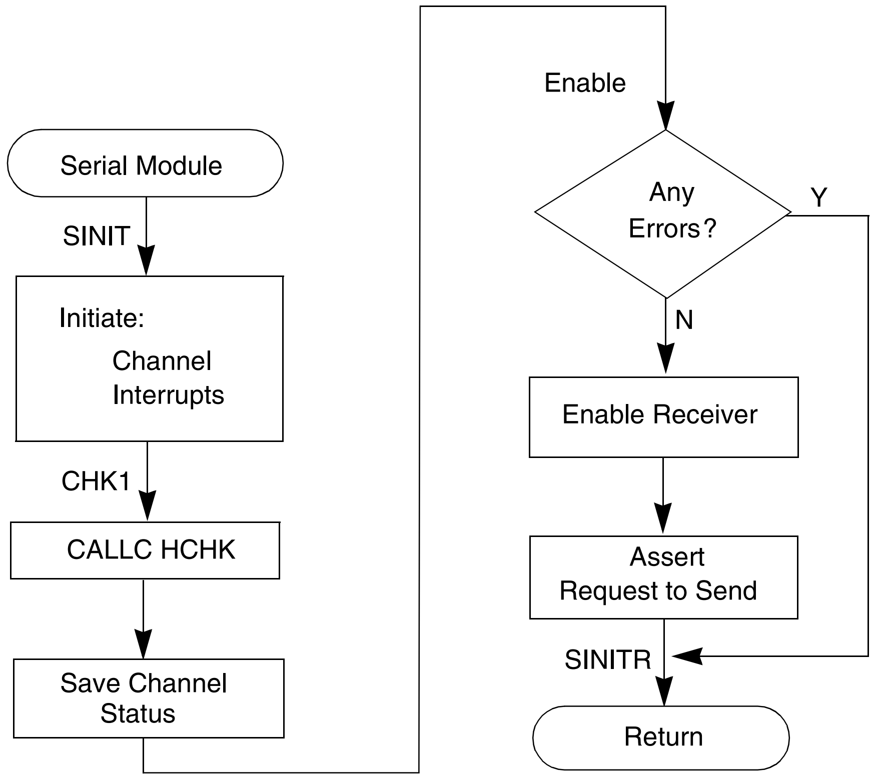


Figure 15-17 Programming Flowchart—UART Mode (sheet 1 of 5)

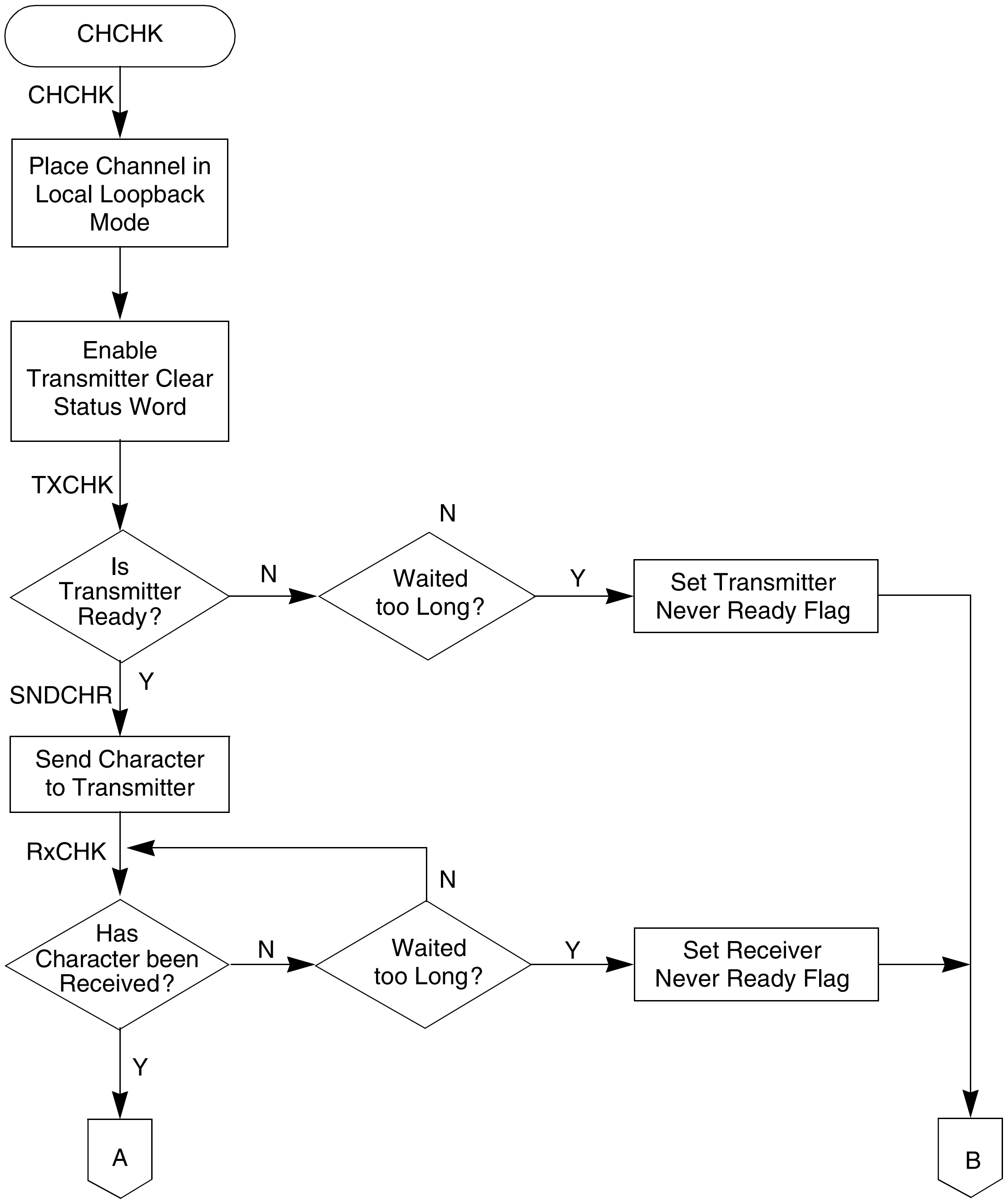

CHCHK
CHCHK
Place Channel in Local Loopback Mode
Enable Transmitter Clear Status Word
TXCHK
Is Transmitter Ready?
N
Waited too Long?
N
Y
Set Transmitter Never Ready Flag
Y
SNDCHR
Send Character to Transmitter
RxCHK
N
Has Character been Received?
N
Waited too Long?
Y
Set Receiver Never Ready Flag
Y
A
B

Figure 15-18 Programming Flowchart—UART Mode (sheet 2 of 5)

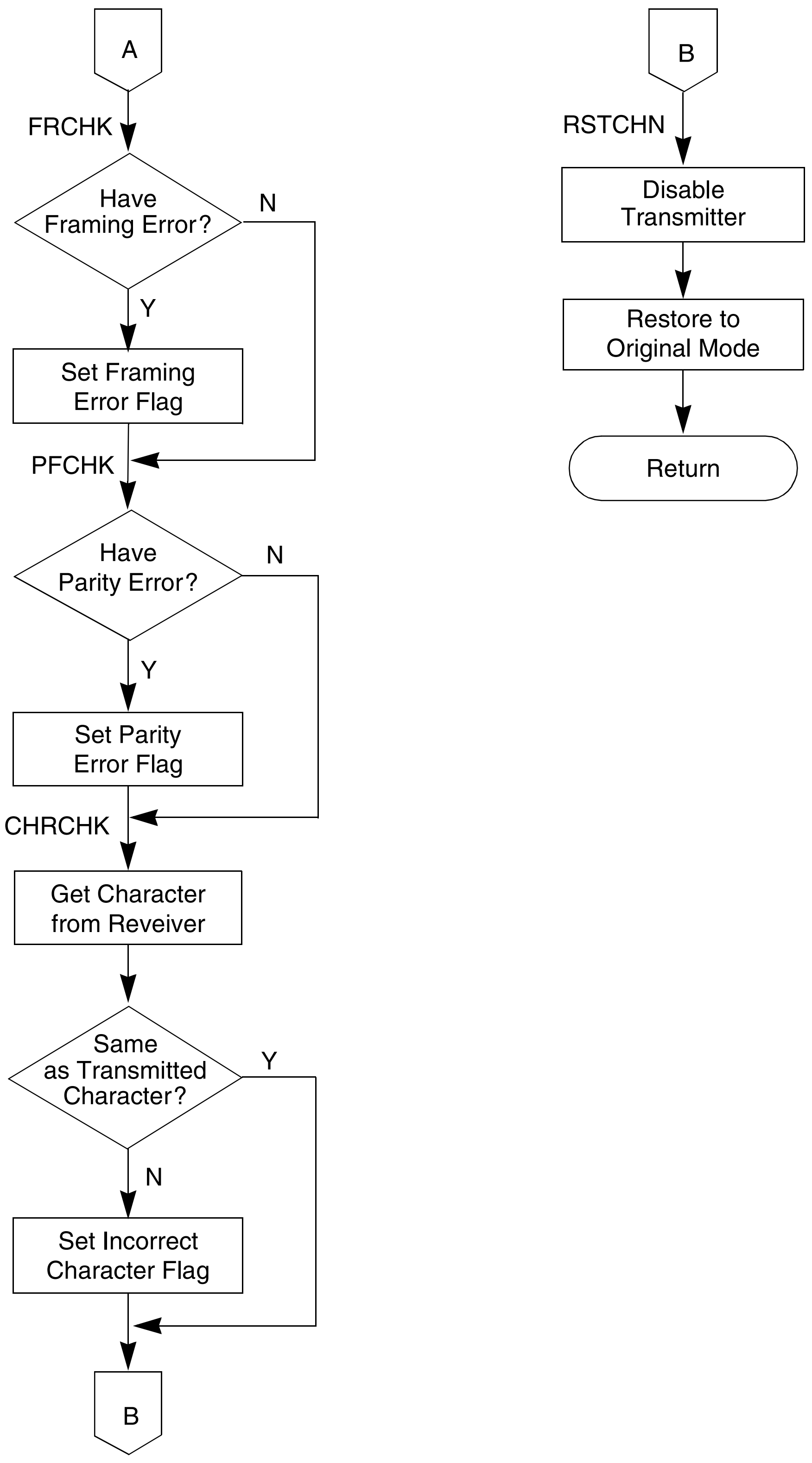


Figure 15-19 Programming Flowchart—UART Mode (sheet 3 of 5)

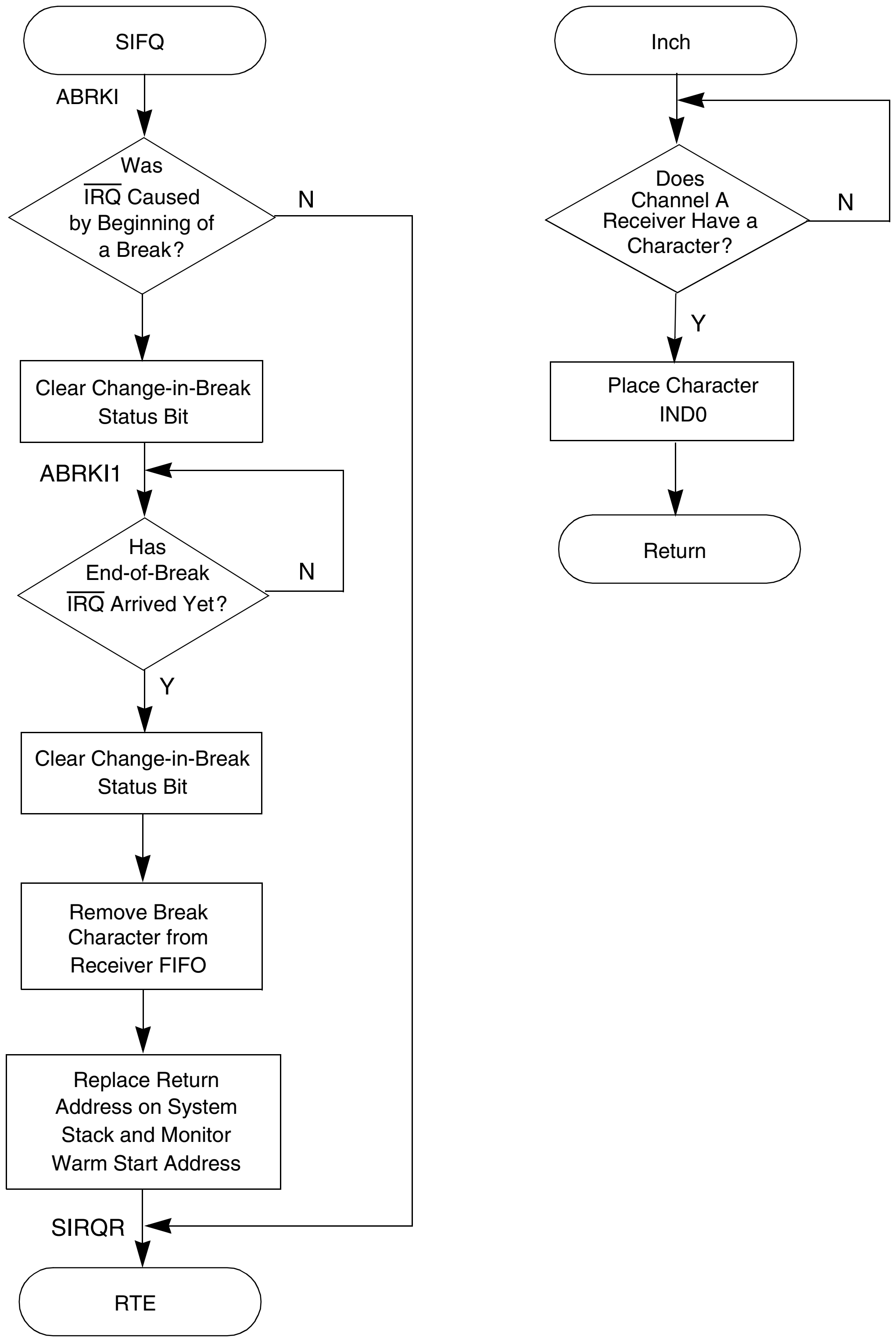

SIFQ
ABRKI
Was $\overline{\text{IRQ}}$ Caused by Beginning of a Break?
N
Clear Change-in-Break Status Bit
ABRKI1
Has End-of-Break $\overline{\text{IRQ}}$ Arrived Yet?
N
Y
Clear Change-in-Break Status Bit
Remove Break Character from Receiver FIFO
Replace Return Address on System Stack and Monitor Warm Start Address
SIRQR
RTE
Inch
Does Channel A Receiver Have a Character?
N
Y
Place Character IND0
Return

Figure 15-20 Programming Flowchart—UART Mode (sheet 4 of 5)

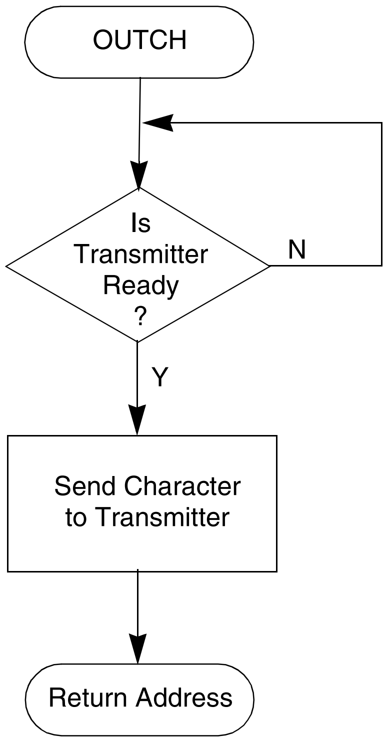


Figure 15-21 Programming Flowchart—UART Mode (sheet 5 of 5)

SECTION 16
INFRARED DATA ASSOCIATION (IRDA) INTERFACE

16.1 Overview

This document contains the following section:

- IrDA Registers—MBAR + 0x2C00

16.1.1 Features

IrDA features include:

- TBD

16.2 IrDA Registers—MBAR + 0x2C00

IrDA uses 35 32-bit registers. These registers are located at an offset from MBAR of 0x2C00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x2C00 + register address**

Hyperlinks to the IrDA registers are provided below:

- SIR Mode 1 (2C00)—MR1
- MIR/FIR Modes (2C00)—MR1
- SIR Mode 2 (2C00)—MR2
- MIR/FIR Modes (2C00)—MR2
- SIR Status (2C04)—SR
- MIR/FIR Status (2C04)—SR
- SIR Mode (2C04)—CSR
- Other Modes (2C04)—CSR
- Command Register, All Modes (2C08)—CR
- SIR/MIR/FIR Rx Buffers (2C0C)—RB
- SIR/MIR/FIR Tx Buffers (2C0C)—TB
- Input Port SIR/MIR/FIR Change (2C10)—IPCR
- Auxiliary Control (2C10)—ACR
- Interrupt SIR Status (2C14)—ISR
- Interrupt MIR/FIR Status (2C14)—ISR
- Interrupt SIR Mask (2C14)—IMR
- Interrupt MIR/FIR Mask (2C14)—IMR
- Counter Timer SIR Upper Bytes (2C18)—CTUR
- Counter Timer MIR/FIR Upper Bytes (2C18)—CTUR
- Counter Timer SIR Lower Bytes (2C1C)—CTLR
- Counter Timer MIR/FIR Lower Bytes (2C1C)—CTLR
- Interrupt Vector (2C30)—IVR
- IrDA Input Port (2C34)—IP
- IrDA Output Port Bit Set (2C38)—OP1
- IrDA Output Port Bit Reset (2C3C)—OP0
- IrDA SIR Control (2C40)—SICR
- IrDA MIR/FIR Control (2C40)—SICR
- Infrared SIR Control 1 (2C44)—IRCR1
- Infrared MIR/FIR Control 1 (2C44)—IRCR1
- Infrared SIR Control 2 (2C48)—IRCR2
- Infrared MIR/FIR Control 2 (2C48)—IRCR2
- Infrared SIR Divide (2C4C)—IRSDR
- Infrared SIR Other Divide (2C4C)—IRSDR
- Infrared MIR Divide (2C50)—IRMDR
- Infrared MIR Other Divide (2C50)—IRMDR

- Infrared FIR Divide (2C54)—IRFDR
- Infrared FIR Other Divide (2C54)—IRFDR

- Rx FIFO Number of Data (2C58)—RFNUM
- Rx FIFO Status (2C64)—RFSTAT
- Rx FIFO Control (2C68)—RFCNTL
- Rx FIFO Alarm (2C6E)—RFALARM
- Rx FIFO Read Pointer (2C72)—RFRPTR
- Rx FIFO Write Pointer (2C76)—RFWPTR
- Rx FIFO Last Read Frame Pointer (2C7A)—RFL-RFPTR
- Rx FIFO Last Write Frame Pointer (2C7C)—RFL-WFPTR
- Tx FIFO Number of Data (2C5C)—TFNUM
- Tx FIFO Status (2C84)—TFSTAT
- Tx FIFO Control (2C88)—TFCNTL
- Tx FIFO Alarm (2C8E)—TFALARM
- Tx FIFO Read Pointer (2C92)—TFRPTR
- Tx FIFO Write Pointer (2C96)—TFWPTR
- Tx FIFO Last Read Frame Pointer (2C9A)—TFL-RFPTR
- Tx FIFO Last Write Frame Pointer (2C9C)—TFL-WFPTR

16.2.1 Mode Register 1 (2C00)—MR1

Table 16-1 SIR Mode 1 (2C00)—MR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	RxRTS	RxIRQ/ FFULL	Reserved					
RESET:	0	0	0	0	0	0	0	0

Table 16-2 MIR/FIR Modes (2C00)—MR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved	RxIRQ/ FFULL	Reserved					
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RxRTS	**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*n*[TxRTS]. 0 = Receiver has no effect on $\overline{\text{RTS}}$. 1 = When a valid start bit is received, $\overline{\text{RTS}}$ is negated if the IrDA's FIFO is full. $\overline{\text{RTS}}$ is re-asserted when the FIFO has an empty position available. **MIR/FIR**—Reserved
1	RxIRQ/ FFULL	Receiver interrupt select 0 = RxRDY is the source that generates IRQ 1 = FFULL is the source that generates IRQ.
2:7	—	Reserved

16.2.2 Mode Register 2 (2C00)—MR2

Table 16-3 SIR Mode 2 (2C00)—MR2

	msb 0	1	2	3	4	5	6	7 lsb
R/W	CM		TxRTS	TxCTS	Reserved			
RESET:	0	0	0	0	0	0	0	0

Table 16-4 MIR/FIR Modes (2C00)—MR2

	msb 0	1	2	3	4	5	6	7 lsb
R/W	CM		Reserved					
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:1	CM	Channel mode—Selects a channel mode. 00 = Normal 01 = Automatic echo 10 = Local loop-back 11 = Remote loop-back
2	TxRTS	**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. 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 OP[RTS] one bit-time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits. **MIR/FIR**—Reserved
3	TxCTS	**SIR**—Transmitter clear-to-send—If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. TxCTS is not used in modem 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. **MIR/FIR**—Reserved
4:7	—	Reserved

16.2.3 SIR Status Register (2C04)—SR

This is a read-only register.

Table 16-5 SIR Status (2C04)—SR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	RB	FE	PE	OE	TxEMP	TxRDY	FFULL	RxRDY	Reserved							
W	**Used by CSR**															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	RB	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 IrDA clock edges.
1	FE	Framing Error 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.
2	PE	Parity Error—valid only if RxRDY = 1. 0 = No parity error occurred. 1 = If MR1_*n*[PM]=0x (with parity or force parity), corresponding FIFO character was received with incorrect parity. If MR1*n*[PM]=11 (multidrop), PE stores received A/D bit.
3	OE	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. OE 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. OE is cleared by the reset error status command in CR*n*.
4	TxEMP	Transmitter Empty—function depends on which mode is used. 0 = Tx buffer not empty. Either a character is being shifted out, or Tx is disabled. Tx is enabled/disabled by programming CR*n*[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.
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 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. TxRDY clears when the number of data bytes in the Tx FIFO becomes greater than the TFALARM value. This bit only asserts if the Tx is enabled.

Bit	Name	Description
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.
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.
8:15	—	Reserved

16.2.4 MIR/FIR Status Register (2C04)—SR

This is a read-only register.

Table 16-6 MIR/FIR Status (2C04)—SR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	EOF	PHYERR	CRCERR	OE	URERR	TxRDY	FFULL	RxRDY	DEOF	Reserved						
W	Used by CSR															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0	EOF	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.
1	PHYERR	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.
2	CRCERR	CRC error. 0 = No error. 1 = The CRC value was not correct. This bit can be cleared by the reset error command in the CR.
3	OE	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. OE 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. OE is cleared by the RESET ERROR STATUS command in CR*n*.

Bit	Name	Description
4	URERR	Underrun Error 0 = No error 1 = Underrun error occurred. When Tx intended to send, there was no data in the TXFIFO. This bit is cleared by the CR RESET ERROR COMMAND.
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 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. TxRDY clears when the number of data bytes in the Tx FIFO becomes greater than the TFALARM value. This bit only asserts if the Tx is enabled.
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.
7	RxRDY	Receiver Ready 0 = CPU has read Rx buffer and no characters remain in FIFO after this read. 1 = One or more characters were received and are waiting in Rx buffer FIFO.
8	DEOF	Detect EOF 0 = RX has not received an EOF. 1 = RX has received EOF at least one time. Reading SR clears this bit.
9:15	—	Reserved

16.2.5 Clock-Select Register (2C04)—CSR

This is a write-only register. Although the IrDA module has an external clock input port, this port does not come out to a pin on the MGT5100. Therefore the user must avoid writing values of ‘1110’ or ‘1111’ to the RCS or TCS fields in the CSR register. The only valid clock source is a prescaled version of the IP Bus clock.

Table 16-7 SIR Mode (2C04)—CSR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Used by SR															
W	RCS				TCS				Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-8 Other Modes (2C04)—CSR

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

Bit	Name	Description
0:3	RCS	**SIR**—Receiver clock select. Selects the clock source for the receiver channel. 0000 -1101 = Prescaled CLKIN 1110 = INVALID 1111 = INVALID **Other Modes**—Reserved
4:7	TCS	**SIR**—Transmitter clock select. Selects the clock source for the transmitter channel. 0000 -1101 = Prescaled CLKIN 1110 = INVALID 1111 = INVALID **Other Modes**—Reserved
8:15	—	Reserved

16.2.6 Command Register (2C08)—CR

The write-only command register (CR), supplies commands to the IrDA. 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 16-9 Command Register, All Modes (2C08)—CR

	msb 0	1	2	3	4	5	6	7 lsb
R								
W	Reserved	MISC			TC		RC	
RESET:	0	0	0	0	0	0	0	0

Bit	Value	Command	Description
0	—		Reserved
1:3 see note	000	no command	—
	001	reset mode register pointer	Causes mode register pointer to point to MR1*n*.
	010	reset receiver	Immediately disables receiver, clears SR*n*[FFULL,RxRDY], and reinitializes 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 reconfiguring the receiver.
	011	reset transmitter	In SIR mode, the TxEMP and TxRDY bits in SR are cleared. In MIR and FIR mode, the URERR bit is not cleared and the TxRDY is asserted due to no holding data in TX-FIFO.
	100	reset error status	In SIR mode, the RB, FE, PE and OE bits in SR are cleared. In MIR and FIR mode, the PHYERR, CRCERR, OE and URERR are cleared. This command has no effect in MIR and FIR mode.

Bit	Value	Command	Description
1:3 cont.	101	reset break change interrupt	Clears the delta break bit, ISR*n*[DB]. This command has not effect in MIR/FIR modes.
	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. This command has not effect in MIR/FIR modes.
	111	stop break	Causes TxD to go high (mark) within two bit-times. Any characters in the Tx buffer are sent.
4:5 see note	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	In MIR and FIR mode, TX-FIFO can be loaded while the TX is disabled, unlike SIR mode. Therefore, TxRDY behaves the same whether the TX is enabled or not, and is not automatically set when the TX is enabled. The ORERR bit is also not automatically set by enabling the TX.
	10	transmitter disable	In SIR mode, the TxEMP and TxRDY bits are negated. In MIR and FIR mode, the TxRDY bit remains as the current condition. In SIR mode, if a character is being transmitted when the TX becomes disabled, the transmission of the character is completed before the TX becomes inactive. In MIR and FIR mode, if the TX is sending and there are any characters in the TX-FIFO when the TX becomes disabled, the TX continues sending data until the last byte (data with EOF mark) in the current frame.
	11	—	Reserved, do not use.
6:7 see note	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 • RECEIVER ENABLE command enables channel's receiver. • If receiver is already enabled, this command has no effect.
	10	receiver disable	This command disables the RX immediately. This command has no effect if the RX is already disabled. In SIR mode, a character being received when the RX becomes disabled is lost. In MIR and SIR mode, the RX continues to receive the input serial data until it finishes receiving the current frame.
	11	—	Reserved, do not use.
NOTE: This field selects a single command.			

16.2.7 Rx Buffers (2C0C)—RB

This is a read-only register.

Table 16-10 SIR/MIR/FIR Rx Buffers (2C0C)—RB

	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

Bit	Name	Description
0:31	RB	Received data. 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 two bytes at a time, data must be read from bits 0:15. Lower-bit data was received before upper-bit data.

16.2.8 Tx Buffers (2C0C)—TB

This is a write-only register.

Table 16-11 SIR/MIR/FIR Tx Buffers (2C0C)—TB

	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

Bit	Name	Description
0:31	TB	Transmit data. 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 two bytes at a time, data must be written to bits 0:15. Lower-bit data is stored before upper-bit data.

16.2.9 Input Port Change (2C10)—IPCR

The read-only IPCR register shows the current state and change-of-state for the modem control input port.

Table 16-12 Input Port SIR/MIR/FIR Change (2C10)—IPCR

	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
0:1	—	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 1bit duration determined by the CSR, CTUR and CTLR) has occurred at scc_dcd_b 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 scc_cts_b 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 DCD port. This input is double latched and same as DCD of IP. 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 CTS port. This input is double latched and same as DCD of IP. 0 = The current state of the CTS input port is low. 1 = The current state of the CTS input port is high.

16.2.10 Auxiliary Control (2C10)—ACR

The write-only ACR register controls Tx/Rx handshaking.

Table 16-13 Auxiliary Control (2C10)—ACR

	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).

16.2.11 Interrupt Status (2C14)—ISR

The read-only ISR register provides status for all potential interrupt sources. Register contents are 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 16-14 Interrupt SIR Status (2C14)—ISR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved				DB	RxRDY FFULL	TxRDY	Reserved							
W	Used by IMR															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-15 Interrupt MIR/FIR Status (2C14)—ISR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	IPC	Reserved					RxRDY FFULL	TxRDY	DEOF	Reserved						
W	Used by IMR															
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.
1:4	—	Reserved
5	DB	**SIR**—Delta Break **MIR/FIR**—Reserved

Bit	Name	Description
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

16.2.12 Interrupt Mask (2C14)—IMR

The write-only IMR register selects corresponding bits in the ISR that cause an interrupt.

- If one ISR bit sets and the corresponding IMR bit also sets, 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 16-16 Interrupt SIR Mask (2C14)—IMR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Used by ISR															
W	IPC	Reserved				DB	RxRDY FFULL	TxRDY	Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-17 Interrupt MIR/FIR Mask (2C14)—IMR

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Used by ISR															
W	IPC	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.
1:4	—	Reserved
5	DB	**SIR**—Delta Break 0 = DB has no effect on the interrupt. 1 = Enable the interrupt for DB **MIR/FIR**—Reserved

Bit	Name	Description
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 = EOF has no effect on the interrupt. 1 = Enable the interrupt for EOF. **SIR**—Reserved
9:15	—	Reserved

16.2.13 Counter Timer Upper Bytes (2C18)—CTUR

These registers hold the upper bytes of the preload value used by the timer to provide a given baud rate.

Table 16-18 Counter Timer SIR Upper Bytes (2C18)—CTUR

	msb 0	1	2	3	4	5	6	7 lsb
R	CT[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Table 16-19 Counter Timer MIR/FIR Upper Bytes (2C18)—CTUR

	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	CT	**SIR**—Baud rate prescale value. The baud rate is calculated as: Baud rate = (system clock frequency) / (CT[15:0] x 16 x 2) The minimum CT value is 1; 0 denotes the counter stop. **MIR/FIR**—Reserved

16.2.14 Counter Timer Lower Bytes (2C1C)—CTLR

These registers hold the lower bytes of the preload value used by the timer to provide a given baud rate.

Table 16-20 Counter Timer SIR Lower Bytes (2C1C)—CTLR

	msb 0	1	2	3	4	5	6	7 lsb
R	CT[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Table 16-21 Counter Timer MIR/FIR Lower Bytes (2C1C)—CTLR

	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	CT	**SIR**—Baud rate prescale value. The baud rate is calculated as: Baud rate = (system clock frequency) / (CT[15:0] x 16 x 2) The minimum CT value is 1; 0 denotes the counter stop. **MIR/FIR**—Reserved

16.2.15 Interrupt Vector (2C30)—IVR

This register is not used since the MGT5100 does not use interrupt vectors supplied by the peripherals.

Table 16-22 Interrupt Vector (2C30)—IVR

	msb 0	1	2	3	4	5	6	7 lsb
R	IVR[0:7]							
W								
RESET:	1	1	1	1	0	0	0	0

Bit	Name	Description
0:7	IVR	Interrupt Vector—Not applicable for MGT5100.

16.2.16 Input Port (2C34)—IP

This read-only IP register shows the current state of the input ports.

Table 16-23 IrDA Input Port (2C34)—IP

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved						DCD	CTS
W	Unused							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:5	—	Reserved
6	DCD	Current state of the scc_dcd_b input. 0 = scc_dcd_b is low 1 = scc_dcd_b is high
7	CTS	Current state of the scc_cts_b input 0 = scc_cts_b is low 1 = scc_cts_b is high

16.2.17 Output Port Bit Set (2C38)—OP1

This is a write-only register. Output ports are asserted by writing to this register.

Table 16-24 IrDA Output Port Bit Set (2C38)—OP1

	msb 0	1	2	3	4	5	6	7 lsb
R	Unused							
W	Reserved						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 scc_res_b (scc_res_b becomes 0).
7	RTS	Assert RTS output. 0 = No operation 1 = Asserts output port scc_rts_b (scc_rts_b becomes 0).

16.2.18 Output Port Bit Reset (2C3C)—OP0

This is a write-only register. Output ports are negated by writing to this register.

Table 16-25 IrDA Output Port Bit Reset (2C3C)—OP0

	msb 0	1	2	3	4	5	6	7 lsb
R	Unused							
W	Reserved						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 scc_res_b (scc_res_b becomes 0).
7	RTS	Assert RTS output. 0 = No operation 1 = Asserts output port scc_rts_b (scc_rts_b becomes 0).

16.2.19 IrDA Control (2C40)—SICR

This register sets the main operation mode.

Table 16-26 IrDA SIR Control (2C40)—SICR

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved				RxDCD	SIM[2:0]		
W								
RESET:	0	0	0	0	0	0	0	0

Table 16-27 IrDA MIR/FIR Control (2C40)—SICR

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					SIM[0:2]		
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4	RxDCD	**SIR**—Receiver DCD control. 0 = scc_dcd_b input is ignored. 1 = scc_dcd_b input is effective. **MIR/FIR**—Reserved
5:7	SIM[0:2]	IrDA operation mode. **CAUTION**: When the operating mode change occurs, all Rx/Tx and error statuses are reset. Rx and Tx are disabled. 000 = illegal 001 = illegal 010 = illegal 011 = illegal 100 = SIR 101 = MIR 110 = FIR 111 = Illegal

16.2.20 Infrared Control 1 (2C44)—IRCR1

This register controls the configuration in IrDA mode.

Table 16-28 Infrared SIR Control 1 (2C44)—IRCR1

	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 16-29 Infrared MIR/FIR Control 1 (2C44)—IRCR1

	msb 0	1	2	3	4	5	6	7 lsb
R/W	Reserved			INVCRC	CRCEN	FD	SIPEN	SPUL
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:2	—	Reserved
3	INVCRC	**MIR/FIR**—Invert CRC output enable. 0 = The MIR/FIR transmitter sends abort for an underrun error. 1 = Send inverted CRC when an underrun error occurred. **SIR**—Reserved
4	CRCEN1	**MIR/FIR**—CRC output enable. 0 = TX doesn't add CRC and the RX doesn't compare the CRC value. 1 = TX adds CRC after the last data byte and the MIR/FIR receiver outputs CRC error if the CRC value of the received flame is incorrect. **SIR**—Reserved
5	FD	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. Bit should not be set in usual operations. In loop-back channel mode, CM=10, bit automatically sets.
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. **SIR**—Reserved
7	SPUL	**SIR**—SIR pulse width. 0 = SIR pulse width is 3/16 of the bit duration. 1 = SIR pulse width is 1.6µs **MIR/FIR**—Reserved

16.2.21 Infrared Control 2 (2C48)—IRCR2

This register sets some requests for the TX or the TX-FIFO.

Table 16-30 Infrared SIR Control 2 (2C48)—IRCR2

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved							
W								
RESET:	0	0	0	0	0	0	0	0

Table 16-31 Infrared MIR/FIR Control 2 (2C48)—IRCR2

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved					SIPREQ	ABORT	NXTEOF
W								
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 TX becomes idle, TX starts to send one SIP pulse. The bit stays high until TX finishes sending a SIP. It automatically goes low when TX finishes sending a SIP. **SIR**—Reserved
6	ABORT	**MIR/FIR**—Abort output. 0 = Stop sending abort sequence. 1 = While TX is sending data or CRC, writing 1 to this bit causes TX to immediately start an 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. **SIR**—Reserved
7	NXTEOF	**MIR/FIR**—Next is the last byte. 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 does 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 TX buffer. This bit is usually set by IP bus write operation. Since the commbus has the transmit_frame_done_b signal, this bit need not be set by the commbus write operation. **SIR**—Reserved

16.2.22 Infrared SIR Divide (2C4C)—IRSDR

This register sets the SIR mode baud rate. This register is reserved when in other modes (i.e., MIR or FIR mode).

Table 16-32 Infrared SIR Divide (2C4C)—IRSDR

	msb 0	1	2	3	4	5	6	7 lsb
R	IRSTIM[0:7]							
W								
RESET:	0	0	0	0	0	0	0	0

Table 16-33 Infrared SIR Other Divide (2C4C)—IRSDR

	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 1.6µs pulse. In SIR mode, this is used to make 1.6µs pulse when SPUL in the IRCR1 is high and SIPREQ in the IRCR2 is high. This value should be set so that: system clock period * IRSTIM = 1.6µs Default value for 33MHz bus clock is 54. **MIR/FIR**—Reserved

16.2.23 Infrared MIR Divide (2C50)—IRMDR

This register sets the MIR mode baud rate. This register is reserved when in other modes (i.e., SIR or FIR mode).

Table 16-34 Infrared MIR Divide (2C50)—IRMDR

	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 16-35 Infrared MIR Other Divide (2C50)—IRMDR

	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	FREQ	**MIR**—0.576Mbps mode. 0 = Baud rate is 1.152 M bps. 1 = If baud rate is 0.576Mbps, this bit should be set high to output 1.6µs SIP. **SIR/FIR**—Reserved
1:7	M_FDIV	Clock divide ratio in MIR mode. The bit frequency is derived by: $f_{bit} = \frac{f_{bit\_clk}}{M\_FDIV + 1}$ This bit frequency should be 0.576 or 1.152MHz. 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 16-36 shows the selectable divide factor and the input clock frequency on ipg_bit_clk port.

Table 16-36 Frequency Selection in MIR Mode

M_FDIV[4:0]	bit_clk Frequency [MHz]	
	1.152Mbps	0.576Mbps
7	9.216	4.6080
11	18.432	9.216
15	36.864	18.432
19	73.728	36.864
23	147.46	73.728
27	294.91	147.46
31	589.82	294.91

16.2.24 Infrared FIR Divide (2C54)—IRFDR

This register sets the baud rate in FIR mode. This register is reserved when in other modes (i.e., SIR or MIR mode).

Table 16-37 Infrared FIR Divide (2C54)—IRFDR

	msb 0	1	2	3	4	5	6	7 lsb
R/W	Reserved				F_FDIV[3:0]			
RESET:	0	0	0	0	0	0	0	0

Table 16-38 Infrared FIR Other Divide (2C54)—IRFDR

	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
0:3	—	Reserved
4:7	F_FDIV	**FIR**—Clock divide ratio in FIR mode. The bit frequency is derived by: $$\int_{bit} = \frac{\int_{bit\_clk}}{F\_FDIV + 1}$$ This bit frequency should be 8MHz. To receive the minimum pulse width described in the IrDA specification, (F_FDIV + 1) should be larger than or equal to 4. Table 16-39 shows several frequency selections. FREQ—0.576Mbps mode. 0 = Baud rate is 1.152Mbps. 1 = If baud rate is 0.576Mbps, this bit should be set high to output 1.6 µs SIP. **SIR/MIR**—Reserved

Table 16-39 Frequency Selection in MIR Mode

F_FDIV[3:0]	bit_clk Frequency [MHz]
3	32.0
4	40.0
5	48.0
6	56.0
—	—

16.2.25 Rx FIFO Number of Data (2C58)—RFNUM

This is a read-only register.

Table 16-40 Rx FIFO Number of Data (2C58)—RFNUM

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

16.2.26 Tx FIFO Number of Data (2C5C)—TFNUM

This is a read-only register.

Table 16-41 Tx FIFO Number of Data (2C5C)—TFNUM

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

16.2.27 Rx FIFO Data (2C60)—RFDATA

The RFDATA register is for test use and not used in normal operation.

16.2.28 Rx FIFO Status (2C64)—RFSTAT

This is a read-only register.

Table 16-42 Rx FIFO Status (2C64)—RFSTAT

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved				Frame[0]	Frame[1]	Frame[2]	Frame[3]	Rsvd	Error	UF	OF	FR	FULL	ALARM	EMPTY
W	Unused															
RESET:	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:7	Frame[0:3]	End-of-Frame indicator. Not applicable to IrDA FIFOs, since the IrDA does 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.

Bit	Name	Description
12	FR	Frame ready. Not applicable to IrDA FIFOs, since the IrDA does 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 SmartComm or CPU. See Section 15.2.25 for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

16.2.29 Rx FIFO Control (2C68)—RFCNTL

Table 16-43 Rx FIFO Control (2C68)—RFCNTL

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved		WFR	COMP	FRAME	GR[2:0]		
RESET:	1	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2	WFR	Write frame. Not applicable to IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.
3	COMP	Re-enable requests on frame transmission completion. Not applicable to IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.
4	FRAME	Frame mode enable. **This bit must be cleared by writing 0 to it**, since the IrDA does 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 15.2.25 for details.

16.2.30 Rx FIFO Alarm (2C6E)—RFALARM

Table 16-44 Rx FIFO Alarm (2C6E)—RFALARM

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				ALARM											
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	“Amost full” threshold level. Amount of empty space remaining in the Rx FIFO at which the ALARM bit in the status register goes high/active.

16.2.31 Rx FIFO Read Pointer (2C72)—RFRPTR

Table 16-45 Rx FIFO Read Pointer (2C72)—RFRPTR

	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.

16.2.32 Rx FIFO Write Pointer (2C76)—RFWPTR

Table 16-46 Rx FIFO Write Pointer (2C76)—RFWPTR

	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.

16.2.33 Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR

Table 16-47 Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR

	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 IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.

16.2.34 Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR

Table 16-48 Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR

	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 IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.

16.2.35 Tx FIFO Data (2C80)—TFDATA

The TFDATA register is for test use and not used in normal operation.

16.2.36 Tx FIFO Status (2C84)—TFSTAT

This is a read-only register.

Table 16-49 Tx FIFO Status (2C84)—TFSTAT

	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	Unused															
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]	End-of-Frame indicator. Not applicable to IrDA FIFOs, since the IrDA does 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 IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.

Bit	Name	Description
13	FULL	Full. The FIFO is completely full of data.
14	ALARM	The FIFO is requesting service from either SmartComm or CPU. See Section 15.2.25 for a detailed description.
15	EMPTY	FIFO Empty. The FIFO is completely empty.

16.2.37 Tx FIFO Control (2C88)—TFCNTL

Table 16-50 Tx FIFO Control (2C88)—TFCNTL

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved		WFR	COMP	FRAME	GR[2:0]		
RESET:	1	0	0	0	0	0	0	0

Bit	Name	Description
0:1	—	Reserved
2	WFR	Write frame. Not applicable to IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.
3	COMP	Re-enable requests on frame transmission completion. Not applicable to IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.
4	FRAME	Frame mode enable. **This bit must be cleared by writing 0 to it**, since the IrDA does 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 15.2.25 for details.

16.2.38 Tx FIFO Alarm (2C8E)—TFALARM

Table 16-51 Tx FIFO Alarm (2C8E)—TFALARM

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	Reserved				ALARM											
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 empty" threshold level. Amount of data remaining in the Tx FIFO at which the ALARM bit in the status register goes high/active.

16.2.39 Tx FIFO Read Pointer (2C92)—TFRPTR

Table 16-52 Tx FIFO Read Pointer (2C92)—TFRPTR

	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.

16.2.40 Tx FIFO Write Pointer (2C96)—TFWPTR

Table 16-53 Tx FIFO Write Pointer (2C96)—TFWPTR

	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.

16.2.41 Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR

Table 16-54 Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR

	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 IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.

16.2.42 Tx FIFO Last Write Frame PTR (2C9C)—TFLWFPTR

Table 16-55 Tx FIFO Last Write Frame Pointer (2C9C)—TFLWFPTR

	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 IrDA FIFOs, since the IrDA does not recognize frame formats in the serial data stream.

SECTION 17
SERIAL PERIPHERAL INTERFACE (SPI)

17.1 Overview

The following sections are contained in this document:

- SPI Signal Description
- SPI Registers—MBAR + 0x0F00

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication between the MGT5100 and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

17.1.1 Features

The SPI has the following features:

- Master mode and slave mode
- Slave-select output
- Double-buffered operation
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode
- Mode fault error flag with CPU interrupt capability

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.

Figure 17-1 shows the SPI block diagram.

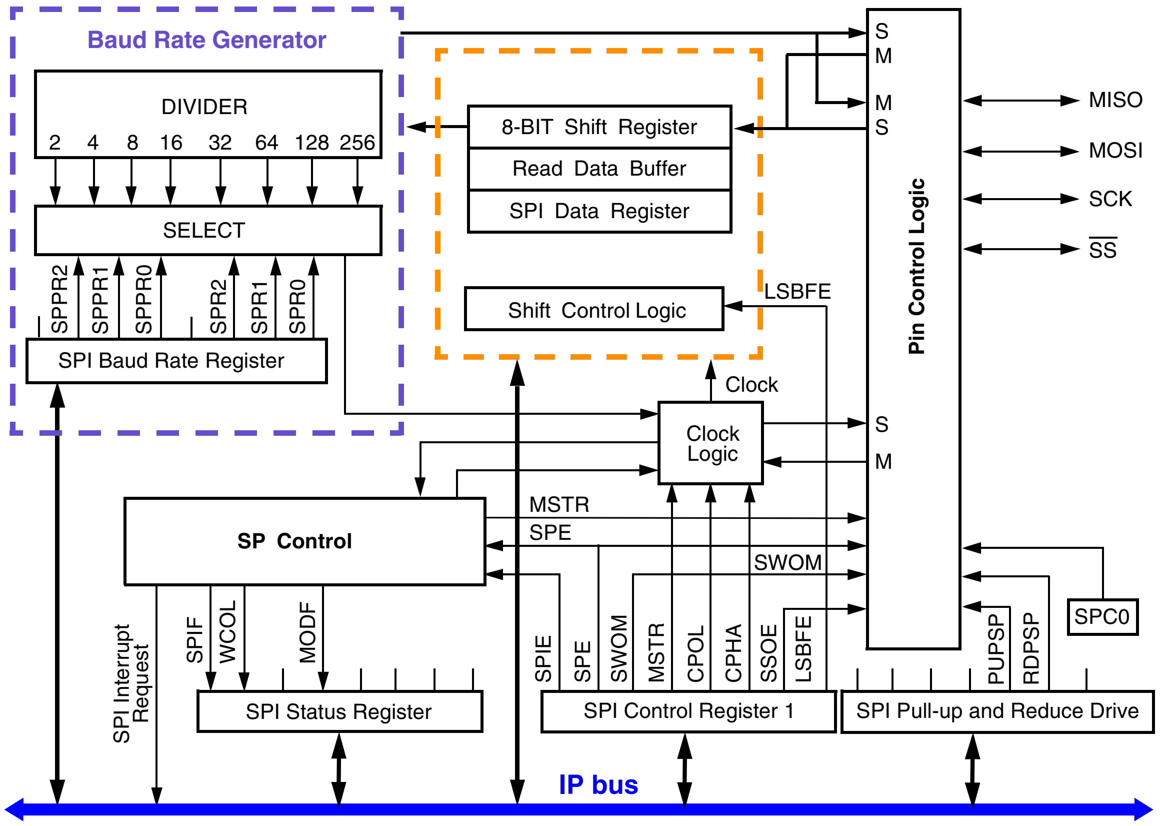


Figure 17-1 Block Diagram—SPI

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 $\overline{\text{SS}}$ 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 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 1 (0F00)—SPICR1
- SPI Control 2 (0F01)—SPICR2
- SPI Baud Rate (0F04)—SPIBR
- SPI Status (0F05)—SPISR
- SPI Data (0F09)—SPIDR
- SPI Port Data (0F0D)—SPIPORT
- SPI Data Direction (0F90)—SPIDDR

17.3.1 Control Register 1 (0F00)—SPICR1

Table 17-2 SPI Control 1 (0F00)—SPICR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	SPIE	SPE	SWOM (unused)	MSTR	CPOL	CPHA	SSOE	LSBFE
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

Bit	Name	Description
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
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 Control Register 2 (0F01)—SPICR2

Table 17-4 SPI Control 2 (0F01)—SPICR2

	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

Table 17-5 Bidirectional Pin Configurations

Pin Mode	SPC0	MSTR	MISO[1]	MOSI[2]	SCK[3]	$\overline{SS}$[4]
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 Baud Rate Register (0F04)—SPIBR

Table 17-6 SPI Baud Rate (0F04)—SPIBR

	msb 0	1	2	3	4	5	6	7 lsb
R W	Reserved	SPPR2	SPPR1	SPPR0	Reserved	SPR2	SPR1	SPR0
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
NOTE: These bits specify the SPI baud rates as shown in Table 17-7.		

Table 17-7 SPI Baud Rate Selection—40MHz Module Clock

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	SPI Module Clock Divisor	Baud Rate
0	0	0	0	0	0	2	20MHz
0	0	0	0	0	1	4	10MHz
0	0	0	0	1	0	8	5MHz
0	0	0	0	1	1	16	2.5MHz
0	0	0	1	0	0	32	1.25MHz
0	0	0	1	0	1	64	625KHz
0	0	0	1	1	0	128	312.5KHz
0	0	0	1	1	1	256	156.3KHz
0	0	1	0	0	0	4	10MHz
0	0	1	0	0	1	8	5MHz
0	0	1	0	1	0	16	2.5MHz
0	0	1	0	1	1	32	1.25MHz
0	0	1	1	0	0	64	625KHz
0	0	1	1	0	1	128	312.5KHz
0	0	1	1	1	0	256	156.3KHz
0	0	1	1	1	1	512	78.1KHz

Table 17-7 SPI Baud Rate Selection—40 MHz Module Clock (continued)

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	SPI Module Clock Divisor	Baud Rate
0	1	0	0	0	0	6	6.67 MHz
0	1	0	0	0	1	12	3.33 MHz
0	1	0	0	1	0	24	1.67 MHz
0	1	0	0	1	1	48	833.3 KHz
0	1	0	1	0	0	96	416.7 KHz
0	1	0	1	0	1	192	208.4 KHz
0	1	0	1	1	0	384	104.2 KHz
0	1	0	1	1	1	768	52.1 KHz
0	1	1	0	0	0	8	5 MHz
0	1	1	0	0	1	16	2.5 MHz
0	1	1	0	1	0	32	1.25 MHz
0	1	1	0	1	1	64	625 KHz
0	1	1	1	0	0	128	312.5 KHz
0	1	1	1	0	1	256	156.3 KHz
0	1	1	1	1	0	512	78.1 KHz
0	1	1	1	1	1	1024	39.1 KHz
1	0	0	0	0	0	10	4 MHz
1	0	0	0	0	1	20	2 MHz
1	0	0	0	1	0	40	1 MHz
1	0	0	0	1	1	80	500 KHz
1	0	0	1	0	0	160	250 KHz
1	0	0	1	0	1	320	125 KHz
1	0	0	1	1	0	640	62.5 KHz
1	0	0	1	1	1	1280	32.3 KHz
1	0	1	0	0	0	12	3.33 MHz
1	0	1	0	0	1	24	1.67 MHz
1	0	1	0	1	0	48	833.3 KHz
1	0	1	0	1	1	96	416.7 KHz
1	0	1	1	0	0	192	208.4 KHz
1	0	1	1	0	1	384	104.2 KHz
1	0	1	1	1	0	768	52.1 KHz
1	0	1	1	1	1	1536	26 KHz
1	1	0	0	0	0	14	2.86 MHz
1	1	0	0	0	1	28	1.43 MHz
1	1	0	0	1	0	56	714.3 KHz
1	1	0	0	1	1	112	357.1 KHz
1	1	0	1	0	0	224	178.6 KHz
1	1	0	1	0	1	448	89.3 KHz
1	1	0	1	1	0	896	44.6 KHz

Table 17-7 SPI Baud Rate Selection—40 MHz Module Clock (continued)

SPPR2	SPPR1	SPPR0	SPR2	SPR1	SPR0	SPI Module Clock Divisor	Baud Rate
1	1	0	1	1	1	1792	22.3 KHz
1	1	1	0	0	0	16	2.5 MHz
1	1	1	0	0	1	32	1.25 MHz
1	1	1	0	1	0	64	625 KHz
1	1	1	0	1	1	128	312.5 KHz
1	1	1	1	0	0	256	156.3 KHz
1	1	1	1	0	1	512	78.1 KHz
1	1	1	1	1	0	1024	39.1 KHz
1	1	1	1	1	1	2048	19.5 KHz

17.3.4 Status Register (0F05)—SPISR

Table 17-8 SPI Status (0F05)—SPISR

	msb 0	1	2	3	4	5	6	7 lsb
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 Data Register (0F09)—SPIDR

Table 17-9 SPI Data (0F09)—SPIDR

	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 Port Reduced Drive/Pull-up Select (0F0C)—SPIPURD

This register is unused.

Table 17-10 SPI Port Reduced Drive/Pull-up Select (0F0C)—SPIPURD

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Unused							
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	—	Unused

17.3.7 Port Data Register (0F0D)—SPIPORT

Table 17-11 SPI Port Data (0F0D)—SPIPORT

	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 (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".		

17.3.8 Data Direction Register (0F90)—SPIDDR

Table 17-12 SPI Data Direction (0F90)—SPIDDR

	msb 0	1	2	3	4	5	6	7 lsb
R / W	DDR7	DDR6	DDR5	DRD4	DDR3	DDR2	DDR1	DDR0
RESET:	0	0	0	0	0	0	0	0

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:** 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

SECTION 18
INTER-INTEGRATED CIRCUIT (I^2C)

18.1 Overview

The following sections are contained in this document:

- I^2C Controller 1
- I^2C Interface Registers—MBAR + 0x3D00
- Initialization Sequence
- Transfer Initiation and Interrupt
- External Signals
- Interface Description
- I^2C Controller 2 (same as I^2C Controller 1)

The Inter-Integrated Circuit (I^2C) is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. This two-wire bus minimizes the interconnection between devices.

The MGT5100 has two I^2C modules. Each has its own dedicated set of pins. The I^2C module is connected to the IP bus, not the SmartComm bus.

Each module operates up to 100Kbps 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 400pF. 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 lost interrupt 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.

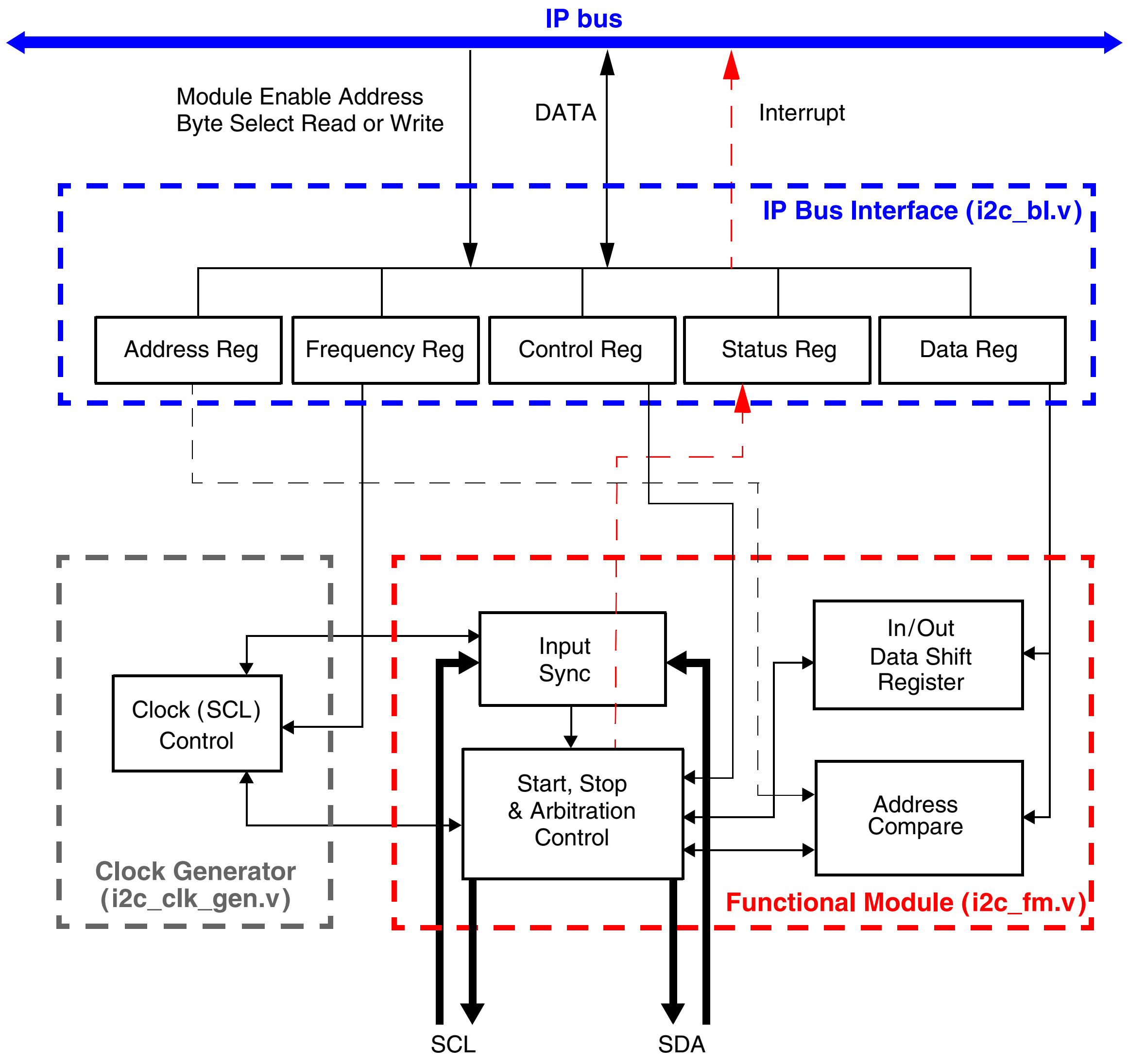

IP bus
Module Enable Address
Byte Select Read or Write
DATA
Interrupt
IP Bus Interface (i2c_bl.v)
Address Reg
Frequency Reg
Control Reg
Status Reg
Data Reg
Input Sync
Clock (SCL) Control
In/Out Data Shift Register
Start, Stop & Arbitration Control
Address Compare
Clock Generator (i2c_clk_gen.v)
Functional Module (i2c_fm.v)
SCL
SDA

Figure 18-1 Block Diagram—I^2C

18.2 I^2C Controller 1

The I^2C module has an IP bus interface with sky blue line signals. No FIFO interface to SmartComm is used. The I^2C 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 MGT5100 and other devices connected to the bus. Each device, including MGT5100, 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 I^2C 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 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 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 at a logical “1”.

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.

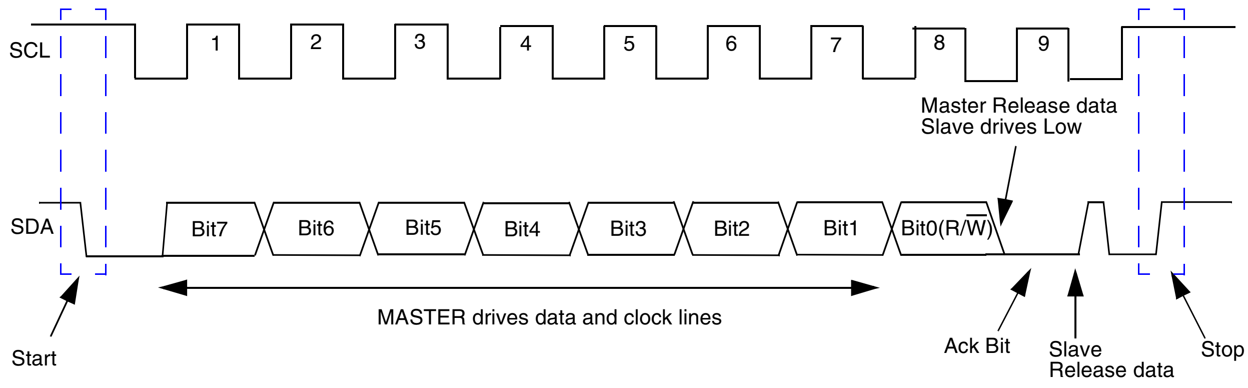


Figure 18-2 Timing Diagram—Start, Address Transfer and Stop Signal

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.

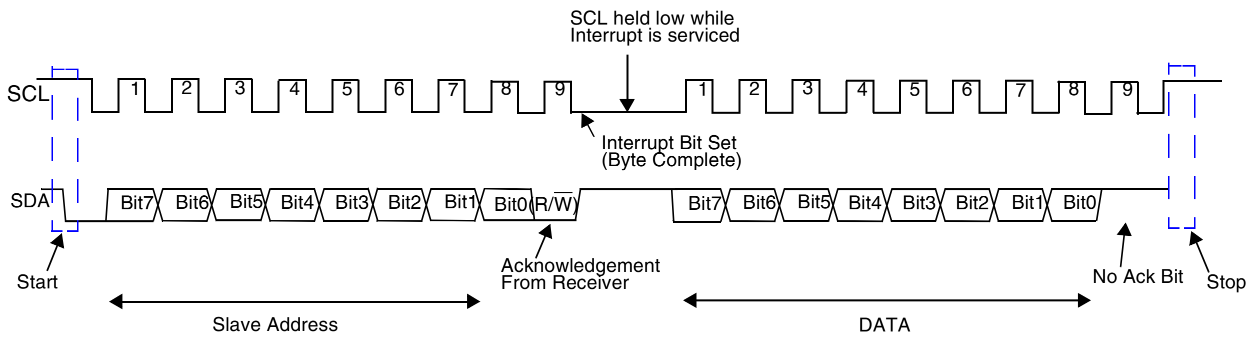


Figure 18-3 Timing Diagram—Data Transfer

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 down 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.

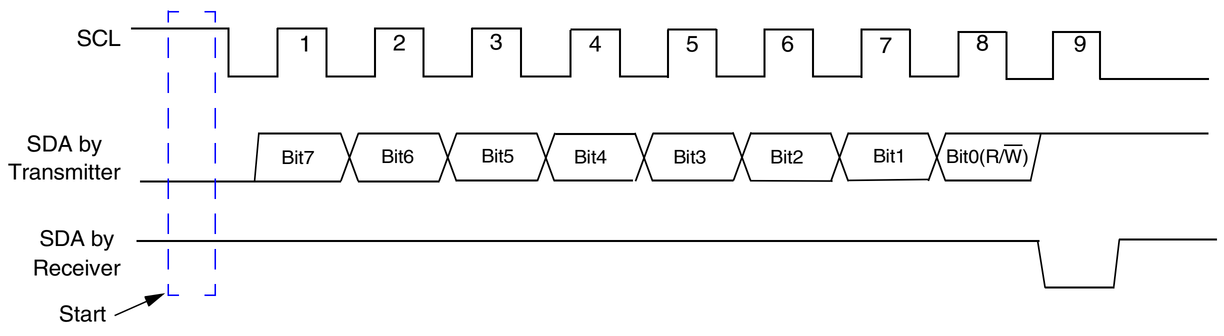


Figure 18-4 Timing Diagram—Receiver Acknowledgement

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.

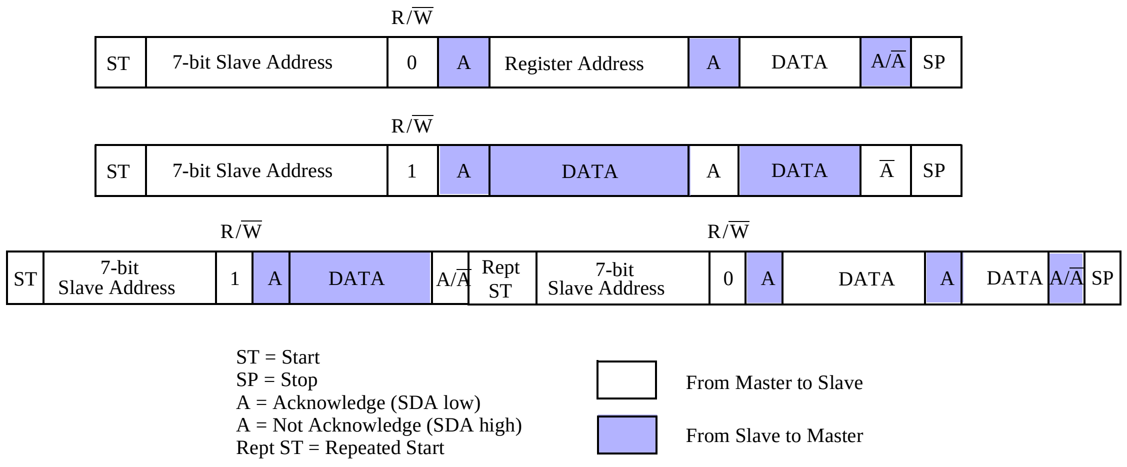

R/W̄
ST | 7-bit Slave Address | 0 | A | Register Address | A | DATA | A/Ā | SP
R/W̄
ST | 7-bit Slave Address | 1 | A | DATA | A | DATA | Ā | SP
R/W̄
R/W̄
ST | 7-bit Slave Address | 1 | A | DATA | A/Ā | Rept ST | 7-bit Slave Address | 0 | A | DATA | A | DATA | A/Ā | SP
ST = Start
SP = Stop
A = Acknowledge (SDA low)
Ā = Not Acknowledge (SDA high)
Rept ST = Repeated Start
From Master to Slave
From Slave to Master

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 done 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.

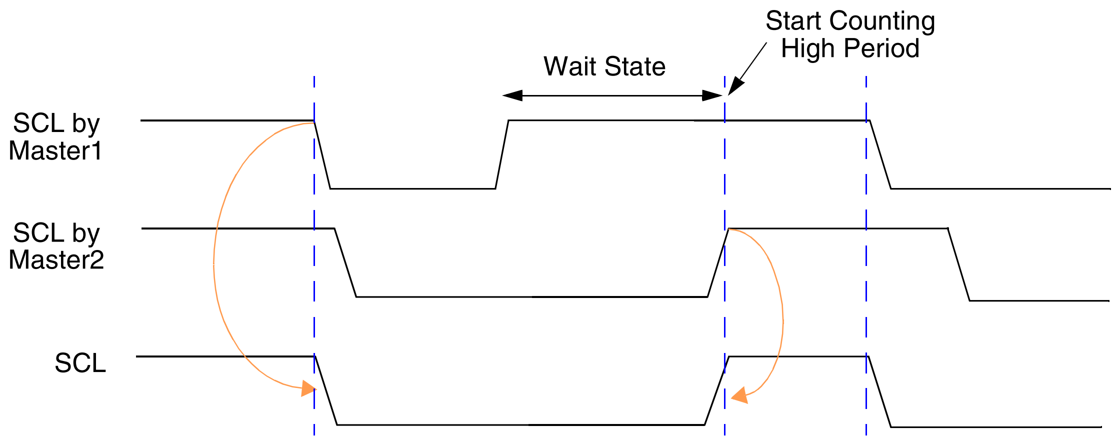


Figure 18-6 Timing Diagram—Clock Synchronization

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.

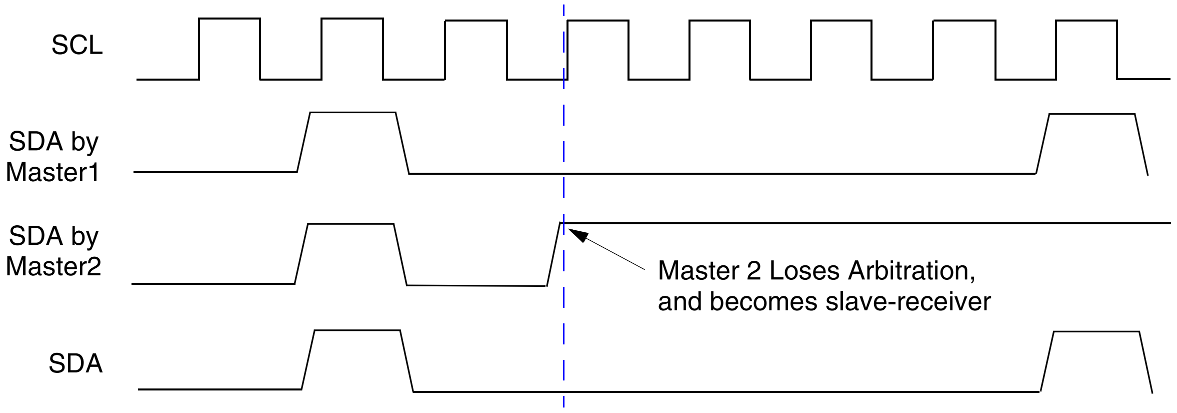


Figure 18-7 Timing Diagram—Arbitration Procedure

18.3 I^2C Interface Registers—MBAR + 0x3D00

The I^2C is controlled by 6 32-bit registers. These registers are located at an offset from MBAR of 0x3d00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x3d00 + register address**

Hyperlinks to the I^2C Interface registers are provided below:

- I^2C Address Register (3D00, 3D40)—I2C[1,2]
- I^2C Frequency Divider Register (3D04, 3D44)—I2C[1,2]
- I^2C Control Register (3D08, 3D48)—I2C[1,2]
- I^2C Status Register (3D0C, 3D4C)—I2C[1,2]
- I^2C Data I/O Register (3D10, 3D50)—I2C[1,2]
- I^2C Interrupt Control Register (3D20)

Internal register configurations are given below.

18.3.1 Address Register (3D00, 3D40)—I2C[*1,2*]

I2C1 3D00 I2C2 3D40

Table 18-2 I^2C Address Register (3D00, 3D40)—I2C[*1,2*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	ADR1	ADR2	ADR3	ADR4	ADR5	ADR6	ADR7	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	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[1:7]	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 Frequency Divider Register (3D04, 3D44)—I2C[*1,2*]

I2C1 3D04 I2C2 3D44

Table 18-3 I^2C Frequency Divider Register (3D04, 3D44)—I2C[*1,2*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		FDR0	FDR1	FDR2	FDR3	FDR4	FDR5	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[0:5]	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

Table 18-4 I^2C Tap and Prescale Values (continued)

FDR 5,1,0 (bin)	SCL_Tap (clocks)	SDA_Tap (clocks)	FDR 4,3,2	scl2tap (clocks)	tap2tap (clocks)
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 0x04(000100) is selected for the FDR[5:0] 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.

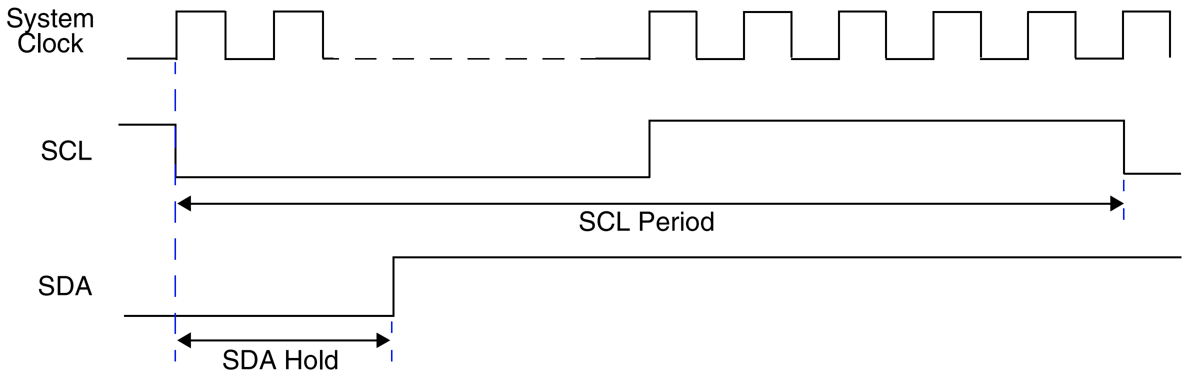


Figure 18-8 Timing Diagram—SCL Period and SDA Hold Time

18.3.3 Control Register (3D08, 3D48)—I2C[*1,2*]

I2C1 3D08 I2C2 3D48

Table 18-5 I^2C Control Register (3D08, 3D48)—I2C[*1,2*]

	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	I^2C Enable—bit controls software reset of entire I^2C module. If I^2C 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 bus is busy. If a start cycle is initiated, current bus cycle may become corrupt. Ultimately this results in current bus master or I^2C 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 = I^2C module is enabled. Bit must be set before other CR bits have any effect.
1	IEN	I^2C Interrupt Enable 0 = Interrupts from I^2C module are disabled. This does not clear currently pending interrupt condition. 1 = Interrupts from I^2C module are enabled. An I^2C 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

Bit	Name	Description
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 I^2C 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 Status Register (3D0C, 3D4C)—I2C[*1,2*]

I2C1 3D0C I2C2 3D4C

Table 18-6 I^2C Status Register (3D0C, 3D4C)—I2C[*1,2*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	CF	AAS	BB	AL	Rsvd	SRW	IF	RXAK	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

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 (I^2C 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 I^2C 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

Bit	Name	Description
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 I^2C 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	I^2C 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 Data I/O Register (3D10, 3D50)—I2C[*1,2*]

I2C1 3D10 I2C2 3D50

Table 18-7 I^2C Data I/O Register (3D10, 3D50)—I2C[*1,2*]

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	D70	D1	D2	D3	D4	D5	D6	D7	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[0:7]	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 following 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 Interrupt Control Register (3D20)

I2CIRQC 3D20

Table 18-8 I^2C Interrupt Control Register (3D20)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	BNBE2	TE2	RE2	IE2	BNBE1	TE1	RE1	IE1	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

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.
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.

Bit	Name	Description
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 MGT5100 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 MTX 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/$\overline{\text{W}}$ 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.

18.5.3 Typical M-bus Interrupt Routine Flow-Chart

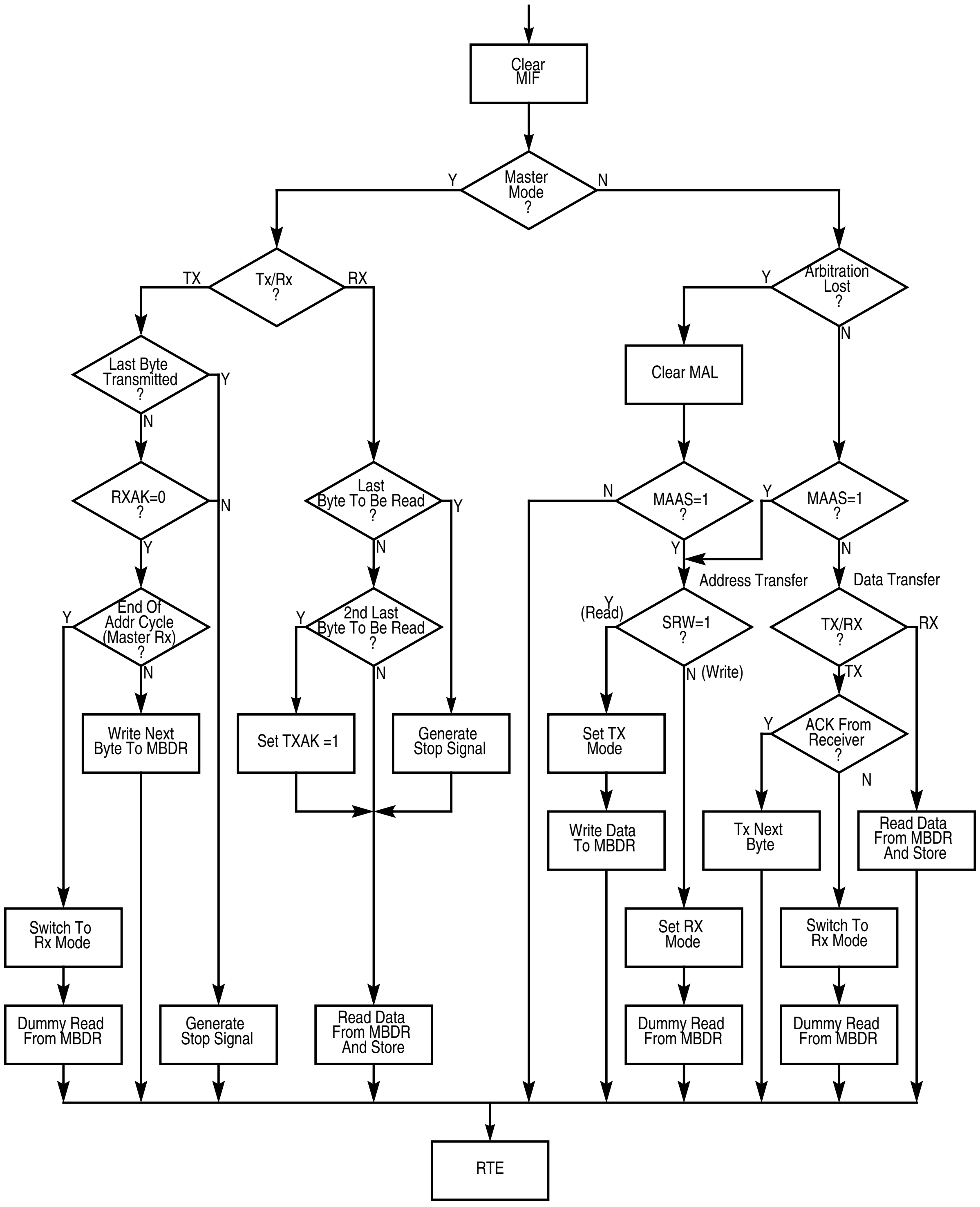


Figure 18-9 Flow Chart—M-bus Interrupt Routine

18.6 External Signals

Table 18-9 I^2C External Signals

Signal Name	I/O	Definition
SCL_Out_I2C1	O	I^2C Clock Output
SCL_Oe_I2C1	O	I^2C Clock Output Enable
SDA_Out_I2C1	O	I^2C Data Output
SDA_Oe_I2C1	O	I^2C Data Output Enable
SCL_In_I2C1	I	I^2C Clock Input
SDA_In_I2C1	I	I^2C Data Input

18.7 Interface Description

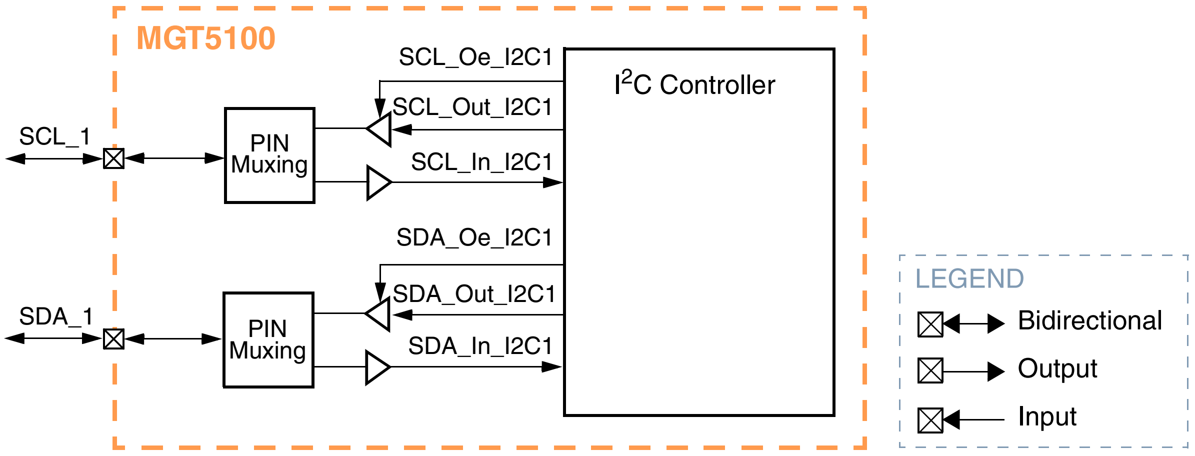


Figure 18-10 I^2C Controller1 External Connections

18.8 I^2C Controller 2

I^2C Controller2 is identical to I^2C Controller1.

SECTION 19
INFRARED (IR) INTERFACE

19.1 Overview

The following sections are contained in this document:

- Block Description
- Signals and Connections
- IR Blaster, includes:
  - IR Registers—MBAR + 0x0E00
- Remote IR Receiver, includes:
  - Remote IR Registers—MBAR + 0x0E00
- IR Keyboard Receiver

The Infrared (IR) interface port consists of four pins and supports the:

- IR blaster
- IR keyboard (simple UART receive)
- Infrared Data Association (IrDA) format to 4 Mbps (SIR, MIR, FIR)
- Consumer remote control standards RC5, RC6 RECS80, etc.

An additional receive input is included to simultaneously receive and process hand-held remote control devices. A dedicated hardware module reduces the need for software intervention in receipt of IR commands.

If the IrDA function is not needed, the available receive input can be used to process IR keyboard input. A simple UART module is provided to accommodate this function. The available transmit pin can be defined as an IR blaster pin.

A hardware module is provided to reduce the software intervention required to generate consumer IR protocols.

The remote IR receiver input and the IR keyboard receiver input (if active) both contain WakeUp functionality. This means the MGT5100 functional blocks remain active during full power down mode and can generate a WakeUp processor interrupt if a specific command word is received.

19.2 Block Description

MGT5100 IrDA functionality is implemented with the SCC/IrDA functional block. This block has an SCC serial port with an attached IrDA modulator. The IrDA block supports the following features:

- IrDA 1.0 SIR mode
  - Baud rate range=2400 to 115200bps
  - Selectable pulse width=either 3/16 bit duration or 1.6µs
- IrDA 1.1 MIR mode
  - Baud rate=0.576Mbps or 1.152Mbps.
  - 16-bit CRC generation (parameter option)
- IrDA 1.1 FIR mode
  - Baud rate=4.0 Mbps
  - 32-bit CRC generation (parameter option)

19.3 Signals and Connections

Table 19-1 Infrared Interface Port External Signals

Signal	I/O	Definition
ITx1	O	IR Transmit Data (Blaster or IRDA)
IRx0	I	IR Receive Data (dedicated to hand-held remote protocols)
IRx1	I	IRDA Receive or IR Keyboard receive
IRCLK	I	Clock input for IRDA
NOTE: No external Inputs connect to CSC.		

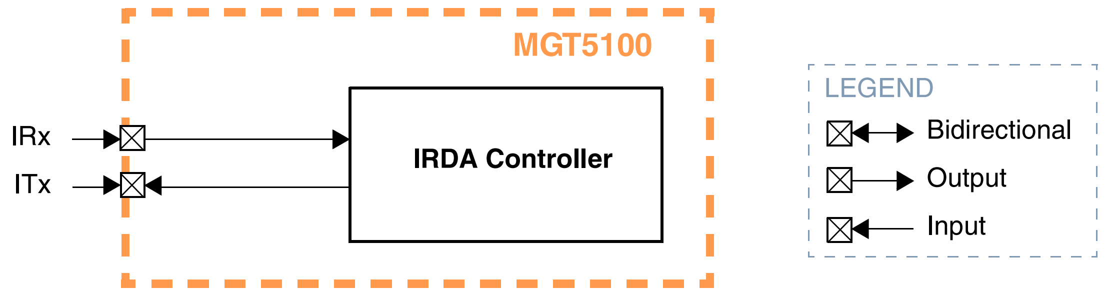


Figure 19-1 IRDA Controller External Connections

19.4 IR Blaster

The IR blaster module is the complement of the remote IR receiver module. However, it has a different function to perform and a slightly different register interface.

A single output is always available to transmit consumer IR protocols. An extensive hardware module has been developed, which allows the least possible software overhead.

To be transmitted, the datastream format must be extensively described. This is a one-time setup function.

Writes to the transmit word registers, followed by setting the enable bit, cause the word to be transmitted. A specified number of frame repeats can be specified and the transmit word is repeatedly transmitted back to back (up to 255 times) before an interrupt is generated and the module goes idle.

Formats with changing control bits at each transmission may require a single-frame send format, with software controlling the changing bits. However, a feature is included which can cause the prescribed control bits to toggle, increment, or decrement with each frame transmission. A status register is provided to store the final control bits state at the end of a repeated transmission format. Thus, software can pickup the end pattern for subsequent transmission.

The IR blaster output generates a carrier signal output, and operates either as:

- carrier-on—toggling at user specified carrier frequency
- carrier-off—static low

Frequency shifted keying is not supported, nor expected to be needed for consumer IR protocols. Format details must be specified, including:

- the number of transmit word bits (of various types)
- how many times the transmit word should be repeated

Writes then start the desired transmit word and enables the module. Transmission begins immediately. When complete, the module generates an interrupt, if enabled, and goes to an idle state. At this point, the module clears the enable bit and must be re-enabled by software to begin another transmission. Registers updates can occur only when the module is disabled (i.e., idle).

NOTE: The IR blaster module does not reside on the CommBus nor is it associated with the SmartComm control system.

Access to or from this module is through the IP bus interface. Interrupts from this module tie directly to the Interrupt Controller and are treated as a very low interrupt priority.

NOTE: VERIFY—All transmissions are LSB first. Software must reorder the bits, if necessary.

19.4.1 IR Registers—MBAR + 0x0E00

The IR interface is controlled by 8 32-bit registers. These registers are located at an offset from MBAR of 0x0e00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0E00 + register address**

Hyperlinks to the IR registers are provided below:

- IR Enable Control Register (0E00)
- IR Data Stream Control Register (0E04)
- IR Data Stream Format Register (0E08)
- IR Data Stream Format Register (0E0C)
- IR Data Stream Format Register (0E10)
- IR Preamble Register (0E14)
- IR Status Register (0E18)
- IR Dataword Transmitted Register (0E1C)

19.4.1.1 Enable Control Register 0 (0E00)

Table 19-2 IR Enable Control Register (0E00)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	Reserved		BEIE	BEEN	Reserved		IntEna	ME	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

Bit	Name	Description
0:1	—	Reserved
2	BEIE	Bus Error Interrupt Enable—bit enables interrupts when bus errors are present. Bus error status bits are updated regardless of BEIE being enabled. Status bit is provided for polling purposes.
3	BEEN	Bus Error Enable—bit enables IP bus interface transfer error acknowledgement (TEA). Provides immediate indication of transfer error. Status bits are provided for polling purposes. 1 = Enabled
4:5	—	Reserved
6	IntEna	Interrupt Enable—bit enables interrupts at receipt of a valid data word (VDW). Valid data words are loaded into output registers regardless of IntEna. A VDW status bit is provided for polling purposes.
7	ME	Master Enable/Module Enable—Module free-runs while enabled, loading (and possibly overwriting) the output registers for each VDW received. 1 = Enabled
8:31	—	Reserved

19.4.1.2 Data Stream Control Register (0E04)

Table 19-3 IR Data Stream Control Register (0E04)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	Reserved						CtlStyle		Rsvd	StopBit	MarkSpB	PLM	CtlCP	AddCP	DataCP	BiPhs
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	CarrierRate															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:5	—	Reserved
6:7	CtlStyle	Control Style—bit for frame repeats. 00 = No Change 01 = Toggle 10 = Increment 11 = Decrement
8	—	Reserved
9	StopBit	For PLM only—bit signals end of transmitted word. • If 1, length of Stop Bit equals value of BitY1. • If 0, length of Stop Bit equals value of BitY0.
10	MarkSpB	For PLM only—bit defines the mark space order within a bit-time. 1 = Mark/Space 0 = Space/Mark).
11	PLM	For PLM only—bit defines the fixed width portion of a bit-time. 1 = Mark fixed 0 = Space fixed).
12	CtlCP	Control Complement—bit defines whether complemented CONTROL data is present in the input datastream. Some protocols send complemented data after each field within a frame. Possible fields are: • control • address • data If any bits are 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the data stream before the dataword is placed in the output registers. See Note.

Bits	Name	Description
13	AddCP	Address Complement—bit defines whether complemented ADDRESS data is present in the input datastream. Some protocols send complemented data after each field within a frame. Possible fields are: • control • address • data If any bits are 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the data stream before the dataword is placed in the output registers. See Note.
14	DataCP	Data Complement—bit defines whether complemented DATA data is present in the input datastream. Some protocols send complemented data after each field within a frame. Possible fields are: • control • address • data If any bits are 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the data stream before the dataword is placed in the output registers. See Note.
15	BiPhs	Bi-Phase—If bit is 1, then Bi-Phase protocol is assumed, else PLM.
16:31	CarrierRate	A divider setting to generate the output carrier signal. At 54MHz system clock, maximum carrier period is 121µs in 18ns steps (frequency down to 1KHz).
NOTE: If defined, preamble or start bits are not complementable. The user must properly define these fields to allow coherency for the particular datastream format.		

19.4.1.3 Data Stream Format Register (0E08)

Table 19-4 IR Data Stream Format Register (0E08)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	numAdd				numData				Rsvd	numStart			numCtl			
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					numPream			Reserved							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	numAdd	Number of address bits (may be 0)—Defined address bits are applied to the address compare mask for interrupt decision making.
4:7	numData	Number of data bits—Defined data bits are applied to the data compare mask. If set to 0, represents maximum value of 16 data bits in the data word. Eight reserved unused bits, writing has no effect, always reads 0.
8	—	Reserved

Bits	Name	Description
9:11	numStar	Number of start bits at start of each frame (may be 0)—Bi-phase protocols typically use start bits. Must be 0 for PLM. Assumed to be high bits
12:15	numCtl	Number of control bits (may be 0)—Control bits are arbitrary bits that precede actual address or data bits. Control bits, if specified, are not compare maskable.
16:20	—	Reserved
21:23	numPream	Number of preamble transitions at start of each frame (may be 0)—1st preamble is expected to be high, followed by a low, then a high, and so on. PLM protocols typically use preamble.
24:31	—	Reserved

19.4.1.4 Data Stream Format Register (0E0C)

Table 19-5 IR Data Stream Format Register (0E0C)

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

Bits	Name	Description
0:7	BitPre-Scale	Prescale count value to be applied to calculation of bit-time. This should be set at a minimum value to accommodate the largest bit-time, but at the greatest resolution.
8:15	frameRPT	Number of times to repeat each frame before going to idle. (may not be 0).
16:31	Frame-PreScale	Prescale count value for calculation of total frame time (clocked by system clock).

19.4.1.5 Data Stream Format Register (0E10)

Table 19-6 IR Data Stream Format Register (0E10)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BitX								BitY0							
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	FrameLength								BitY1							
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	BitX	Prescaled bit-time • For Bi-phase protocols this is the total bit-time. • For PLM protocols this represents the fixed time of either the mark or space (whichever is fixed).
8:15	BitY0	PLM only—the varying mark or space time which represents that data = 0.
16:23	FrameLength	Number of frame prescaled counts that indicate total frame time. At 54 MHz, total frame time maximum is 310 ms. Frame length is the minimum amount of time it would take for any sequence of bits to be transmitted. it is assumed that there is an identifiable and fixed frame length for any given protocol. This should include some guaranteed idle time at the end of each frame.
24:31	BitY1	PLM only—the varying mark or space time which represents that data = 1.

19.4.1.6 Preamble Register (0E14)

Table 19-7 IR Preamble Register (0E14)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PreamLO								PreamHI							
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	PreamLO	Preamble low time, if applicable. (Preamble may be different from normal bit-time).
8:15	PreamHI	Preamble high time, if applicable.
16:31	—	Reserved

19.4.1.7 Status Register (0E18)

Table 19-8 IR Status Register (0E18)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STATUS								Reserved		BE2	BE1	Reserved		EOFSet	IntSet
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	STATUS	Status is read-only—Writing has no effect and does not set BusError3 for write to read-only. There are currently no transmitter status bits. A status register is provided to store the final state of the control bits at the end of a repeated transmission format. Software can then pickup the end pattern for subsequent transmission.
8:9	—	Reserved
10	BE2	Bus Error type 2—flag sets: • when an IP bus transaction writes to an unimplemented register. • regardless of Bus Error Enable Bit (BEEN). If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
11	BE1	Bus Error type 1—flag sets: • when an IP bus transaction reads an unimplemented register. • regardless of BEEN. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
12:13	—	Reserved
14	EOFSet	End of Frame—flag sets: • when transmitter reaches end of frame. • regardless of other control bits. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
15	IntSet	Interrupt Set—flag sets: • when an Interrupt condition occurred. • regardless of BEIE or IntEna. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
16:31	—	Reserved

19.4.1.8 Dataword Transmitted Register (0E1C)

Table 19-9 IR Dataword Transmitted Register (0E1C)

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

Bits	Name	Description
0:7	Control	Control word (8 bits)—Received control bits, if any (significant bits according to numCtl). Right justified LSB.
8:15	Address	Address word (8 bits)—Received address bits, if any (according to numAdd). Right justified LSB.
16:31	Data	Data word (16 bits)—Received data bits (according to numData). Right justified LSB.

19.5 Remote IR Receiver

A single input is always available to receive and process consumer IR protocols. An extensive hardware module has been developed to allow the least possible software overhead for this function. An extensively description is not necessary for the particular IR protocol support. This is a one-time step that involves writing several registers to describe bit-time, frame-time, preamble, start bits, number of address, and so on. This approach has the advantage of providing a very flexible hardware module which hopefully can support all the myriad IR protocols which exist.

Mask and compare registers are available for address and data bits. This lets the user “filter” what devices and device commands generate a processor interrupt. The filtering process can be specified to a single address/data combination to be used in conjunction with a WakeUp function. For example, in a powered down mode, the user can specify that only one command word, when received, generates an interrupt to the processor. Typically this would be the “Power” command. Whatever the filter parameters are, when they are matched, the received data word is put in an output register and an interrupt, if enabled, is generated.

Additional features are the stripping of preamble, start, and/or complement data prior to the data word being written to the output register. Separate registers are provided for control bits, address bits, and data bits. Thus, software can fetch any or all of the desired information, providing very low cycle counts for decision processing. A programmable spike filter is included. The needed incoming datastream timing precision can be degraded via a “Slop” register setting. The slop register essentially opens the window of opportunity for an input edge transition to be considered valid.

NOTE: At the MGT5100 input, it is assumed an external IR detector converts the IR carrier to a static high or low indication. This module operates independent of the carrier frequency and expects an input signal indication of either carrier-on (1) or carrier-off (0).

Fairly detailed status information is provided for initial software debugging. This status must be polled, since no interrupt is generated until a valid word is received.

NOTE: The remote IR receiver module does not reside on the CommBus, nor is it associated with the SmartComm control system. Access to or from this module is through the IP bus interface only. Module interrupts tie directly to the Interrupt Controller and are treated as a very low interrupt priority.

19.5.1 Remote IR Registers—MBAR + 0x0E00

The Remote IR interface is controlled by 10 32-bit registers. These registers are located at an offset from MBAR of 0x0e00. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0E00 + register address**

Hyperlinks to the Remote IR registers are provided below:

- Remote IR Enable Control Register (0E00)
- Remote IR Data Stream Control Register (0E04)
- Remote IR Data Stream Format Register (0E08)
- Remote IR Data Stream Format Register (0E0C)
- Remote IR Data Stream Format Register (0E10)
- Remote IR Data Compare/Mask Register (0E14)
- Remote IR Data Compare/Mask Register (0E18)
- Remote IR Data Stream Format Register (0E1C)
- Remote IR Status Register (0E20)
- Remote IR Dataword Received Register (0E24)

19.5.1.1 Enable Control Register (0E00)

Table 19-10 Remote IR Enable Control Register (0E00)

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

Bits	Name	Description
0:1	—	Reserved
2	BEIE	Bus Error Interrupt Enable—bit enables interrupts when bus errors are present. Bus error status bits are updated regardless of BEIE being enabled. Status bit is provided for polling purposes.
3	BEEN	Bus Error Enable—bit enables IP bus interface Transfer Error Acknowledgement (TEA). Provides immediate indication of transfer error. Status bits are provided for polling purposes. 1 = Enabled.
4	—	Reserved
5	Latch	A diagnostic control bit, which causes the module to halt at the first VDW. 1 = Latch operation (Enable autoclears at VDW).
6	IntEna	Interrupt Enable— bit enables interrupts at receipt of a valid data word (VDW). Valid data words are loaded into output registers regardless of IntEna, a VDW status bit is provided for polling purposes.
7	ME	Master Enable/Module Enable—Module free-runs while enabled, loading (and possibly overwriting) the output registers for each VDW received. 1 = Enabled
8:15	—	Reserved

19.5.1.2 Data Stream Control Register (0E04)

Table 19-11 Remote IR Data Stream Control Register (0E04)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R	Reserved					SLOP			Reserved		MarkSpB	PLM	CtlCP	AddCP	DataCP	BiPhs
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	SLOP	This is a special register that provides for input datastream timing parameter variations. The above bit-times are applied to each input transition, the SLOP setting provides for a plus and minus variation window around each input transition. If the input transition falls within this window, it is deemed to be a valid transition. The SLOP setting must be non-zero, but not so large as to allow false valid indications on “foreign” data streams. There are a lot of IR signals bouncing around, it is important to set this discrimination factor as tightly as possible.
8:9	—	Reserved
10	MarkSpB	PLM only—bit defines the mark space order within a bit-time. 1 = Mark/Space 0 = Space/Mark
11	PLM	PLM only—bit defines the fixed width portion of a bit-time. 1 = Mark fixed 0 = Space fixed
12	CtlCP	Control Complement—bit defines whether complemented CONTROL data is present in the input data stream. Some protocols send complemented data after each field within a frame. The possible fields (as defined above) are control, address, and data. If any bit is 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the data stream before the data word is placed in the output registers. See Note.
13	AddCP	Add Complement—bit defines whether complemented ADDRESS data is present in the input datastream. Some protocols send complemented data after each field within a frame. The possible fields (as defined above) are control, address, and data. If any bit is 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the data stream before the data word is placed in the output registers. See Note.
14	DataCP	Data Complement—bit defines whether complemented DATA data is present in the input datastream. Some protocols send complemented data after each field within a frame. The possible fields (as defined above) are control, address, and data. If any bit is 1, the module looks for complement data following each field not defined as 0. Complement data is checked for integrity and stripped from the datastream before the data word is placed in the output registers. See Note.
15	BiPhs	If 1, then Bi-phase protocol is assumed, else PLM.
NOTE: If defined, preamble or start bits are not complementable. The user must properly define these fields to allow coherency for the particular data stream format.		

19.5.1.3 Data Stream Format Register (0E08)

Table 19-12 Remote IR Data Stream Format Register (0E08)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	numAdd				numData				Rsvd	numStart			numCtl			
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					numPream			Reserved							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:3	numAdd	Number of address bits (may be 0)—Defined address bits are applied to the address compare mask for Interrupt decision making.
4:7	numData	Number of data bits—Defined data bits are applied to the data compare mask. If set to 0, represents maximum value of 16 data bits in the data word. Eight reserved unused bits, writing has no effect, always reads 0.
8	—	Reserved
9:11	numStar	Number of start bits at start of each frame (may be 0)—Bi-phase protocols typically use start bits. Must be 0 for PLM. Assumed to be high bits
12:15	numCtl	Number of control bits (may be 0)—Control bits are arbitrary bits that precede actual address or data bits. Control bits, if specified, are not compare maskable.
16:20	—	Reserved
21:23	numPream	Number of preamble transitions at start of each frame (may be 0)—1st preamble is expected to be high, followed by a low, then a high, and so on. PLM protocols typically use preamble.
24:31	—	Reserved

19.5.1.4 Data Stream Format Register (0E0C)

Table 19-13 Remote IR Data Stream Format Register (0E0C)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	BitPreScale								Reserved				SpikeCnt			
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	FramePreScale															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	BitPre-Scale	Prescale count value to be applied to bit-time calculation. This should be set at a minimum value to accommodate the largest bit-time, but at the greatest resolution.
8:11	—	Reserved
12:15	SpikeCnt	Number of system clocks to apply to spike suppression.
16:31	Frame-PreScale	Prescale count value for calculation of total frame time (clocked by system clock).

19.5.1.5 Data Stream Format Register (0E10)

Table 19-14 Remote IR Data Stream Format Register (0E10)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R / W	BitX								BitY0							
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	FrameLength								BitY1							
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	BitX	Prescaled bit-time • For Bi-phase protocols, this is the total bit-time. • For PLM protocols, this represents the fixed time of either the mark or space, whichever is fixed.
8:15	BitY0	PLM only—varying mark or space time that represents that data = 0.
16:23	FrameLength	Number of Frame prescaled counts to indicate total frame time. At 54 MHz total frame time, maximum is 310 ms. Frame length is the minimum amount of time it takes for any sequence of bits to be transmitted. It is assumed there is an identifiable and fixed frame length for any given protocol. This should include some guaranteed idle time at the end of each frame.
24:31	BitY1	PLM only—varying mark or space time that represents that data = 1.

19.5.1.6 Data Compare/Mask Register (0E14)

Table 19-15 Remote IR Data Compare/Mask Register (0E14)

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

Bits	Name	Description
0:15	DataComp	Any "1" in this register causes the corresponding received data bit to be compared to the corresponding bit in the data mask register.
16:31	DataMask	Any compare enabled data bit must match the corresponding bit in this register before an Interrupt can be generated. The IntEna bit must be set and the AddComp/AddMask registers must also pass.

19.5.1.7 Data Compare/Mask Register (0E18)

Table 19-16 Remote IR Data Compare/Mask Register (0E18)

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

Bits	Name	Description
0:7	AddComp	Any "1" in this register causes the corresponding received data bit to be compared to the corresponding bit in the DataMask register.
8:15	AddMask	Any compare enabled data bit must match the corresponding bit in this register before an Interrupt can be generated. The IntEna bit must be set and the AddComp/AddMask registers must also pass.

19.5.1.8 Data Stream Format Register (0E1C)

Table 19-17 Remote IR Data Stream Format Register (0E1C)

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

Bits	Name	Description
0:7	PreamLO	Preamble low time, if applicable. Preamble may be different from normal bit-time.
8:15	PreamHI	Preamble high time, if applicable.

19.5.1.9 Status Register (0E20)

Table 19-18 Remote IR Status Register (0E20)

	msb 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 lsb
R / W	STATUS								Rsvd	BE3	BE2	BE1	Reserved		AbortSet	IntSet
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description
0:7	STATUS	Status is read-only—Writing has no effect and does not set BusError3 for write to read-only. Currently there are NO transmitter status bits. A status register is provided to store the final state of the control bits at the end of a repeated transmission format. Thus, software can pickup the end pattern for subsequent transmission.
8	—	Reserved
9	BE3	Bus Error type 3—flag sets: • when an IP bus transaction writes to a read-only register. • regardless of the bus error enable bit (BEEN). If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
10	BE2	Bus Error type 2—flag sets: • when an IP bus transaction writes to an unimplemented register. • regardless of BEEN. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
11	BE1	Bus Error type 1—flag sets: • when an IP bus transaction reads an Unimplemented register. • regardless of BEEN. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.
12:13	—	Reserved
14	AbortSet	Abort Set—flag sets: • when an Abort condition occurs. • regardless of other control bits. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.

Bits	Name	Description
15	IntSet	Interrupt Set—flag sets: • when an Interrupt condition occurs. • regardless of BEIE or IntEna. If software is polling this byte and wishes to disregard this error, it must be masked. This bit field is sticky, clear with write to 1.

19.5.1.10 Dataword Received Register (0E24)

Table 19-19 Remote IR Dataword Received Register (0E24)

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

Bits	Name	Description
0:7	Control	ControlWord (8 bits)—Received control bits, if any (significant bits according to numCtl). Right justified LSB.
8:15	Address	AddressWord (8 bits)—Received address bits, if any (according to numAdd). Right justified LSB.
16:31	Data	Dataword (16 bits)—Received Data bits (according to numData). Right justified LSB.

19.6 IR Keyboard Receiver

The IR keyboard receiver consists of a single pin to receive basic UART signals. Some custom functionality must be wrapped around this UART to make it usable and software friendly. If this function is to support WakeUp (it is currently defined as such) then some additional support is needed.

The main premise is that IR keyboards transmit a standard UART-capable datastream (quite different from typical consumer remote protocols).

NOTE: The IR keyboard receiver module does not reside on the CommBus nor is it associated with the SmartDMA control system. Access to or from this module is only available through the IP bus interface. Interrupts from this module are tied directly to the Interrupt Controller and are treated as a very low interrupt priority.

SECTION 20
MOTOROLA SCALABLE CAN (MSCAN)

20.1 Overview

The following sections are contained in this document:

- Features
- MSCAN Registers—MBAR + 0900
- External Pin Descriptions

The Motorola Scalable Controller Area Network (MSCAN) module is a communication controller that implements the CAN 2.0A/B protocol. This CAN protocol is a serial data bus design that meets the following requirements:

- real-time processing
- reliable operation in an EMI environment
- cost-effectiveness
- required bandwidth

MSCAN uses an advanced buffer arrangement resulting in predictable, real-time behavior that simplifies application software.

Figure 20-1 shows the MSCAN block diagram.

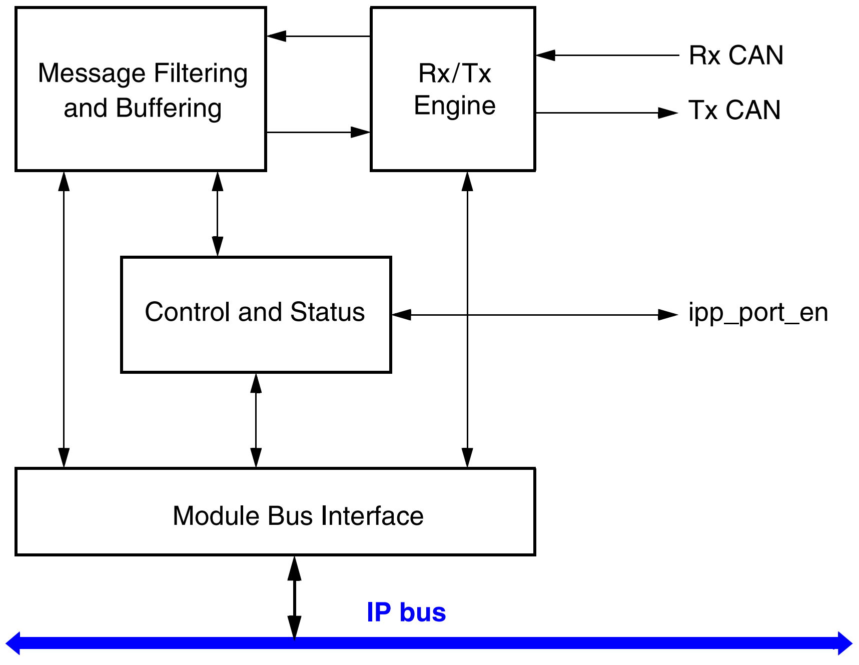


Figure 20-1 Block Diagram—MSCAN

20.2 Features

MSCAN basic features are:

- Modular architecture
- Implementation of the CAN protocol—Version 2.0 A/B
  - Standard and extended data frames
  - 0–8 Bytes data length
  - Programmable bit-rate up to 1 Mbps
    (depends on actual bit timing and PLL clock jitter)
  - Support for remote frames
- 4 Rx buffers with FIFO storage scheme
- 3 Tx buffers with internal prioritization using a "local priority" concept
- Flexible maskable identifier filter supports two full size extended identifier filters:
  - 2 32-bit filters
  - 4 16-bit filters
  - 8 8-bit filters
- Programmable WakeUp 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 Rx and Tx error states
  - warning
  - error-passive
  - bus-off
- Internal timer for time stamping of Rx and Tx messages
- Three low power modes:
  - sleep
  - power-down
  - MSCAN enable
- Global initialization of configuration registers

20.3 MSCAN Registers—MBAR + 0900

All register bits in this module are completely synchronous to internal clocks during a register read.

MSCAN is controlled by 32-bit registers. These registers are located at an offset from MBAR of 0x0200. Register addresses are relative to this offset. Therefore, the actual register address is: **MBAR + 0x0200 + register address**

Hyperlinks to the MSCAN registers are provided below:

- MSCAN Control Register 0 (0900, 0980)—CAN[1,2]CTL0
- MSCAN Control Register 1 (0901, 0981)—CAN[1,2]CTL1
- MSCAN Bus Timing Register 0 (0904, 0984)—CAN[1,2]BTR0
- MSCAN Bus Timing Register 1 (0905, 0985)—CAN[1,2]BTR1
- MSCAN Rx Flag (0908, 0988)—CAN[1,2]RFLG
- MSCAN Rx Interrupt Enable (0909, 0989)—CAN[1,2]RIER
- MSCAN Tx Flag (090C, 098C)—CAN[1,2]TFLG
- MSCAN Tx Interrupt Enable (090D, 098D)—CAN[1,2]TIER
- MSCAN Tx Message Abort Request (0910, 0990)—CAN[1,2]TARQ
- MSCAN Tx Message Abort Ack (0911, 0991)—CAN[1,2]TAAK
- MSCAN Tx Buffer Select (0914, 0994)—CAN[1,2]BSEL
- MSCAN ID Acceptance Control (0915, 0995)—CAN[1,2]IDAC
- MSCAN Rx Error (091C, 099C)—CAN[1,2]RXERR
- MSCAN Tx Error (091D, 099D)—CAN[1,2]TXERR
- MSCAN ID Acceptance (0920–0935)—CAN[1,2]IDAR[1–7]
- MSCAN ID Mask (0928–09BD)—CAN[1,2]IDMR[0–7]
- MSCAN Rx ID Register 0 (0940–09C0)—CAN[1,2]RXIDR0
- MSCAN Rx ID Register 1 (0941–09C1)—CAN[1,2]RXIDR1
- MSCAN Rx ID Register 0 (0940, 09C0)—CAN[1,2]RXIDR0
- MSCAN Rx ID Register 1 (0941, 09C1)—CAN[1,2]RXIDR1
- MSCAN Rx ID Register 2 (0944, 09C4)—CAN[1,2]RXIDR2
- MSCAN Rx ID Register 3 (0945,09C5)—CAN[1,2]RXIDR3
- MSCAN Rx Data Segment (0948–09D5)—CAN[1,2]RXDSR[0–7]
- MSCAN Rx Data Length (0958, 09D8)—CAN[1,2]RXDLR
- MSCAN Rx Time Stamp High (095C, 09DC)—CAN[1,2]RXTIMH
- MSCAN Rx Time Stamp Low (095D, 09DD)—CAN[1,2]RXTIML
- MSCAN Tx Buffer Priority (0979, 09F9)—CAN[1,2]TXTBPR
- MSCAN Tx Time Stamp High (097C, 09FC)—CAN[1,2]TXTIMH
- MSCAN Tx Time Stamp Low (097D, 09FD)—CAN[1,2]TXTIML
- MSCAN Tx ID Register 0 (0960, 09E0)—CAN[1,2]TXIDR0
- MSCAN Tx ID Register 1 (0961, 09E1)—CAN[1,2]TXIDR1
- MSCAN Tx ID Register 2 (0964, 09E4)—CAN[1,2]TXIDR2
- MSCAN Tx ID Register 3 (0965, 09E5)—CAN[1,2]TXIDR3
- MSCAN Tx Data Segment (0968–09F5)—CAN[1,2]TXDSR[0–7]
- MSCAN Tx Data Length (0978, 09F8)—CAN[1,2]TXDLR

20.3.1 Control Register 0 (0900, 0980)—CAN[*1,2*]CTL0

CAN1CTL0 0900 CAN2CTL0 0980

Table 20-1. MSCAN Control Register 0 (0900, 0980)—CAN[*1,2*]CTL0

	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	0

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

Bit	Name	Description
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

20.3.2 Control Register 1 (0901, 0981)—CAN[*1,2*]CTL1

CAN1CTL1 0901 CAN2CTL1 0981

Table 20-2. MSCAN Control Register 1 (0901, 0981)—CAN[*1,2*]CTL1

	msb 0	1	2	3	4	5	6	7 lsb
R	CANE	CLKSRC	LOOPB	LISTEN	0	WUPM	SLPAK	INITAK
W	CANE	CLKSRC	LOOPB	LISTEN	▨	WUPM	▨	▨
RESET:	0	0	0	0	0	0	0	0

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 oscillator clock (OSC_CLK) 1 = MSCAN clock source is the ungated IP bus clock (CLK)
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

Bit	Name	Description
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 T_{wup} 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

20.3.3 Bus Timing Register 0 (0904, 0984)—CAN[*1,2*]BTR0

CAN1BTR0 0904 CAN2BTR0 0984

Table 20-3. MSCAN Bus Timing Register 0 (0904, 0984)—CAN[*1,2*]BTR0

	msb 0	1	2	3	4	5	6	7 lsb
R / W	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
RESET:	0	0	0	0	0	0	0	0

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 20-4.

Table 20-4. 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

20.3.4 Bus Timing Register 1 (0905, 0985)—CAN[*1,2*]BTR1

CAN1BTR1 0905 CAN2BTR1 0985

Table 20-5. MSCAN Bus Timing Register 1 (0905, 0985)—CAN[*1,2*]BTR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
RESET:	0	0	0	0	0	0	0	0

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 20-7.
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 20-6.

Bit-time, as shown below, is determined by:

- oscillator frequency
- baud rate prescaler
- number of time quanta (Tq) clock cycles per bit

Table 20-6 and Table 20-7 give time segment values.

$$\text{Bit Time} = \frac{(\text{Prescaler value})}{f_{\text{CANCLK}}} \cdot (\text{Number of Time Quanta})$$

Table 20-6. 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 20-7. 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

20.3.5 Rx Flag (0908, 0988)—CAN[*1,2*]RFLG

CAN1RFLG 0908 CAN2RFLG 0988

Table 20-8. MSCAN Rx Flag (0908, 0988)—CAN[*1,2*]RFLG

	msb 0	1	2	3	4	5	6	7 lsb
R	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
W			▨	▨	▨	▨		
RESET:	0	0	0	0	0	0	0	0

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.

Bit	Name	Description
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: $0 \leq$ Receive Error Counter ≤ 96 01 = RxERR: $127 <$ Receive Error Counter 10 = RxWRN: $96 <$ Receive Error Counter ≤ 127 11 = BusOff1: $255 >$ 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: $0 \leq$ Transmit Error Counter ≤ 96 01 = TxERR: $127 <$ Transmit Error Counter ≤ 255 10 = TxWRN: $96 <$ Transmit Error Counter ≤ 127 11 = BusOff: $255 >$ Transmit Error Counter
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.		

20.3.6 Rx Interrupt Enable (0909, 0989)—CAN[*1,2*]RIER

CAN1RIER	0909	CAN2RIER	0989

Table 20-9. MSCAN Rx Interrupt Enable (0909, 0989)—CAN[*1,2*]RIER

	msb 0	1	2	3	4	5	6	7 lsb
R	WUPIE	CSCIE	RSTAT1	RSTAT0	TSTAT1	TSTAT0	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, RSTAT 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 “RxErr” or “BusOff”1 state. Discard other Rx state changes for generating CSCIF interrupt. 10 = Generate CSCIF interrupt only if receiver enters or leaves “BusOff” state. Discard other Rx state changes for generating CSCIF interrupt. 11 = Generate CSCIF interrupt on all state changes.
4:5	TSTAT[1:0]	Transmitter Status Change Enable—bits control sensitivity level in which Tx state changes cause CSCIF interrupts. Independent of the chosen sensitivity level, TSTAT 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 “TxErr” or “BusOff” state. Discard other Tx state changes for generating CSCIF interrupt. 10 = Generate CSCIF interrupt only if transmitter enters or leaves “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

20.3.7 Tx Flag (090C, 098C)—CAN[*1,2*]TFLG

CAN1TFLG 090C CAN2TFLG 098C

Table 20-10. MSCAN Tx Flag (090C, 098C)—CAN[*1,2*]TFLG

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	TXE2	TXE1	TXE0
W	▨	▨	▨	▨	▨			
RESET:	0	0	0	0	0	0	0	0

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)

20.3.8 Tx Interrupt Enable (090D, 098D)—CAN[*1,2*]TIER

CAN1TIER 090D CAN1TIER 098D

Table 20-11. MSCAN Tx Interrupt Enable (090D, 098D)—CAN[*1,2*]TIER

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	TXEIE2	TXEI1	TXEI0
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.

20.3.9 Tx Message Abort Request (0910, 0990)—CAN[*1,2*]TARQ

CAN1TARQ 0910 CAN2TARQ 0990

Table 20-12. MSCAN Tx Message Abort Request (0910, 0990)—CAN[*1,2*]TARQ

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
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

20.3.10 Tx Message Abort Ack (0911, 0991)—CAN[*1,2*]TAAK

CAN1TAAK 0911 CAN2TAAK 0991

Table 20-13. MSCAN Tx Message Abort Ack (0911, 0991)—CAN[*1,2*]TAAK

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
W	▨	▨	▨	▨	▨			
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:4	—	Reserved
5:7	ABTRQ[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

20.3.11 Tx Buffer Select (0914, 0994)—CAN[*1,2*]BSEL

CAN1BSEL 0914 CAN2BSEL 0994

Table 20-14. MSCAN Tx Buffer Select (0914, 0994)—CAN[*1,2*]BSEL

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	0	0	0	TX2	TX1	TX0
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

20.3.12 ID Acceptance Control (0915, 0995)—CAN[*1,2*]IDAC

CAN1IDAC 0915 CAN2IDAC 0995

Table 20-15. MSCAN ID Acceptance Control (0915, 0995)—CAN[*1,2*]IDAC

	msb 0	1	2	3	4	5	6	7 lsb
R	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
W	▨	▨			▨			
RESET:	0	0	0	0	0	0	0	0

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 20-17.
4	—	Reserved
5:7	IDHIT[2:1]	Identifier Acceptance Hit Indicator—MSCAN sets these flags to indicate an identifier acceptance hit. See Table 20-16.

Table 20-16. 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

Table 20-16. Identifier Acceptance Hit Indication (continued)

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance 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 20-17. 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

20.3.13 Rx Error (091C, 099C)—CAN[*1,2*]RXERR

CAN1RXERR 091C CAN2RXERR 099C

Table 20-18. MSCAN Rx Error (091C, 099C)—CAN[*1,2*]RXERR

	msb 0	1	2	3	4	5	6	7 lsb
R / W	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	RxERR[7:0]	Read-only when in sleep mode (SLPRQ=1 and SLPAK=1) or initialization mode (INITRQ=1 and INITAK=1). Write unimplemented. Reading this register when in any mode other than sleep or initialization, may return an incorrect value. For MCUs with dual CPUs this may cause a CPU fault condition. Writing to this register when in special modes can alter MSCAN functionality.

20.3.14 Tx Error (091D, 099D)—CAN[*1,2*]TXERR

CAN1TXERR 091D CAN2TXERR 099D

Table 20-19. MSCAN Tx Error (091D, 099D)—CAN[*1,2*]TXERR

	msb 0	1	2	3	4	5	6	7 lsb
R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	TxERR[7:0]	Read-only when in sleep mode (SLPRQ=1 and SLPAK=1) or initialization mode (INITRQ=1 and INITAK=1). Write unimplemented. Reading this register when in any mode other than sleep or initialization, may return an incorrect value. For MCUs with dual CPUs this may cause a CPU fault condition. Writing to this register when in special modes can alter MSCAN functionality.

20.3.15 ID Acceptance (0920–09B5)—CAN[*1,2*]IDAR[*0–7*]

CAN1IDAR0	0920	CAN2IDAR0	09A0
CAN1IDAR1	0921	CAN2IDAR1	09A1
CAN1IDAR2	0924	CAN2IDAR2	09A4
CAN1IDAR3	0925	CAN2IDAR3	09A5
CAN1IDAR4	0930	CAN2IDAR4	09B0
CAN1IDAR5	0931	CAN2IDAR5	09B1
CAN1IDAR6	0934	CAN2IDAR6	09B4
CAN1IDAR7	0935	CAN2IDAR7	09B5

Table 20-20. MSCAN ID Acceptance (0920–0935)—CAN[*1,2*]IDAR[*1–7*]

	msb 0	1	2	3	4	5	6	7 lsb
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.

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).

20.3.16 ID Mask (0928–09BD)—CAN[*1,2*]IDMR[*0–7*]

CAN1IDMR0	0928	CAN2IDMR0	09A8
CAN1IDMR1	0929	CAN2IDMR1	09A9
CAN1IDMR2	092C	CAN2IDMR2	09AC
CAN1IDMR3	092D	CAN2IDMR3	09AD
CAN1IDMR4	0938	CAN2IDMR4	09B8
CAN1IDMR5	0939	CAN2IDMR5	09B9
CAN1IDMR6	093C	CAN2IDMR6	09BC
CAN1IDMR7	093D	CAN2IDMR7	09BD

Table 20-21. MSCAN ID Mask (0928–09BD)—CAN[*1,2*]IDMR[*0–7*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
RESET:	0	0	0	0	0	0	0	0

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

20.3.17 Rx ID Register 0 (0940, 09C0)—CAN[*1,2*]RXIDR0

CAN1RXIDR0	0940	CAN2RXIDR0	09C0

Table 20-22. MSCAN Rx ID Register 0 (0940–09C0)—CAN[*1,2*]RXIDR0

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ID[10:3]	Standard format identifier—has 11 bits (ID[0:10]) for standard format. ID10 is most significant bit and is transmitted first on bus during arbitration procedure. ID priority is defined to be highest for the smallest binary number.

20.3.18 Rx ID Register 1 (0941, 09C1)—CAN[*1,2*]RXIDR1

CAN1RXIDR1	0941	CAN2RXIDR1	09C1

Table 20-23. MSCAN Rx ID Register 1 (0941–09C1)—CAN[*1,2*]RXIDR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID2	ID1	ID0	RTR	IDE(=0)	Reserved		
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:2	ID[2:0]	Standard format identifier—has 11 bits (ID[0:10]) for standard format. ID10 is most significant bit and is transmitted first on bus during arbitration procedure. ID priority is defined to be highest for the smallest binary number.
3	RTR	Remote Transmission Request—flag reflects status of remote transmission request bit in CAN frame. If RX buffer, flag indicates Rx frame status and supports transmission of an answering frame in software. If Tx buffer, flag defines RTR bit setting to be sent. 1 = Remote frame 0 = Data frame
4	IDE	ID Extended—flag indicates whether extended or standard identifier format is applied in this buffer. If Rx buffer, flag is set as Rx and indicates to CPU how to process buffer identifier registers. If Tx buffer, flag indicates to MSCAN what type of identifier to send. 1 = Extended format (29 bits) 0 = Standard format (11 bits)
5:7	—	Reserved

20.3.19 Rx ID Register 0 (0940, 09C0)—CAN[*1,2*]RXIDR0

CAN1RXIDR0	0940	CAN2RXIDR0	09C0

Table 20-24. MSCAN Rx ID Register 0 (0940, 09C0)—CAN[*1,2*]RXIDR0

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ID[28:21]	Standard format identifier—has 11 bits (ID[0:10]) for standard format. ID10 is most significant bit and is transmitted first on the bus during the arbitration procedure. ID priority is defined to be highest for the smallest binary number.

20.3.20 Rx ID Register 1 (0941, 09C1)—CAN[*1,2*]RXIDR1

CAN1RXIDR1	0941	CAN2RXIDR1	09C1

Table 20-25. MSCAN Rx ID Register 1 (0941, 09C1)—CAN[*1,2*]RXIDR1

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID20	ID19	ID18	SRR(=1)	IDE(=1)	ID17	ID16	ID15
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:2	ID[20:18]	Extended Format Identifier [20:18]
3	SRR	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.
4	IDE	ID Extended—flag indicates whether extended or standard identifier format is applied in this buffer. If Rx buffer, flag is set as Rx and indicates to the CPU how to process the buffer identifier registers. If Tx buffer, flag indicates to MSCAN what type of identifier to send. 1 = Extended format (29 bits) 0 = Standard format (11 bits)
5:7	ID[17:15]	Extended Format Identifier [17:15]

20.3.21 Rx ID Register 2 (0944, 09C4)—CAN[*1,2*]RXIDR2

CAN1RXIDR2	0944	CAN2RXIDR2	09C4

Table 20-26. MSCAN Rx ID Register 2 (0944, 09C4)—CAN[*1,2*]RXIDR2

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ID[14:7]	Read anytime for Tx buffers—only when RxF flag is set for Rx buffers. Write anytime for Tx buffers—when TxEx flag is set and corresponding Tx buffer is selected in CANTBSEL. They are unimplemented for Rx buffers.

20.3.22 Rx ID Register 3 (0945,09C5)—CAN[*1,2*]RXIDR3

CAN1RXIDR3	0945	CAN2RXIDR3	09C5

Table 20-27. MSCAN Rx ID Register 3 (0945,09C5)—CAN[*1,2*]RXIDR3

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	ID[6:0]	Read anytime for Tx buffers—only when RxF flag is set for Rx buffers. Write anytime for Tx buffers—when TxEx flag is set and corresponding Tx buffer is selected in CANTBSEL. They are unimplemented for Rx buffers.
7	RTR	Remote Transmission Request—flag reflects status of remote transmission request bit in CAN frame. If RX buffer, flag indicates Rx frame status and supports transmission of an answering frame in software. If Tx buffer, flag defines RTR bit setting to be sent. 1 = Remote frame 0 = Data frame

20.3.23 Rx Data Segment (0948–09D5)—CAN[*1,2*]RXDSR[*0–7*]

CAN1RXDSR0	0948	CAN2RXDSR0	09C8
CAN1RXDSR1	0949	CAN2RXDSR1	09C9
CAN1RXDSR2	094C	CAN2RXDSR2	09CC
CAN1RXDSR3	094D	CAN2RXDSR3	09CD
CAN1RXDSR4	0950	CAN2RXDSR4	09D0
CAN1RXDSR5	0951	CAN2RXDSR5	09D1
CAN1RXDSR6	0954	CAN2RXDSR6	09D4
CAN1RXDSR7	0955	CAN2RXDSR7	09D5

Table 20-28. MSCAN Rx Data Segment (0948–09D5)—CAN[*1,2*]RXDSR[*0–7*]

	msb 0	1	2	3	4	5	6	7 lsb
R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	DB[7:0]	8 data segment registers, each with bits DB[0:7], contains Tx or Rx data. Number of Tx or Rx bytes is determined by data length code in corresponding DLR register.

20.3.24 Rx Data Length (0958, 09D8)—CAN[*1,2*]RXDLR

CAN1RXDLR	0958	CAN2RXDLR	09D8

Table 20-29. MSCAN Rx Data Length (0958, 09D8)—CAN[*1,2*]RXDLR

	msb 0	1	2	3	4	5	6	7 lsb
R	Reserved				DLC3	DLC2	DLC1	DLC0
W								
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved
4:7	DLC[3:0]	Data Length Code—contains CAN frame data length field. DLC contains the number of bytes (data byte count) of the respective message. During a remote frame transmission, DLC is transmitted as programmed, while the number of transmitted data bytes is always 0. Data Byte count ranges from 0 to 8 for a data frame.

20.3.25 Rx Time Stamp High (095C, 09DC)—CAN[*1,2*]RXTIMH

CAN1RXTIMH	095C	CAN2RXTIMH	09DC

Table 20-30. MSCAN Rx Time Stamp High (095C, 09DC)—CAN[*1,2*]RXTIMH

	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

Bit	Name	Description
0:7	TSR[15:8]	If the TIME bit is enabled, MSCAN writes a special time stamp to the respective registers in the 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.

20.3.26 Rx Time Stamp Low (095D, 09DD)—CAN[*1,2*]RXTIML

CAN1RXTIML 095D CAN2RXTIML 09DD

Table 20-31. MSCAN Rx Time Stamp Low (095D, 09DD)—CAN[*1,2*]RXTIML

	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

Bit	Name	Description
0:7	TSR[7:0]	If TIME bit is enabled, MSCAN writes a special time stamp to the respective registers in the 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.

20.3.27 Tx Buffer Priority (0979, 09F9)—CAN[*1,2*]TXTBPR

CAN1TXTBPR 0979 CAN2TXTBPR 09F9

Table 20-32. MSCAN Tx Buffer Priority (0979, 09F9)—CAN[*1,2*]TXTBPR

	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.

20.3.28 Tx Time Stamp High (097C, 09FC)—CAN[*1,2*]TXTIMH

CAN1TXTIMH 097C CAN2TXTIMH 09FC

Table 20-33. MSCAN Tx Time Stamp High (097C, 09FC)—CAN[*1,2*]TXTIMH

	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

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.

20.3.29 Tx Time Stamp Low (097D, 09FD)—CAN[*1,2*]TXTIML

CAN1TXTIML 097D CAN2TXTIML 09FD

Table 20-34. MSCAN Tx Time Stamp Low (097D, 09FD)—CAN[*1,2*]TXTIML

	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

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.

20.3.30 Tx ID Register 0 (0960, 09E0)—CAN[*1,2*]TXIDR0

CAN1TXIDR0	0960	CAN2TXIDR0	09E0

Table 20-35. MSCAN Tx ID Register 0 (0960, 09E0)—CAN[*1,2*]TXIDR0

	msb 0	1	2	3	4	5	6	7 lsb
R/W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ID[10:3]	Standard format identifier—has 11 bits (ID[0:10]) for standard format. ID10 is most significant bit and is transmitted first on bus during arbitration procedure. Identifier priority is defined to be highest for the smallest binary number.

20.3.31 Tx ID Register 1 (0961, 09E1)—CAN[*1,2*]TXIDR1

CAN1TXIDR1	0961	CAN2TXIDR1	09E1

Table 20-36. MSCAN Tx ID Register 1 (0961, 09E1)—CAN[*1,2*]TXIDR1

	msb 0	1	2	3	4	5	6	7 lsb
R/W	ID2	ID1	ID0	RTR	IDE(=0)	Reserved		
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:2	ID[2:0]	Reserved
3	RTR	Remote Transmission Request—flag reflects status of remote transmission request bit in CAN frame. If Rx buffer, it indicates status of received frame and supports transmission of an answering frame in software. If Tx buffer, flag defines RTR bit setting to be sent. 1 = Remote frame 0 = Data frame

Bit	Name	Description
4	IDE	ID Extended—flag indicates whether extended or standard identifier format is applied in this buffer. If Rx buffer, flag is set as received and indicates to CPU how to process buffer identifier registers. If Tx buffer, flag indicates to MSCAN what type of identifier to send. 1 = Extended format (29 bits) 0 = Standard format (11 bits)
5:7	—	Reserved

20.3.32 Tx ID Register 2 (0964, 09E4)—CAN[*1,2*]TXIDR2

CAN1TXIDR2 0964 CAN2TXIDR2 09E4

Table 20-37. MSCAN Tx ID Register 2 (0964, 09E4)—CAN[*1,2*]TXIDR2

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	ID[14:7]	Read anytime for Tx buffers—only when RXF flag is set for Rx buffers. Write anytime for Tx buffers—when TXEx flag is set and corresponding Tx buffer is selected in CANTBSEL. They are unimplemented for Rx buffers.

20.3.33 Tx ID Register 3 (0965, 09E5)—CAN[*1,2*]TXIDR3

CAN1TXIDR3 0965 CAN2TXIDR3 09E5

Table 20-38. MSCAN Tx ID Register 3 (0965, 09E5)—CAN[*1,2*]TXIDR3

	msb 0	1	2	3	4	5	6	7 lsb
R / W	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:6	ID[6:0]	Read anytime for Tx buffers—only when RXF flag is set for Rx buffers. Write anytime for Tx buffers—when TXEx flag is set and corresponding Tx buffer is selected CANTBSEL. They are unimplemented for Rx buffers.

Bit	Name	Description
7	RTR	Remote Transmission Request—flag reflects status of remote transmission request bit in CAN frame. If Rx buffer, it indicates status of received frame and supports transmission of an answering frame in software. If Tx buffer, flag defines RTR bit setting to be sent. 0 = Remote frame 1 = Data frame

20.3.34 Tx Data Segment (0968–09F5)—CAN[*1,2*]TXDSR[*0–7*]

CAN1TXDSR0	0968	CAN2TXDSR0	09E8
CAN1TXDSR1	0969	CAN2TXDSR1	09E9
CAN1TXDSR2	096C	CAN2TXDSR2	09EC
CAN1TXDSR3	096D	CAN2TXDSR3	09ED
CAN1TXDSR4	0970	CAN2TXDSR4	09F0
CAN1TXDSR5	0971	CAN2TXDSR5	09F1
CAN1TXDSR6	0974	CAN2TXDSR6	09F4
CAN1TXDSR7	0975	CAN2TXDSR7	09F5

Table 20-39. MSCAN Tx Data Segment (0968–09F5)—CAN[*1,2*]TXDSR[*0–7*]

	msb 0	1	2	3	4	5	6	7 lsb
R / W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:7	DB[7:0]	8 data segment registers, each with bits DB[0:7], contain data to be transmitted or received. Number of bytes transmitted or received is determined by data length code in corresponding DLR register.

20.3.35 Tx Data Length (0978, 09F8)—CAN[*1,2*]TXDLR

CAN1TXDLR	0978	CAN2TXDLR	09F8

Table 20-40. MSCAN Tx Data Length (0978, 09F8)—CAN[*1,2*]TXDLR

	msb 0	1	2	3	4	5	6	7 lsb
R / W	Reserved				DLC3	DLC2	DLC1	DLC0
RESET:	0	0	0	0	0	0	0	0

Bit	Name	Description
0:3	—	Reserved

Bit	Name	Description
4:7	DLC[3:0]	Data Length Code—contains the number of bytes (Data Byte count) of the respective message. During transmission of a remote frame, data length code is transmitted as programmed while number of transmitted Data Bytes is always 0. Data Byte count ranges from 0 to 8 for a data frame. Table 20-41 shows the effect of setting DLC bits.

Table 20-41. 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

20.4 External Pin Descriptions

MSCAN uses two external pins:

- one input—RxCAN
- one output—TxCAN

The TxCAN output pin represents the CAN logic level:

- 0=dominant state
- 1=recessive state

When MSCAN is enabled (CANE=1) via the CANCTL1 register, TxCAN and RxCAN pins are enabled within the port module. This is indicated to the port module using a dedicated enable line (ipp_port_en).

When MSCAN is disabled (CANE=0), the pins are available as GPIO in the port module.

A typical CAN system with MSCAN is shown in Figure 20-2. 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.

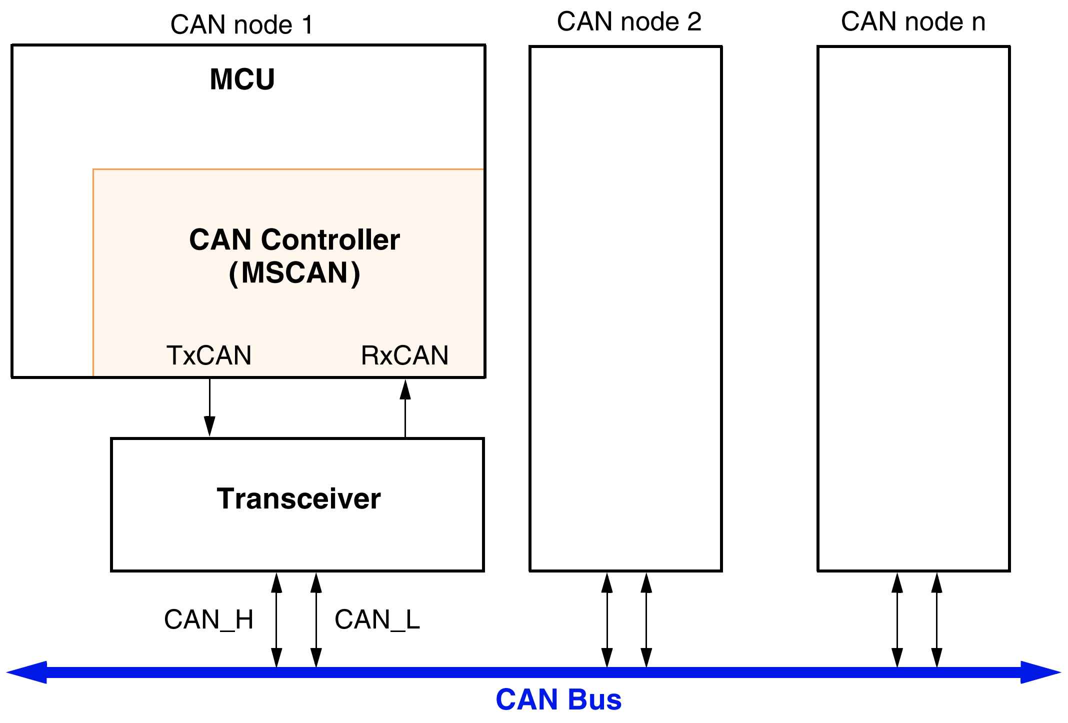


Figure 20-2 The CAN System

SECTION 21
DEBUG SUPPORT AND JTAG INTERFACE

21.1 Overview

The following sections are contained in this document:

- TAP Link Module (TLM) and Slave TAP Implementation
- TLM and TAP Signal Descriptions
- Slave Test Reset (STRST)
- TAP State Machines
- Harpo Core JTAG/COP Serial Interface
- TLM Link DR Instructions
- TLM Test Instructions, includes:
  - IDCODE
  - Device ID Register
- HARPO COP/BDM Interface

The MGT5100 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 MGT5100's imbedded Motorola MPC603e 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 MGT5100 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 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.

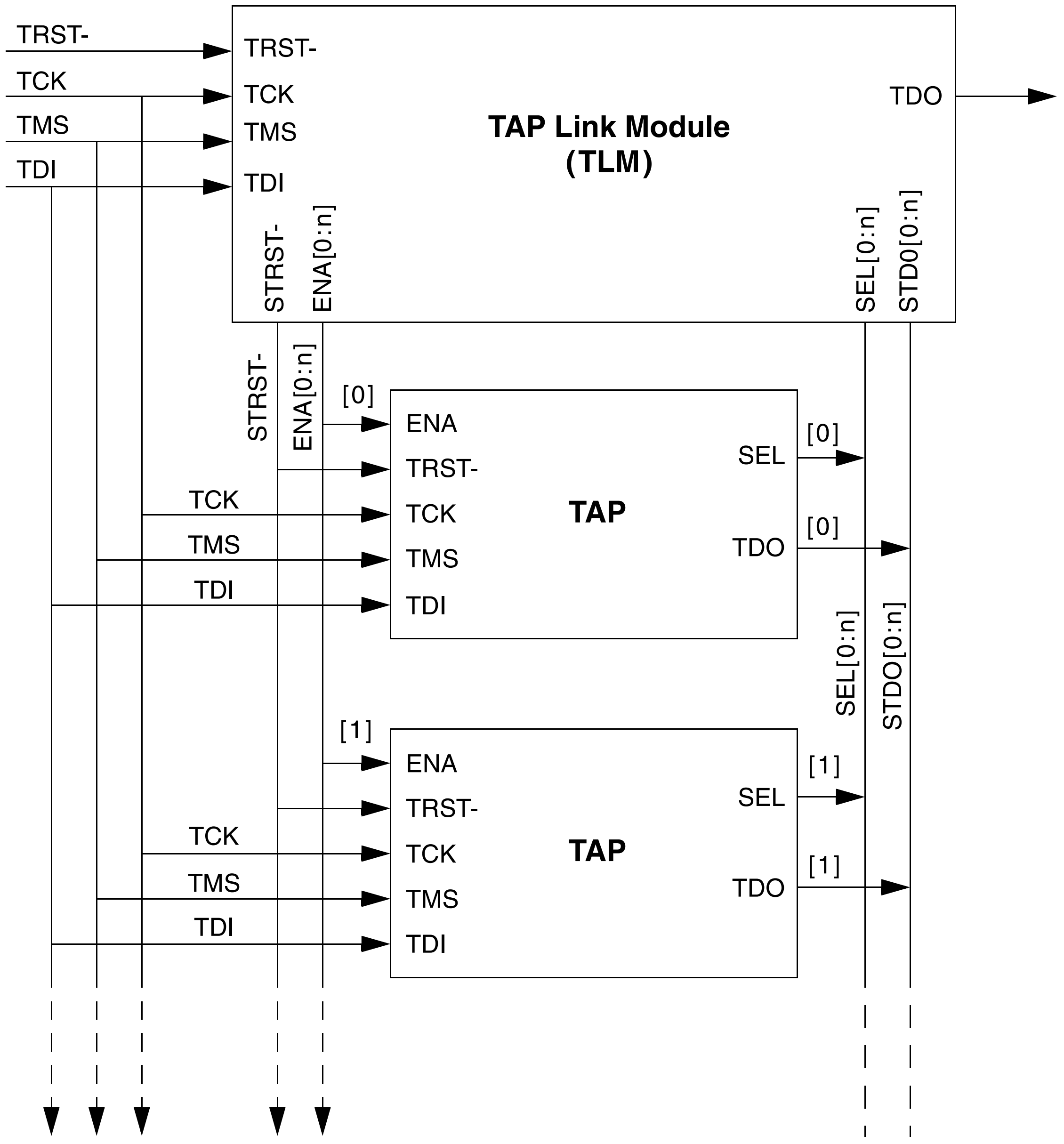


Figure 21-1 Generic TLM/TAP Architecture Diagram

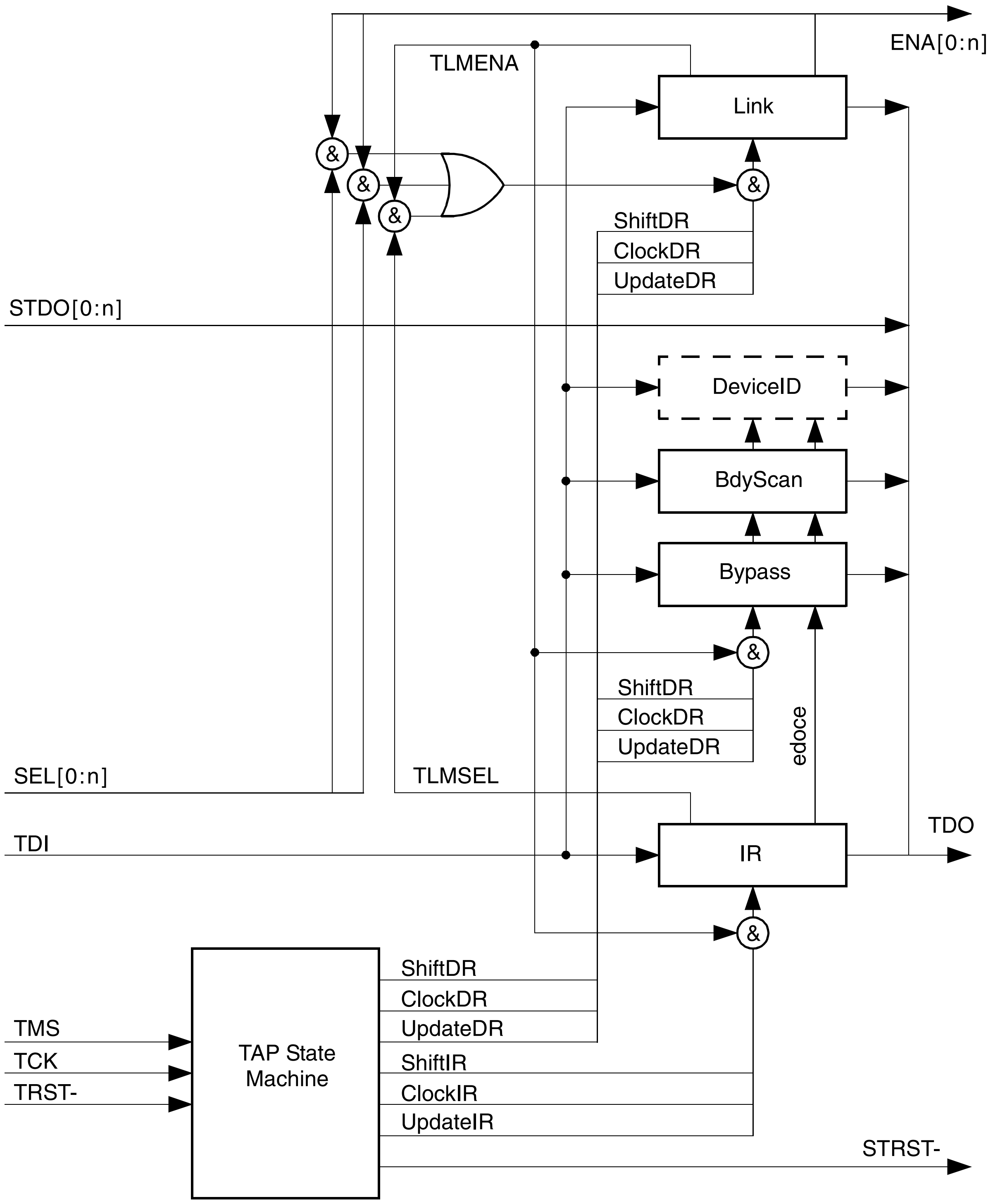


Figure 21-2 Generic TAP Link Module (TLM) Diagram

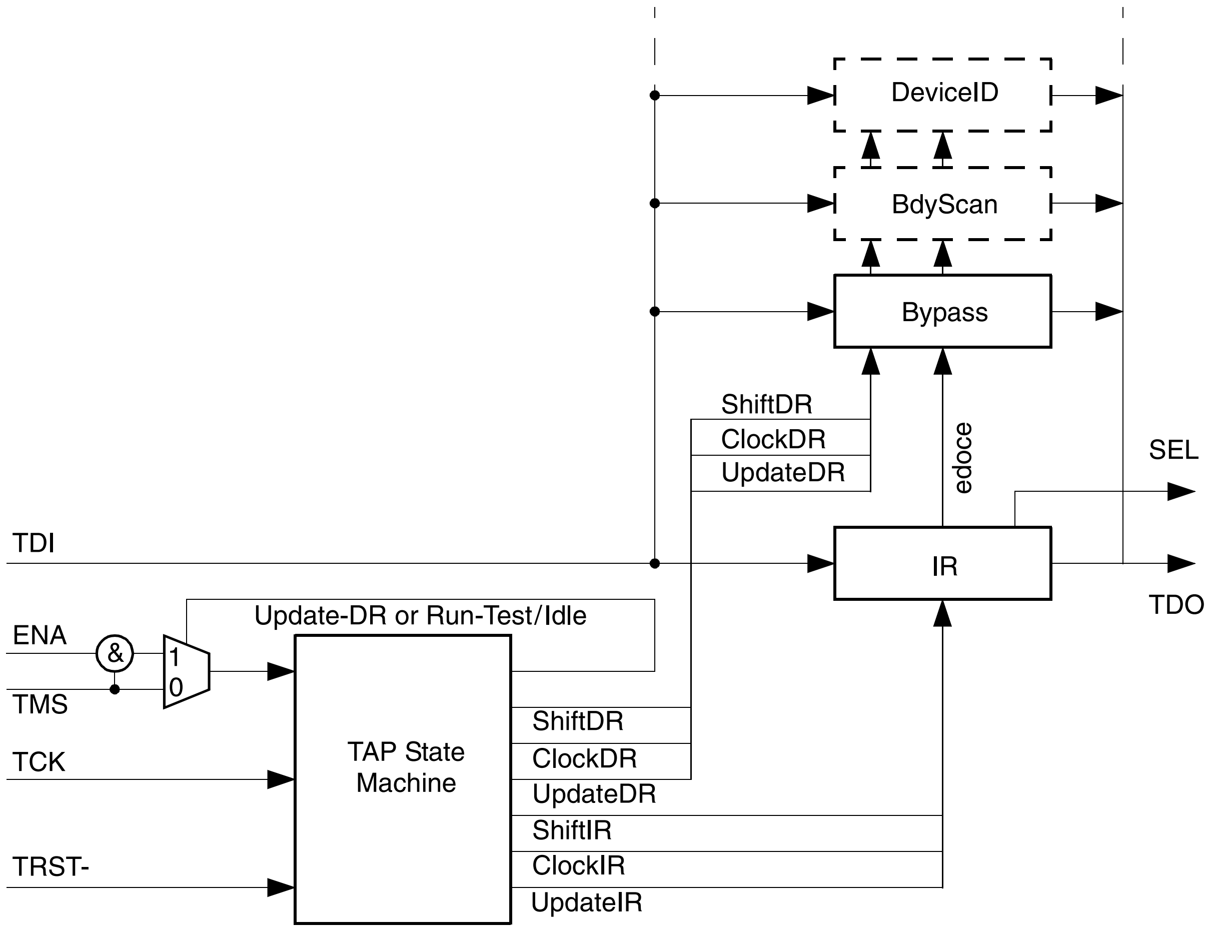


Figure 21-3 Generic Slave TAP

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. MGT5100 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.

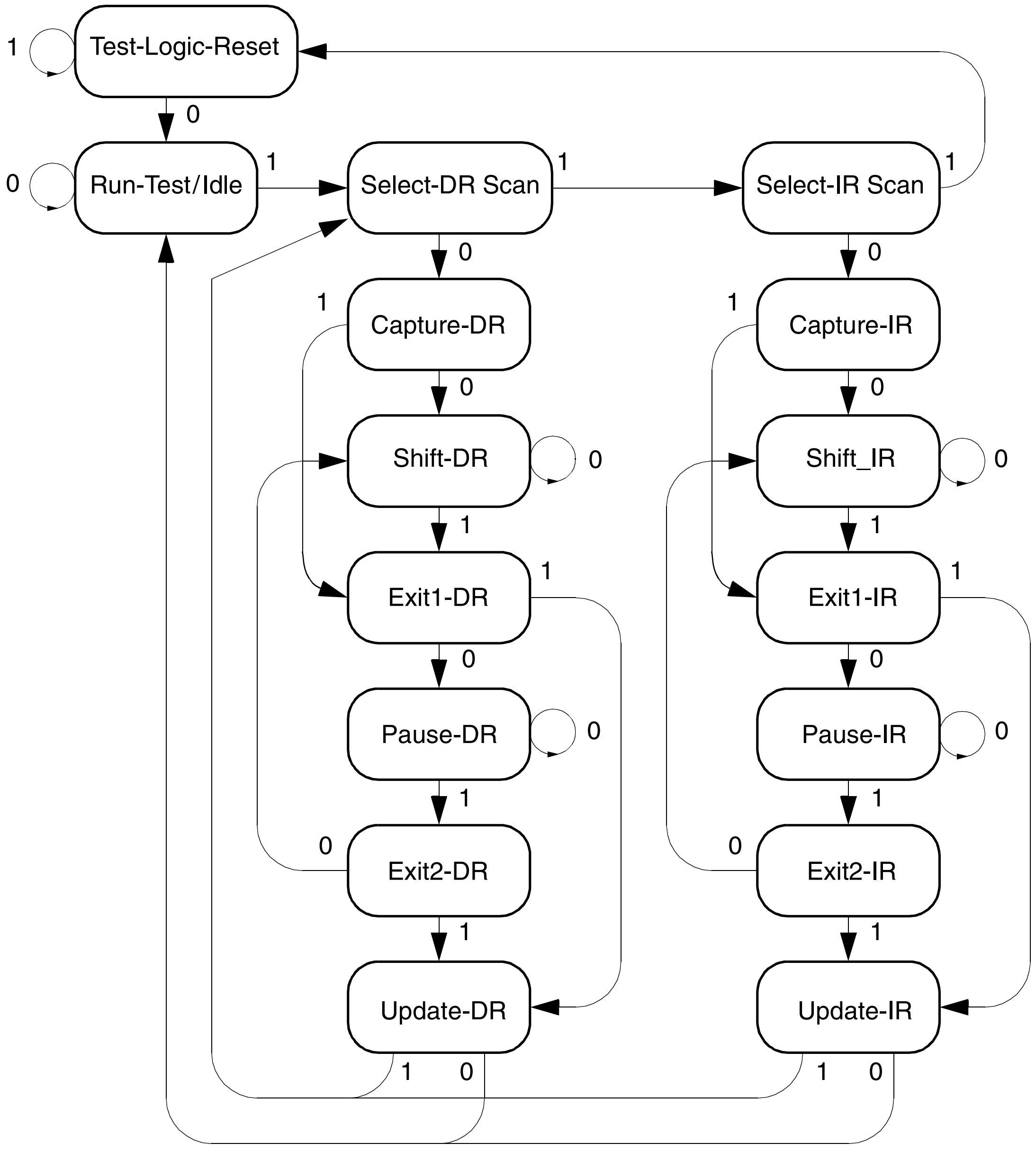


Figure 21-4 State Diagram—TAP Controller

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 Harpo 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.

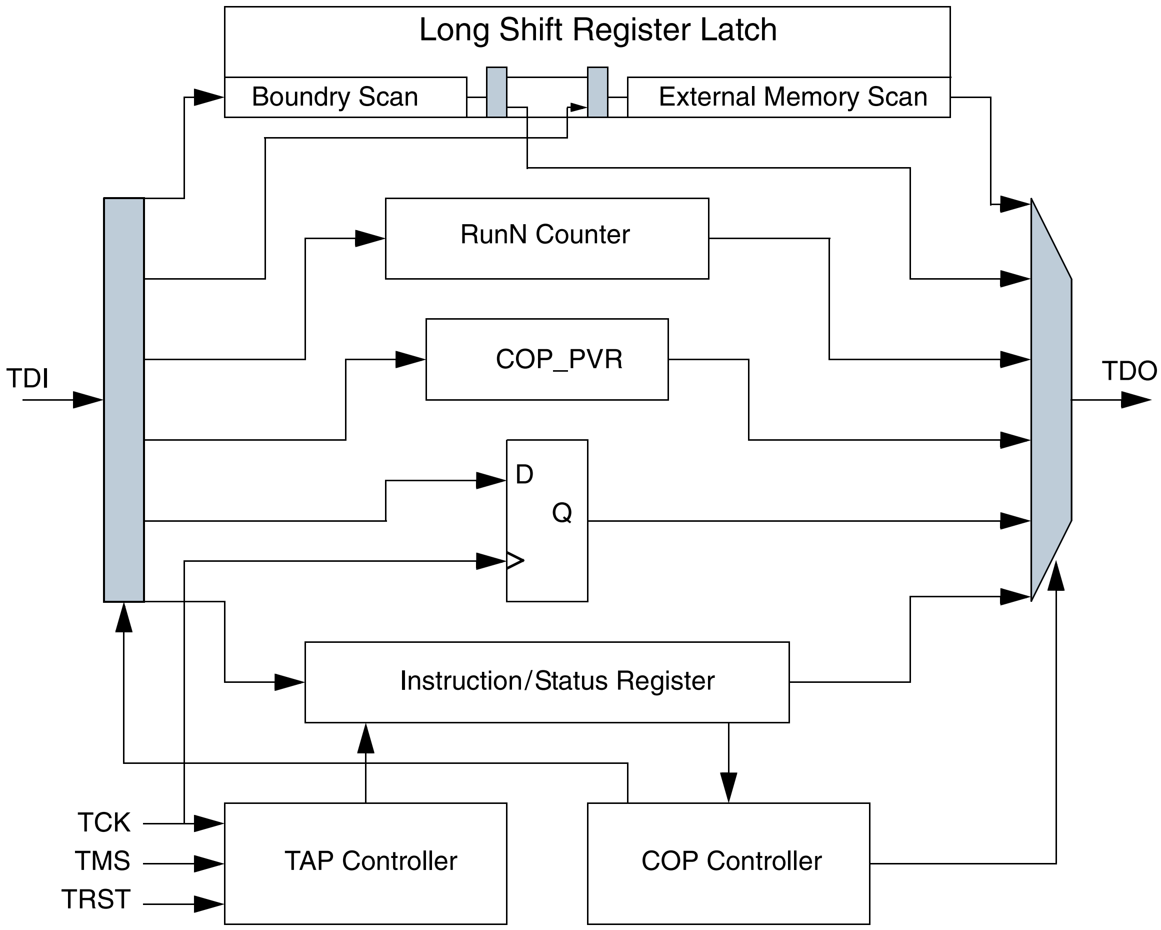


Figure 21-5 Harpo Core JTAG/COP Serial Interface

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	N[3]
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 (Motorola), device type (MGT5100), and device revision level.

21.8.1.1 Device ID Register

Table 21-3 Device ID Register = 01C5301D hex

Version	Device (MGT5100 – Initial Release)	Manufacturer (Motorola)	
000	0001 1100 0101 0011	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 504-bit 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 504-bit 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 HARPO COP/BDM Interface

The MGT5100 functional pin interface and internal logic provides access to the embedded HARPO G2 processer core through the Motorola standard COP/BDM interface. Table 21-4 gives the COP/BDM interface signals. The pin order shown reflects only the COP/BDM connector order. See Figure 21-6 for more information.

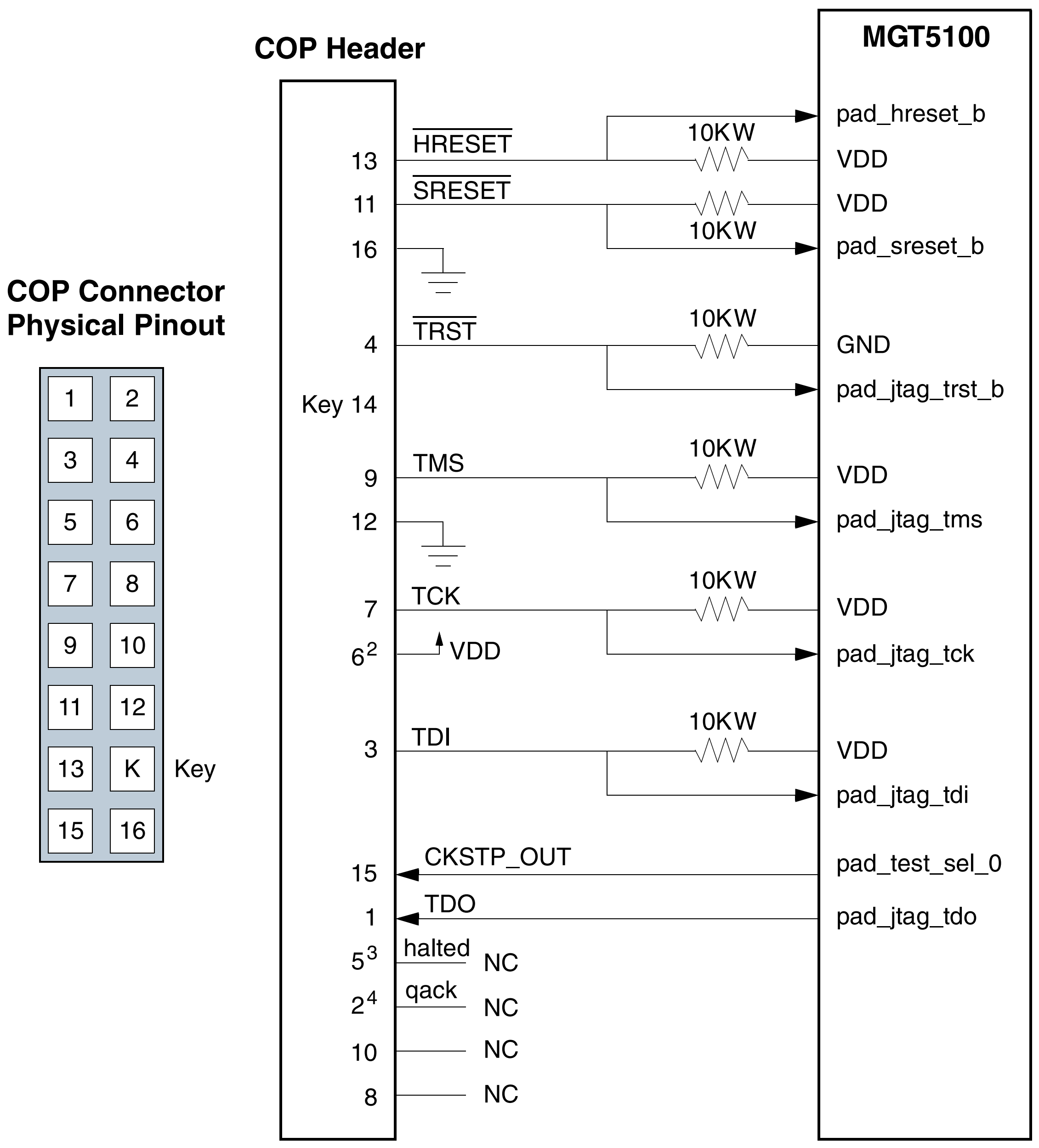
Table 21-4 COP/BDM Interface Signals

BDM Pin #	MGT5100 I/O Pin	Microprocessor Pin	BDM Connector	Internal PullUp/Down	External PullUp/Down	I/O[1]
16	—	—	GND	—	—	—
15	pad_test_sel_0	core_ckstp_out_	ckstp_out	—	—	I
14	—	—	KEY	—	—	—
13	pad_hreset_	core_hreset_	hreset		10k Pull-Up	O

Table 21-4 COP/BDM Interface Signals (continued)

BDM Pin #	MGT5100 I/O Pin	Microprocessor Pin	BDM Connector	Internal PullUp/Down	External PullUp/Down	I/O[1]
12	—	—	GND	—	—	—
11	pad_sreset_	core_sreset_	sreset		10k Pull-Up	O
10	—	—	N/C	—	—	—
9	pad_jtag_tms	core_tms	tms	100k Pull-Up	10k Pull-Up	O
8	—	—	N/C	—	—	—
7	pad_jtag_tck	core_tck	tck	100k Pull-Up	10k Pull-Up	O
6	—	—	VDD[2]	—	—	—
5	See Note [3].	—	halted[3]	—	—	I
4	pad_jtag_trst_b	core_trst_	trst	100k Pull-Up	10k Pull-Up	O
3	pad_jtag_tdi	core_tdi	tdi	100k Pull-Up	10k Pull-Up	O
2	See Note [4].	core_qack_	qack[4]	—	—	O
1	pad_jtag_tdo	core_tdo	tdo	—	—	I

NOTE:
1. With respect to the emulator tool's perspective:
   - Input is really an output from the embedded G2 core.
   - Output is really an input to the core.
2. From the board under test, power sense for chip power.
3. HALTED is not available from HARPO G2 core, from the processor, if present (i.e., MPC604 family).
4. Input to the G2 core to enable/disable soft-stop condition during breakpoints. If not present, must be internal to the design integration tied/forced low or connected to core_qreq_. MGT5100 internal ties core_qack_ to GND in its normal/functional mode (always asserted).



NOTE: (See Table 21-4 for more information.)

1. With respect to the emulator tool's perspective:
   - Input is really an output from the embedded G2 core.
   - Output is really an input to the core.
2. From the board under test, power sense for chip power.
3. HALTED is not available from HARPO G2 core, from the processor, if present (i.e., MPC604 family).
4. Input to the G2 core to enable/disable soft-stop condition during breakpoints. If not present, must be internal to the design integration tied/forced low or connected to core_qreq_. MGT5100 internal ties core_qack_ to GND in its normal/functional mode (always asserted).

Figure 21-6 COP Connector Diagram

SECTION A
TIMING AND ELECTRICAL SPECIFICATIONS

A.1 AC Timing Quick Reference

Hyperlinks to the indicated timing specification sections are provided below.

- Clock, Section A.2.1
- Reset, Section A.2.2
- SDRAM, Section A.2.3
- PCI, Section A.2.4
- LP–CS, Section A.2.5
- ATA, Section A.2.6
- Ethernet, Section A.2.7
- IR, Section A.2.8
- IrDA, Section A.2.9
- JTAG, Section A.2.10
- USB, Section A.2.11
- SPI, Section A.2.12
- I^2C, Section A.2.13
- MSCAN, Section A.2.14
- PSC, Section A.2.15
- GPIOs and Timers, Section A.2.16

A.2 AC Timing Specifications

A.2.1 Clock

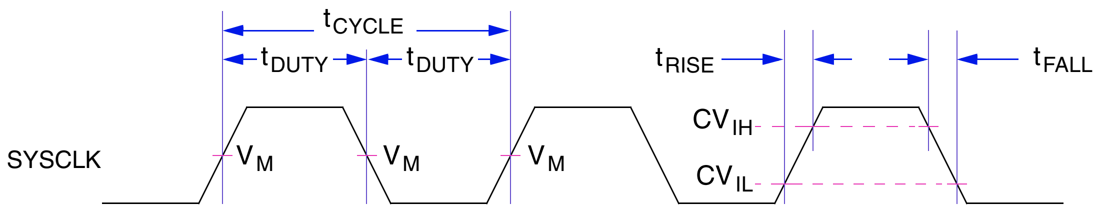


Figure A-1 Timing Diagram—SYS_XTAL_IN

Table A-1 SYS_XTAL_IN Timing

Sym	Description	Min	Max	Units
t_{CYCLE}	SYS_XTAL_IN cycle time. See Note 1.	30	40	ns
t_{RISE}	SYS_XTAL_IN rise time.	—	5.0	ns
t_{FALL}	SYS_XTAL_IN fall time.	—	5.0	ns
t_{DUTY}	SYS_XTAL_IN duty cycle (measured at V_M). See Note 2.	40.0	60.0	%
CV_{IH}	SYS_XTAL_IN input voltage high	2.0	—	V
CV_{IL}	SYS_XTAL_IN input voltage low	—	0.8	V

NOTE:
1. CAUTION—The SYS_XTAL_IN frequency and PLL_CFG[0-6] settings must be chosen such the resulting G2 processor core frequency and the processor bus frequency (xlb_clk) do not exceed there respective maximum or minimum operating frequencies. For more information see Section 4.6, Reset Configuration and Section 5, Clocks and Power Management.
2. SYS_XTAL_IN duty cycle is measured at V_M. Timing is guaranteed by design and characterization and is not tested.

Table A-2 Clock Frequencies

		Min	Max	Units
1	G2 Processor Core	—	231	MHz
2	SDRAM Clock	—	66	MHz
3	Local Plus bus	—	66	MHz
4	PLL Input Range	27	33	MHz

NOTE: Maximum Local Plus Bus frequency is 33MHz with SmartComm enabled.

Table A-3 Power Consumption

HARPO Core	**Min**	**Max**	**Units**
VDD (3.3V ±5%)	1.71	1.89	V
Run Idd @ 231 MHz – 1.8V supply	—	600	mA
Run Idd @ 100 MHz – 1.8V supply	—	300	mA

I/O and Memory Interfaces	Min	Max	Units
run Idd – 3.3V supply	—	330	mA
Deep Sleep mode	—	3	mA

A.2.2 Reset

The MGT5100 has 3 reset signals:

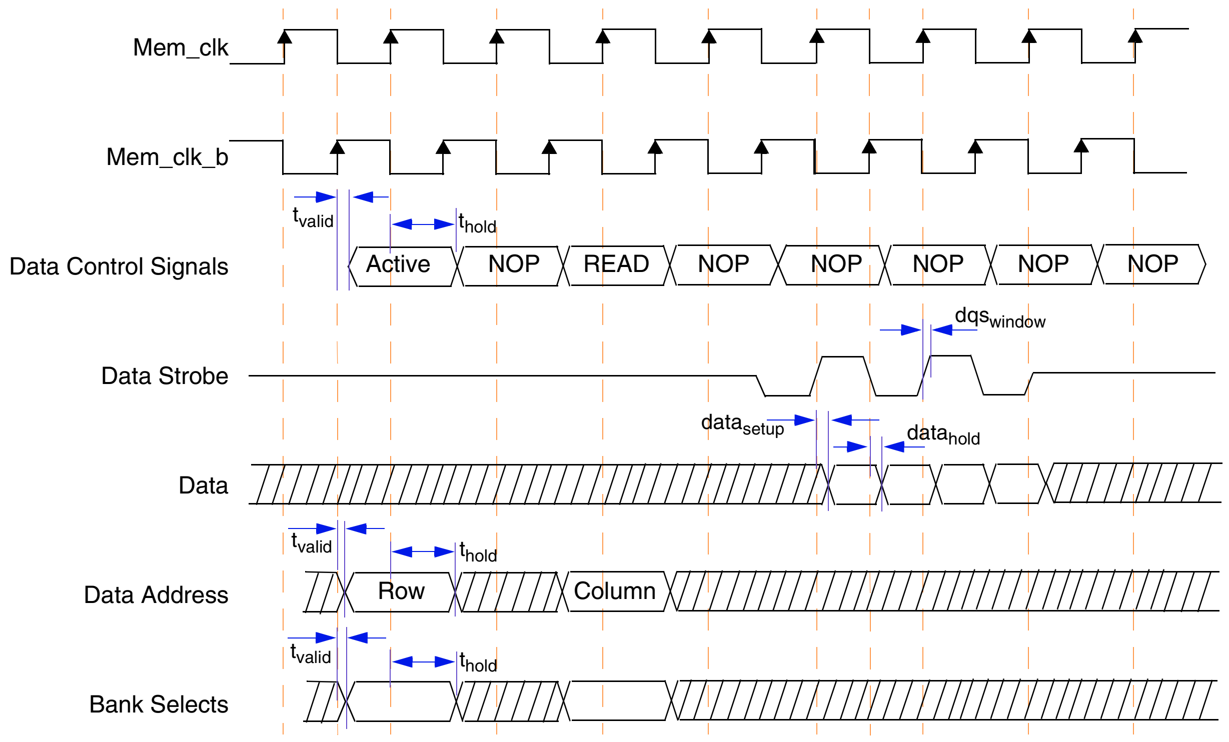
- $\overline{\text{PORRESET}}$
- $\overline{\text{HRESET}}$
- $\overline{\text{SRESET}}$

These signals are asynchronous I/O signals and can be asserted at any time. The input side uses a Schmitt trigger and requires the same input characteristics as other MGT5100 inputs, as specified in the DC Electrical Specifications section.

A.2.3 SDRAM

A.2.3.1 Memory Interface Timing—DDR SDRAM Read Command

The Memory Controller uses a skewable clock for reading and has a configurable register used to account for unknown board delays. Table A-1 shows the varying setup and hold that can be achieved by changing the Address Select Register in the Memory Controller.



NOTE: Data Control Signals signals are composed of RAS, CAS, WE, CS and CKE

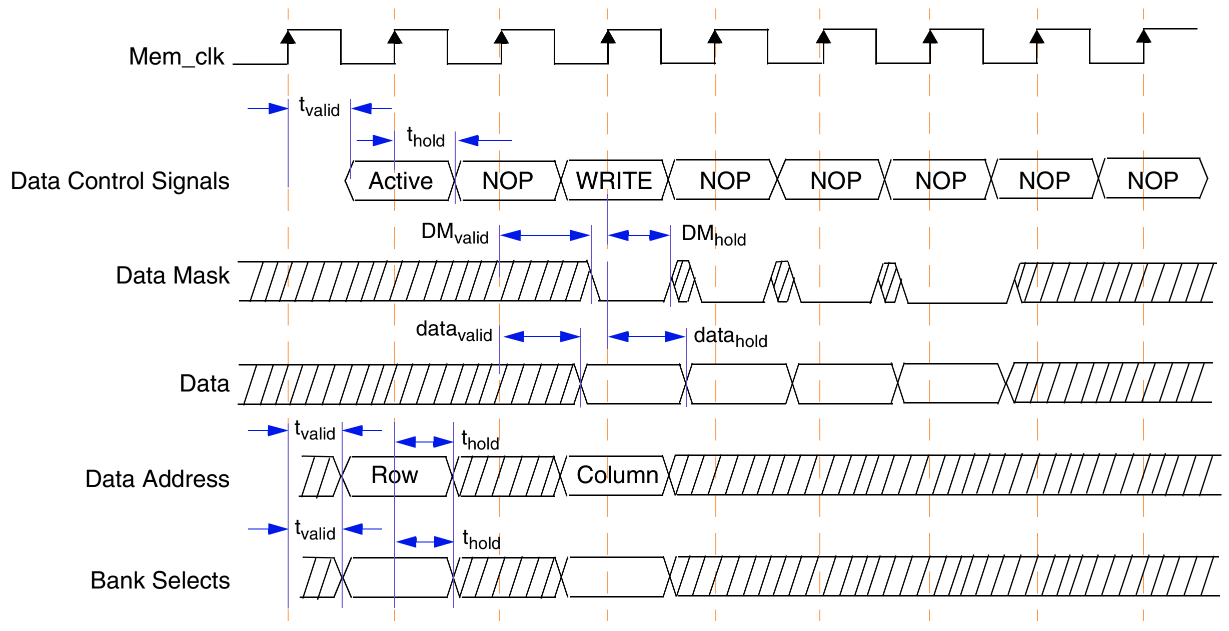
Figure A-2 Timing Diagram—DDR SDRAM Memory Read Timing

Table A-4 DDR SDRAM Memory Read Timing

Sym	Description	Min	Max	Units
t_{valid}	Control Signals, Address and BA Valid after rising edge of Mem_clk_b	—	0.10	ns
t_{hold}	Control Signals, Address and BA Hold after rising edge of Mem_clk	3.76	—	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b000	—	0.73	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b001	—	0.96	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b010	—	1.19	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b011	—	1.42	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b100	—	1.65	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b101	—	1.88	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b110	—	2.11	ns
$data_{setup}$	Setup timing skewed by Address Select Register = 0b111	—	2.34	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b000	2.34	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b001	2.11	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b010	1.88	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b011	1.65	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b100	1.42	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b101	1.19	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b110	0.96	—	ns
$data_{hold}$	Hold timing skewed by Address Select Register = 0b111	0.73	—	ns
dqs_{window}	Data Strobe window transitions around either Mem_clk or Mem_clk_b	–0.20	0.20	ns

A.2.3.2 Memory Interface Timing—Standard SDRAM Write Command

In Standard SDRAM, all signals are activated on the Mem_clk from the Memory Controller and captured on the Mem_clk clock at the memory device.



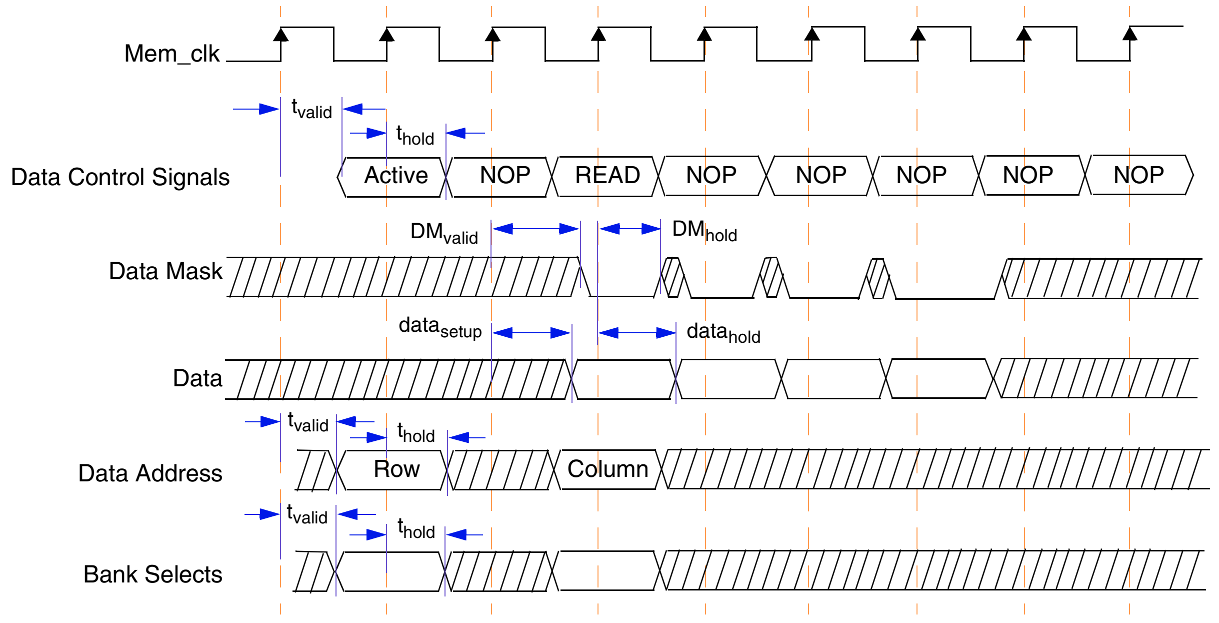
NOTE: Data Control Signals signals are composed of RAS, CAS, WE, CS and CKE

Figure A-3 Timing Diagram—Standard SDRAM Memory Write Timing

Table A-5 Standard SDRAM Memory Write Timing

Sym	Description	Min	Max	Units
t_{valid}	Control Signals, Address and BA Valid after rising edge of Mem_clk	—	4.0	ns
t_{hold}	Control Signals, Address and BA Hold after rising edge of Mem_clk	4.0	—	ns
DM_{valid}	Data Mask Valid after rising edge of Mem_clk	—	4.0	ns
DM_{hold}	Data Mask Hold after rising edge of Mem_clk	4.0	—	ns
$data_{valid}$	Data Valid after rising edge of Mem_clk	—	4.0	ns
$data_{hold}$	Data Hold after rising edge of Mem_clk	4.0	—	ns

A.2.3.3 Memory Interface Timing—Standard SDRAM Read Command



NOTE: Data Control Signals signals are composed of RAS, CAS, WE, CS and CKE

Figure A-4 Timing Diagram—Standard SDRAM Memory Write Timing

Table A-6 Standard SDRAM Memory Write Timing

Sym	Description	Min	Max	Units
t_{valid}	Control Signals, Address and BA Valid after rising edge of Mem_clk	—	4.0	ns
t_{hold}	Control Signals, Address and BA Hold after rising edge of Mem_clk	4.0	—	ns
DM_{valid}	Data Mask Valid after rising edge of Mem_clk	—	4.0	ns
DM_{hold}	Data Mask Hold after rising edge of Mem_clk	4.0	—	ns
$data_{setup}$	Data Valid after rising edge of Mem_clk	—	4.0	ns
$data_{hold}$	Data Hold after rising edge of Mem_clk	4.0	—	ns

A.2.4 PCI

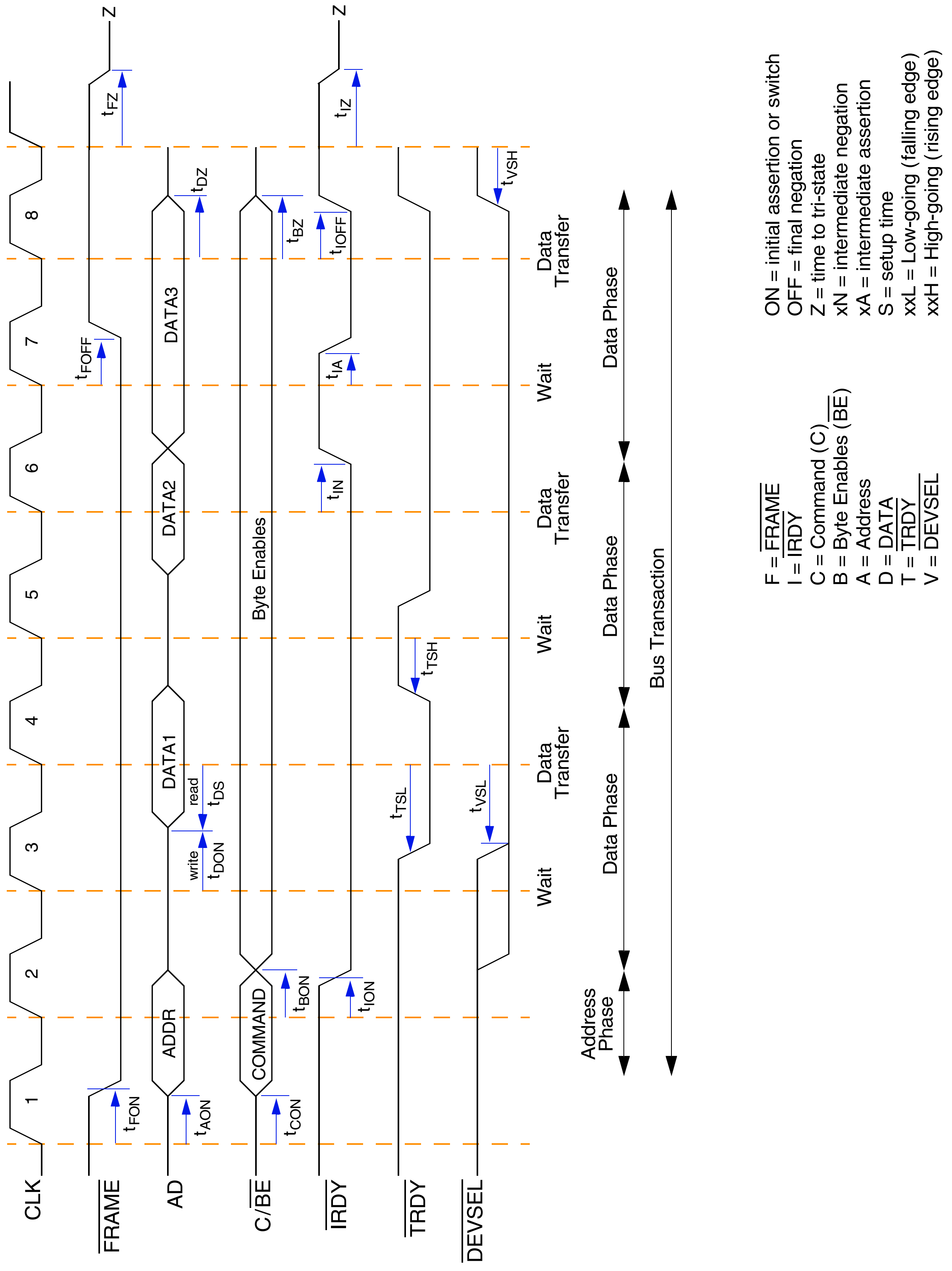


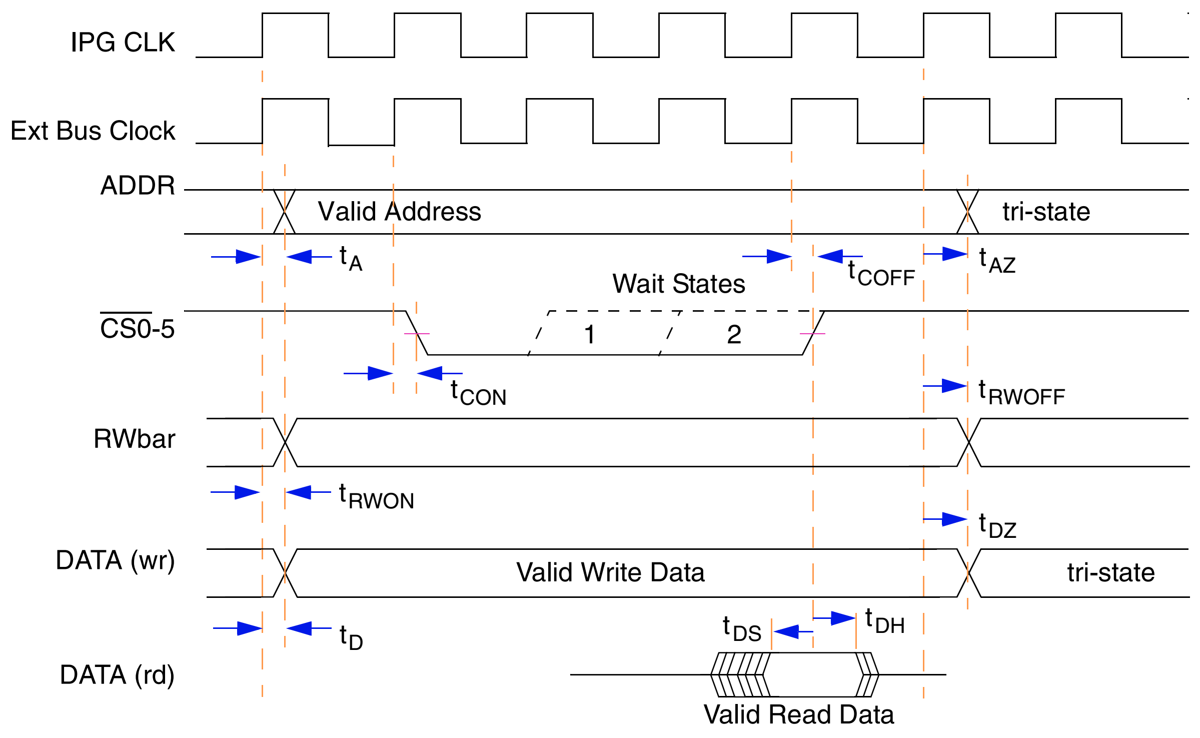
Figure A-5 PCI Timing Diagram—Basic Read/Write

Table A-7 PCI Electrical Characteristics

Sym	Description	Min	Max	Units	Notes
t_{FON}	Clock to FRAME assertion			nS	
t_{AON}	Clock to stable Address			nS	
t_{CON}	Clock to stable Command			nS	
t_{ION}	Clock to first IRDY assertion			nS	
t_{BON}	Clock to stable Byte Enables			nS	
t_{DON}	Clock to stable Write Data			nS	
t_{IN}	Clock to IRDY negation			nS	
t_{IA}	Clock to IRDY assertion			nS	
t_{FOFF}	Clock to FRAME negation			nS	
t_{IOFF}	Clock to final IRDY negation			nS	
t_{FZ}	Clock to FRAME tri-state			nS	
t_{IZ}	Clock to IRDY tri-state			nS	
t_{DS}	Read Data setup to Clock			nS	
t_{TSL}	TRDY falling setup to Clock		—	nS	
t_{TSH}	TRDY rising setup to Clock		—	nS	
t_{VSL}	DEVSEL falling setup to Clock		—	nS	
t_{VSH}	DEVSEL rising setup to Clock			nS	
t_{DZ}	Clock to Data tri-state			nS	
t_{BZ}	Clock to Byte Enables tri-state			nS	

A.2.5 LP–CS

A.2.5.1 Chip Select Non-MUXed Timing—1:1



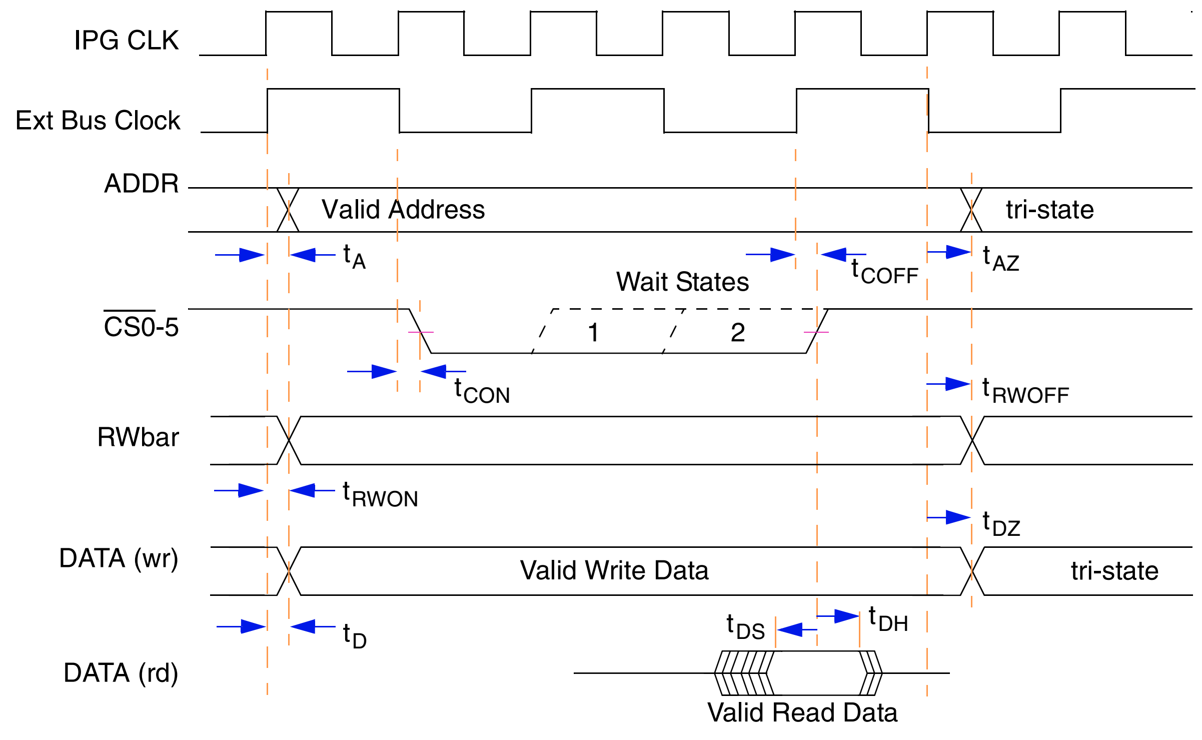
NOTE:
1. Wait states are as programmed for corresponding access and chip select.
2. Read data has nominal setup and hold requirements around the CS negation.
3. Signals are driven with one-clock setup and hold outside of CS active.

Figure A-6 Timing Diagram—Chip Select Access (non-MUXed)

Table A-8 Chip Select Access Electrical Characteristics

Sym	Description	Min	Max	Units	Notes
t_A	ExtBus clock to valid address		14	nS	
t_{AZ}	ExtBus clock to tri-state address			nS	
t_{CON}	ExtBus clock to CS assertion		14	nS	
t_{COFF}	ExtBus clock to CS negation	1		nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change in command			nS	
t_D	ExtBus clock to valid write data		4	nS	
t_{DZ}	ExtBus clock to write data tri-state		1	nS	
t_{DS}	Read data set up to ExtBus clock		3	nS	
t_{DH}	Read data hold time from ExtBus clock	1		nS	

A.2.5.2 Chip Select Non-MUXed Timing—2:1 Phase A



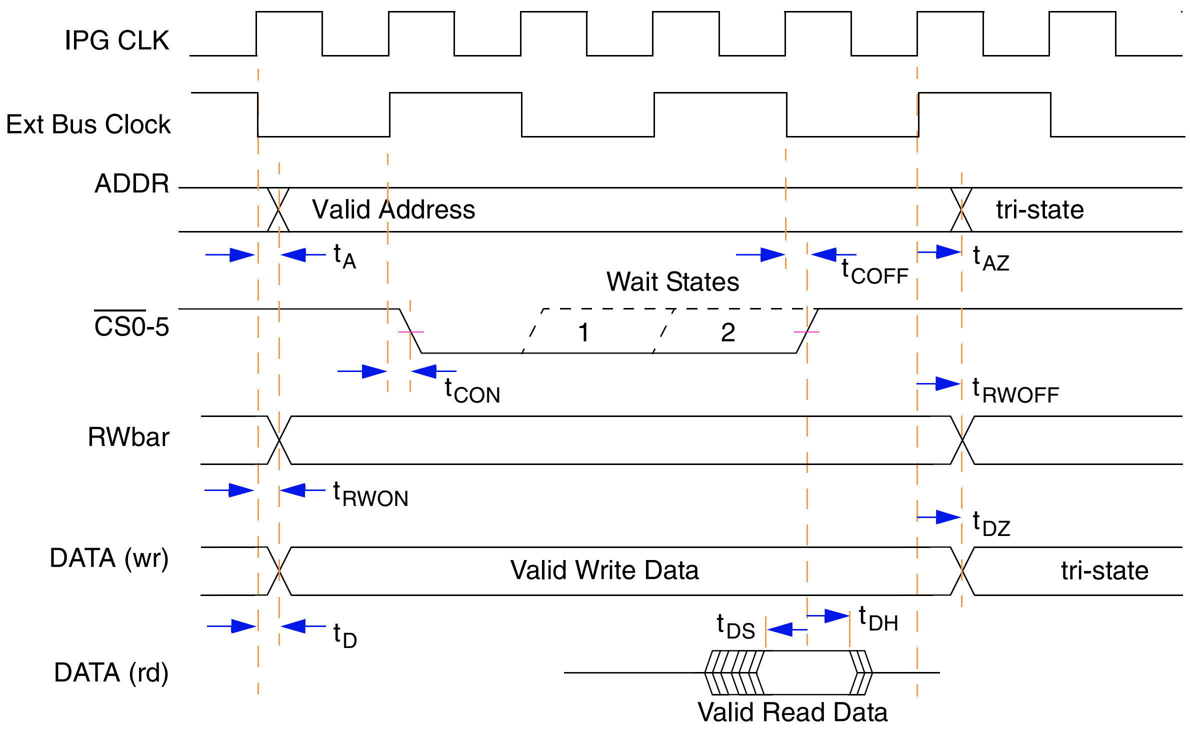
NOTE:
1. Wait states are as programmed for corresponding access and chip select.
2. Read data has nominal setup and hold requirements around the CS negation.
3. Signals are driven with one-clock setup and hold outside of CS active.

Figure A-7 Timing Diagram—Chip Select Access (non-MUXed)

Table A-9 Chip Select Access Electrical Characteristics

Sym	Description	Min	Max	Units	Notes
t_A	ExtBus clock to valid address		14	nS	
t_{AZ}	ExtBus clock to tri-state address			nS	
t_{CON}	ExtBus clock to CS assertion		14	nS	
t_{COFF}	ExtBus clock to CS negation	1		nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change in command			nS	
t_D	ExtBus clock to valid write data		4	nS	
t_{DZ}	ExtBus clock to write data tri-state		1	nS	
t_{DS}	Read data set up to ExtBus clock		3	nS	
t_{DH}	Read data hold time from ExtBus clock	1		nS	

A.2.5.3 Chip Select Non-MUXed Timing—2:1 Phase B



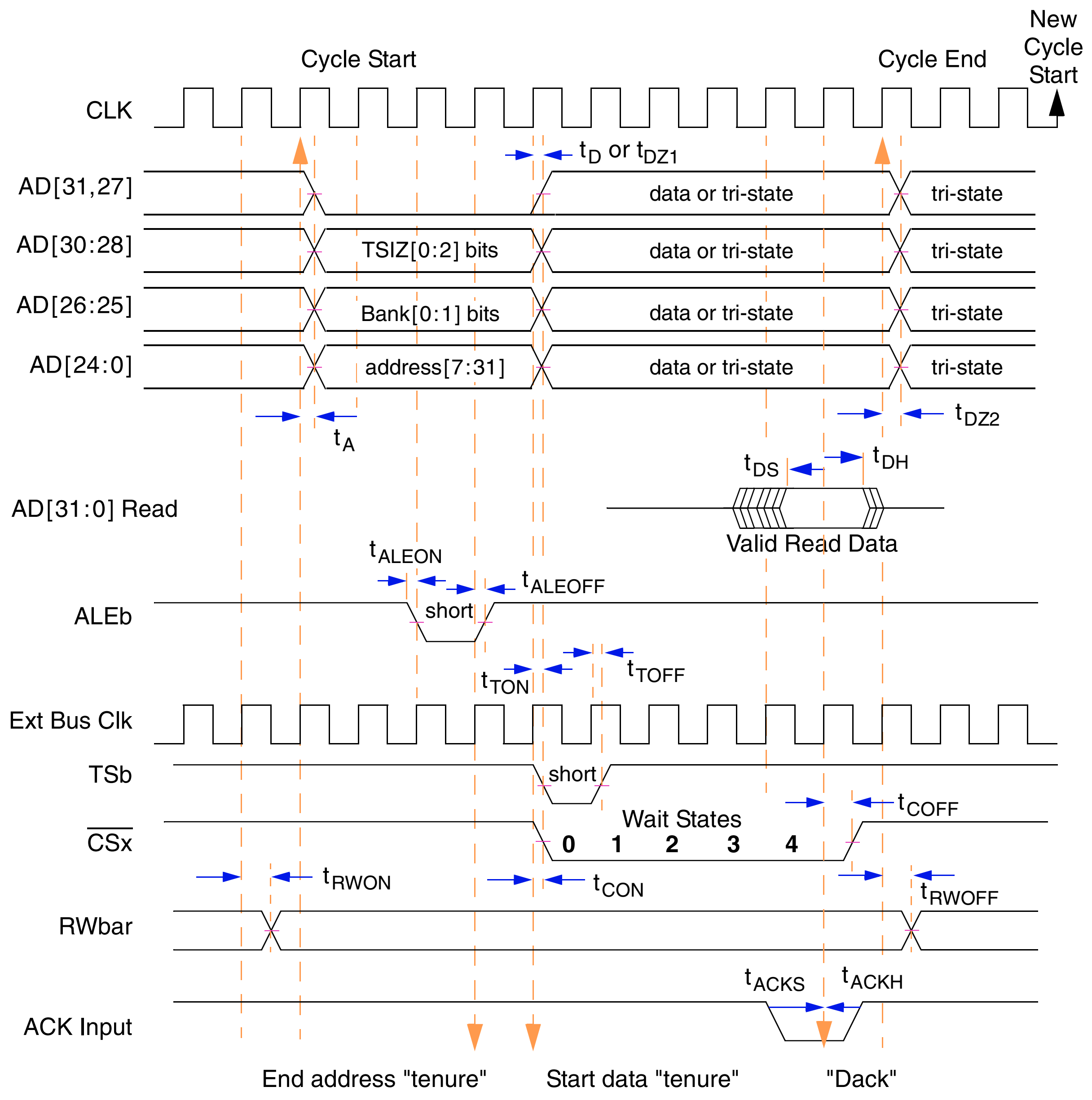
NOTE:
1. Wait states are as programmed for corresponding access and chip select.
2. Read data has nominal setup and hold requirements around the CS negation.
3. Signals are driven with one-clock setup and hold outside of CS active.

Figure A-8 Timing Diagram—Chip Select Access (non-MUXed)

Table A-10 Chip Select Access Electrical Characteristics

Sym	Description	Min	Max	Units	Notes
t_A	ExtBus clock to valid address		14	nS	
t_{AZ}	ExtBus clock to tri-state address			nS	
t_{CON}	ExtBus clock to CS assertion		14	nS	
t_{COFF}	ExtBus clock to CS negation	1		nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change in command			nS	
t_D	ExtBus clock to valid write data		4	nS	
t_{DZ}	ExtBus clock to write data tri-state		1	nS	
t_{DS}	Read data set up to ExtBus clock		3	nS	
t_{DH}	Read data hold time from ExtBus clock	1		nS	

A.2.5.4 Chip Select MUXed Timing—1:1



NOTE:

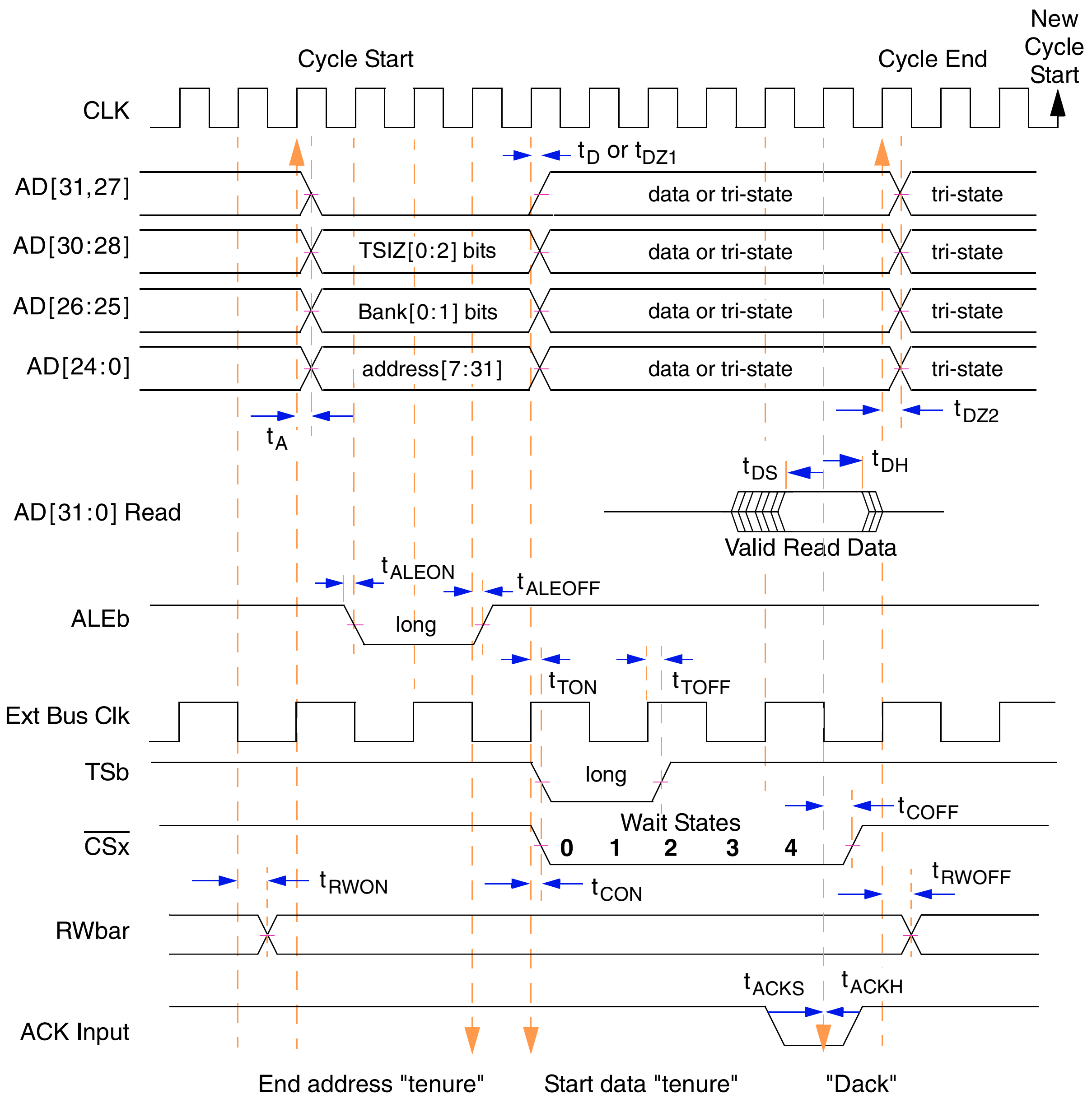
1. During data tenure, the decision between data being driven or the AD bit being tri-stated is dependent on whether the transaction is a Read or Write and what the programmed Data Size is. Unused bits in the data tenure (i.e., those in excess of the programmed data size) are driven to zero by the LPC as a precaution to avoid a floating bus condition.
2. The cycle terminates without an ACK, if the internal wait-state condition expires. In either case, the data and control signals are maintained one clock cycle beyond CSx negation to assure hold time.
3. Use of ACK for termination is software programmable.
4. Ext Bus Clk, if 1/2 internal CLK, may occur 180 degrees phase-shifted for any given transaction.

Figure A-9 Timing Diagram—Chip Select Access (MUXed, 1:1)

Table A-11   CS MUXed Electrical Characteristics—1:1

Sym	Description	Min	Max	Units	Notes
t_{DH}	Data hold time from ExtBus clock	1		nS	
t_{AH}	Address hold time from ExtBus clock	1		nS	
t_D	ExtBus clock to write data stable	12		nS	
tZ_{D1}	ExtBus clock to read data tri-state			nS	
t_{DZ2}	ExtBus clock to data tri-state	2		nS	
t_{TON}	ExtBus clock to TSb asserted	2		nS	
t_{TOFF}	ExtBus clock to TSb negated		14	nS	
t_{CON}	ExtBus clock to $\overline{\text{CSx}}$ assertion	1		nS	
t_{COFF}	ExtBus clock to $\overline{\text{CSx}}$ negation			nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change incommand			nS	
t_{ACKS}	ACK input setup time to ExtBus clock	5		nS	
t_{ACKH}	ACK input hold time from ExtBus clock	2		nS	
t_{ALEON}	ExtBus clock to ALEb asserted	10		nS	
t_{ALEOFF}	ExtBus clock to ALEb negated	10		nS	

A.2.5.5 Chip Select MUXed Timing—2:1 Phase A



NOTE:

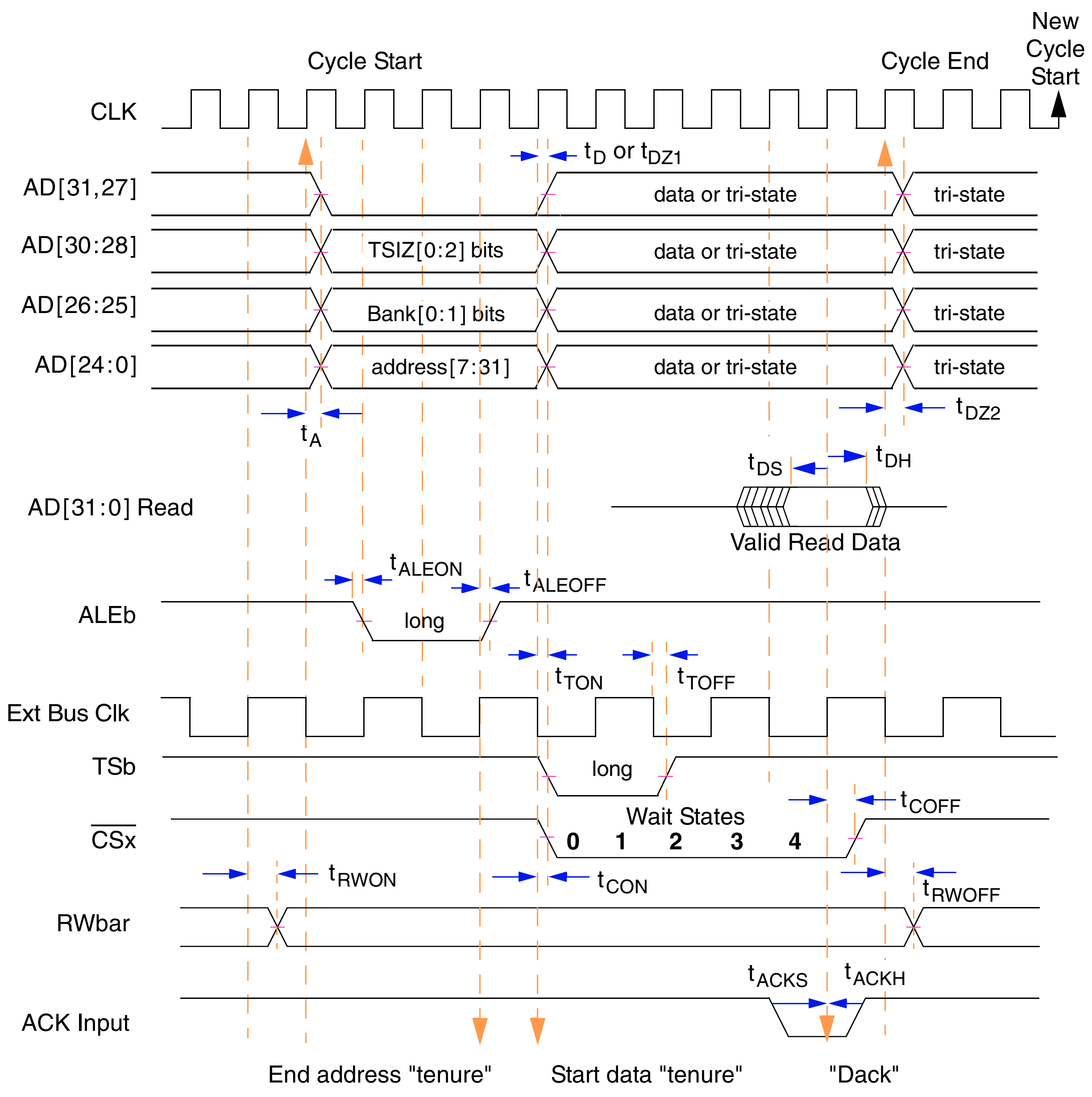
1. During data tenure, the decision between data being driven or the AD bit being tri-stated is dependent on whether the transaction is a Read or Write and what the programmed Data Size is. Unused bits in the data tenure (i.e., those in excess of the programmed data size) are driven to zero by the LPC as a precaution to avoid a floating bus condition.
2. The cycle terminates without an ACK, if the internal wait-state condition expires. In either case, the data and control signals are maintained one clock cycle beyond CSx negation to assure hold time.
3. Use of ACK for termination is software programmable.
4. Ext Bus Clk, if 1/2 internal CLK, may occur 180 degrees phase-shifted for any given transaction.

Figure A-10 Timing Diagram—Chip Select (MUXed, 2:1 Phase A)

Table A-12 CS MUXed Electrical Characteristics—2:1 Phase A

Sym	Description	Min	Max	Units	Notes
t_{DH}	Data hold time from ExtBus clock	1		nS	
t_{AH}	Address hold time from ExtBus clock	1		nS	
t_{D}	ExtBus clock to write data stable	12		nS	
tZ_{D1}	ExtBus clock to read data tri-state			nS	
t_{DZ2}	ExtBus clock to data tri-state			nS	
t_{TON}	ExtBus clock to TSb asserted	2		nS	
t_{TOFF}	ExtBus clock to TSb negated	2		nS	
t_{CON}	ExtBus clock to $\overline{\text{CSx}}$ assertion		14	nS	
t_{COFF}	ExtBus clock to $\overline{\text{CSx}}$ negation	1		nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change incommand			nS	
t_{ACKS}	ACK input setup time to ExtBus clock	5		nS	
t_{ACKH}	ACK input hold time from ExtBus clock	2		nS	
t_{ALEON}	ExtBus clock to ALEb asserted	10		nS	
t_{ALEOFF}	ExtBus clock to ALEb negated	10		nS	

A.2.5.6 Chip Select MUXed Timing—2:1 Phase B



NOTE:
1. During data tenure, the decision between data being driven or the AD bit being tri-stated is dependent on whether the transaction is a Read or Write and what the programmed Data Size is. Unused bits in the data tenure (i.e., those in excess of the programmed data size) are driven to zero by the LPC as a precaution to avoid a floating bus condition.
2. The cycle terminates without an ACK, if the internal wait-state condition expires. In either case, the data and control signals are maintained one clock cycle beyond CSx negation to assure hold time.
3. Use of ACK for termination is software programmable.
4. Ext Bus Clk, if 1/2 internal CLK, may occur 180 degrees phase-shifted for any given transaction.

Figure A-11 Timing Diagram—Chip Select (MUXed, 2:1 Phase B)

Table A-13 CS MUXed Electrical Characteristics—2:1 Phase B

Sym	Description	Min	Max	Units	Notes
t_{DH}	Data hold time from ExtBus clock	1		nS	
t_{AH}	Address hold time from ExtBus clock	1		nS	
t_{D}	ExtBus clock to write data stable	12		nS	
tZ_{D1}	ExtBus clock to read data tri-state			nS	
t_{DZ2}	ExtBus clock to data tri-state			nS	
t_{TON}	ExtBus clock to TSb asserted	2		nS	
t_{TOFF}	ExtBus clock to TSb negated	2		nS	
t_{CON}	ExtBus clock to $\overline{\text{CSx}}$ assertion		14	nS	
t_{COFF}	ExtBus clock to $\overline{\text{CSx}}$ negation	1		nS	
t_{RWON}	ExtBus clock to RWbar stable command			nS	
t_{RWOFF}	ExtBus clock to RWbar change incommand			nS	
t_{ACKS}	ACK input setup time to ExtBus clock	5		nS	
t_{ACKH}	ACK input hold time from ExtBus clock	2		nS	
t_{ALEON}	ExtBus clock to ALEb asserted	10		nS	
t_{ALEOFF}	ExtBus clock to ALEb negated	10		nS	

A.2.6 ATA

The MGT5100 ATA Controller is completely software programmable. It can be programmed to operate with ATA protocols using their respective timing, as described in the ANSI ATA-4 specification. The ATA interface is completely asynchronous in nature. Signal relationships are based on specific fixed timing in terms of timing units (nano seconds).

ATA data setup and hold times, with respect to Read/Write strobes, are software programmable inside the ATA Controller. Data setup and hold times are implemented using counters. The counters count the number of ATA clock cycles needed to meet the ANSI ATA-4 timing specifications. For details, refer to ANSI ATA-4 specification and how to program an ATA Controller and ATA drive for different ATA protocols and their respective timing. For more information see Section 11, ATA Controller.

The MGT5100 ATA Host Controller design makes data available coincident with the active edge of the WRITE strobe in PIO and Multiword DMA modes.

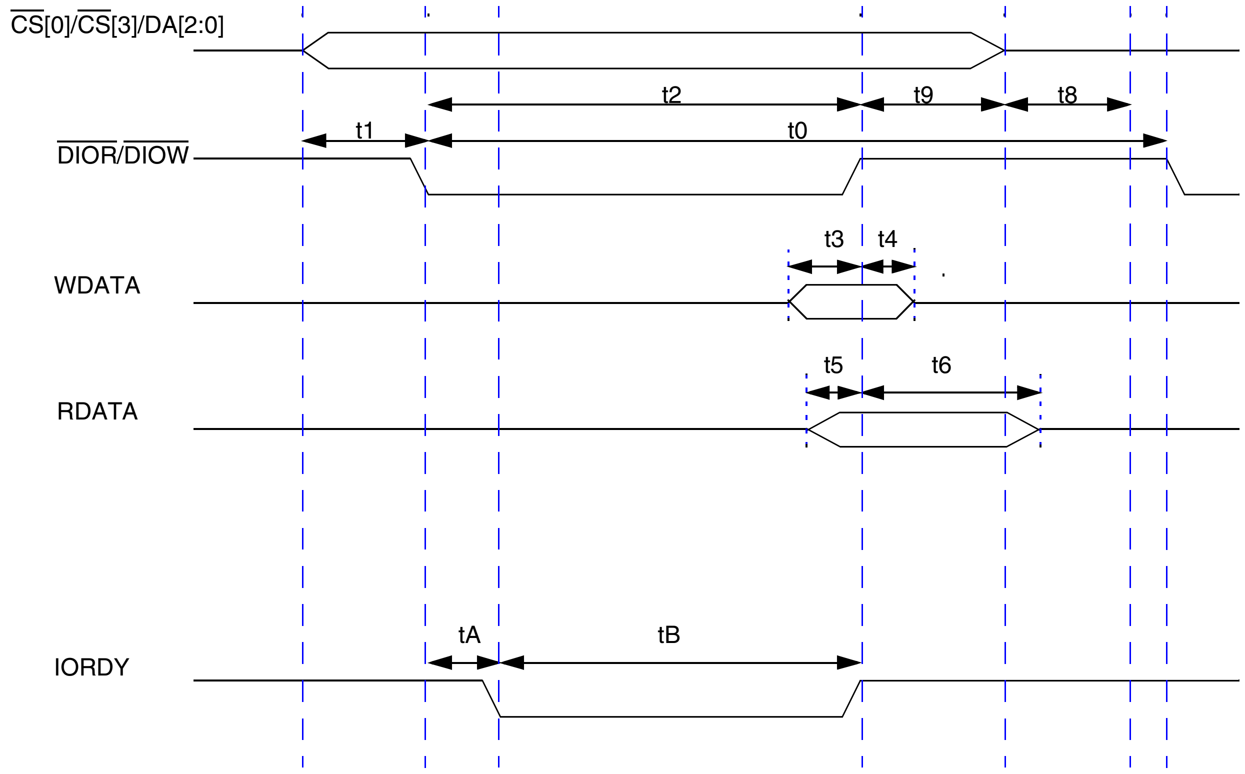
- Write data is latched by the drive at the inactive edge of the WRITE strobe. This gives ample setup-time beyond that required by the ATA-4 specification.
- Data is held unchanged until the next active edge of the WRITE strobe. This gives ample hold-time beyond that required by the ATA-4 specification.

All ATA transfers are programmed in terms of system clock cycles (IP bus clocks) in the ATA Host Controller timing registers. This puts constraints on the ATA protocols and their respective timing modes in which the ATA Controller can communicate with the drive.

Faster ATA modes (i.e., UDMA 0, 1, 2) are supported when the system is running at a sufficient frequency to provide adequate data transfer rates. Adequate data transfer rates are a function of:

- the MGT5100 operating frequency (IP bus clock frequency).
- internal MGT5100 bus latencies.
- other system load dependent variables.

The ATA clock is the same frequency as the IP bus clock in MGT5100. Section 11.4, ATA Host Controller Operation, gives more information on programming the ATA Controller for ATA protocol and mode-specific timing.

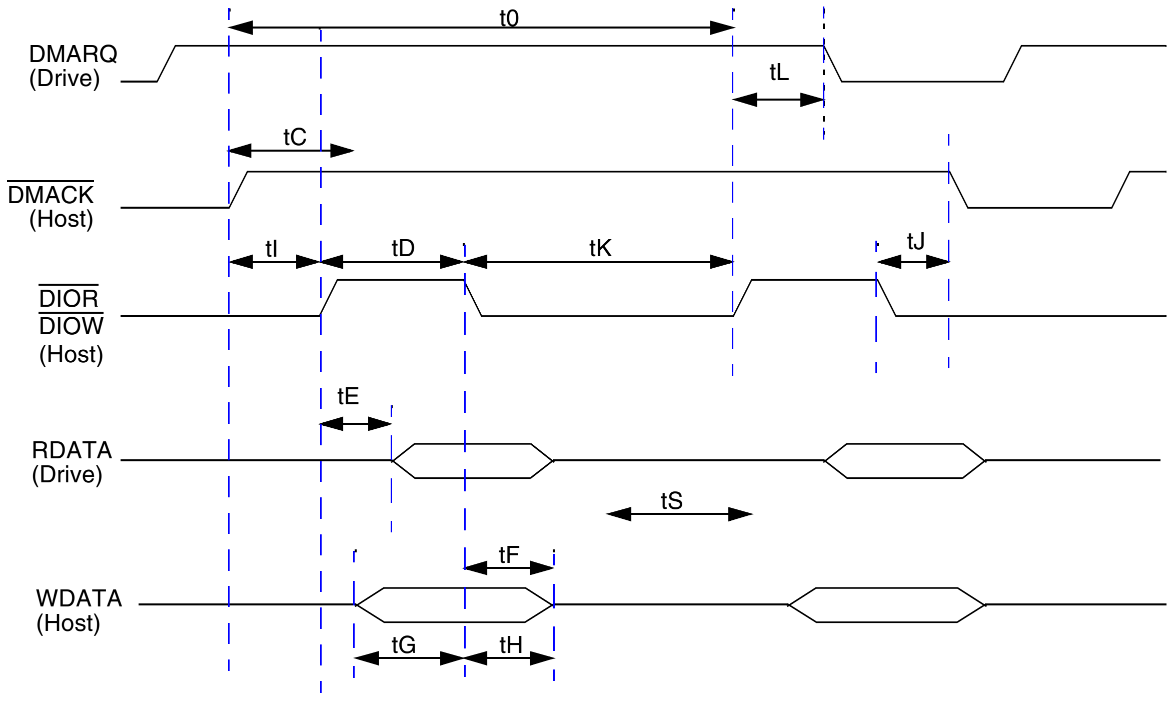


$\overline{CS}$[0]/$\overline{CS}$[3]/DA[2:0]
$\overline{DIOR}$/$\overline{DIOW}$
WDATA
RDATA
IORDY
t0
t1
t2
t3
t4
t5
t6
t8
t9
tA
tB

Figure A-12 PIO Mode Timing

Table A-14 PIO Mode Timing Specifications

	PIO Timing Parameter	Min/ Max (ns)	Mode 0 (ns)	Mode 1 (ns)	Mod e2 (ns)	Mode 3 (ns)	Mode 4 (ns)
t0	Cycle Time	min	600	383	240	180	120
t1	Address valid to $\overline{\text{DIOR}}/\overline{\text{DIOW}}$ setup	min	70	50	30	30	25
t2	$\overline{\text{DIOR}}/\overline{\text{DIOW}}$ pulse width 16-bit 8-bit	min min	165 290	125 290	100 290	80 80	70 70
t2i	$\overline{\text{DIOR}}/\overline{\text{DIOW}}$ recovery time	min	—	—	—	70	25
t3	$\overline{\text{DIOW}}$ data setup	min	60	45	30	30	20
t4	$\overline{\text{DIOW}}$ data hold	min	30	20	15	10	10
t5	$\overline{\text{DIOR}}$ data setup	min	50	35	20	20	20
t6	$\overline{\text{DIOR}}$ data hold	min	5	5	5	5	5
t9	$\overline{\text{IOR}}/\overline{\text{DIOW}}$ to address valid hold	min	20	15	10	10	10
tA	IORDY setup	max	35	35	35	35	35
tB	IORDY pulse width	max	1250	1250	1250	1250	1250



NOTE: The direction of signal assertion is towards the top of the page, and the direction of negation is towards the bottom of the page, irrespective of the electrical properties of the signal.

Figure A-13 Multiword DMA Timing

Table A-15 Multiword DMA Timing Specifications

	Multiword DMA Timing Parameters	Min/Max	Mode 0(ns)	Mode1(ns)	Mode 2(ns)
t0	Cycle Time	min	480	150	120
tC	$\overline{\text{DMACK}}$ to DMARQ delay	max	—	—	—
tD	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ pulse width (16-bit)	min	215	80	70
tE	$\overline{\text{DIOR}}$ data access	max	150	60	50
tG	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ data setup	min	100	30	20
tF	$\overline{\text{DIOR}}$ data hold	min	5	5	5
tH	$\overline{\text{DIOW}}$ data hold	min	20	15	10
tI	DMACK to $\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ setup	min	0	0	0
tJ	$\overline{\text{DIOR}}$/$\overline{\text{DIOW}}$ to $\overline{\text{DMACK}}$ hold	min	20	5	5
tKr	$\overline{\text{DIOR}}$ negated pulse width	min	50	50	25
tKw	$\overline{\text{DIOW}}$ negated pulse width	min	215	50	25
tLr	$\overline{\text{DIOR}}$ to DMARQ delay	max	120	40	35
tLw	$\overline{\text{DIOW}}$ to DMARQ delay	max	40	40	35

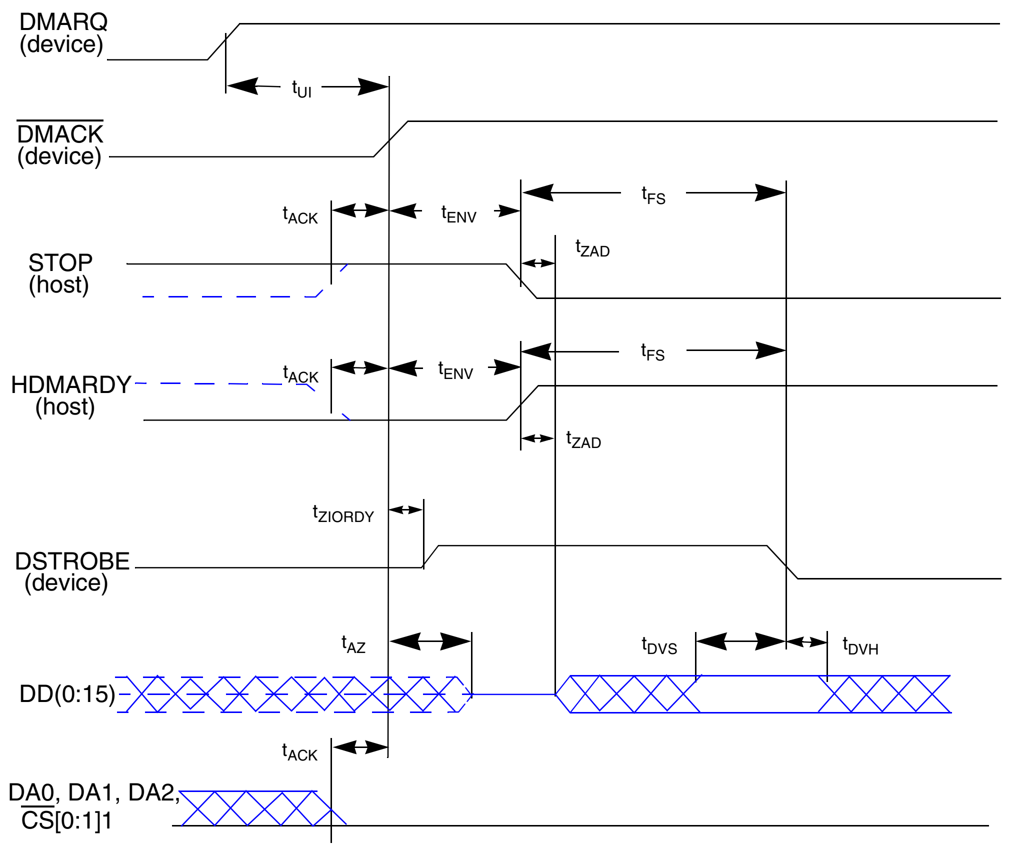
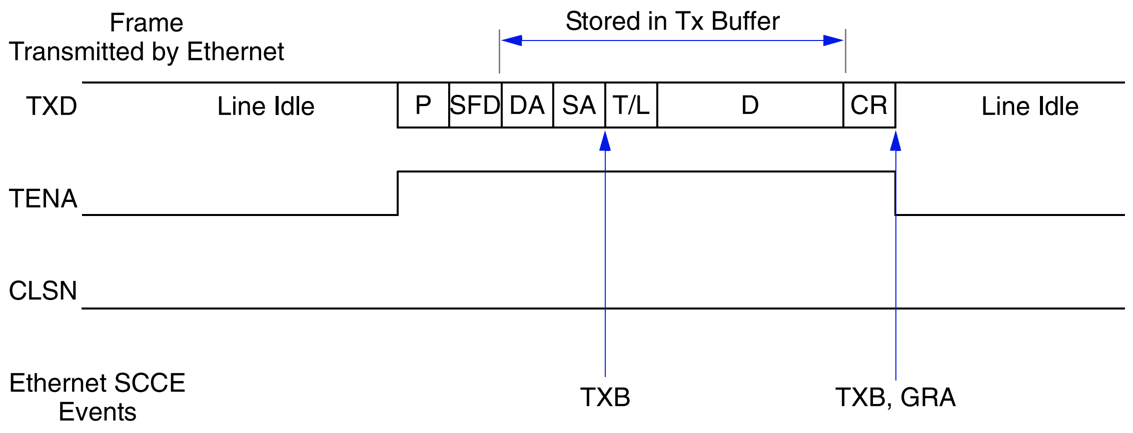


Figure A-14 Initiating an Ultra DMA Data In Burst

A.2.7 Ethernet

LEGEND:

T/L = Type/Length	P = Preamble
D = Data	SFD = Start frame delimiter
CR = CRC Bytes	DA and SA = Source/Destination address

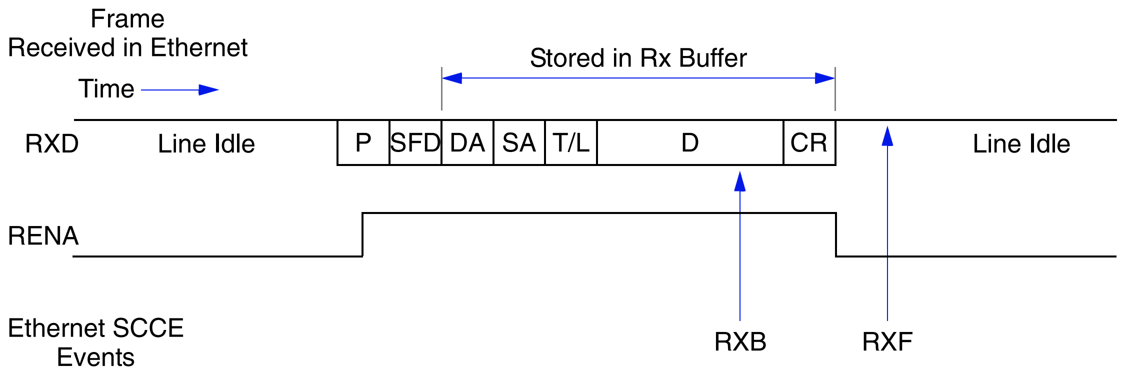


NOTES:
1. TXB events assume the frame requires 2 Txbuffers.
2. GRA event assumes a GRACEFUL STOP TRANSMIT command was issued during frame transmission.
3. TENA or CLSN events, if required, must be programmed in the port C parallel I/O, not in SCC.

Figure A-15 Ethernet Timing Diagram—Interrupt Events Tx Example

LEGEND:

T/L = Type/Length	P = Preamble
D = Data	SFD = Start frame delimiter
CR = CRC Bytes	DA and SA = Source/Destination address



NOTES:
1. RXB event assumes receive buffers are 64Bytes each.
2. RENA events, if required, must be programmed in the port C parallel I/O, not in SCC.
3. RxF interrupt may occur later than RENA due to receive FIFO latency.

Figure A-16 Ethernet Timing Diagram—Interrupt Events Rx Example

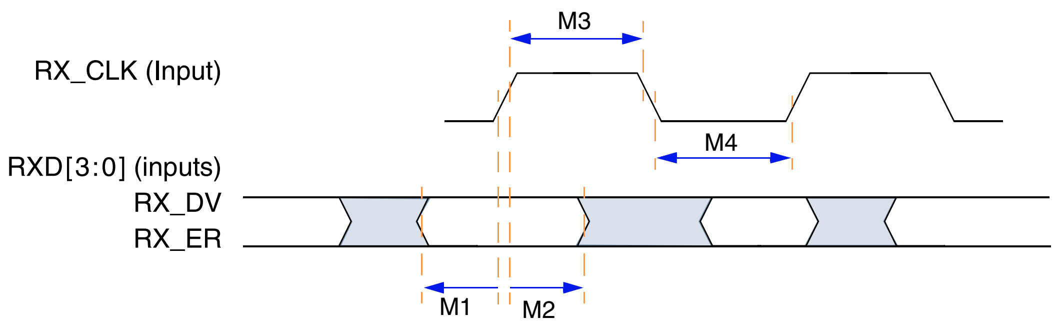


Figure A-17 Ethernet Timing Diagram—MII Rx Signal

Table A-16 MII Rx Signal Timing

Sym	Description	Min	Max	Unit
M1	RXD[3:0], RX_DV, RX_ERR to RX_CLK setup	5	—	ns
M2	RX_CLK to RXD[3:0], RX_DV, RX_ERR hold	5	—	ns
M3	RX_CLK pulse width high	35%	65%	RX_CLK Period
M4	RX_CLK pulse width low	35%	65%	RX_CLK Period

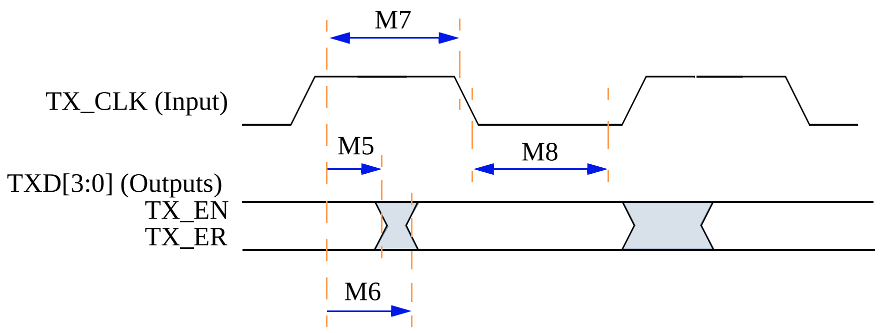


Figure A-18 Ethernet Timing Diagram—MII Tx Signal

Table A-17 MII Tx Signal Timing

Sym	Description	Min	Max	Unit
M1	RXD[3:0], RX_DV, RX_ERR to RX_CLK setup	5	—	ns
M2	RX_CLK to RXD[3:0], RX_DV, RX_ERR hold	5	—	ns
M3	RX_CLK pulse width high	35%	65%	RX_CLK Period
M4	RX_CLK pulse width low	35%	65%	RX_CLK Period

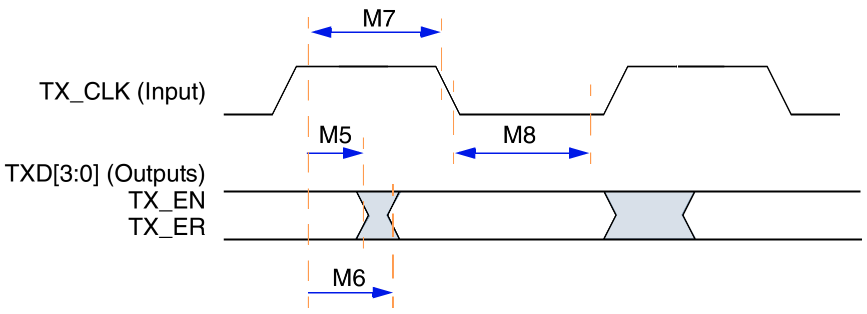


Figure A-19 Ethernet Timing Diagram—MII Tx Signal

Table A-18 MII Tx Signal Timing

Sym	Description	Min	Max	Unit
M5	TX_CLK to TXD[3:0], TX_EN, RX_ER Invalid	5	—	ns
M6	TX_CLK to TXD[3:0], TX_EN, RX_ER Valid	—	25	ns
M7	TX_CLK pulse width high	35%	65%	TX_CLK Period
M8	TX_CLK pulse width low	35%	65%	TX_CLK Period

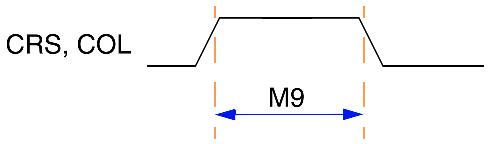


Figure A-20 Ethernet Timing Diagram—MII Async

Table A-19 MII Async Signal Timing

Sym	Description	Min	Max	Unit
M9	CRS, COL minimum pulse width	1.5	—	TX_CLK Period

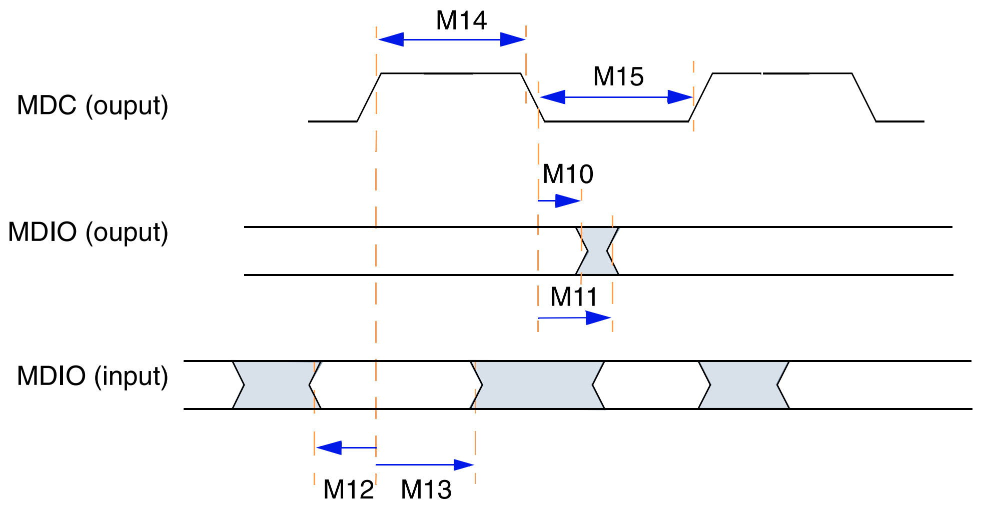


Figure A-21 Ethernet Timing Diagram—MII Serial Management

Table A-20 MII Serial Management Channel Signal Timing

Sym	Description	Min	Max	Unit
M10	MDC falling edge to MDIO output invalid (min prop delay)	0	—	ns
M11	MDC falling edge to MDIO output valid (max prop delay)	—	25	—
M12	MDIO (input) to MDC rising edge setup	10	—	ns
M13	MDIO (input) to MDC rising edge setup	0	—	—
M14	MDC pulse width high	40%	60%	MDC Period
M15	MDC pulse width low	40%	60%	MDC Period

A.2.8 IR

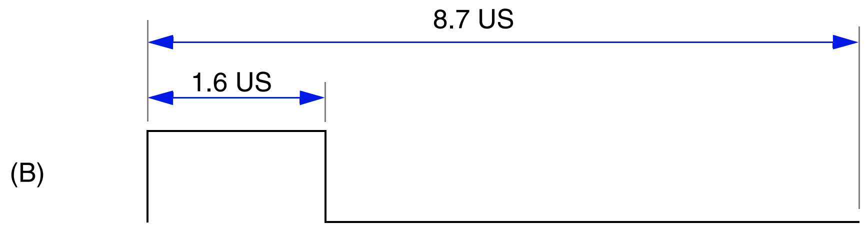


Figure A-22 IR Timing Diagram—SIP Waveform

A.2.9 IrDA

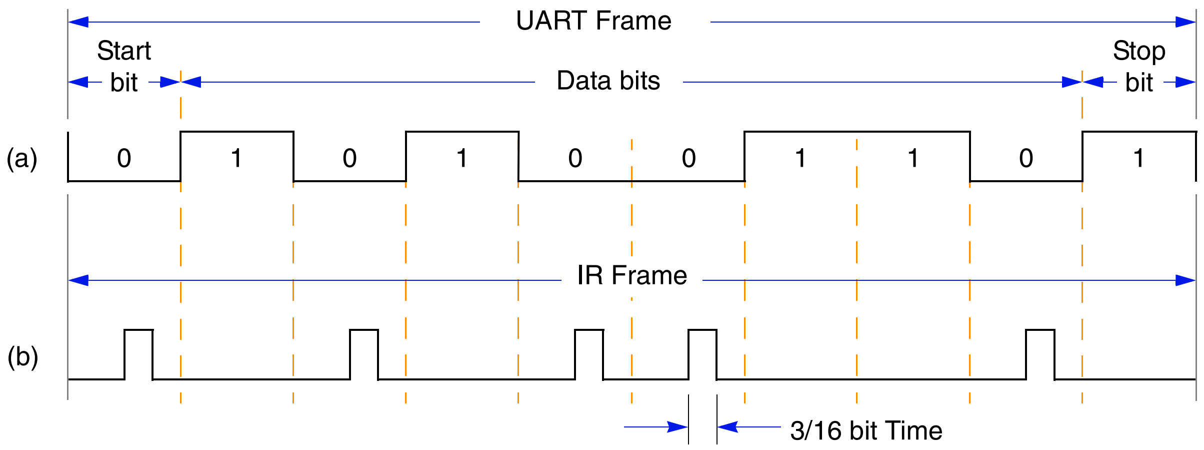


Figure A-23 IrDA Timing Diagram—Low-Speed Data Format

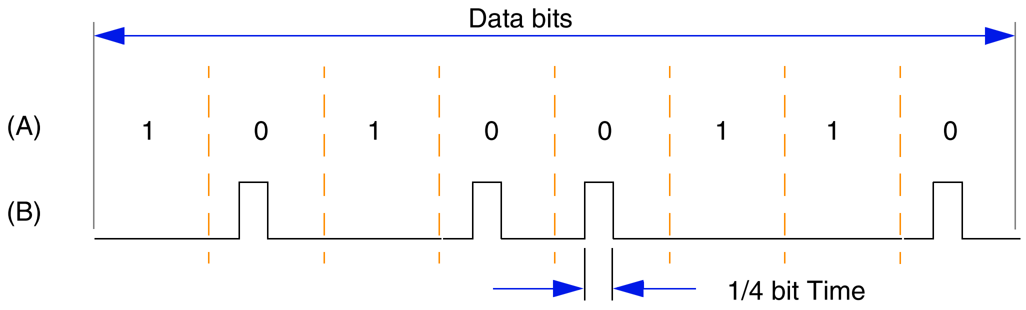


Figure A-24 IrDA Timing Diagram—Middle-Speed Data Format

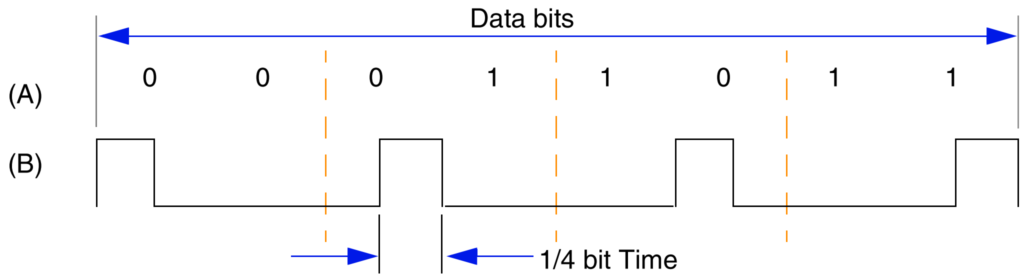
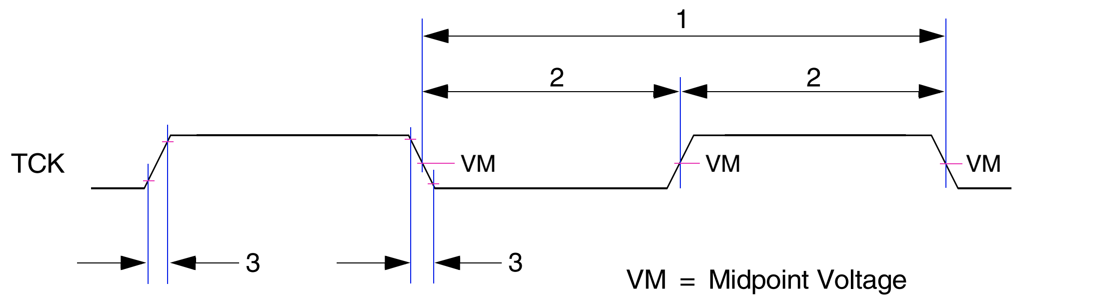


Figure A-25 IrDA Timing Diagram—High-Speed Data Format

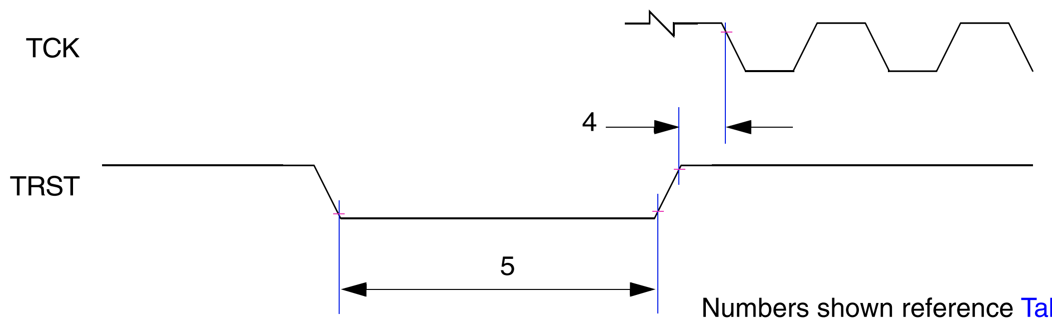
A.2.10 JTAG

A.2.10.1 IEEE 1149.1 (JTAG) AC Timing Specification



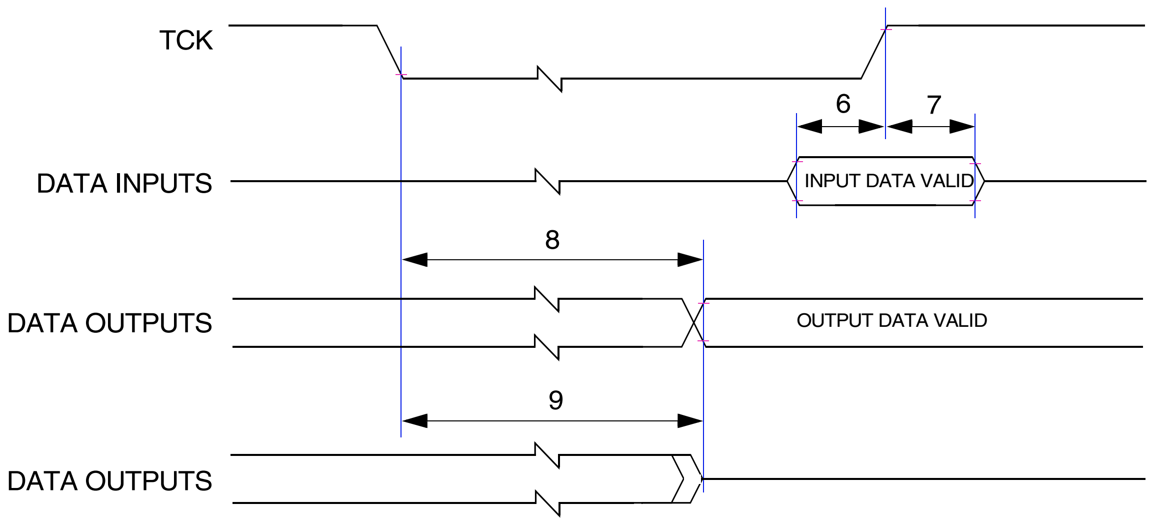
VM = Midpoint Voltage
Numbers shown reference Table A-21.

Figure A-26 Timing Diagram—JTAG Clock Input



Numbers shown reference Table A-21.

Figure A-27 Timing Diagram—JTAG TRST



Numbers shown reference Table A-21.

Figure A-28 Timing Diagram—JTAG Boundary Scan

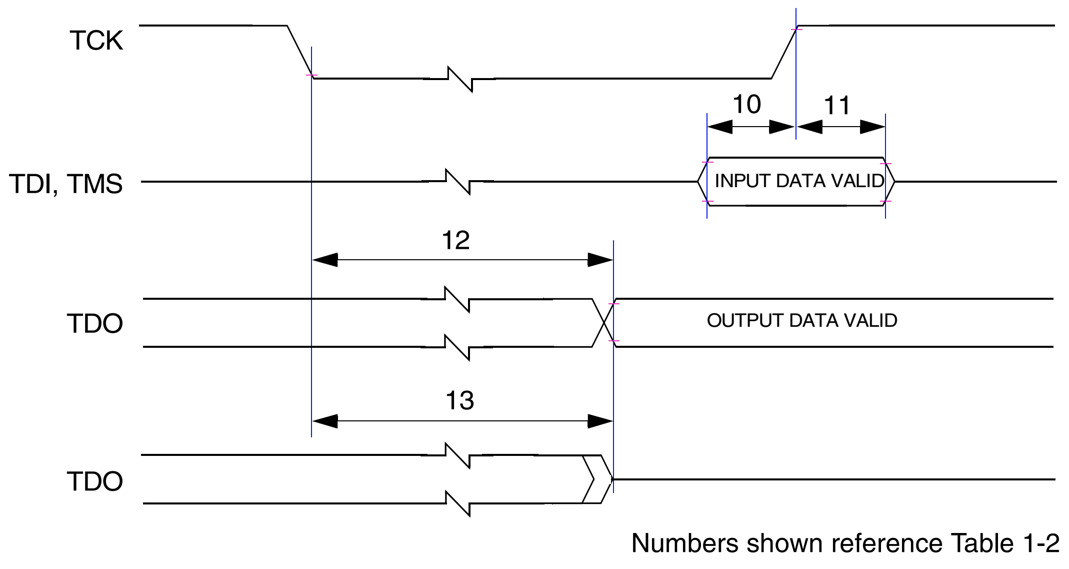


Figure A-29 Timing Diagram—Test Access Port

Table A-21 JTAG Timing Specification

Sym	Characteristic	Min	Max	Unit
—	TCK frequency of operation.	0	25	MHz
1	TCK cycle time.	40	—	nS
2	TCK clock pulse width measured at 1.5V.	1.08	—	nS
3	TCK rise and fall times.	0	3	nS
4	TRST_ setup time to tck falling edge. See Note 1.	10	—	nS
5	TRST_ assert time.	5	—	nS
6	Input data setup time. See Note 2.	5	—	nS
7	Input data hold time.See Note 2.	15	—	nS
8	TCK to output data valid. See Note 3.	0	30	nS
9	TCK to output high impedance. See Note 3.	0	30	nS
10	TMS, TDI data setup time.	5	—	nS
11	TMS, TDI data hold time.	1	—	nS
12	TCK to TDO data valid.	0	15	nS
13	TCK to TDO high impedance.	0	15	nS

NOTE:
1. $\overline{\text{TRST}}$ is an asynchronous signal. The setup time is for test purposes only.
2. Non-test, other than TDI and TMS, signal input timing with respect to TCK.
3. Non-test, other than TDO, signal output timing with respect to TCK.

A.2.11 USB

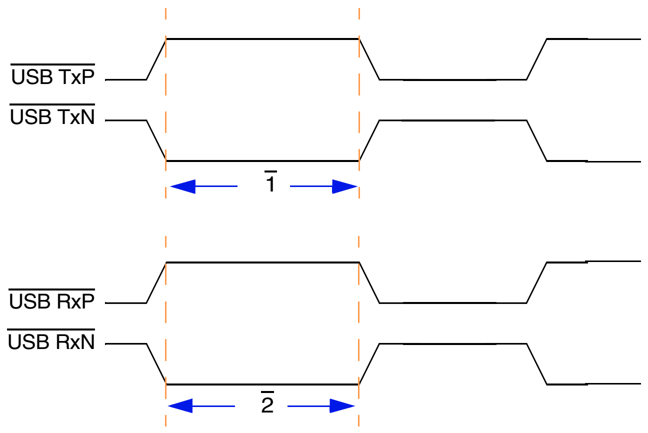


Figure A-30 Timing Diagram—USB

Table A-22 USB Timing Specifications

Sym	Description	All Frequencies		Units
		Min	Max	
1	USB Tx Bit Time	8.3	—	ns
2	USB Rx Bit Time	8.3	—	ns

A.2.12 SPI

A.2.12.1 SPI Master AC Timing Specifications

Figure A-31 and Figure A-32 shows the SPI Master AC Timing Diagrams. Table A-23 gives the timing specifications.

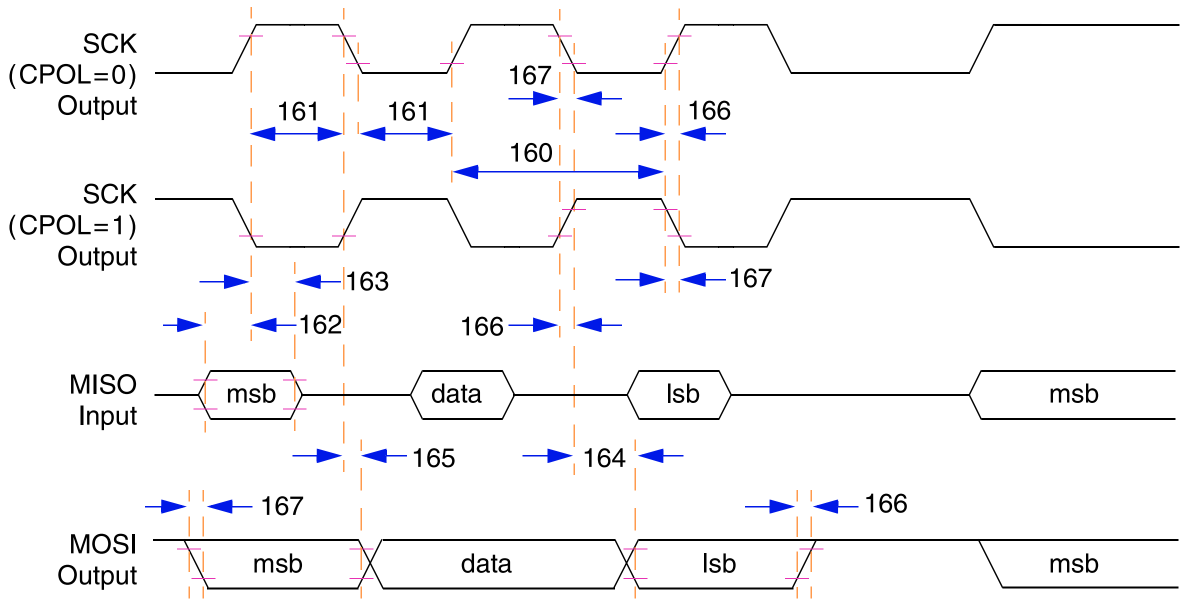


Figure A-31 Timing Diagram—SPI Master (CPHA=0)

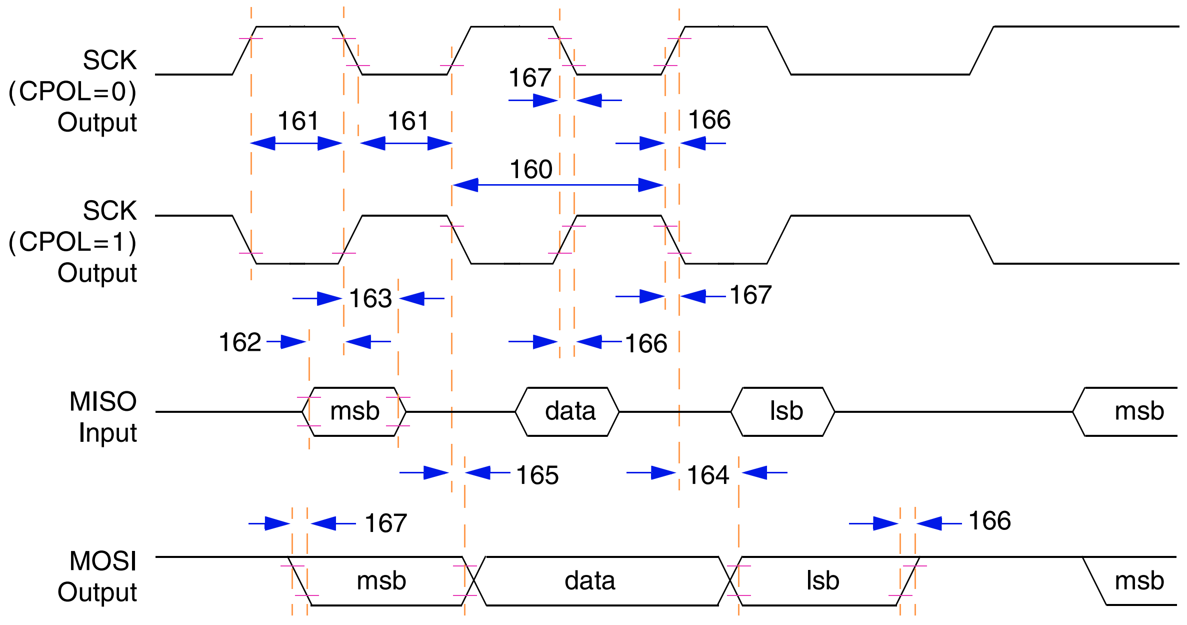


Figure A-32 Timing Diagram—SPI Master (CPHA=1)

Table A-23 SPI Master AC Timing Specifications

Sym	**Description**	**All Frequencies**		**Units**
		Min	**Max**	
160	Master cycle time	4	1024	t_{cyc}
161	Master clock (SCK) high or low time	2	512	t_{cyc}
162	Master data setup time (inputs)	50.00	—	nS
163	Master data hold time (inputs)	0.00	—	nS
164	Master data valid (after SCK edge)	—	20.00	nS
165	Master data hold time (outputs)	0.00	—	nS
166	Rise time output	—	15.00	nS
167	Fall time output	—	15.00	nS

A.2.12.2 SPI Slave AC Timing Specifications

Figure A-33 shows the SPI Slave AC Timing Diagram. Table A-24 gives the timing specifications.

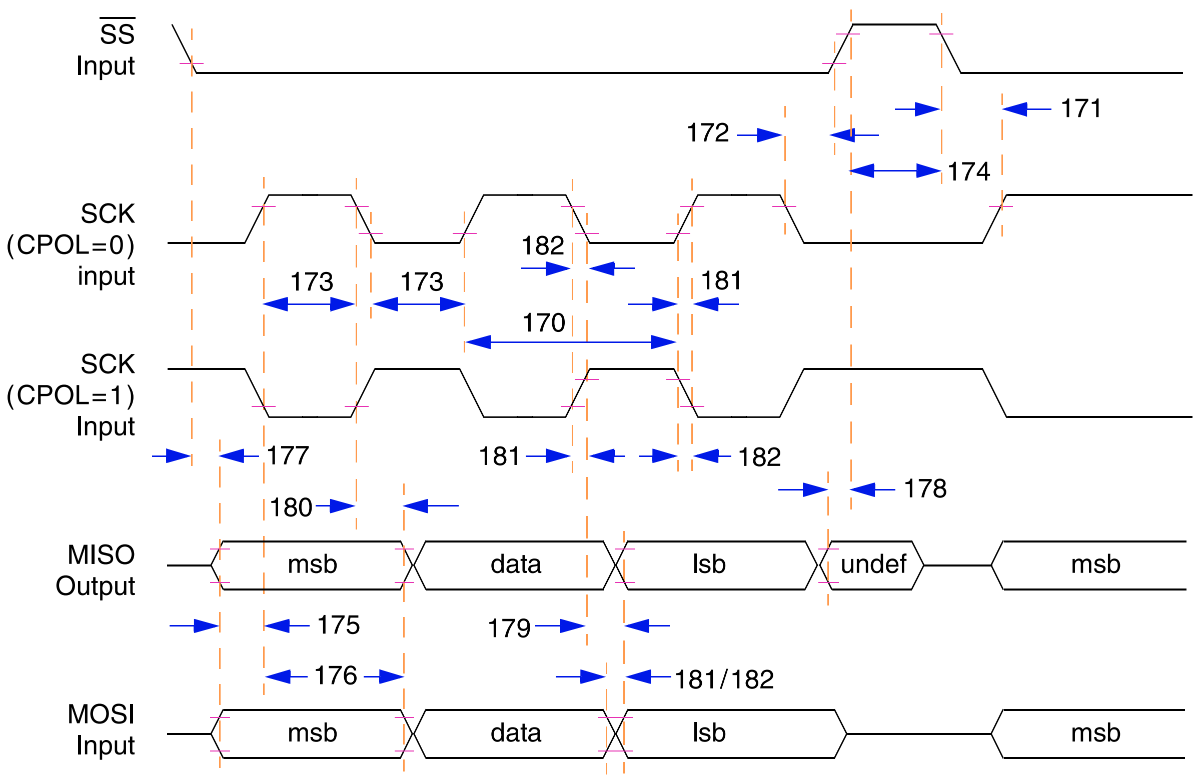


Figure A-33   Timing Diagram—SPI Slave (CPHA=0)

Table A-24   SPI Slave AC Timing Specifications

Sym	Description	All Frequencies Min	All Frequencies Max	Units
170	Slave cycle time	2	—	t_{cyc}
171	Slave enable lead time	15.00	—	nS
172	Slave enable lag time	15.00	—	nS
173	Slave clock (SCK) high or low time	1	—	t_{cyc}
174	Slave sequential transfer delay (does not require deselect)	1	—	t_{cyc}
175	Slave data setup time (inputs)	20.00	—	nS
176	Slave data hold time (inputs)	20.00	—	nS
177	Slave access time	—	50.00	nS
178	Slave SPI MISO disable time	—	50.00	nS
179	Slave data valid (after SCK edge)	—	50.00	nS
180	Slave data hold time (outputs)	0.00	—	nS
181	Rise time (input)	—	15.00	nS
182	Fall time (input)	—	15.00	nS

A.2.13 I^{2}C

Figure A-34 shows the I^{2}C Input/Output timing diagram. Table A-25 and Table A-26 gives the timing specifications.

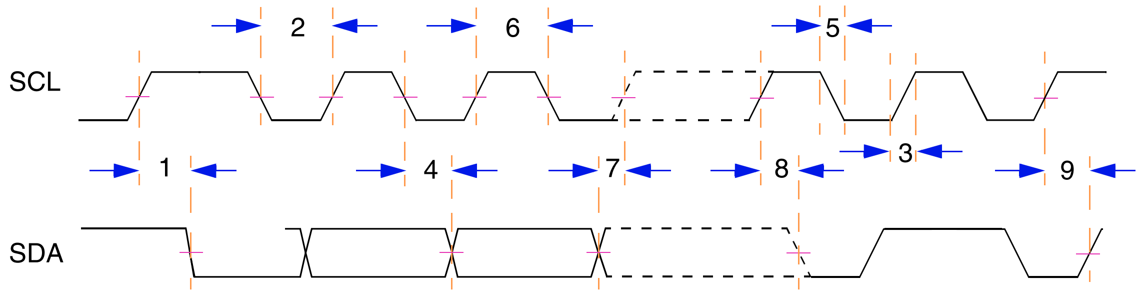


Figure A-34 Timing Diagram—I^{2}C Input/Output

Table A-25 I^{2}C Input Timing Specifications—SCL and SDA

Sym	Description	54MHz CLKIN		Units
		Min	Max	
1	Start condition hold time	2	—	Bus clocks
2	Clock low period	8	—	Bus clocks
3	SCL/SDA rise time ($V_{IL} = 0.5V$ to $V_{IH} = 2.4V$)	—	1	mS
4	Data hold time	0	—	mS
5	SCL/SDA fall time ($V_{IL} = 2.4V$ to $V_{IH} = 0.5V$)	—	1	mS
6	Clock high time	4	—	Bus clocks
7	Data setup time	0	—	nS
8	Start condition setup time (for repeated start condition only)	2	—	Bus clocks
9	Stop condition setup time	2	—	Bus clocks

Table A-26 I^{2}C Output Timing Specifications—SCL and SDA

Sym	Description	54MHz CLKIN		Units
		Min	Max	
1[1]	Start condition hold time	6	—	Bus clocks
2[1]	Clock low period	10	—	Bus clocks
3[2]	SCL/SDA rise time ($V_{IL} = 0.5V$ to $V_{IH} = 2.4V$)	—	—	—
4[1]	Data hold time	7	—	Bus clocks
5[3]	SCL/SDA fall time ($V_{IL} = 2.4V$ to $V_{IH} = 0.5V$)	—	3	nS
6[1]	Clock high time	10	—	Bus clocks
7[1]	Data setup time	2	—	Bus clocks

Table A-26 I²C Output Timing Specifications—SCL and SDA (continued)

Sym	Description	54 MHz CLKIN		Units
		Min	Max	
8[1]	Start condition setup time (for repeated start condition only)	20	—	Bus clocks
9[1]	Stop condition setup time	10	—	Bus clocks

NOTE:
1. Programming IFDR with the maximum frequency (IFDR=0x20) results in the minimum output timeings listed. The I2C interface is designed to scale the data transition time, moving it to the middle of the SCL low period. The actual position is affected by the prescale and division values programmed in IFDR.
2. Because SCL and SDA are open-collector-type outputs, which the processor can only actively drive low, the time SCL or SDA takes to reach a high level depends on external signal capacitance and pull-up resistor values.
3. Specified at a nominal 50pF load.

A.2.14 MSCAN

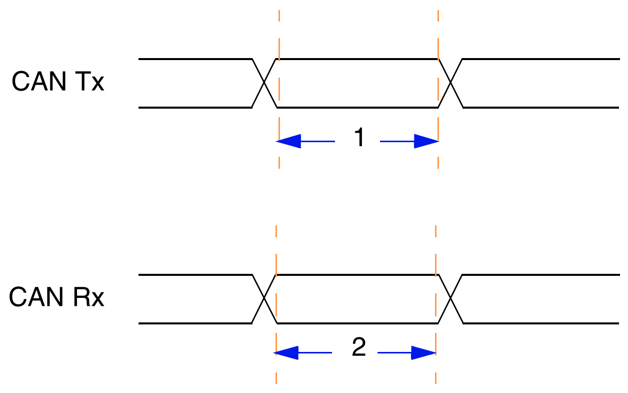


Figure A-35 Timing Diagram—CAN

Table A-27 CAN Timing Specifications

Sym	Description	All Frequencies		Units
		Min	Max	
1	CAN Tx Bit Time	1	—	ns
2	CAN Rx Bit Time	1	—	ns

A.2.15 PSC

Figure A-36 through Figure A-37 show the PSC Module AC Timing Diagrams. Table A-28 gives the timing specifications.

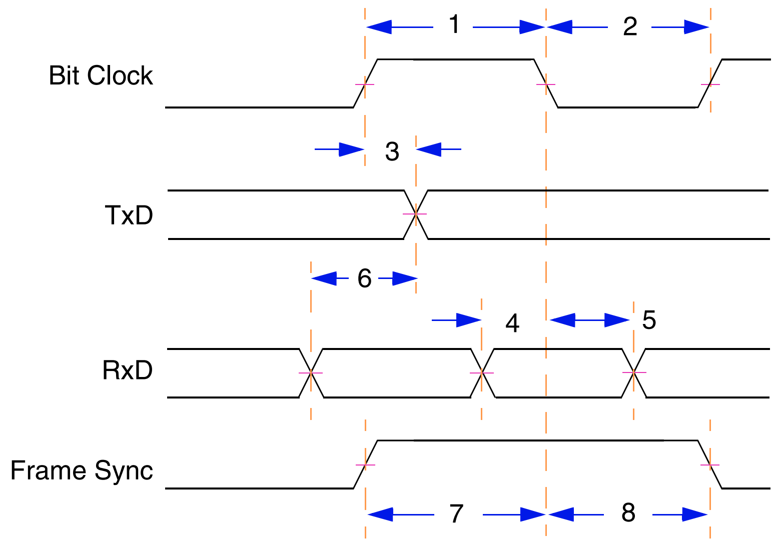


Figure A-36 Timing Diagram—8- and 16-bit CODEC Mode

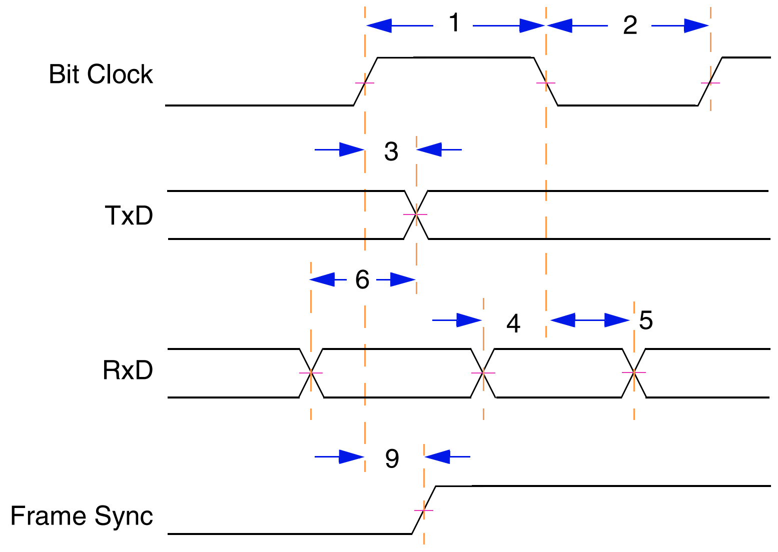


Figure A-37 Timing Diagram—AC97 Mode

Table A-28 PSC Module AC Timing Specifications

Sym	Description	54 MHz CLKIN		Units
		Min	Max	
1	Bit Clock high time	38	—	nS
2	Bit Clock low time	38	—	nS
3	Bit Clock rising to TxD valid	—	20	nS
4	RxD setup to Bit Clock falling	10	—	nS
5	RxD hold from Bit Clock falling	—	5	nS
6	RxD to TxD (remote loop back)	—	15	nS
7	Frame setup to Bit Clock falling	10	—	nS
8	Frame hold from Bit Clock falling	—	5	nS
9	Bit Clock rising to Frame asserted	—	20	nS

A.2.16 GPIOs and Timers

Figure A-38 show the GPIO Timing Diagram. Table A-28 gives the timing specifications.

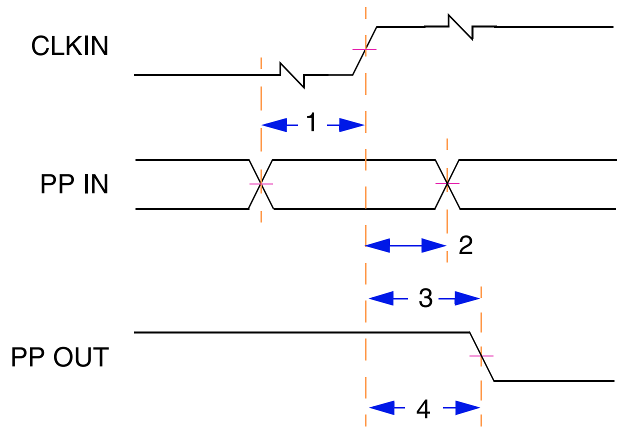


Figure A-38 Timing Diagram—General-Purpose I/O

Table A-29 General-Purpose I/O Timing Specifications

Sym	Description	54 MHz CLKIN		Units
		Min	Max	
1	PP valid to CLKIN (input setup)	7.5	—	nS
2	CLKIN to PP invalid (input hold)	1.0	—	nS
3	CLKIN to PP valid (output valid)	—	10	nS
4	CLKIN to PP invalid (output hold)	1.0	—	nS

A.3 DC Electrical Specifications

The tables in this section describe the MGT5100 DC Electrical characteristics. Table A-30 gives the absolute maximum ratings.

Table A-30 Absolute Maximum Ratings[1]

Characteristic	Symbol	Value	Unit
Core supply voltage [2]	VDD_CORE	–0.3 to 2.10	V
I/O supply voltage [2]	VDD_IO, VDD_MEM_IO	–0.3 to 3.6	V
System APLL supply voltage [2]	AVDD1	–0.3 to 2.1	V
Harpo APLL supply voltage [2]	AVDD2	–0.3 to 2.1	V
Input signal voltage (during Power-ON sequence) [2]	Vin	VDD_CORE/AVDDx + 2.0V; VDD_IO/VDD_MEM_IO + 0.4V	V
Input signal voltage overshoot [3]	Vinos	1.0	V
Input signal voltage undershoot [3]	Vinus	1.0	V
Storage temperature	Tstg	–55 to 150	^{o}C
Maximum operating ambient temperature	Tmax_op	75	^{o}C

NOTE:
1. Functional operating conditions are given in Table A-31. Absolute maximum ratings are stress ratings only, and functional operation at the maximums is not guaranteed. Stresses beyond those listed may affect device reliability or cause permanent device damage.
2. **Caution**—Vin must not exceed VDD_CORE/AVDDx by more than 2.0V at any time, including time during the Power-ON sequence, and must not exceed VDD_IO/VDD_MEM_IO by more than 0.4V during the Power-ON sequence. VDD_IO/VDD_MEM_IO must not exceed VDD_CORE/AVDDx by more than 2.0V at any time including time during the Power-ON sequence. VDD_CORE/AVDDx must not exceed VDD_IO/VDD_MEM_IO by more than 0.4 at any time including time during Power-ON reset.
3. **Caution**—After Power-ON, during normal operation, Vin must not exceed 1.0V above VDD_IO/VDD_MEM_IO or 1.0V below VSS_IO. Excursions above VDD_IO/VDD_MEM_IO or below VSS_IO must not exceed 20% of the reference clock frequency (SYS_XTAL_IN).

Table A-31 gives the recommended operating conditions.

Table A-31 Recommended Operating Conditions

Characteristic	Symbol	Value	Unit
Core supply voltage	VDD_CORE	1.8 ± 5%	V
I/O supply voltage (3.3V)	VDD_IO	3.3 ± 5%	V
I/O supply voltage (2.5V)	VDD_MEM_IO	2.5 +5% –3%	V
System APLL supply voltage[1]	AVDD1	1.8 ± 5%	V
Harpo APLL supply voltage[1]	AVDD2	1.8 ± 5%	V

Table A-31 Recommended Operating Conditions (continued)

Characteristic	Symbol	Value	Unit
Input signal voltage	Vin	VSS_IO to VDD_IO/ VDD_MEM_IO	V
Die junction temperature	Tj	0 to 105	^{o}C
NOTE: 1. These are recommended and tested operating conditions. Proper device operation outside these conditions is not guaranteed			

Table A-32 gives the DC Electrical characteristics for the MGT5100.

Table A-32 DC Electrical Specifications

Characteristic	Symbol	Min	Max	Unit
Input high voltage (3.3V I/Os)	V_{IH}	2.0	3.6	V
Input high voltage (2.5V I/Os)	V_{IH}	1.7	2.626	V
Input high voltage (SYS_XTAL_IN)	CV_{IH}	2.0	3.6	V
Input low voltage (3.3V I/Os)	V_{IL}	VSS_IO	0.8	V
Input low voltage (2.5V I/Os)	V_{IL}	VSS_IO	0.7	V
Input low voltage (SYS_XTAL_IN)	CV_{IL}	VSS_IO	0.8	V
Midpoint Reference Voltage (3.3V I/Os)	V_M	1.4		V
Midpoint Reference Voltage (2.5V I/Os)	V_M	VDD_MEM_IO/2		V
Input leakage current, $V_{IN} = V_{IN}$Max	I_{IN}	—	10[1]	uA
Hi-Z (off-state) leakage current, $V_{IN} = V_{IN}$Max	I_{TSI}	—	10[1]	uA
Output high voltage, $I_{OH} = -7$ mA (3.3V I/Os)	V_{OH}	2.4	—	V
Output high voltage, $I_{OH} = -5$ mA (2.5V I/Os)	V_{OH}	1.7	—	V
Output low voltage, $I_{OL} = 7$ mA (3.3V I/Os)	V_{OL}	—	0.4	V
Output low voltage, $I_{OL} = 5$ mA (2.5V I/Os)	V_{OL}	—	0.7	V
Capacitance, $V_{IN} = 0$V, f = 1 MHz	C_{in}	—	10.0[2]	pF
Capacitance, $V_{IN} = 0$V, f = 1 MHz (open drain)	C_{in}	—	15.0[2]	pF
NOTE: 1. The leakage is measured for nominal VDD_IO/VDD_MEM_IO and VDD_CORE. 2. Capacitance is periodically sampled rather than 100% tested.				

A.3.1 Electrostatic Discharge

— CAUTION —

This device contains circuitry that protects against damage due to high-static voltage or electrical fields. However, it is advised that normal pre-cautions be taken to avoid application of any voltages higher than max-imum-rated voltages. Operational reliability is enhanced if unused inputs are tied to an appropriate logic voltage level (i.e., either GND or V_{CC}). Table A-34 gives package thermal characteristics for this device.

Table A-33 Electrostatic Discharge

Sym	Parameter	Min	Typ	Unit
ESD	HBM (Human Body Model)—MIL-STD-883C method 3015-7	2000	—	V

A.3.2 Thermal Characteristics

Table A-34 Package Thermal Characteristics

Symbol	Characteristic	Description	Value	Units
$R_{\theta JA}$	Junction to Ambient[1,] Natural Convection	A 4-layer board (2s2p).	19	^{o}C/W
$R_{\theta JB}$	Junction to Board[3]		10	^{o}C/W
$R_{\theta JC}$	Junction to Case[4]		7	^{o}C/W
Y_{JT}	Junction to Package Top[5]	Natural Convection	2	^{o}C/W

NOTE:
1. Junction temperature is a function of on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and board thermal resistance.
2. Per JEDEC JESD51-6 with the board horizontal.
3. Thermal resistance between the die and the printed circuit board per JEDEC JESD51-8. Board temperature is measured on the top surface of the board near the package.
4. Indicates the average thermal resistance between the die and the case top surface as measured by the cold plate method (MIL SPEC-883 Method 1012.1) with the cold plate temperature used for the case temperature.
5. Thermal characterization parameter indicating the temperature difference between package top and the junction temperature per JEDEC JESD51-2. When Greek letters are not available, the thermal characterization parameter is written as Psi-JT.

A.3.3 Heat Dissipation

An estimation of the chip junction temperature, T_J, can be obtained from the following equation:

$$T_J = T_A + (R_{\theta JA} \times P_D)$$

where:

T_A = ambient temperature for the package (°C)

$R_{\theta JA}$ = junction to ambient thermal resistance (°C/W)

P_D = power dissipation in package (W)

The junction to ambient thermal resistance is an industry standard value which provides a quick and easy estimation of thermal performance. Unfortunately, there are two values in common usage: the value determined on a single layer board and the value obtained on a board with two planes. For packages such as the PBGA, these values can be different by a factor of two. Which value is correct depends on the power dissipated by other components on the board. The value obtained on a single layer board is appropriate for the tightly packed printed circuit board. The value obtained on the board with the internal planes is usually appropriate if the board has low power dissipation and the components are well separated.

Historically, the thermal resistance has frequently been expressed as the sum of a junction to case thermal resistance and a case to ambient thermal resistance:

$$R_{\theta JA} = R_{\theta JC} + R_{\theta CA}$$

where:

$R_{\theta JA}$ = junction to ambient thermal resistance (°C/W)

$R_{\theta JC}$ = junction to case thermal resistance (°C/W)

$R_{\theta CA}$ = case to ambient thermal resistance (°C/W)

$R_{\theta JC}$ is device related and cannot be influenced by the user. The user controls the thermal environment to change the case to ambient thermal resistance, $R_{\theta CA}$. For instance, the user can change the air flow around the device, add a heat sink, change the mounting arrangement on printed circuit board, or change the thermal dissipation on the printed circuit board surrounding the device. This description is most useful for ceramic packages with heat sinks where some 90% of the heat flow is through the case to the heat sink to ambient. For most packages, a better model is required.

A more accurate thermal model can be constructed from the junction to board thermal resistance and the junction to case thermal resistance[1-3]. The junction to case covers the situation where a heat sink will be used or where a substantial amount of heat is dissipated from the top of the package. The junction to board thermal resistance describes the thermal performance when most of the heat is conducted to the printed circuit board. This model can be used for either hand estimations or for a computational fluid dynamics (CFD) thermal model.

To determine the junction temperature of the device in the application after prototypes are available, the Thermal Characterization Parameter (Ψ_{JT}) can be used to determine the junction temperature with a measurement of the temperature at the top center of the package case using the following equation:

$$T_J = T_T + (\Psi_{JT} \times P_D)$$

where:

T_T = thermocouple temperature on top of package (°C)

Ψ_{JT} = thermal characterization parameter (°C/W)

P_D = power dissipation in package (W)

The thermal characterization parameter is measured per JESD51-2 specification using a 40 gauge type T thermocouple epoxied to the top center of the package case. The thermocouple should be positioned so that the thermocouple junction rests on the package. A small amount of epoxy is placed over the thermocouple junction and over about 1mm of wire extending from the junction. The thermocouple wire is placed flat against the package case to avoid measurement errors caused by cooling effects of the thermocouple wire.

A.3.4 Power Dissipation

Table A-35 gives power dissipation information.

Table A-35 Power Dissipation

Sym	Characteristic	Die Revision	Frequency	Typ	Max[1]	Unit
P_D	Power dissipation[2]	All	All	TBD	TBD	mW

NOTE:
1. Maximum power dissipation is measured at VDD_CORE = 1.98V and VDD_IO = 3.6V.
2. Typical power dissipation is measured at VDD_CORE = 1.8V and VDD_IO = 3.3V.

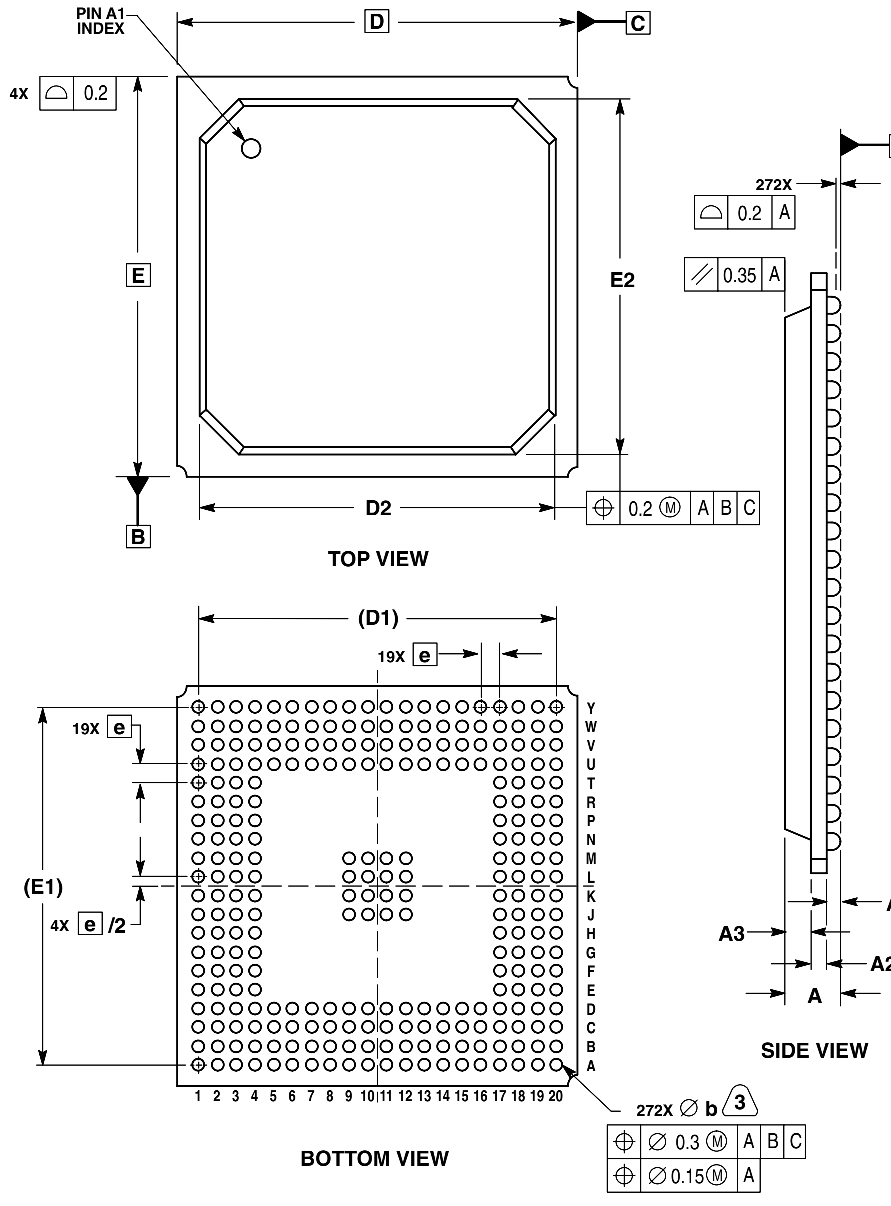
SECTION B
MECHANICAL SPECIFICATIONS

B.1 Overview

To Be Determined

B.2 Case Diagrams

B.2.1 272-Pin PBGA



NOTES:
1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M, 1994.
2. DIMENSIONS IN MILLIMETERS.
3. DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL TO PRIMARY DATUM A.
4. PRIMARY DATUM A AND THE SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.

	MILLIMETERS	
DIM	MIN	MAX
A	2.05	2.65
A1	0.50	0.70
A2	0.50	0.70
A3	1.05	1.25
b	0.60	0.90
D	27.00 BSC	
D1	24.13 REF	
D2	23.30	24.70
E	27.00 BSC	
E1	24.13 REF	
E2	23.30	24.70
e	1.27 BSC	

CASE 1135A–01
ISSUE B

Figure B-1 Case Diagram—272-Pin PBGA

SECTION C
ADDENDUM

C.1 Overview

To Be Determined

SECTION D TROUBLESHOOTING

D.1 Solutions to Known Problems

Table D-1 Solutions or Work-Arounds

Problem	Solution
I^2C E2PROM holds SDA line low. System is stuck in a lock-up mode and can't escape unless the board is repowered.	The problem is a faulty E2PROM. The only known solution is to use a non-faulty E2PROM. The MGT5100 I^2C block(s) are functioning correctly and according to the I^2C specification. The problem is related to the I^2C arbitration scheme. In this scheme the master can lose arbitration if SDA is held low. The circumstances where I^2C masters can lose arbitration due to SDA, are in #1 and #2 of the 5 arbitration lost possibilities shown below: 1. The master samples SDA low when driving a high during data and address cycles. 2. The master samples SDA low when driving a high during the acknowledge of a master-receiver cycle. 3. A start cycle is attempted when the bus is busy. 4. A repeated start cycle is requested in slave mode. 5. A stop condition is detected when the master did not request it. When the master loses arbitration, the hardware automatically and immediately switches to slave mode. If the master is programmed to start-up again, it immediately fails arbitration due to #3 above. All subsequent starts by the master will result in arbitration lost until the board is powered off and back on.
With regards to the MGT5100 Microprocessor Memory Management Unit (MMU)—what impact does address translation have on command execution time—assuming the necessary page is in the Translation Look-aside Buffer (TLB)?	There are 2 ways to answer this question concerning how the MMU affects program execution time. 1. Some overhead is involved in maintaining the TLBs when task switching occurs. This is a system design issue, which depends on issues such as: • How many tasks will be running? • How often do task switches occur? • How much TLB data must be changed each time a task switch occurs? etc. These are questions only the user can answer. 2. As an example, let us say the TLB is programmed only once (which is probably unrealistic, but works for this example). If there is no overhead associated with maintaining the TLBs, then the MMU does not affect machine performance. As long as the software does not need to change TLB data, software performance is not affected by whether the MMU is turned ON or OFF.

Table D-1 Solutions or Work-Arounds (continued)

Problem	Solution
In UDMA2 mode, the READ_DMA command sometimes fails when reading a large number of sectors, say 256. The probability of the READ_DMA command failure is about 30%. Although the ATA driver has the ability to retry the failed command, retry takes place only after timeout (3s). Therefore, performance in the UDMA2 mode will be very poor.	UDMA2 mode requires IP bus clock (ipb_clk) frequency be at least 66MHz frequency. If the IP bus clock is only 33MHz, the problem occurs.

SECTION E
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 execut-

	ing 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. **NOTE:** 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^{2}C, 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

00000000 00000000 00000100 00000001		
Addr	**Big-Endian**	**Little-Endian**
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 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 Capture
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
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 immediate index addressing	A form of addressing that specifies an immediate value to be added to the contents of a specified GPR to form the target address for the load or store.
Register indirect with index addressing	A form of addressing that specifies that the contents of two 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
SCPCI	PCI_SC (PCI SmartComm)
SCR	Sustained Cell Rate
SCTMR	SmartComm Timer
SDLC	Synchronous Data Link Control
SDMA, SCDMA	Smart Direct Memory Access
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 83MHz RAM would be equivalent to 12ns.
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 MGT5100 and peripheral devices. It supports master and slave mode, double-buffered operation and can operate in a polling or interrupt driven environment.
SPLL	System Phase-Locked Loop
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-Rate 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 12Mbps.
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.

SECTION F
LIST OF REGISTERS

3.3.1 Memory Map Registers—MBAR + 0x0000 3-4
3.3.1.1 Module Base Address (0x0000)—MBAR 3-4
3.3.1.2 CS0 Start Address (0004–002C)—CS0STR 3-5
3.3.1.3 CS0 Stop Address (0008)—CS0STP 3-5
3.3.1.4 CS1 Start Address (000C)—CS1STR 3-5
3.3.1.5 CS1 Stop Address (0010)—CS1STP 3-6
3.3.1.6 CS2 Start Address (0014)—CS2STR 3-6
3.3.1.7 CS2 Stop Address (0018)—CS2STP 3-7
3.3.1.8 CS3 Start Address (001C)—CS3STR 3-7
3.3.1.9 CS3 Stop Address (0020)—CS3STP 3-7
3.3.1.10 CS4 Start Address (0024)—CS4STR 3-8
3.3.1.11 CS4 Stop Address (0028)—CS4STP 3-8
3.3.1.12 CS5 Start Address (002C)—CS5STR 3-9
3.3.1.13 CS5 Stop Address (0030)—CS5STP 3-9
3.3.1.14 SDRAM Start Address (0034)—SDRAMSTR 3-9
3.3.1.15 SDRAM Stop Address (0038)—SDRAMSTP 3-10
3.3.1.16 PCI1 Start Address (003C)—PCI1STR 3-10
3.3.1.17 PCI1 Stop Address (0040)—PCI1STP 3-11
3.3.1.18 PCI2 Start Address (0044)—PCI2STR 3-11
3.3.1.19 PCI2 Stop Address (0048)—PCI2STP 3-11
3.3.1.20 Boot Start Address (004C)—BOOTSTR 3-12
3.3.1.21 Boot Stop Address (0050)—BOOTSTP 3-12
3.3.1.22 Address Space Enable (0054)—ADREN 3-13

3.4 Register Summaries 3-14
3.4.1 PCI XLB Configuration Registers 3-14
3.4.2 SmartComm DMA Registers 3-16
3.4.3 PSC Registers—Quick Reference 3-19
3.4.4 IrDA Registers—Quick Reference 3-23
3.4.5 SPI Registers—Quick Reference 3-26
3.5 SDRAM Memory Controller Register Map (MBAR + 0x0100) 3-27
3.6 Slice Timers Register Map (MBAR + 0x0700) 3-27

5.6 CDM Registers—MBAR + 0x0200 5-14
5.6.1 JTAG ID Number (0200)—JTAGID 5-14
5.6.2 Power ON Reset Configuration (0204)—PORCFG 5-15
5.6.3 Bread Crumb (0208)—BC 5-15
5.6.4 Configuration (020C)—CFG 5-16
5.6.5 48 MHz Fractional Divider Configuration (0210)—FDCFG 5-17
5.6.6 Clock Enable (0214)—CLKEN 5-18
5.6.7 System Oscillator Configuration (0218)—OSCCFG 5-20
5.6.8 Clock Control Sequencer Configuration (021C)—CCSCFG 5-20
5.6.9 Soft Reset (0220)—SFTRST 5-21
5.6.10 System PLL Status (0224)—PLLSTA (33-27 MHz) 5-21

6.3 Arbiter Registers—MBAR + 0x0080 6-2
6.3.1 Configuration Register (00C0)—ACFG 6-3
6.3.2 Version Register (00C4)—VER 6-4

6.3.3 Status Register (00C8)—STA 6-4
6.3.4 Interrupt Enable Register (00CC)—INTEN 6-5
6.3.5 Address Capture Register (00D0)—ADRCAP 6-6
6.3.6 Bus Signal Capture Register (00D4)—SIGCAP 6-6
6.3.7 Address Tenure Time-Out Register (00D8)—ADRTO 6-7
6.3.8 Data Tenure Time-Out Register (00DC)—DATTO 6-8
6.3.9 Bus Activity Time-Out Register (00E0)—BUSTO 6-8
6.3.10 Master Priority Enable Register (00E4)—PRIEN 6-9
6.3.11 Master Priority Register (00E8)—PRI 6-10
6.3.12 Base Address Register (00EC)—BAR 6-11
6.3.13 Reserved Registers (0080–00BF, 00EC, 00F0–00FC) 6-12
7.2.4 Interrupt Controller Registers—MBAR + 0x0500 7-6
7.2.4.1 Peripheral Interrupt Mask (0500)—Register 0 7-6
7.2.4.2 Peripheral Priority and HI / LO Select 1 (0504)—Register 1 7-7
7.2.4.3 Peripheral Priority and HI / LO Select 2 (0508)—Register 2 7-8
7.2.4.4 Peripheral Priority and HI / LO Select 3 (050C)—Register 3 7-9
7.2.4.5 External Enable and External Types (0510)—Register 4 7-9
7.2.4.6 Critical Priority and Main Interrupt Mask (0514)—Register 5 7-11
7.2.4.7 Main Interrupt Priority and INT / SMI Select 1 (0518)—Register 6 7-12
7.2.4.8 Main Interrupt Priority and INT / SMI Select 2 (051C)—Register 7 7-13
7.2.4.9 PerStat, MainStat, CritStat Encoded (0524)—Register 9 7-14
7.2.4.10 Critical Interrupt Status All (0528)—Register A 7-16
7.2.4.11 Main Interrupt Status All (052C)—Register B 7-16
7.2.4.12 Peripheral Interrupt Status All (0530)—Register C 7-17
7.2.4.13 Peripheral Interrupt Status All (0538)—Register E 7-18
7.3.2.1 GPIO Standard Registers—MBAR + 0x0B00 7-26
7.3.2.1.1 Port Configuration Register (0B00)—GPIOPCR 7-28
7.3.2.1.2 Simple GPIO Enables (0B04)—GPIOSEN 7-30
7.3.2.1.3 Simple GPIO Open Drain Type (0B08)—GPIOSOD 7-31
7.3.2.1.4 Simple GPIO Data Direction (0B0C)—GPIOSDD 7-32
7.3.2.1.5 Simple GPIO Data Output Values (0B10)—GPIOSDO 7-34
7.3.2.1.6 Simple GPIO Data Input Values (0B14)—GPIOSDI 7-35
7.3.2.1.7 Output-Only Enables (0B18)—GPIOOE 7-36
7.3.2.1.8 Output-Only Data Value Out (0B1C)—GPIOODO 7-37
7.3.2.1.9 Simple Interrupt Enables (0B20)—GPIOSIE 7-38
7.3.2.1.10 Simple Interrupt Open-Drain Emulation (0B24)—GPIOSIOD 7-38
7.3.2.1.11 Simple Interrupt Data Direction (0B28)—GPIOSIDD 7-39
7.3.2.1.12 Simple Interrupt Data Value Out (0B2C)—GPIOSIDO 7-40
7.3.2.1.13 Simple Interrupt Interrupt Enable (0B30)—GPIOSIIE 7-40
7.3.2.1.14 Simple Interrupt Interrupt Types (0B34)—GPIOSIIT 7-41
7.3.2.1.15 Simple Interrupt Master Enable (0B38)—GPIOSIME 7-42
7.3.2.1.16 Simple Interrupt Status (0B3C)—GPIOSIST 7-42
7.3.2.2 WakeUp GPIO Registers—MBAR + 0x0C00 7-44
7.3.2.2.1 WakeUp GPIO Enables (0C00)—GPIOWE 7-44
7.3.2.2.2 WakeUp GPIO Open Drain Emulation (0C04)—GPIOWOD 7-45
7.3.2.2.3 WakeUp GPIO Data Direction (0C08)—GPIOWDD 7-46
7.3.2.2.4 WakeUp GPIO Data Value Out (0C0C)—GPIOWDO 7-46
7.3.2.2.5 WakeUp GPIO Interrupt Enable (0C10)—GPIOWUE 7-47
7.3.2.2.6 WakeUp GPIO Simple Interrupt Enable (0C14)—GPIOWSIE 7-48

7.3.2.2.7 WakeUp GPIO Interrupt Types (0C18)—GPIOWT 7-49
7.3.2.2.8 WakeUp GPIO Master Enables (0C1C)—GPIOWME 7-50
7.3.2.2.9 WakeUp GPIO Data Input Values (0C20)—GPIOWI 7-50
7.3.2.2.10 WakeUp GPIO Status Register (0C24)—GPIOWS 7-51

7.4.4 GPT Registers—MBAR + 0x0600 7-54
7.4.4.1 GPT[0–7] Enable and Mode Select (0600–0670)—Register 0 7-54
7.4.4.2 GPT[0–7] Counter Input (0604–0674)—Register 1 7-57
7.4.4.3 GPT[0–7] PWM Configuration (0608–0678)—Register 2 7-58
7.4.4.4 GPT[0–7] Status (060C–067C)—Register 3 7-59

7.5.1 SLT Registers—MBAR + 0x0700 7-61
7.5.1.1 SLT[0, 1] Terminal Count (0700, 0710)—Register 0 7-61
7.5.1.2 SLT[0, 1] Control (0704, 0714)—Register 1 7-62
7.5.1.3 SLT[0, 1] Count (0708, 0718)—Register 2 7-62
7.5.1.4 SLT[0, 1] Count (0708, 0718)—Register 2 7-63

7.6.3 RTC Interface Registers—MBAR + 0x0800 7-65
7.6.3.1 Time Set (0800)—Reg 0 7-66
7.6.3.2 Date Set (0804)—Reg 1 7-67
7.6.3.3 New Year and Stopwatch (0808)—Reg 2 7-68
7.6.3.4 Alarm and Interrupt Enable (080C)—Reg 3 7-68
7.6.3.5 Current Time (0810)—Reg 4 7-69
7.6.3.6 Current Date (0814)—Reg 5 7-70
7.6.3.7 Alarm and Stopwatch Interrupt (0818)—Reg 6 7-71
7.6.3.8 Periodic Interrupt and Bus Error (081C)—Reg 7 7-72

8.6 SDRAM Memory Controller Registers (MBAR + 0x0100) 8-19
8.6.1 Mode Register (MOD) 8-19
8.6.2 Control Register (CTR) 8-21
8.6.3 Configuration Register 1 (CFG1) 8-23
8.6.4 Configuration Register 2 8-25
8.6.5 Address Select Register (ADRSEL) 8-26

9.4.2 Chip Select / LocalPlus Bus Registers—MBAR + 0x0300 9-13
9.4.2.1 CS Boot ROM, (0300)—Register 0 9-13
9.4.2.2 CS Configuration (0304–0314)—Registers 1–5 9-16
9.4.2.3 CS Control (0318)—Register 6 9-18
9.4.2.4 CS Status (031C)—Register 7 9-19

10.3.3.1 PCI XLB Configuration Registers—MBAR + 0x0D00 10-6
10.3.3.1.1 PCI Header: Device ID/Vendor ID (0D00)—Register 0 10-6
10.3.3.1.2 PCI Header: Status/Command (0D04)—Register 1 10-7
10.3.3.1.3 PCI Header: Class Code/Revision (0D08)—Register 2 10-9
10.3.3.1.4 PCI Header: BIST/Type/Latency/Cache (0D0D)—Register 3 10-10
10.3.3.1.5 PCI Header: BAR0 (0D10)—Register 4 10-11
10.3.3.1.6 PCI Header: BAR1 (0D14)—Register 5 10-11
10.3.3.1.7 PCI Header: Reserved (0D18–0D27)—Registers 6–9 10-12
10.3.3.1.8 PCI Header: CardBus CIS Pointer (0D28)—Register 10 10-12
10.3.3.1.9 PCI Header: SubSystem Vendor ID (0D2C)—Register 11 10-12
10.3.3.1.10 PCI Header: Unused (0D30)—Register 12 10-12
10.3.3.1.11 PCI Header: Unused (0D34–0D3B)—Registers 13–14 10-12
10.3.3.1.12 PCI Header: Max Lat, Min Grant, Interrupt (0D3C)—Register 15 10-13
10.3.3.1.13 PCI Header: Unused (0D40–0D5F)—Registers 16–23 10-13

10.3.3.2 MPC4200 Application Interface—MBAR + 0x0D00 10-13
10.3.3.2.1 PCI Interrupt Enable (0D60)—Custom Register 24 10-14
10.3.3.2.2 PCI Status (0D64)—Custom Register 25 10-14
10.3.3.2.3 PCI Control (0D68)—Custom Register 26 10-15
10.3.3.2.4 PCI Mask/Value Read (0D6C)—Custom Register 27 10-16
10.3.3.2.5 PCI Mask/Value Write (0D70)—Custom Register 28 10-17
10.3.3.2.6 PCI Subwindow 1 (0D74)—Custom Register 29 10-18
10.3.3.2.7 PCI Subwindow 2 (0D78)—Custom Register 30 10-18
10.3.3.2.8 PCI Window Command/Control (0D7C)—Custom Register 31 10-19
10.3.4.2 PCI Transmit (Tx) Registers—MBAR + 0x3800 10-21
10.3.4.2.1 PCI Tx Packet Size (3800)—Register 0 10-21
10.3.4.2.2 PCI Tx Start Address (3804)—Register 1 10-22
10.3.4.2.3 PCI Tx Transaction Control (3808)—Register 2 10-22
10.3.4.2.4 PCI Tx Enables (380C)—Register 3 10-23
10.3.4.2.5 PCI Tx Next Address (3810)—Register 4 10-25
10.3.4.2.6 PCI Tx Last Word (3814)—Register 5 10-25
10.3.4.2.7 PCI Tx Done Counts (3818)—Register 6 10-26
10.3.4.2.8 PCI Tx Status Bits (381C)—Register 7 10-26
10.3.4.2.9 PCI Tx FIFO Data (3840)—Register 10 10-28
10.3.4.2.10 PCI Tx FIFO Status (3844)—Register 11 10-28
10.3.4.2.11 PCI Tx FIFO Control (3848)—Register 12 10-29
10.3.4.2.12 PCI Tx Alarm (384E)—Register 13 10-30
10.3.4.2.13 PCI Tx Read Pointer (3852)—Register 14 10-30
10.3.4.2.14 PCI Tx Write Pointer (3856)—Register 15 10-31
10.3.4.4 PCI Receive Registers—MBAR + 0x3880 10-32
10.3.4.4.1 PCI Rx Packet Size (3880)—Register 0 10-32
10.3.4.4.2 PCI Rx Start Address (3884)—Register 1 10-33
10.3.4.4.3 PCI Rx Transaction Command (3888)—Register 2 10-33
10.3.4.4.4 PCI Rx Enables (388C)—Register 3 10-34
10.3.4.4.5 PCI Rx Next Address (3890)—Register 4 10-35
10.3.4.4.6 PCI Rx Done Counts (3898)—Register 6 10-36
10.3.4.4.7 PCI Rx Status Bits (389C)—Register 7 10-36
10.3.4.4.8 PCI Rx FIFO Data (38C0)—Register 16 10-38
10.3.4.4.9 PCI Rx FIFO Status (38C4)—Register 17 10-38
10.3.4.4.10 PCI Rx FIFO Control (38C8)—Register 18 10-39
10.3.4.4.11 PCI Rx Alarm (38CC)—Register 19 10-40
10.3.4.4.12 PCI Rx Read Pointer (38D0)—Register 20 10-40
10.3.4.4.13 PCI Rx Write Pointer (38D4)—Register 21 10-41
11.3.1 ATA Host Registers—MBAR + 0x3A00 11-3
11.3.1.1 Host Configuration (3A00)—HCFG 11-3
11.3.1.2 Host Status (3A04)—HSR 11-4
11.3.1.3 PIO Timing 1 (3A08)—PIO1 11-4
11.3.1.4 PIO Timing 2 (3A0C)—PIO2 11-5
11.3.1.5 Multiword DMA Timing 1 (3A10)—DMA1 11-5
11.3.1.6 Multiword DMA Timing 2 (3A14)—DMA2 11-6
11.3.1.7 Ultra DMA Timing 1 (3A18)—UDMA1 11-6
11.3.1.8 Ultra DMA Timing 2 (3A1C)—UDMA2 11-7
11.3.1.9 Ultra DMA Timing 3 (3A20)—UDMA3 11-8
11.3.1.10 Ultra DMA Timing 4 (3A24)—UDMA4 11-8

11.3.1.11 Ultra DMA Timing 5 (3A28)—UDMA5 11-9
11.3.2 ATA FIFO Registers—MBAR + 0x3A00 11-9
11.3.2.1 Rx / Tx FIFO Data Word (3A3C)—RTFDWR 11-10
11.3.2.2 Rx / Tx FIFO Status (3A40)—RTFSR 11-10
11.3.2.3 Rx / Tx FIFO Control (3A44)—RTFCR 11-11
11.3.2.4 Rx / Tx Alarm (3A48)—RTFAR 11-12
11.3.2.5 Rx / Tx FIFO Read Pointer (3A4C)—RTFRPR 11-12
11.3.2.6 Rx / Tx FIFO Write Pointer (3A50)—RTFWPR 11-13
11.3.3 ATA Drive Registers—MBAR + 0x3A00 11-13
11.3.3.1 Device Control (3A5C)—DCTR 11-14
11.3.3.2 Drive Alternate Status (3A5C)—DASR 11-14
11.3.3.3 Drive Data (3A60)—DDR 11-15
11.3.3.4 Drive Features (3A64)—DFR 11-15
11.3.3.5 Drive Error (3A64)—DER 11-16
11.3.3.6 Drive Sector Count (3A68)—DSCR 11-16
11.3.3.7 Drive Sector Number (3A6C)—DSNR 11-17
11.3.3.8 Drive Cylinder Low (3A70)—DCLR 11-17
11.3.3.9 Drive Cylinder High (3A74)—DCHR 11-18
11.3.3.10 Drive Device / Head (3A78)—DDHR 11-18
11.3.3.11 Drive Command (3A7C)—DCR 11-19
11.3.3.12 Device Status (3A80)—DSR 11-20
11.7.3.1 ATA Register Addressing 11-30
12.4 Host Control (HC) Operational Registers 12-6
12.4.1 Control and Status Partition—MBAR + 0x1000 12-7
12.4.1.1 HC Revision (1000)—HcRevision 12-7
12.4.1.2 HC Control (1004)—HcControl 12-7
12.4.1.3 HC Command Status (1008)—HcCommandStatus 12-9
12.4.1.4 HC Interrupt Status (100C)—HcInterruptStatus 12-11
12.4.1.5 HC Interrupt Enable (1010)—HcInterruptEnable 12-12
12.4.1.6 HC Interrupt Disable (1014)—HcInterruptDisable 12-13
12.4.2 Memory Pointer Partition—MBAR + 0x1000 12-14
12.4.2.1 HC Communication (1018)—HcHCCA 12-14
12.4.2.2 HC Period Current ED (101C)—HcPeriodCurrentED 12-15
12.4.2.3 HC Control Head ED (1020)—HcControlHeadED 12-16
12.4.2.4 HC Control Current ED (1024) HcControlCurrentED 12-16
12.4.2.5 HC First Bulk Head List ED (1028)—HcBulkHeadED 12-17
12.4.2.6 HC Bulk List Current ED (102C)—HcBulkCurrentED 12-17
12.4.2.7 HC Last Completed Transfer (1030)—HcDoneHead 12-18
12.4.3 Frame Counter Partition—MBAR + 0x1000 12-19
12.4.3.1 HC Frame Interval (1034)—HcFmInterval 12-19
12.4.3.2 HC Frame Bit-time Remaining (1038)—HcFmRemaining 12-20
12.4.3.3 HC Timing Reference (103C)—HcFmNumber 12-20
12.4.3.4 HC Start Processing Periodic List (1040)—HcPeriodicStart 12-21
12.4.3.5 HC Commit Transfer (1044)—HcLSThreshold 12-22
12.4.4 Root Hub Partition—MBAR + 0x1000 12-22
12.4.4.1 HC Root Hub Descriptor A (1048)—HcRhDescriptorA 12-23
12.4.4.2 HC Root Hub Descriptor B (104C)—HcRhDescriptorB 12-24
12.4.4.3 HC Root Hub Status (1050)—HcRhStatus 12-25

12.4.4.4 HC RH Port Status (1054)—Hc Rh Port Status [1 : NDP] 12-26
13.3 SmartComm DMA Registers—MBAR + 0x1200 13-3
13.3.1 Task Bar (1200)—taskBar 13-4
13.3.2 Current Pointer (1204)—currentPointer 13-4
13.3.3 End Pointer (1208)—endPointer 13-4
13.3.4 Variable Pointer (120C)—variablePointer 13-5
13.3.5 Interrupt Vector, PTD Control (1210)—IntVect1/2, PtdCntrl 13-5
13.3.6 Interrupt Pending (1214)—IntPend 13-6
13.3.7 Interrupt Mask (1218)—IntMask 13-6
13.3.8 Task Control 0, 1 (121C)—TCR0, TCR1 13-7
13.3.9 Task Control 2, 3 (1220)—TCR2, TCR3 13-7
13.3.10 Task Control 4, 5 (1224)—TCR4, TCR5 13-8
13.3.11 Task Control 6, 7 (1228)—TCR6, TCR7 13-8
13.3.12 Task Control 8, 9 (122C)—TCR8, TCR9 13-8
13.3.13 Task Control A, B (1230)—TCRA, TCRB 13-9
13.3.14 Task Control C, D (1234)—TCRC, TCRD 13-9
13.3.15 Task Control E, F (1238)—TCRE, TCRF 13-10
13.3.16 Initiator Priority 0–3 (123C)—IPR0–3 13-10
13.3.17 Initiator Priority 4–7 (1240)—IPR4–7 13-10
13.3.18 Initiator Priority 8–11 (1244)—IPR8–11 13-11
13.3.19 Initiator Priority 12–15 (1248)—IPR12–15 13-11
13.3.20 Initiator Priority 16–19 (124C)—IPR16–19 13-12
13.3.21 Initiator Priority 20–23 (1250)—IPR20–23 13-12
13.3.22 Initiator Priority 24–27 (1254)—IPR24–27 13-13
13.3.23 Initiator Priority 28–31 (1258)—IPR28–31 13-13
13.3.24 Reserved Register 1 (125C)—res 1 13-14
13.3.25 Reserved Register 2 (1260)—res 2 13-14
13.3.26 Reserved Register 3 (1264)—res 3 13-14
13.3.27 Reserved Register 4 (1268)—res 4 13-15
13.3.28 Reserved Register 5 (126C)—res 5 13-15
13.3.29 Debug Module Comparator 1 (1270)—Value 1 13-16
13.3.30 Debug Module Comparator 2 (1274)—Value 2 13-16
13.3.31 Debug Modulator Control (1278)—Control 13-16
13.3.32 Debug Module Status (127C)—Status 13-17
13.5 SmartComm Timer Registers (SCTMR)—MBAR + 0x0400 13-17
14.3.1.2.1 MII Management Register Set 14-8
14.4 FEC Memory Map and Registers 14-8
14.5 FEC Registers—MBAR + 3000 14-12
14.5.1 FEC ID (3000)—FEC_ID 14-13
14.5.2 Interrupt Event (3004)—IEVENT 14-14
14.5.3 Interrupt Enable (3008)—IMASK 14-16
14.5.4 Ethernet Control (3024)—ECNTRL 14-17
14.5.5 MII Management Frame (3040)—MII_DATA 14-18
14.5.6 MII Speed Control (3044)—MII_SPEED 14-20
14.5.7 MIB Control (3064)—MIB_CONTROL 14-21
14.5.8 Receive Control (3084)—R_CNTRL 14-22
14.5.9 Hash (3088)—R_HASH 14-23
14.5.10 Rx Destination Address Low (309C)—R_DA_LOW 14-24

14.5.11 Rx Destination Address High (30A0)—R_DA_HIGH 14-24
14.5.12 Tx Control (30C4)—X_CNTRL 14-25
14.5.13 Tx Status (30D0)—XMIT.X_STATUS 14-26
14.5.14 Physical Address Low (30E4)—PADDR1 14-27
14.5.15 Physical Address High (30E8)—PADDR2 14-27
14.5.16 Opcode / Pause Duration (30EC)—OP_PAUSE 14-28
14.5.17 Descriptor Individual Address 1 (3118)—IADDR1 14-28
14.5.18 Descriptor Individual Address 2 (311C)—IADDR2 14-29
14.5.19 Descriptor Group Address 1 (3120)—GADDR1 14-30
14.5.20 Descriptor Group Address 2 (3124)—GADDR2 14-30
14.5.21 Tx FIFO Watermark (3144)—X_WMRK 14-31
14.5.22 Tx FIFO Watermark (3144)—X_WMRK 14-32

15.2 PSC Registers—MBAR + 0x2000, 0x2400, 0x2800 15-3

15.2.1 Mode Register 1 (2x00)—MR1_[1, 2, 3] 15-5
15.2.2 Mode Register 2 (2x00)—MR2_[1, 2, 3] 15-6
15.2.3 Status Register (2x04)—SR[1, 2, 3] 15-8
15.2.4 Modem Mode (2x04)—SR[1, 2, 3] 15-9
15.2.5 Clock-Select Register (2x04)—CSR[1, 2, 3] 15-10
15.2.6 Command Register (2x08)—CR[1, 2, 3] 15-11
15.2.7 Rx Buffer Registers (2x0C)—RB[1, 2, 3] 15-14
15.2.8 Tx Buffer Registers (2x0C)—TB[1, 2, 3] 15-15
15.2.9 Input Port Change Register (2x10)—IPCR[1, 2, 3] 15-16
15.2.10 Auxiliary Control Register (2x10)—ACR[1, 2, 3] 15-17
15.2.11 Interrupt Status Register (2x14)—ISR[1, 2, 3] 15-17
15.2.12 Interrupt Mask Register (2x14)—IMR[1, 2, 3] 15-18
15.2.13 Counter Timer Upper Register (2x18)—CTUR[1, 2, 3] 15-19
15.2.14 Counter Timer Lower Register (2x1C)—CTLR[1, 2, 3] 15-20
15.2.15 Interrupt Vector Register (2x30)—IVR[1, 2, 3] 15-20
15.2.16 Input Port (2x34)—IP[1, 2, 3] 15-20
15.2.17 Output Port 1 Bit Set (2x38)—OP1_[1, 2, 3] 15-21
15.2.18 Output Port 0 Bit Reset (2x3C)—OP0_[1, 2, 3] 15-22
15.2.19 SCC / IrDA Control Register (2x40)—SICR[1, 2, 3] 15-23
15.2.20 Rx FIFO Number of Data (2x58)—RFNUM[1, 2, 3] 15-24
15.2.21 Tx FIFO Number of Data (2x5C)—TFNUM[1, 2, 3] 15-24
15.2.22 Rx FIFO Data (2x60)—RFDATA[1, 2, 3] 15-25
15.2.23 Rx FIFO Status (2x64)—RFSTAT[1, 2, 3] 15-25
15.2.24 Rx FIFO Control (2x68)—RFCNTL[1, 2, 3] 15-25
15.2.25 Rx FIFO Alarm (2x6E)—RFALARM[1, 2, 3] 15-26
15.2.26 Rx FIFO Read Pointer (2x72)—RFRPTR[1, 2, 3] 15-28
15.2.27 Rx FIFO Write Pointer (2x76)—RFWPTR[1, 2, 3] 15-28
15.2.28 Rx FIFO Last Read Frame PTR (2x7A)—RFLRFPTR[1, 2, 3] 15-29
15.2.29 Rx FIFO Last Write Frame PTR (2x7E)—RFLWFPTR[1, 2, 3] 15-29
15.2.30 Tx FIFO Data (2x80)—TFDATA[1, 2, 3] 15-29
15.2.31 Tx FIFO Status (2x84)—TFSTAT[1, 2, 3] 15-29
15.2.32 Tx FIFO Control (2x88)—TFCNTL[1, 2, 3] 15-30
15.2.33 Tx FIFO Alarm (2x8E)—TFALARM[1, 2, 3] 15-31
15.2.34 Tx FIFO Read Pointer (2x92)—TFRPTR[1, 2, 3] 15-31
15.2.35 Tx FIFO Write Pointer (2x96)—TFWPTR[1, 2, 3] 15-31
15.2.36 Tx FIFO Last Read Frame PTR (2x9A)—TFLRFPTR[1, 2, 3] 15-32

15.2.37 Tx FIFO Last Write Frame PTR (2x9E)—TFLWFPTR[1, 2, 3] 15-32
16.2 IrDA Registers—MBAR + 0x2C00 16-1
16.2.1 Mode Register 1 (2C00)—MR1 16-2
16.2.2 Mode Register 2 (2C00)—MR2 16-3
16.2.3 SIR Status Register (2C04)—SR 16-4
16.2.4 MIR / FIR Status Register (2C04)—SR 16-5
16.2.5 Clock-Select Register (2C04)—CSR 16-6
16.2.6 Command Register (2C08)—CR 16-7
16.2.7 Rx Buffers (2C0C)—RB 16-9
16.2.8 Tx Buffers (2C0C)—TB 16-9
16.2.9 Input Port Change (2C10)—IPCR 16-10
16.2.10 Auxiliary Control (2C10)—ACR 16-10
16.2.11 Interrupt Status (2C14)—ISR 16-11
16.2.12 Interrupt Mask (2C14)—IMR 16-12
16.2.13 Counter Timer Upper Bytes (2C18)—CTUR 16-13
16.2.14 Counter Timer Lower Bytes (2C1C)—CTLR 16-13
16.2.15 Interrupt Vector (2C30)—IVR 16-14
16.2.16 Input Port (2C34)—IP 16-14
16.2.17 Output Port Bit Set (2C38)—OP1 16-15
16.2.18 Output Port Bit Reset (2C3C)—OP0 16-15
16.2.19 SCC / IrDA Control (2C40)—SICR 16-16
16.2.20 Infrared Control 1 (2C44)—IRCR1 16-17
16.2.21 Infrared Control 2 (2C48)—IRCR2 16-18
16.2.22 Infrared SIR Divide (2C4C)—IRSDR 16-19
16.2.23 Infrared MIR Divide (2C50)—IRMDR 16-19
16.2.24 Infrared FIR Divide (2C54)—IRFDR 16-20
16.2.25 Rx FIFO Number of Data (2C58)—RFNUM 16-21
16.2.26 Tx FIFO Number of Data (2C5C)—TFNUM 16-22
16.2.27 Rx FIFO Data (2C60)—RFDATA 16-22
16.2.28 Rx FIFO Status (2C64)—RFSTAT 16-22
16.2.29 Rx FIFO Control (2C68)—RFCNTL 16-23
16.2.30 Rx FIFO Alarm (2C6E)—RFALARM 16-23
16.2.31 Rx FIFO Read Pointer (2C72)—RFRPTR 16-24
16.2.32 Rx FIFO Write Pointer (2C76)—RFWPTR 16-24
16.2.33 Rx FIFO Last Read Frame Pointer (2C7A)—RFLRFPTR 16-24
16.2.34 Rx FIFO Last Write Frame Pointer (2C7C)—RFLWFPTR 16-25
16.2.35 Tx FIFO Data (2C80)—TFDATA 16-25
16.2.36 Tx FIFO Status (2C84)—TFSTAT 16-25
16.2.37 Tx FIFO Control (2C88)—TFCNTL 16-26
16.2.38 Tx FIFO Alarm (2C8E)—TFALARM 16-26
16.2.39 Tx FIFO Read Pointer (2C92)—TFRPTR 16-27
16.2.40 Tx FIFO Write Pointer (2C96)—TFWPTR 16-27
16.2.41 Tx FIFO Last Read Frame Pointer (2C9A)—TFLRFPTR 16-27
16.2.42 Tx FIFO Last Write Frame PTR (2C9C)—TFLWFPTR 16-28
17.3 SPI Registers—MBAR + 0x0F00 17-4
17.3.1 Control Register 1 (0F00)—SPICR1 17-4
17.3.2 Control Register 2 (0F01)—SPICR2 17-5
17.3.3 Baud Rate Register (0F04)—SPIBR 17-6
17.3.4 Status Register (0F05)—SPISR 17-8

17.3.5 Data Register (0F09)—SPIDR 17-9
17.3.6 Port Reduced Drive/Pull-up Select (0F0C)—SPIPURD 17-9
17.3.7 Port Data Register (0F0D)—SPIPORT 17-9
17.3.8 Data Direction Register (0F90)—SPIDDR 17-10

18.3 I2C Interface Registers—MBAR + 0x3D00 18-8

18.3.1 Address Register 0 (3D00, 3D40)—I2C[1, 2] 18-8
18.3.2 Frequency Divider Register 1 (3D04, 3D44)—I2C[1, 2] 18-9
18.3.3 Control Register 2 (3D08, 3D48)—I2C[1, 2] 18-11
18.3.4 Status Register 3 (3D0C, 3D4C)—I2C[1, 2] 18-12
18.3.5 Data I / O Register 4 (3D10, 3D50)—I2C[1, 2] 18-13
18.3.6 Interrupt Control Register 8 (3D20) 18-14

19.4.1 IR Registers—MBAR + 0x0E00 19-3

19.4.1.1 Enable Control Register 0 (0E00) 19-4
19.4.1.2 Data Stream Control Register 1 (0E04) 19-5
19.4.1.3 Data Stream Format Register 2 (0E08) 19-6
19.4.1.4 Data Stream Format Register 3 (0E0C) 19-7
19.4.1.6 Preamble Register 5 (0E14) 19-8
19.4.1.7 Status Register 6 (0E18) 19-8
19.4.1.8 Dataword Transmitted Register 7 (0E1C) 19-9

19.5.1 Remote IR Registers—MBAR + 0x0e00 19-11

19.5.1.1 Enable Control Register 0 (0Exx) 19-11
19.5.1.2 Data Stream Control Register 1 (0Exx) 19-12
19.5.1.3 Data Stream Format Register 2 (0Exx) 19-13
19.5.1.4 Data Stream Format Register 3 (0Exx) 19-13
19.5.1.5 Data Stream Format Register 4 (0Exx) 19-14
19.5.1.6 Data Compare / Mask Register 5 (0Exx) 19-15
19.5.1.7 Data Compare / Mask Register 6 (0Exx) 19-15
19.5.1.8 Data Stream Format Register 7 (0Exx) 19-15
19.5.1.9 Status Register 8 (0Exx) 19-16
19.5.1.10 Dataword Received Register 9 (0Exx) 19-17

20.3 MSCAN Registers—MBAR + 0900 20-2

20.3.1 Control Register 0 (0900, 0980)—CAN[1, 2]CTL0 20-4
20.3.2 Control Register 1 (0901, 0981)—CAN[1, 2]CTL1 20-5
20.3.3 Bus Timing Register 0 (0904, 0984)—CAN[1, 2]BTR0 20-6
20.3.4 Bus Timing Register 1 (0905, 0985)—CAN[1, 2]BTR1 20-7
20.3.5 Rx Flag (0908, 0988)—CAN[1, 2]RFLG 20-8
20.3.6 Rx Interrupt Enable (0909, 0989)—CAN[1, 2]RIER 20-10
20.3.7 Tx Flag (090C, 098C)—CAN[1, 2]TFLG 20-11
20.3.8 Tx Interrupt Enable (090D, 098D)—CAN[1, 2]TIER 20-11
20.3.9 Tx Message Abort Request (0910, 0990)—CAN[1, 2]TARQ 20-12
20.3.10 Tx Message Abort Ack (0911, 0991)—CAN[1, 2]TAAK 20-12
20.3.11 Tx Buffer Select (0914, 0994)—CAN[1, 2]BSEL 20-13
20.3.12 ID Acceptance Control (0915, 0995)—CAN[1, 2]IDAC 20-13
20.3.13 Rx Error (091C, 099C)—CAN[1, 2]RXERR 20-14
20.3.14 Tx Error (091C, 099C)—CAN[1, 2]TXERR 20-14
20.3.15 ID Acceptance (0920–09B5)—CAN[1, 2]IDAR[0–7] 20-15
20.3.16 ID Mask (0928–09BD)—CAN[1, 2]IDMR[0–7] 20-16
20.3.17 Rx ID Register 0 (0940, 09C0)—CAN[1, 2]RXIDR0 20-17

20.3.18 Rx ID Register 1 (0941, 09C1)—CAN[1, 2]RXIDR1 20-17
20.3.19 Rx ID Register 0 (0940, 09C0)—CAN[1, 2]RXIDR0 20-18
20.3.20 Rx ID Register 1 (0941, 09C1)—CAN[1, 2]RXIDR1 20-18
20.3.21 Rx ID Register 2 (0944, 09C4)—CAN[1, 2]RXIDR2 20-18
20.3.22 Rx ID Register 3 (0945,09C5)—CAN[1, 2]RXIDR3 20-19
20.3.23 Rx Data Segment (0948–09D5)—CAN[1, 2]RXDSR[0–7] 20-19
20.3.25 Rx Time Stamp High (095C, 09DC)—CAN[1, 2]RXTIMH 20-20
20.3.26 Rx Time Stamp Low (095D, 09DD)—CAN[1, 2]RXTIML 20-21
20.3.27 Tx Buffer Priority (0979, 09F9)—CAN[1, 2]TXTBPR 20-21
20.3.28 Tx Time Stamp High (097C, 09FC)—CAN[1, 2]TXTIMH 20-22
20.3.29 Tx Time Stamp Low (097D, 09FD)—CAN[1, 2]TXTIML 20-22
20.3.30 Tx ID Register 0 (0960, 09E0)—CAN[1, 2]TXIDR0 20-23
20.3.31 Tx ID Register 1 (0961, 09E1)—CAN[1, 2]TXIDR1 20-23
20.3.32 Tx ID Register 2 (0964, 09E4)—CAN[1, 2]TXIDR2 20-24
20.3.33 Tx ID Register 3 (0965, 09E5)—CAN[1, 2]TXIDR3 20-24
20.3.34 Tx Data Segment (0968–09F5)—CAN[1, 2]TXDSR[0–7] 20-25
20.3.35 Tx Data Length (0978, 09F8)—CAN[1, 2]TXDLR 20-25
21.8.1.1 Device ID Register 21-10

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

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

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

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

freescale™
semiconductor